HUMAN-CENTRIC COMPUTER VISION FOR THE ARTIFICIAL INTELLIGENCE OF THINGS

by

Christopher Neff

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Electrical Engineering

Charlotte

2023

Approved by:

Dr. Hamed Tabkhi

Dr. Arun Ravindran

Dr. Ahmed Arafa

Dr. Vasily Astratov

Copyright Notes

promotional purposes or for creating new collective works for resale or redistribution, please go to here[1] to learn how to obtain a License from RightsLink.

In reference to Springer copyrighted material which is used with permission in this dissertation, the Springer does not endorse any of the University of North Carolina at Charlotte's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing Springer copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to here[2] to learn how to obtain a License from Springer.

[1]http://www.ieee.org/publications_standards/publications/rights/rights_link.html
[2]https://www.springer.com/gp/rights-permissions/obtaining-permissions/882

ABSTRACT

CHRISTOPHER NEFF. Human-Centric Computer Vision for the Artificial Intelligence of Things. (Under the direction of DR. HAMED TABKHI)

This dissertation presents a comprehensive exploration of innovative approaches and systems at the intersection of edge computing, deep learning, and real-time video analytics, with a focus on real-world computer vision for the Artificial Intelligence of Things (AIoT). The research comprises four distinct articles, each contributing to the advancement of AIoT systems, intelligent surveillance, lightweight human pose estimation, and real-world domain adaptation for person re-identification.

The first article, **REVAMP$_2$T: Real-time Edge Video Analytics for Multi-camera Privacy-aware Pedestrian Tracking**, introduces REVAMP^{2}T, an integrated end-to-end IoT system for privacy-built-in decentralized situational awareness. REVAMP^{2}T presents novel algorithmic and system constructs to push deep learning and video analytics next to IoT devices (i.e. video cameras). On the algorithm side, REVAMP^{2}T proposes a unified integrated computer vision pipeline for detection, re-identification, and tracking across multiple cameras without the need for storing the streaming data. At the same time, it avoids facial recognition, and tracks and re-identifies pedestrians based on their key features at runtime. On the IoT system side, REVAMP^{2}T provides infrastructure to maximize hardware utilization on the edge, orchestrates global communications, and provides system-wide re-identification, without the use of personally identifiable information, for a distributed IoT network. For the results and evaluation, this article also proposes a new metric, Accuracy•Efficiency (Æ), for holistic evaluation of AIoT systems for real-time video analytics based on accuracy, performance, and power efficiency. REVAMP^{2}T outperforms current state-of-the-art by as much as thirteen-fold Æ improvement.

The second article, **Ancilia: Scalable Intelligent Video Surveillance for the**

Artificual Intelligence of Things, presents an end-to-end scalable intelligent video surveillance system tailored for the Artificial Intelligence of Things. Ancilia brings state-of-the-art artificial intelligence to real-world surveillance applications while respecting ethical concerns and performing high-level cognitive tasks in real-time. Ancilia aims to revolutionize the surveillance landscape, to bring more effective, intelligent, and equitable security to the field, resulting in safer and more secure communities without requiring people to compromise their right to privacy.

The third article, **EfficientHRNet: Efficient and Scalable High-Resolution Networks for Real-Time Multi-Person 2D Human Pose Estimation**, focuses on the increasing demand for lightweight multi-person pose estimation, a vital component of emerging smart IoT applications. Existing algorithms tend to have large model sizes and intense computational requirements, making them ill-suited for real-time applications and deployment on resource-constrained hardware. Lightweight and real-time approaches are exceedingly rare and come at the cost of inferior accuracy. This article presents EfficientHRNet, a family of lightweight multi-person human pose estimators that are able to perform in real-time on resource-constrained devices. By unifying recent advances in model scaling with high-resolution feature representations, EfficientHRNet creates highly accurate models while reducing computation enough to achieve real-time performance. The largest model is able to come within 4.4% accuracy of the current state-of-the-art, while having 1/3 the model size and 1/6 the computation, achieving 23 FPS on Nvidia Jetson Xavier. Compared to the top real-time approach, EfficientHRNet increases accuracy by 22% while achieving similar FPS with $\frac{1}{3}$ the power. At every level, EfficientHRNet proves to be more computationally efficient than other bottom-up 2D human pose estimation approaches, while achieving highly competitive accuracy.

The final article introduces the concept of **$\mathbf{R_2}$OUDA: Real-world Real-time Online Unsupervised Domain Adaptation for Person Re-identification**. Fol-

lowing the popularity of Unsupervised Domain Adaptation (UDA) in person re-identification, the recently proposed setting of Online Unsupervised Domain Adaptation (OUDA) attempts to bridge the gap towards practical applications by introducing a consideration of streaming data. However, this still falls short of truly representing real-world applications. The R^{2}OUDA setting sets the stage for true real-world real-time OUDA, bringing to light four major limitations found in real-world applications that are often neglected in current research: system generated person images, subset distribution selection, time-based data stream segmentation, and a segment-based time constraint. To address all aspects of this new R^{2}OUDA setting, this paper further proposes Real-World Real-Time Online Streaming Mutual Mean-Teaching (R^{2}MMT), a novel multi-camera system for real-world person re-identification. Taking a popular person re-identification dataset, R^{2}MMT was used to construct over 100 data subsets and train more than 3000 models, exploring the breadth of the R^{2}OUDA setting to understand the training time and accuracy trade-offs and limitations for real-world applications. R^{2}MMT, a real-world system able to respect the strict constraints of the proposed R^{2}OUDA setting, achieves accuracies within 0.1% of comparable OUDA methods that cannot be applied directly to real-world applications.

Collectively, this dissertation contributes to the evolution of intelligent surveillance, lightweight human pose estimation, edge-based video analytics, and real-time unsupervised domain adaptation, advancing the capabilities of real-world computer vision in AIoT applications.

Keywords: Computer Vision, Deep Learning, AIoT, Video Surveillance, Edge Computing, Video Analytics, Pedestrian Tracking, Action Recognition, Anomaly Detection, Privacy, Ethics, Human Pose Estimation, High-Resolution Networks, Model Scaling, Real-Time, Lightweight, Person Re-identification, Online Learning, Unsupervised Learning, Domain Adaptation, Real-World, Real-Time, Domain Shift

ACKNOWLEDGEMENTS

I am profoundly grateful for the unwavering support and guidance of my advisor, Dr. Hamed Tabkhi. His mentorship not only enabled the research within this dissertation but also ignited my passion for graduate studies. The profound impact of his unwavering commitment, dedication, and inspiration on my academic journey cannot be overstated. Without him, I would not be where I am today.

I extend my heartfelt appreciation to my esteemed committee members: Dr. Arun Ravindran, Dr. Ahmed Arafa, and Dr. Vasily Astratov, for their invaluable insights and contributions to this work.

My academic journey has been enriched by countless individuals, each contributing in their unique way. I owe sincere thanks to Reza Baharani, Samuel Rogers, Matias Mendieta, Aneri Sheth, Steve Furgurson, Justin Sanchez, Amrin Danesh Pazho, Ghazal Alinezhad Noghre, Vinit Kataria, Babak Ardabili, Shanle (Eric) Yao, Jeri Guido, and Dr. Shannon Reid. Your collaboration, mentorship, and camaraderie have been instrumental in my growth and success.

I would like to acknowledge the generous support of the National Science Foundation (NSF) under Award No. 1831795 and the NSF Graduate Research Fellowship Award No. 1848727. This funding provided essential resources for conducting the research presented in this dissertation.

As I move forward in my academic and professional journey, I am deeply aware that the influence of these remarkable individuals will continue to shape my future endeavors. Their contributions have been the cornerstone of my academic achievements, and for that, I am profoundly grateful.

DEDICATION

In heartfelt gratitude, I extend special thanks to:

My brother and sister-in-law, Sean and Lauren, who have been unwavering pillars of strength throughout this journey. In the midst of life's challenges and triumphs, including loss of our closest family members, they stood by me with resilience and unwavering support.

To Greg and Elizabeth, true allies and confidants, whose friendship and encouragement have been the cornerstone of my academic pursuit. Without their unwavering support, I would not have reached this momentous milestone.

I also want to acknowledge my extensive circle of friends, both those I have the privilege of knowing in person and those I've had the pleasure of meeting virtually. Their unwavering support, camaraderie, and occasional delightful distractions from the world of academia have nurtured my mental and social well-being throughout this journey.

As I embark on new horizons and future endeavors, I carry with me the enduring impact of these remarkable individuals, who have shared in my challenges and celebrated my victories. Their presence in my life has been an irreplaceable source of inspiration and strength, and for that, I am profoundly thankful.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

2D	Two Dimensional.
5G	Fifth Generation of cellular networks.
Æ	Accuracy•Efficiency.
ADG	Automated Dataset Generation.
AI	Artificial Intelligence.
AIoT	Artificial Intelligence of Things
AP	Average Precision.
AUC	Area Under the Curve
AWS	Amazon Web Services
BN	Batch-Normalization.
CMC	Cumulative Matching Characteristics.
CMU	Carnegie Mellon University.
CNN	Convolutional Neural Network.
CPU	Central Processing Unit.
CUDA	Compute Unified Device Archetecture.
DA	Domain Adaptation
DBSCAN	Density-based Spatial Clustering of Applications with Noise
DeConv	Deconvolution
EER	Equal Error Rate

FLOPs	Floating Point Operations.
FPS	Frames per Second.
GAN	Generative Adversarial Netwok
GCN	Graph Convolutional Network.
GeMM	General Matrix Multiply.
GPU	Graphical Processing Unit.
GUI	Graphical User Interface.
ID	Identity.
IDF1	Identity F1-score.
IDP	Identity Precision.
IDR	Identity Recall.
IoT	Internet of Things.
IoU	Intersection Over Union.
LR	Learning Rate.
LSTM	Long Short-Term Memory.
LTE	Long-Term Evolution.
mAP	Mean Average Precision.
MMT	Mutual Mean-Teaching
MSE	Mean Squared Error.
MTMC	Multi-Target Multi-Camera.

ONNX	Open Neural Network Exchange.
ONNX	Open Neural Network Exchange
OUDA	Online Unsupervised Domain Adaptation
Params	Parameters
PII	Personally Identifiable Information.
PNG	Portable Network Graphics.
PoE	Power over Ethernet.
PR	Precision-Recall
R^2MMT	Real-world Real-time Mutual Mean-Teaching
R^2OUDA	Real-world Real-time Online Unsupervised Domain Adaptation
RAM	Random Access Memory
ReID	Re-Identification.
ReLU	Rectified Linear Unit.
REVAMP^{2}T	Real-time Edge Video Analytics for Multi-person Privacy-aware Pedestrian Tracking.
RGB	Red Green Blue.
RMPE	Regional Multi-Person Pose Estimation
RNN	Recurrent Neural Network.
ROC	Region of Convergence
RoI	Region of Interest.

RW-GCN	Real-World Graph Convolutional Network.
SDS	Subset Distribution Selection
SGD	Stochastic Gradient Descent.
SoC	System on Chip.
SotA	State-of-the-Art.
SQL	Structured Query Language.
ST-GCN	Spatial-Temporal Graph Convolutional Network.
TCP	Transmission Control Protocol.
UDA	Unsupervised Domain Adaptation
VRAM	Video Random Access Memory

CHAPTER 1: Introduction

1.1 Motivation

In the ever-evolving landscape of technology, the advent of the Internet of Things (IoT) has ushered in a new era of computational challenges, punctuated by the scarcity of bandwidth resources and an unwavering demand for low-latency responsiveness. At the forefront of this transformation stands edge computing, complemented by the concept of fog computing [10], which collectively define a paradigm shift towards cooperative computation at the network's edge [11, 12]. However, the convergence of these technologies finds its true purpose in the realm of ambient computer vision and real-time video analytics. These are the domains where human-like vision processing across vast geographic areas has become not just a technical aspiration, but an imperative for applications ranging from smart cities to surveillance [12, 13, 14, 15].

The stage for this transformation is set by the remarkable advances in machine learning, especially the realm of deep learning, which have propelled the development of sophisticated video analytics and surveillance technologies. These encompass an array of applications, from license plate recognition to the intricate domains of facial recognition and pedestrian tracking. Historically, these applications relied heavily on a cloud computing paradigm, where video collection and processing were concentrated on centralized servers. Yet, this paradigm introduces pressing technical and ethical concerns.

From a technical perspective, the cloud-based approach imposes significant burdens. It leads to the massive recording, transmission, and storage of raw video data, incurring substantial costs and imposing constraints on scalability. Equally, it falls short in addressing the critical requirements of real-time and time-sensitive video

analytics, where split-second decisions can make all the difference.

On the ethical front, the indiscriminate capture of vast volumes of personal information has elicited substantial resistance from privacy advocates. This has, in turn, culminated in outright bans on facial recognition and tracking technologies in numerous U.S. cities [16], as well as heightened regulatory scrutiny in the European Union [17]. In response, there is a growing consensus that a holistic approach is imperative. It must encompass the realms of IoT system design, privacy preservation, computation mapping, communication, and system-level synchronization, all the way from the inception of algorithms to their real-time execution.

This is the backdrop against which REVAMP$_2$T (Real-time Edge Video Analytics for Multi-person Privacy-aware Pedestrian Tracking) emerges as a beacon of innovation. This solution envisions a future where human-like vision processing occurs seamlessly and securely at the network's edge, without infringing upon individuals' privacy. It stands as a testament to our commitment to striking the delicate balance between real-time tracking, privacy preservation, and computational efficiency.

Yet, REVAMP$_2$T is just one facet of our mission. The demand for effective surveillance to protect our cities, citizens, and vital infrastructure remains unabated. As incidents in Itaewon, South Korea [18], and Moore County, North Carolina [19], have shown, the need for proactive, intelligent surveillance systems is undeniable. In many of these cases cameras are already prevalent in the target environment, but installing edge based processing on top of the existing infrastructure is not a feasible solution. In this context, there is a need for a scalable solution that can run in the fog - that is a centralized location on premise located between the edge and the cloud. In this context, Ancilia emerges as an innovative surveillance system, designed to offer real-time situational awareness and threat detection using scalable fog intelligence that can retrofit to existing installations. Ancilia leverages advancements in computer vision to perform high level cognitive tasks, such as action recognition and anomaly

detection, advancing the field towards truly autonomous surveillance.

For such applications, human motion data is critical. In REVAMP$_2$T, OpenPose [20] was used to collect human pose data, but its large computational demands dominated the system design and greatly limited performance on edge devices. In Ancilia, the much smaller HRNet [21] was used, but as a top-down method it scaled poorly with the number of people in the scene, becoming a performance bottleneck in high traffic environments. This highlights the need for real-time multi-person pose estimation that is both lightweight enough to run at the IoT edge and accurate enough to provide quality data to downstream tasks. It is here that EfficientHRNet enters the arena, aiming to bridge the gap by delivering real-time, lightweight human pose estimation models that rival the accuracy of state-of-the-art approaches.

Lastly, person re-identification (ReID) emerges as a pivotal task with applications spanning video surveillance, public safety, and smart health [22]. While deep learning has elevated accuracy to remarkable levels, the challenge remains in adapting these models to real-world scenarios riddled with domain shift and streaming data complexities. Models trained on datasets have greatly diminished accuracy in real-world scenarios, due to the highly context specific nature of person re-identification. Manually labeling data for each new environment is a monumentally expensive and practically infeasible task, while current unsupervised methods have diminished accuracy compared to supervised approaches. Online unsupervised domain adaptation shows promise, but current method fail to account for the many challenges of real-world implementation. R$_2$OUDA, or Real-World Real-Time Online Unsupervised Domain Adaptation, steps up to address these domain adaptation challenges in the dynamic context of streaming data, aligning our research more closely with the demands of practical applications.

Collectively, these research efforts weave the narrative of an evolving Computer Vision landscape in the Artificial Intelligence of Things (AIoT). It is a landscape

that demands not only technological prowess but also a profound commitment to ethics, privacy, and real-world feasibility. As we confront the complexities of AIoT algorithmic and system design and the imperative of safeguarding privacy, we seek to pave the way for an interconnected, intelligent, and secure future. It is a future where the boundaries of possibility are redrawn by innovation, one algorithm at a time.

1.2 Contributions to the Body of Knowledge

Overall, this dissertation encompasses a broad range of transformative research, each contributing significantly to the evolving landscape of the Artificial Intelligence of Things (AIoT). The following contributions represent the key findings and innovations presented in this work:

- This dissertation introduces the novel Real-time Edge Video Analytics for Multi-camera Privacy-aware Pedestrian Tracking, or REVAMP^{2}T. This innovative system achieves multi-camera pedestrian tracking without transferring raw video or personally identifiable information. REVAMP^{2}T prioritizes privacy rights and complies with recent EU legislation, generating unique pedestrian identities for tracking and safeguarding privacy. It demonstrates exceptional pedestrian re-identification accuracy, competing with the state-of-the-art while operating at significantly higher real-time frames per second (FPS) and lower power consumption. The novel metric Accuracy•Efficiency (Æ) is introduced to balance algorithmic accuracy and system efficiency, highlighting REVAMP^{2}T's scalability and privacy-awareness in multi-camera IoT environments.

- This work presents Ancilia, the first end-to-end scalable intelligent video surveillance system designed to perform high-level cognitive tasks in real-time. Ancilia leverages existing IoT camera ecosystems, processing video data on the edge to minimize latency and privacy concerns associated with cloud computing. It addresses ethical concerns by strictly avoiding the storage of personally identifi-

able information and invasive techniques like facial recognition. Ancilia focuses on pose and locational information, contributing to unbiased surveillance. Empirical evaluations demonstrate Ancilia's capability to achieve state-of-the-art results, delivering high-level cognitive tasks with minimal accuracy deviation. Ancilia serves as a privacy-conscious, effective, and scalable solution for real-world intelligent surveillance.

- This dissertation introduces EfficientHRNet, a family of lightweight scalable networks for high-resolution and efficient real-time bottom-up multi-person pose estimation. EfficientHRNet combines the principles of EfficientNet and HRNet to provide near-state-of-the-art human pose estimation while being more computationally efficient than other bottom-up methods. It achieves higher accuracy with lower computational complexity and parameter requirements, improving real-time performance. Empirical evaluations on the COCO dataset demonstrate EfficientHRNet's competitive accuracy and efficiency, highlighting its suitability for resource-constrained edge devices.

- To bridge the gap between research and real-world applications, this work introduces the setting of Real-World Real-Time Online Unsupervised Domain Adaptation (R^2OUDA). R^2OUDA addresses the unique challenges of real-world applications, considering algorithmic generation of person images, data distribution selection, time-based data streams, and segment-based time constraints. Real-World Real-Time Online Streaming Mutual Mean-Teaching (R^2MMT) is proposed as an end-to-end multi-camera system designed for real-world person re-identification within the R^2OUDA setting. R^2MMT captures the complexities of real-world applications, achieving high accuracy while accommodating noisy data and real-time constraints. Exhaustive experimentation demonstrates R^2MMT's potential to meet the demands of real-world applications, achieving

highly competitive accuracy within the R^2OUDA setting.

In summary, this dissertation contributes significantly to the evolving field of Computer Vision for the AIoT by introducing novel technologies and methodologies in privacy-aware tracking, intelligent surveillance, lightweight human pose estimation, and real-world domain adaptation for person re-identification. These contributions offer innovative solutions, address ethical concerns, and enhance the efficiency of real-time applications, providing valuable insights for future research and real-world applications.

CHAPTER 2: REVAMP^{2}T: Real-time Edge Video Analytics for Multi-camera Privacy-aware Pedestrian Tracking

2.1 Introduction

The emerging wave of Internet of Things (IoT), scarcity of bandwidth resources, and tight latency awareness is pushing system designers to extend cloud computing to the edge of the network. Edge computing (and also fog computing [10]) refers to a group of technologies allowing cooperative computation at the edge of the network [11, 12]. Ambient computer vision and real-time video analytics are the major classes of applications that requires edge computing for human-like vision processing over a large geographic area [12, 13, 14, 15].

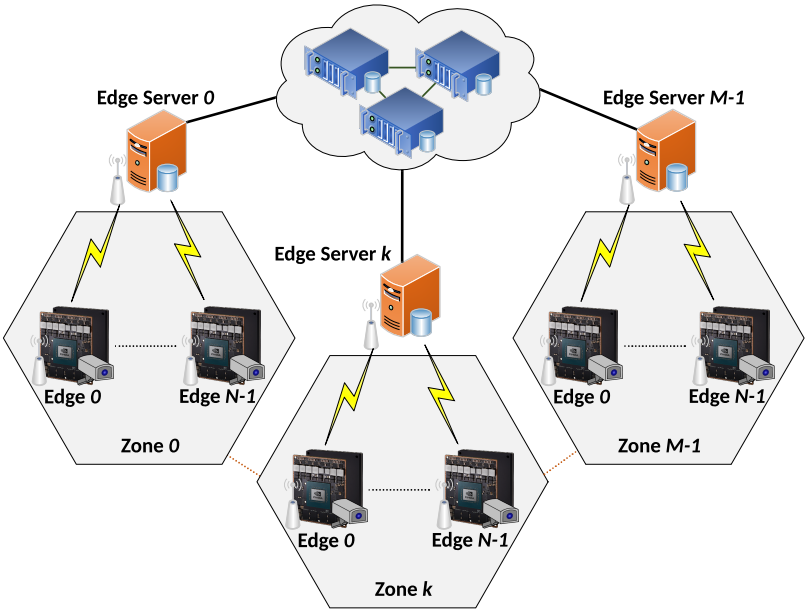

Figure 2.1: Hierarchical System Overview

Recent advances in machine learning, particularly deep learning, have driven the development of more advanced video analytics and surveillance technologies. This

includes everything from simple license plate scanners that search for stolen vehicles, to facial recognition and pedestrian tracking. These applications often rely on a cloud computing paradigm for mass video collection and processing on a centralized computing server. The cloud computing paradigm introduces significant technical and social/ethical concerns for such applications. On the technical side, cloud computing leads to mass recording and storage of raw video data, which result in significant costs and limits scalability. At the same time, cloud computing is not applicable to many inherently real-time and time-sensitive video analytics.

On the social perspective, the broad net cast by typical surveillance approaches means that large amounts of personal information are incidentally collected and stored. This has led to significant push-back by privacy advocates against any expansions to video surveillance systems As an example, multiple cities in the US have imposed bans on all deployment of facial recognition and tracking technologies [16]. European Union regulators are also considering new restrictions on AI-driven surveillance [17]. To address both technical and ethical concerns, novel approaches are required to address both IoT systems design and privacy challenges in a holistic manner across entire computing stack from algorithm design to computation mapping, communication, and system-level synchronization.

This paper introduces novel Real-time Edge Video Analytics for Multi-camera Privacy-aware Pedestrian Tracking, or REVAMP^{2}T. REVAMP^{2}T is able to track pedestrians across multiple cameras without ever transferring raw video or other forms of personally identifiable information. Fig.2.1 presents our proposed REVAMP^{2}T IoT system. Each IoT device (edge nodes) contains cameras equipped with the NVIDIA AGX Xavier [23] embedded platform, running a deep learning based video analytics pipeline, for real-time pedestrian detection and tracking over streaming pixels. In keeping with the concept of the "right to be forgotten" that was recently enshrined in EU law, our system does not rely on a static identity database. Instead, unique

identities are generated when pedestrians first enter the view of a camera in our system and forgotten when those individuals are no longer being actively tracked by any part of our system.

Overall, REVAMP^{2}T achieves a pedestrian re-identification accuracy of 74.8% (only 4.3% below the current state-of-the-art [24]) on the DukeMTMC dataset [25], while achieving more than two times the real-time FPS and consuming 1/5th of the power compared to [24]. A balance was struck between algorithmic Accuracy and system Efficiency, measured by Accuracy•Efficiency (Æ). Our system has high scalability potential in a multi-camera IoT environment while never sacrificing personal privacy.

2.2 Related Work

2.2.1 Pedestrian Detection, Re-Identification, and Tracking

With the rapid advancements made in deep learning, a plethora of work has been published on pedestrian detection. Such models include region proposal networks like Faster-RCNN [26], single shot detectors like SSD [27] and YOLO [28], as well as pose-estimation models like DeeperCut [29] and OpenPose [20]. When analyzing these algorithms in light of edge-capable real-time performance, MobileNet-SSD, TinyYOLOv3, and OpenPose show promising results.

The heart of pedestrian tracking is consistent re-identification (ReID) of those pedestrians throughout the frames of videos across multiple cameras. Similarly, on the re-identification side, recent methods leverage CNNs to extract unique features among persons [30, 31, 32, 33, 34, 35, 36, 37]. The work in [38] learns the spatial and temporal behavior of objects by translating the feature map of the Region of Interest (RoI) into an adaptive body-action unit. [39] uses bidirectional Long-Short-Term-Memory (LSTM) neural networks to learn the spatial and temporal behavior of people throughout the video. Triplet loss [40, 41, 42] is another promising technique to train the network with the goal of clustering classes in a way that IDs with the same type have minimum distance among each other, while examples from different

categories are separated by a large margin.

Pedestrian tracking systems often rely on prediction models to create insight on the changes in movement over time and empowers object re-identification. Object tracking has been tried using spatial masking and Kalman filter techniques for single and multiple object tracking [43, 44, 45]. In contrast, there is an interest in leveraging LSTM networks for prediction and tracking. One pronounced example is ROLO [46], which uses YOLOv1 as its feature extractor, combined with LSTMs. Similarly, [47] uses VGG-16 for feature extraction and inputs the 500x1 feature vector into an LSTM. LSTM networks have been shown to provide lower Mean Squared Error in single object and fewer ID switches in multi-object tasks. However, the approaches in [47] and [46] often show very low accuracy as they are not customized for human objects.

Overall, the current state of pedestrian tracking algorithms struggle with limited focus and lack of privacy preservation. First, they look at the problem of pedestrian tracking in isolation, whether solely by detection, only re-identification using image crops, or just tracking with trajectories. However, these approaches do not analyze the problem in a holistic manner, which would require designing a pipeline to understand, integrate, and correlate these three functions into a single intelligent system. Second, the idea of privacy preservation and online functionality are lost with this narrowly focused approach. The previous works typically rely on the storage of large time segments of video data or image crops, degrading privacy preservation. Similarly, many works propose facial recognition techniques [48, 49, 50, 51, 52], which also gravely compromises the privacy of tracked persons, requiring the pre-loaded and long-term storage of personally identifiable information like a facial database. At the same time, existing approaches typically analyzes the data offline with the ability to move forward and backward in time to maximize their algorithm accuracy scores, making edge deployable operation of these approaches impractical. In contrast to

existing work, this article proposes a shift to non-personal and data private pedestrian tracking, improving upon our previous work in re-identification [53] and LSTM tracking [54] for a holistic algorithm pipeline and fully edge capable design.

2.2.2 IoT Systems for Edge Video Analytics

The concept and motivation behind edge computing has been described in a number of recent publications [10, 12, 13, 14, 15, 55, 56, 57, 58]. However, there are very few works that present a distributed IoT system for video analytics and real-time tracking. [59] proposes a basic system framework for vehicle detection and tracking across multiple cameras. The approach uses positional matching for re-identification, relying on GPS coordinates, known distances, and time synchronization between cameras. The Gabriel project from CMU [60] is a wearable cognitive assistance system where the images captured by a mobile device are processed by the edge node to analyze what the user is seeing, and provide the user with cues as to what is in the scene (for example, recognizing a person). In the VisFlow project from Microsoft, Lu et al. [61] describe a system that can analyze feeds from multiple cameras. In particular, they describe a dataflow platform for vision queries that is built on top of the SCOPE dataflow engine, that offers general SQL syntax, and supports added user-defined operators such as extractors, processors, reducers, and combiners. [62] proposed a method for single camera multi-target tracking in terms of the Binary Integer Program, and can incur online, real-time results on hardware. However, the system does not scale to multi-camera systems. The current state-of-the-art in Multi-Target Multi-Camera (MTMC) systems is DeepCC [24]. This approach uses OpenPose for detections, a deep learning triplet loss ReID network for visual data association, and trajectory tracklets.

In contrast to Gabriel, REVAMP^{2}T targets machine vision at the edge involving multiple cameras distributed across a geographical area. Unlike VisFlow, where all processing is done at the cloud, REVAMP^{2}T performs a considerable amount of the

processing at the edge nodes (next to the camera) by custom-designed deep learning based vision engines, thereby decreasing bandwidth requirements. This also allows REVAMP^{2}T to protect the privacy of the tracked individuals, On the other hand, cloud-based systems must transfer personally identifiable information across large, interconnected networks and store that information in the cloud, where it is all too vulnerable. Additionally, the proposed edge vision system scales easily to a large number of cameras distributed over a wide geographic region.

REVAMP^{2}T accomplishes all these tasks online, in real-time, on low-power edge devices.

2.3 Privacy Requirements and Threat Modeling

This section describes the privacy threat models which REVAMP^{2}T is designed to address. In safeguarding privacy, we wish to protect the identities and Personally Identifiable Information (PII) of the individuals being viewed by our system. This is most commonly in the form of raw image data, but can also refer to meta-data that can be used to determine the race, gender, nationality, or even identity of an individual. There are three main threats to this that we attempt to address:

- The external threat of someone getting unintended access to network communications and retrieving image data or Personally Identifiable Information (PII).
- The internal threat of someone with authorized access to the system viewing image data or PII - even someone who is supposed to have access to the system should not be able to discern the identities of individuals or have access to their personal information.
- The physical threat of someone getting physical access to the edge device.

To safeguard against these threats, we impose two major policies for designing REVAMP^{2}T:

(1) REVAMP^{2}T will not store any image data or transfer it across the network. As soon as the image is processed on the edge node, it is destroyed. This protects any PII in the images from being viewed by anyone with access to the system. Even with direct access to the edge node, image data never touches non-volatile memory, so accessing it is impossible without fundamentally changing the semantics of our system.

(2) REVAMP^{2}T re-identification algorithm will work on an encoded feature representation of an individual (without using facial recognition algorithms). These features represent the visual and structural attributes of an individual, but can not be interpreted by humans and has zero meaning outside the constraints of our system. This means that even if a person gains access to this feature representation, intended or otherwise, they can not learn anything about the appearance or identity of the individual it was derived from. By utilizing these feature representations, REVAMP^{2}T is able to focus on *differentiation* between people rather than personal *identification*. This is in contrast to common methods that rely on facial recognition or other PII [48, 49, 50, 51, 52].

We design REVAMP^{2}T, with respect to defined privacy protection policies. Section 2.4 and Section 2.5 present algorithmic constructs and IoT system design of REVAMP^{2}T.

2.4 REVAMP^{2}T: Algorithmic Constructs

This section presents algorithmic constructs to enable real-time pedestrian re-identification and tracking while satisfying our privacy model detailed in Section 2.3.

Figure. 2.2 outlines the full algorithmic pipeline. The pipeline consists of three primary phases: (1) Detection, (2) Re-identification, and (3) Tracking and Prediction. For the detection part, we chose OpenPose [20] from the CMU Perceptual Computing Lab. OpenPose is a pose prediction framework that uses part affinity fields to understand the image input and provide person detections with marked keypoint

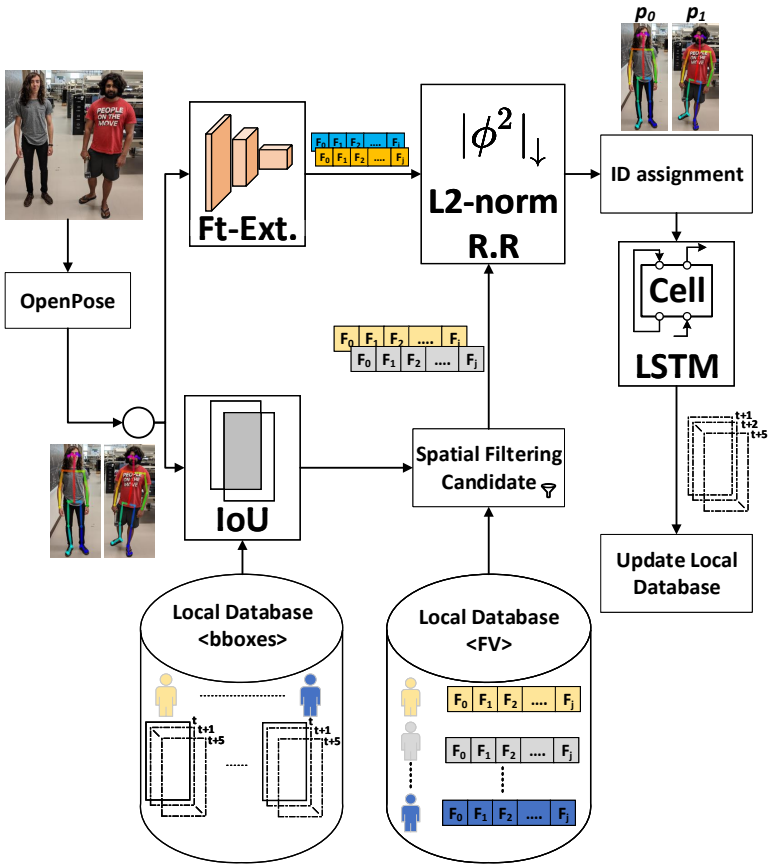

Figure 2.2: Algorithm Pipeline on the Edge

locations. In addition, it provides keypoints that reveal the motion of the human body, making them useful for motion prediction and action recognition. In the re-identification portion, discriminative features are generated for detection comparison and matching in the Local Database. Once the re-identification has been completed, an LSTM network is applied to predict future positions of known detections. The rest of this section discusses the technical details of our proposed re-identification, tracking, and integration.

2.4.1 Feature Extractor Network

The core of the re-identification is the feature extraction network to extract discriminative features from each detection, represented by the Ft-Ext. box in Figure. 2.2. For this task, a deep convolution network had to be developed for accurate,

real-time performance. Most deep convolution networks have a massive number of parameters and operations, which makes them computationally expensive for use in mobile and embedded platforms. MobileNet-V1 [63] and MobileNet-V2 [64] are two developed light-weight deep convolution networks which effectively break down a standard convolution into a depth-wise and point-wise convolution to decrease the network parameters and operations. MobileNet-V2 further improved the MobileNet-V1 architecture by adding linear bottleneck layers and inverted residual connections.

In this article, we use the MobileNet-V2 model and change the fully connected layer to a 2D average pooling with the kernel size of (8, 4) in order to make the output of the network a 1x1280 vector as the embedded appearance features. We also use the triplet loss function [42] to train the MobileNet-V2 for extraction of discriminative features based on person appearance. The underlying architecture of a triplet loss network is consisted of three identical networks which transform the cropped RoI into embedding on a lower dimensional space. One RoI is the anchor image, the second is a positive sample of the anchor and third is a negative sample. The basic concept here is to minimize the distance between the anchor and the positive samples and maximize the distance between the anchor and the negative samples in the lower dimensional embedding space. To facilitate such learning, a suitable loss function is used after the embeddings are extracted from the RoIs:

$$Loss = \sum_{i=1}^{n} \left[\alpha + \|f_i^a - f_i^p\|^2 - \|f_i^a - f_i^n\|^2 \right]_+, \tag{2.1}$$

where α is margin, f^a, f^p, and f^n are embedded appearance features of the anchor, positive, and negative samples for the class i, respectively. Minimizing $Loss$ function will force all samples of class i to be inside of hypersphere of radius α. The dimension of the hypersphere is equal to the size of the network output (1280 for MobileNet-V2).

To further improve the performance of MobileNet-V2, we have assigned error-friendly operations, such as convolution and General Matrix Multiply (GeMM) operations, to half precision which is 16-bit floating point accuracy and applied mixed precision training [65] to minimize the error caused by half precision operations.

2.4.2 Pedestrians Tracking

After the re-identification process, we send the current detections to the LSTM network to get their respective bounding box predictions for the coming five frames. In this way, we can handle miss-detections that the detection network might suffer as it is running at the edge (lower detection network resolution). At the same time, we are able to handle short-term occlusions as we know the position of the occluded pedestrian and can ReID them once they are back in the scene.

To efficiently train our LSTM, we chose the DukeMTMC [25] dataset for training the module as it focuses on pedestrian tracking. The dataset involves multiple targets, and we curate single target instances from the dataset so that we do not involve the re-identification pipeline for the training phase. This results in the network being trained on single instances and carrying out inference on multiple targets. Also, using this technique helps us to re-use the same model parameters for multiple pedestrians when predicting their future positions, saving us redundant computation and making our LSTM tracking module scalable. We leverage the sequence learning capabilities of LSTM by providing it with consecutive frame keypoints of a set sequence length and minimizing the mean-squared error between the obtained predictions and the ground-truth positions of the next five frames.

Fig 2.3 shows our LSTM module in detail. We provide it with the keypoints of three (our sequence length) consecutive frames and send the last step output to the fully-connected layer which encodes these keypoints to the bounding box position of the pedestrian for the next five frames. The size of our trained LSTM model is under 1.5 MB, making it suitable to run on edge devices.

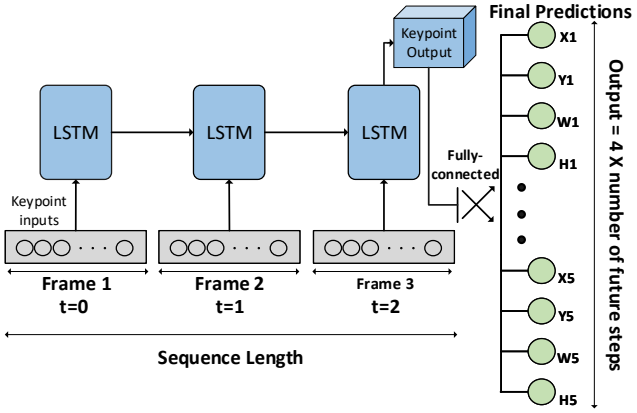

Figure 2.3: LSTM training: Feeding a sequence of frames and getting the predicted bounding box predictions

2.4.3 Integration of Video Analytic Pipeline

In order for the entire re-identification task to be accomplished on the edge, these modules must be integrated seamlessly together. Referring back to Figure. 2.2, a frame is inputted from the camera feed directly into the detection network. The resulting detections are received, scaled to the appropriate size and aspect ratio, and batched through the feature extractor (FT-Ext. box, Figure. 2.2). The output of this network provides the encoded 1x1280 Feature Vector for each detection. In parallel, spatial filtering is being done on the Local Database (southern portion of Figure. 2.2). For each detection, a subset of candidate matches are chosen based on IoU with last known or predicted bounding boxes from previous Local Database entries. The intuition behind this filtering mechanism is to ensure that detections are matched with entries that not only match in the embedded space (Feature Vector), but also in location and trajectory. Because the entire pipeline is running many times per second, the likelihood of a pedestrian traversing a substantial amount of distance or drastically changing trajectory between processed frames is low. Therefore, we avoid including entries that do not make sense from a positional standpoint in the candidate pool for ReID on a new detection.

Within the subset of candidates, the L2-norm operation can be done between Feature Vectors to differentiate between entries, and make final matching decisions using a re-ranking approach to ensure optimal ID assignment (L2-norm R.R box in Figure. 2.2). As described above, the feature extraction network was trained to maximize the euclidean distance between Feature Vectors of different pedestrians, and minimize the distance between vectors of the same pedestrian. This training and inference methodology provides a privacy-aware approach to ReID, as per our threat models. Rather than using specialized, personally identifiable blocks of information to continually re-identify a pedestrian, our model simply encodes the current visual features of a detection to an abstract representation, and focuses on *differentiation* between entries rather than *personal* identification. Once all detections in the scene are assigned, the LSTM described previously takes in the detection keypoints and generates predicted bounding boxes for the next five time intervals. Finally, the Local Database is updated with assigned labels, keypoints, Feature Vectors, and generated predictions from the processed frame.

2.5 REVAMP^{2}T: System Constructs

Creating algorithms that can effectively solve issues while running on low-power devices is of vital importance to enable inference on the edge. However, there are many system-level considerations that must be taken into account when developing a robust end-to-end system. How data flows between algorithms, when and how to utilize said algorithms, how to handle communications between the edge node and edge server, and how to map and optimize processes to and for the underlying hardware available on the edge. All of these are system-level design decisions that greatly impact the efficiency and viability of the end-to-end system. With REVAMP^{2}T's focus on privacy, it was important that the IoT system was designed around never storing any personally identifiable information. Algorithmic selections and the design of the system's processing flow hinged around that constraint.

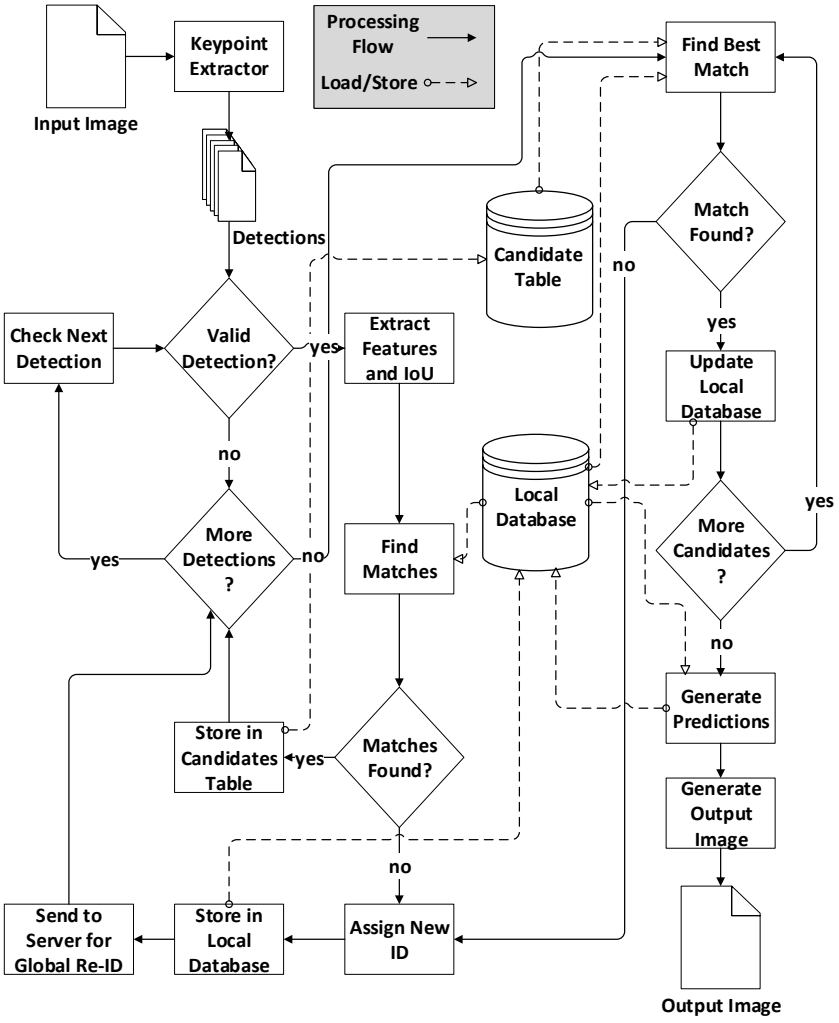

Figure 2.4: Processing Flow of the Edge

2.5.1 System Hyperparameters and Processing Flow

Figure. 2.4 shows the logical processing flow of one frame of data on the edge, beginning at when the image is extracted from the camera to when the final output is displayed on the edge device. First, the image is run through the keypoint extractor, which outputs a vector of detections. To remove false detections, each detected pedestrian should have a minimum number of keypoints equal to θ_{key}, and each of those keypoints a confidence value of at least θ_{conf}.

Table 2.1 presents the system configurable hyperparameters. For every valid detec-

Table 2.1: System Parameters

Parameter	Description
θ_{key}	Minimum keypoints for valid detection
θ_{conf}	Confidence threshold for valid keypoint
θ_{euc}	Euclidean threshold with IoU
β_{euc}	Euclidean threshold without IoU
θ_{IoU}	IoU threshold for updating Feature Vector
$\bar{D}$	Detection from keypoint extractor
$\bar{V}$	Valid Detections
$\widehat{DB}$	Local database
$\widehat{C}$	Candidate matrix
ζ	Global variable to keep track of new ID

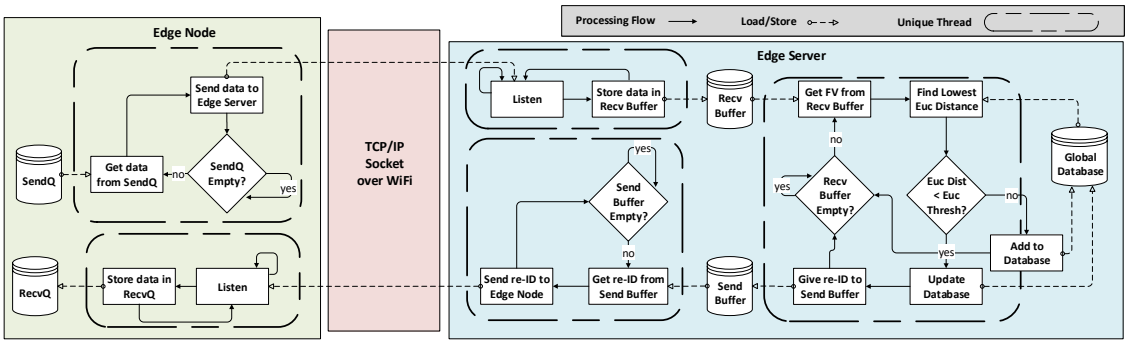

Figure 2.5: Edge Node to Edge Server Communications

tion, all possible matches for ReID are gathered from the Local Database, as discussed in Section 2.4.3. When a potential match is found, the detection, database entry index, and the Euclidean Distance between the two are stored in the Candidate Table. If no potential matches were found for a detection, it is considered to be a new person, assigned a new Local ID, stored in the Local Database, and sent to the server for Global ReID.

After all detections have been processed and the Candidate Table filled, the ReID processing is completed and IDs assigned, as shown in Algorithm 2. The lowest Euclidean Distance score in the Candidate Table is found, the detection assigned the ID it was matched to, and the Local Database updated accordingly. Then all entries in the Candidate Table corresponding to that detection and Local Table entry

are removed. This process is repeated until there are no suitable matches in the Candidate Table, after which all remaining detections are assigned new IDs.

For updating the Local Database on a successful ReID, the system always updates the spatial location of the person (bbox coordinates). However, it only updates the Feature Vector if the IoU score is less than θ_{IoU} and the new Feature Vector is better representative of the object (meaning obtained with more keypoints than previously had). Whenever the Local Database is updated, a message is sent to the server to update its contents accordingly. Once ReID is complete, the system uses the LSTM to generate predictions on all applicable detections, as detailed in Section 2.4.2 and Section 2.4.3.

2.5.2 Databases

On the edge node, a "Local Database" is responsible for storing all pedestrians in the current scene. This database is filled with objects that contain IDs, bounding box coordinates, feature vectors, keypoints, and a parameter called life which keeps track of how many frames it has seen since that pedestrian has been detected. When an object has not been seen by the system after some time (as indicated by life), the object's ID is sent to the edge sever, informing the server of the object's removal from the Local Database. This has two main benefits. Reducing the length of time an object's data is stored on the edge increases the effectiveness of spatial reasoning through IoU, as well as ensuring any single person's data is not stored on the edge when they are not active in the current scene. It also acts as an efficient replacement policy without complex computation.

On the edge server, there is a "Global Database" that functions very similarly to the Local Database. It stores the exact same information, with the addition of knowing which edge node's scene, if any, the object is currently active in. When an object is active in an edge node's scene, that edge node gains ownership of that object, blocking it from being ReID'd by other edge nodes. This ownership is cleared when

the server receives notification of the object's removal from the local database of the owning edge node, allowing the object to be included in ReID from all edge nodes. The size of both the Local Databases and the Global Database are easily configurable, allowing for customization to fit the requirements of individual applications.

Algorithm 1 Validating Detections

Input: $\bar{D}, \theta_{conf}, \theta_{key}$
Output: $\bar{V}$
1: $\bar{V} \leftarrow \emptyset$
2: **for** d in $\bar{D}$ **do**
3: $numKeyPoints$ = findValidKeyPoints(d, θ_{conf})
4: **if** $numKeyPoints \geqslant \theta_{key}$ **then**
5: $\bar{V} \leftarrow \bar{V} \cup \{d\}$
6: **end if**
7: **end for**

Algorithm 2 Finding Best Matches

Input: $\widehat{C}$
Output: $\widehat{newID}, \widehat{reID}$
1: $\widehat{newID} \leftarrow \Phi, \widehat{reID} \leftarrow \Phi$
2: $\tilde{C}$ = sortBasedOnL2Norm($\widehat{C}$)
3: **for** (v, e, ϕ) in $\tilde{C}$ **do**
4: **if** $e \neq null$ **then**
5: $\widehat{reID} \leftarrow \widehat{reID} \cup \{e.\text{id}\}$
6: removeValidEntryAndCandidate(v,e, $\tilde{C}$)
7: **end if**
8: **end for**
9: **while** thereIsCandidate($\tilde{C}$) **do**
10: $\zeta = \zeta + 1$
11: $\widehat{newID} \leftarrow \widehat{newID} \cup \{\zeta\}$
12: removeValidEntryAndCandidate(v,$\emptyset$, $\tilde{C}$)
13: **end while**

2.5.3 System Communication / Synchronization

A vital aspect of REVAMP^{2}T's communication design hinges around exactly what data is sent to the server, according to our privacy threat models described in Section 2.3. The only data transmitted across the network is encoded Feature Vectors,

impersonal IDs, and system metadata. The current iteration of REVAMP^{2}T's communication protocol leverages Wifi, but is adaptable and can be expanded to other communication protocols, such as LTE and 5G. This allows REVAMP^{2}T to keep up with the ever-changing communications landscape, no matter what technologies emerge.

Figure. 2.5 shows communication synchronization between edge nodes and edge server. Communications between the edge node and edge server are handled asynchronously. Both the edge node and edge server have separate threads to handle sending and receiving ReID information and storing it in separate buffers. These buffers hold the data until the main threads are ready to work on it. In addition to the main thread and the Global Database, the server has a separate set of these buffers and threads for each node. The main thread of the server processes this data in a round robin fashion and based on the metadata it receives from the edge node. It will either update the Global Database with a new Feature Vector for a global ID, release ownership of a global ID, or check the Global Database for ReID matches for the provided Feature Vector. Communications are only sent back to the edge node when a ReID match is successfully found.

By handling all communications on separate threads unrelated to inference, communications are entirely decoupled from the processing pipeline, eliminating pipeline stalls that would normally result from inline communications. This decoupling also means that edge node throughput is not dependent on network latency; Local ReID will always perform at a constant FPS. Additionally, if the network goes down and communications are completely lost, the buffers will allow a level of data synchronization after communications are restored. The system efficiently utilizes edge resources, which has a monumental impact on inference time. On the server side, the proposed system achieves greater scalability, as edge nodes do not fight over available sockets and communications does not take away from ReID resources.

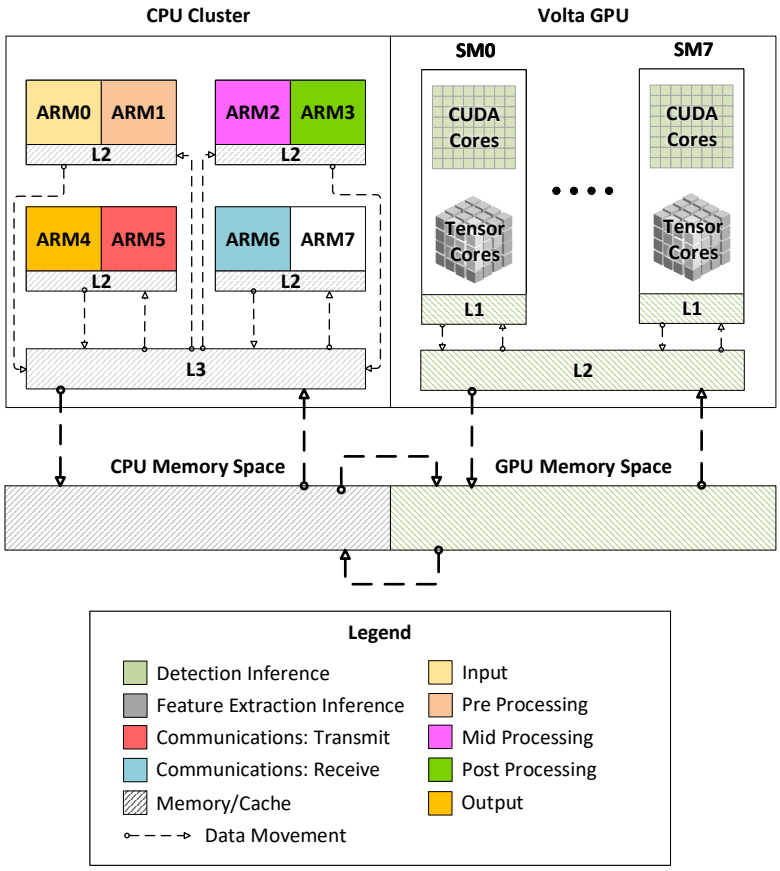

Figure 2.6: Mapping of Processes to Edge Resources

2.5.4 Computation and Optimization

To achieve real-time performance on the edge, we chose Nvidia AGX Xavier SoCs [23]. The Xavier is equipped with many advanced components that are leveraged for REVAMP^{2}T, including eight ARM Core processors, two Nvidia Deep Learning Accelerators (NVDLA), and a Volta GPU with Tensor Cores optimized for FP16 Multiply and Accumulate.

Figure. 2.6 shows the how the different processes in REVAMP^{2}T are mapped to the Xavier resources. Each stage of the detection framework is mapped to a separate ARM Core. The transmit and receive threads are mapped to their own cores as well. This leaves one ARM Core free to handle the OS and any background processes running outside of the system. Detection inference runs on the CUDA Cores of the Volta GPU. ReID inference is run on Tensor Cores. To enable this, the ReID

network model was is converted from ONNX to use half precision through TensorRT. Batch normalization layers are also fused into the convolutional layers, reducing data migration. Detections are batched for ReID inference each frame, allowing a ReID throughput above 20 FPS. The NVDLAs were not used for ReID due to a lack of support for the level of group convolution in MobileNetv2. All code on the edge was developed in C++ for computational efficiency, enhanced execution, and mapping control.

2.6 Experimental Results and Evaluation

The experimental setups and results will be split into four subsections: Algorithm, System, Scalability, and Design Flexibility. All project code for simulations and full system implementation is provided on GitHub[1].

2.6.1 Algorithm Evaluation

2.6.1.1 Feature Extractor Network

We used DukeMTMC-reID [25, 66], CUHK03 [67], and Market1501 [68] for evaluating the performance of two networks with different training methods. Table 2.2 summarizes the hyperparameters of our network. We decreased learning rate exponentially after 150 epochs and used Adam optimizer to train both networks.

Table 2.2: Training Parameters

Description	Value	Description	Value
Batch size	128	Input shape (H×W)	(256×128)
IDs per batch	32	Margin	0.3
Instances per ID	4	Epoch	300
Initial learning rate	2×10^{-4}		

We used the baseline ResNet as used in DeepCC for the sake of performance evaluation of MobileNet-V2. We applied Cumulative Match Characteristic (CMC) [69, 70] as a metric to evaluate and compare the identification performance of the two net-

[1]https://github.com/TeCSAR-UNCC/Edge-Video-Analytic

works. Each dataset consists of a gallery $\mathcal{G}$ as a set of various person images, and a query $\mathcal{Q}$ as a set of various person images that we want to identify. $\mathcal{P}_\mathcal{G}$ is a *probe* set, a subset of $\mathcal{Q}$, and for each of its images there are matches in $\mathcal{G}$. As the gallery embeddings are extracted, they are ranked (sorted) based on the similarity ($L2-Norm$ distance) across the current query image features. Then a set of the matched cases at rank r can be defined as in [70]:

$$C(r) = \{p_j \mid rank(p_j) \leqslant r\} \; \forall p_j \in \mathcal{P}_\mathcal{G} \tag{2.2}$$

Based on Eq. 2.2 CMC at rank r is calculated by following equation:

$$CMC(r) = \frac{|C(r)|}{|\mathcal{P}_\mathcal{G}|} \tag{2.3}$$

It should be noted that the CMC calculation can still be different for each dataset. For example, in Market-1501, $\mathcal{Q}$ and $\mathcal{G}$ can share the same camera view. However, for each individual query image, the individual's samples in $\mathcal{G}$ from the same camera are excluded [68].

Another evaluation metric which gives a representation of network performance over a set of queries $\mathcal{Q}$ is mean Average Precision (mAP), which can be extracted by:

$$mAP = \frac{\sum_{i=1}^{|\mathcal{Q}|} AP(q_i)}{|\mathcal{Q}|}, q_i \in \mathcal{Q} \tag{2.4}$$

$$AP(q) = \frac{1}{TP_{gt}} \sum_{j}^{|\mathcal{G}|} \frac{TP_{detected}}{j} \tag{2.5}$$

where TP_{gt} is the number of ground-truth true positives, and $TP_{detected}$ is the number of true positives detected by the network. CMC(1), CMC(5), and mAP are computed and compared side-by-side in Fig. 2.7 for both ResNet and MobileNet-V2 half precision networks. We can realize that the ModilbeNet-v2 half precision is 6.1%

less than ResNet for mean value of CMC(1) across all three datasets, while reaching an 18.92× model size compression ratio for MobileNet-V2 (5.0MB) over the baseline model (94.6MB).

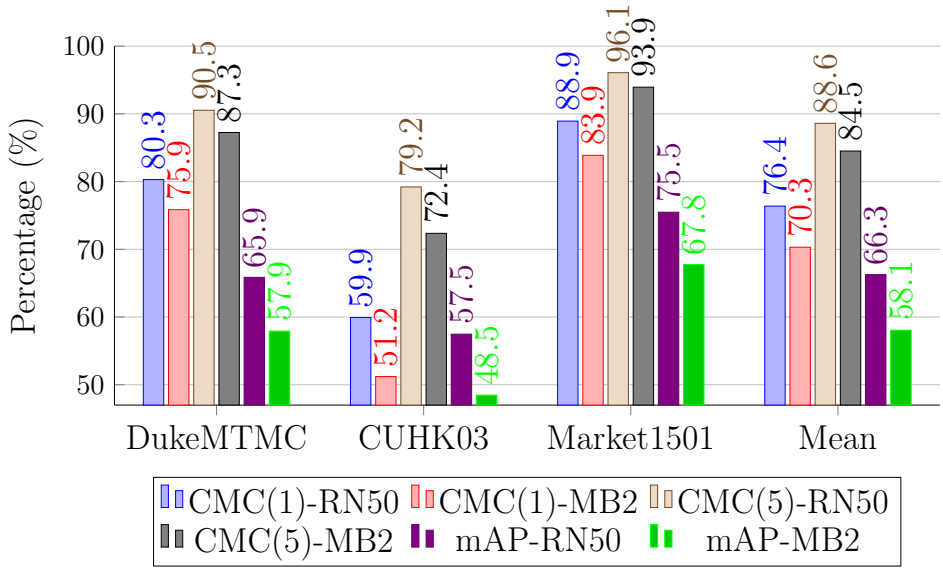

Figure 2.7: ResNet-50 single precision and MobileNetV2 half precision accuracy evaluation on three different benchmarks

2.6.1.2 LSTM Prediction Network

A total of 120 single object sequences, 15 from each camera, were used to train the LSTM model for 200 epochs. For testing the model we used 3 sequences per camera. The network takes around 14 hours to train on an Nvidia V100 GPU and was implemented using PyTorch. To evaluate the performance of the network we use the Intersection over Union (IoU) of the predicted bounding boxes with the ground truth bounding boxes and average it for all frames in the sequence, as shown in Figure. 2.8. This average IoU shows that we maintain performance above the 0.3 IoU detection threshold typically used for evaluation. We do not compare these results with DeepCC because their approach uses tracklets rather than IoU for tracking evaluation.

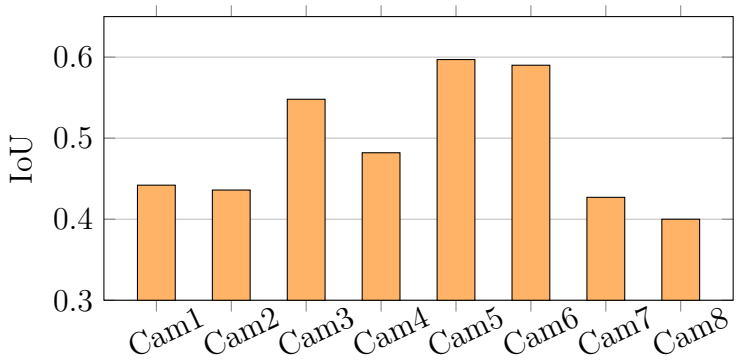

Figure 2.8: Average IoU for each camera on the testing sequences

2.6.1.3 Algorithm Pipeline

In order to validate the accuracy of the full algorithm pipeline, the edge algorithms and edge server functionality were ported to MATLAB and compiled into a simulation testbed to gather results. For these experiments, we used the DukeMTMC dataset, which includes 85 minutes of 1080p footage from 8 different cameras on the Duke University campus. Specifically, the trainval_mini frame set was used for validation. For comparison, we also ran the current state-of-the-art in MTMC work, DeepCC, on the same trainval_mini validation set. For all experiments, we measure ID Precision (IDP), ID Recall (IDR), and ID F1 score (IDF1) with truth-to-result matching, as proposed in [71]. Intuitively, IDP measures the percentage of attempted ReIDs that were correct, and IDR measures the percentage of possible ReIDs completed, regardless of number of attempts. IDF1 is simply the harmonic mean of IDP and IDR. Detection misses are computed in accordance to the truth-to-matching method, with the IoU threshold at 0.3. In accordance with Table 2.1, the values for system hyperparameters are as follows: $\theta_{key} = 5$, $\theta_{conf} = 0.5$, $\theta_{euc} = 5$, $\theta_{euc\beta} = 2$, $\theta_{IoU} = 0.3$.

Figure. 2.9 shows IDP versus IDR for REVAMP^{2}T and DeepCC. Analyzing the results, DeepCC maintains groupings around 80% for both IDP and IDR. REVAMP^{2}T maintains high IDP, always above 90%; however, the IDR is less consistent across cameras. The reasons for this problem are two fold. First, because REVAMP^{2}T is

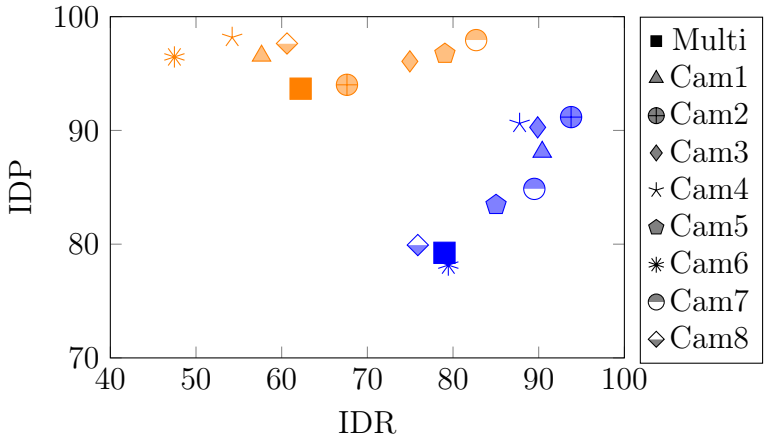

Figure 2.9: Precision (IDP) and Recall (IDR) for DeepCC (blue) and REVAMP^{2}T (orange)

an online system, it was designed to ReID within a short temporal window, in accordance with the spontaneous nature of online operation. Second, many of our false negatives are simply the result of missed detections. For the full 8-camera (shown as Multi) scenario, 59% of the false negatives incurred were from missed detections from the first stage of the pipeline. As mentioned in Section 2.4, we chose to run the detection network at a relatively low resolution at an attempt to balance reasonable runtime speed and detection accuracy. Nonetheless, despite the challenges of edge-capable algorithmic development, REVAMP^{2}T maintains reasonably close IDF1 in comparison to DeepCC. Figure. 2.10 shows the IDF1 for each camera individually, as well as the complete 8-camera global system (Multi) for both approaches. Overall, REVAMP^{2}T only drops 4.3% IDF1 in the full multi-camera system compared to the offline DeepCC algorithm, with DeepCC at 79.1% and REVAMP^{2}T at 74.8%.

2.6.2 System Evaluation

For all measurements, REVAMP^{2}T is run in real-time. We also compare against DeepCC [24]. For both, 16 detections per frame is assumed. As DeepCC was not built as a real-time system, it would be unfair to include the latency incurred through gallery matching in these comparisons, so we ignore the effect of this on power and latency. Real-time candidate matching is built into REVAMP^{2}T, so it is included in

all reported measurements. For measuring the power consumption on the Xavier NX, Tegrastats was used.

2.6.2.1 Power Consumption and Computation Efficiency

For power consumption on the Titan V and V100 GPUs, we utilized the NVIDIA System Management Interface. AMD μProf was used to measure CPU idle power for the edge server. For REVAMP^{2}T, 1080p 30 FPS video was pulled directly from a webcam. For DeepCC, 1080p 60 FPS video was read from memory. In both cases, a brief warm up of 20 frames was allowed before power was sampled over 100 frames. Measurements for FPS were taken directly from the OpenPose GUI.

Table 2.3: FPS and Power Consump. of Real-Time Inference

System	REVAMP^{2}T	DeepCC	DeepCC	DeepCC
Device	Xavier	Titan V	2xTitan V	V100
FPS ↑	5.7	2.5	4.7	2.7
Power ↓	34.4W	200W	365W	224W
Detailed Xavier Power Consumption				
CPU	GPU	DDR	SOC	Total
4.2W	22.1W	2.75W	5.35W	34.4W

Table 2.3 presents the power consumption and FPS for REVAMP^{2}T and DeepCC.

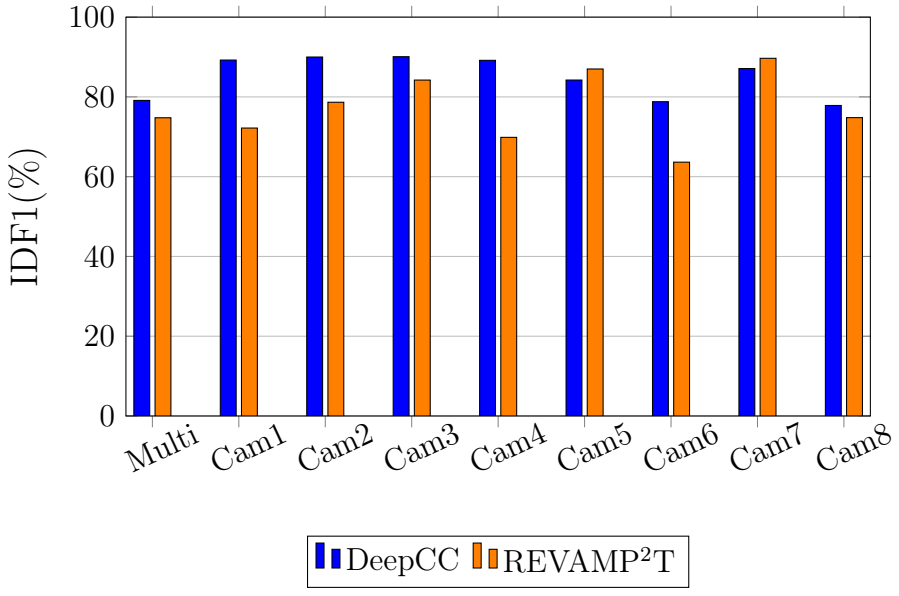

Figure 2.10: IDF1 Results for Multi-Camera and Single Camera

Here we can see that for real-time applications, REVAMP^{2}T out performs DeepCC on each GPU setup we tested. Even using two Titan V's, DeepCC is only able to reach 4.7 FPS. Meanwhile, REVAMP^{2}T can reach 5.7 FPS. In addition, REVAMP^{2}T consumes only 17% of the power of DeepCC on a single Titan V, or 9% for the dual Titan setup. Figure. 2.11 presents computation efficiency, which is FPS processing per watt. DeepCC has an Efficiency between 0.0147 and 0.0161 FPS/Watt in all configurations. In comparison, REVAMP^{2}T has an Efficiency of 0.166 FPS/Watt. When looking at Efficiency, REVAMP^{2}T performs an order of magnitude better than DeepCC for real-time applications. This is because REVAMP^{2}T was built from the ground up to perform in real-time, both algorithmically and systemically.

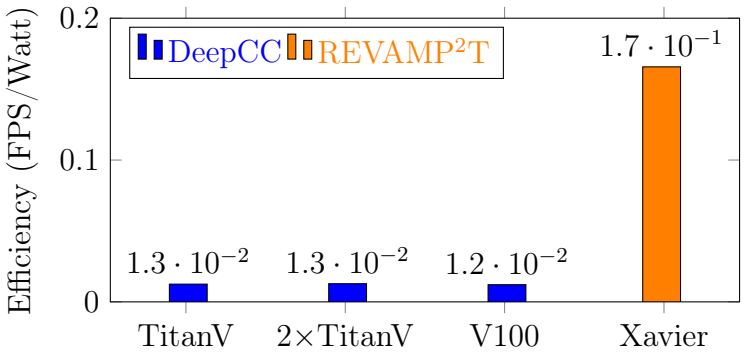

Figure 2.11: Efficiency of each test case.

2.6.2.2 Accuracy•Efficiency (Æ)

To enable real-time AI applications on the edge, we propose a new metric with which to measure edge performance; that is Accuracy•Efficiency (Æ). With Æ, we combine the algorithmic measurement of Accuracy with the systemic measurement of Efficiency to measure how well an application will perform in a real-time edge environment. Æ has two parts: an Æ mark, which is a score measured by the product of Accuracy and Efficiency, and Æ coverage, which is measured in area, as determined by all the components of an Æ mark. The components in Æ coverage, when not already reported as a percentage, are normalize to be so. In the case of power, this

normalized value is subtracted from one, as lower power consumption is preferable.

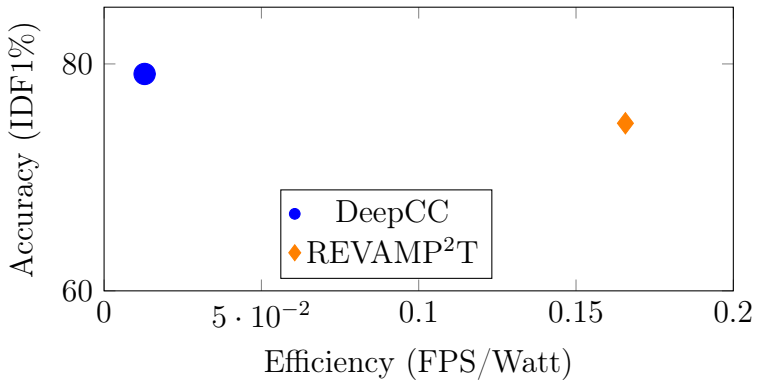

Figure 2.12: Æ of DeepCC on Titan V and REVAMP^{2}T on Xavier.

Table 2.4: Scalability Evaluation Results

	Server Processing					Split Processing				Edge Processing			
Nodes	GPUs	Cost	Power (W)	End-to-end Latency (ms)	Network Latency (ms)	Cost	Power (W)	End-to-end Latency (ms)	Network Latency (ms)	Cost	Power (W)	End-to-end Latency (ms)	Network Latency (ms)
1	1	$8,300	201	**83.2**	21	$8,900	122	506.6	29	**$5,700**	**34**	540.6	17
2	1	$8,400	202	**105.0**	35	$9,600	203	571.6	53	**$6,400**	**69**	549.6	26
4	2	$11,800	359	**168.8**	51	$11,000	306	678.6	82	**$7,800**	**138**	559.7	36
8	3	$15,400	518	**223.8**	107	$13,800	455	941.6	175	**$10,600**	**275**	601.9	78
16	6	$25,000	946	**374.8**	258	$19,400	742	1547.1	414	**$16,200**	**550**	700.3	175
32	11	$48,400	1827	972.8	856	$30,600	1301	2723.2	858	**$27,400**	**1101**	**866.0**	338
64	22	$91,800	3609	1202.8	1086	$53,000	2406	6020.9	2074	**$49,800**	**2202**	**1095.7**	559

In the case of REVAMP^{2}T, Accuracy would take the form of F1, while Efficiency is measured in FPS/Watt. Figure. 2.12 shows the Æ Mark for REVAMP^{2}T and DeepCC, while Æ Coverage can be seen in Figure. 2.13. Here you can see that while DeepCC outperforms REVAMP^{2}T in terms of IDR and IDF1 Accuracy, REVAMP^{2}T has a significantly higher Æ Mark (12.39 vs 1.02) and almost double the total Æ Coverage (81.25% vs 42.75%). This is because our optimizations allow us to operate at twice the framerate, 17% of the power, and we only lose by 4.3% in F1 accuracy.

2.6.3 Scalability

To measure scalability, we compare REVAMP^{2}T, which pushes all local processing to the edge nodes, against two other configurations: (1) "Server Processing" which streams the edge video frames to the edge server to handle all the local processing, and (2) "Split Processing" which splits local processing between the edge node and

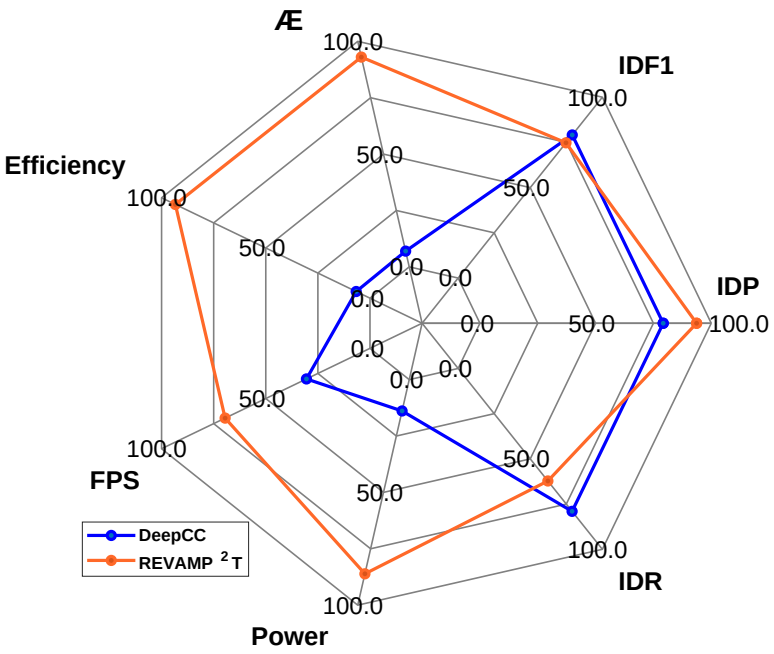

Figure 2.13: Æ Coverage

the edge server. Table 2.4 lists the number of GPUs, cost, power, end-to-end latency, and network latency over increasing number of edge nodes for all three scenarios. Power was measured using the same methods as described previously. The latency for a single frame was measured from when it was grabbed from the camera/video, to right before being displayed (using chrono library in C++), averaged over 100 frames, after a 20 frame warm up. For Server Processing, the edge nodes are Nvidia Jetson Nano SoCs [72], as they only stream video to the server. For the other cases, the Nvidia Xavier was used. In all cases we assume an edge server with a 12 core CPU, at least 32 GB of memory, and the capacity to support up to eight Nvidia Titan-V GPUs. The server processes all data at 30 FPS. We assume video data from separate nodes can be interwoven to allow a single instance of REVAMP^{2}T to support two edge nodes. Network latency was simulated using NS-3 Discrete Network Simulator, using 802.11ac, TCP, and 600Gbps throughput. H.264 compression for video, PNG for images, and 16 detections per frame are assumed.

From Table 2.4 we are able to see that Server Processing is able to achieve the

lowest latency for smaller node counts. Past 16 edge nodes, the network latency from streaming video tips the scales latency to favor Edge Processing. However, Edge Processing always wins out in terms of cost and power consumption, making it a promising option particularly for high node counts. At 64 edge nodes, Edge Processing has the best latency while only requiring about half the cost and 60% the power of Server Processing. We expect this trend to continue on to even higher node counts, increasingly favoring Edge Processing. Split Processing fares the worst in the comparison, due mostly to network latency. PNG compression is not nearly as efficient as H.264, meaning far more data is sent across the network, leading to the large latencies seen in the table. Overall, this shows that even for computationally intensive applications, edge processing is the only truly scalable solution for real-time IoT applications.

Table 2.5: Design Configuration Analysis

Config.	**Power(W)** $\downarrow$	**FPS** $\uparrow$	**Accuracy** $\uparrow$
P_2	**12.48**	3	71.37%
P_3	15.51	4	73.67%
C_D	34.40	5	74.80%
R_{720}	36.47	3	**74.91%**
R_{256}	30.09	15	73.77%
R_{128}	26.01	**24**	0.97%

2.6.4 Design Flexibility and Adaptation

REVAMP^{2}T can further be configured to prioritize accuracy, FPS, or power consumption, as illustrated Table 2.5. The default configuration of REVAMP^{2}T is shown as C_D, with an input resolution of 496x368 for the keypoint extractor, and power consumption as seen in Table 2.3. We analyze five additional design configurations by modifying the input resolution, as well as the power restriction levels on the Xavier device. Configuration R_{720}, R_{256}, and R_{128} are the proposed REVAMP^{2}T with modified input resolutions at 720x544, 256x192, and 128x96. Configuration P_2 and P_3 are the proposed REVAMP^{2}T configured with Power Mode 2 and 3 (C_D uses Power

Mode 0) provided by the Xavier device [73].

While R_{720} does provide slightly higher accuracy than other configurations, it does incurs loss in FPS and increased power consumption. The keypoint extraction resolution for R_{128} cannot properly extract keypoints for persons further than ∼30 feet from the camera, resulting in low accuracy for DukeMTMC. R_{256} offers an option for additional throughput at an accuracy loss. While R_{256} performs well on the tested dataset, we found that in real-world testing, this configuration was only able to extract keypoints for persons within ∼50 feet from the camera. Therefore, for robustness to real-world situations and its balance across all areas, configuration C_D was chosen for the analysis of this report. With deployment in an IoT environment, C_D would likely require PoE Type 3. The P_3 and P_2 configurations show how REVAMP^{2}T could be adapted to the power levels of PoE Type 2 and Type 1 for deployment, with minimal loss in accuracy.

2.7 Conclusions

This report proposes REVAMP^{2}T as an integrated end-to-end IoT system to enable decentralized edge cognitive intelligence for situational awareness. For the results and evaluation, this report also proposes a new two-part metric, Accuracy•Efficiency (Æ). REVAMP^{2}T outperforms current state-of-the-art by as much as a thirteen-fold improvement in Æ.

CHAPTER 3: Ancilia: Scalable Intelligent Video Surveillance for the Artificial Intelligence of Things

3.0 Individual Contributions to Jointly Authored Work

This section highlights my individual contributions to the collaborative work on **Ancilia: Scalable Intelligent Video Surveillance for the Artificial Intelligence of Things**, undertaken alongside Armin Danesh Pazho, with whom I share co-first authorship. The work presented in Ancilia is the result of a synergistic collaboration, with each author bringing unique skills and perspectives to the table. My direct contributions to this body of work are outlined below:

Algorithmic Design and Implementation: I was responsible for the conceptualization, design, and implementation of the Ancilia algorithmic framework, as detailed in Sec. 3.4. My direct contributions include:

- Crafting the initial concept and design of the Ancilia algorithmic framework, ensuring its alignment with the objectives of scalable and real-time video surveillance in the realm of the Artificial Intelligence of Things.
- Developing the underlying algorithmic pipeline that forms the core of Ancilia and allows for concurrent and overlapping execution to increase real-time performance.

Experimental Results: My contribution extended to the empirical evaluation of Ancilia, encompassing a comprehensive suite of experiments to validate its performance and real-world efficacy. Seen in Sec. 3.6.1 and Sec. 3.6.2, my direct contributions include

- Performing evaluations of the individual algorithms that make up the Algorithmic Core of Ancilia compared to their state-of-the-art counterparts, ensuring that accuracy remains high enough to limit the amount of noise generated by the system while still allowing real-time performance.

- Data extraction, training, and evaluation of two state-of-the-art action recognition algorithms (PoseConv3D [3] and CTR-GCN [4]) and two state-of-the-art anomaly detection algorithms (GEPC [6] and MPED-RNN [7]) using both ground truth and Ancilia generated data, both at full and half throughput, to demonstrate the effect of both system generated noise and reduced throughput on end-to-end accuracy in action recognition, and validating the efficacy of Ancilia for such high-level tasks.

In summary, my contributions to Ancilia encompass both the design of the algorithmic framework, the practical aspects of its implementation, and a thorough evaluation of its suitability for high-level cognative tasks common in the video surveillance industry. These contributions are vital to Ancilia's success, ensuring it stands as a robust and scalable solution for intelligent video surveillance within the AIoT paradigm.

3.1 Introduction

There is a growing need for effective and efficient surveillance technologies that can be deployed to protect our cities, people, and infrastructure. For example, in Itaewon, South Korea, a holiday celebration left over 150 dead due to severe overcrowding, with many blaming the tragedy on careless government oversight [18]. In Moore County, North Carolina, directed attacks against two power substations left over 45,000 residents without power for days as technicians rushed to restore power and authorities struggled to find the source of the attacks [19]. With enough forewarning through smart video surveillance, they could have been prevented.

With the recent emergence of the Artificial Intelligence of Things (AIoT), some surveillance solution providers have started adding basic forms of artificial intelligence to their systems. However, their methods are still naive and unable to enhance security in a truly meaningful way [74]. This is because, while a lot of research is conducted on tasks that would benefit surveillance systems, most works focus on algorithmic improvements in a lab environment instead of paying attention to factors that are prevalent in real-world scenarios [75, 76]. Most research focuses on a single algorithm and how to tweak it to get the best possible results on readily available datasets that often do not reflect a real surveillance environment. Few works explore how different algorithms affect the performance of other downstream algorithms in multi-algorithm systems. Few still explore the effects of noise (both data derived and the system produced) in end-to-end accuracy. Beyond this, real-world intelligent surveillance necessitates real-time performance. The cognitive abilities of advanced artificial intelligence are only helpful if they can be provided to security personnel quickly enough to take appropriate action before it is too late.

In this article, we present Ancilia, the first end-to-end scalable, intelligent video surveillance system able to perform high-level cognitive tasks in real-time while achieving state-of-the-art results. Ancilia takes advantage of the prevalence of cameras in the

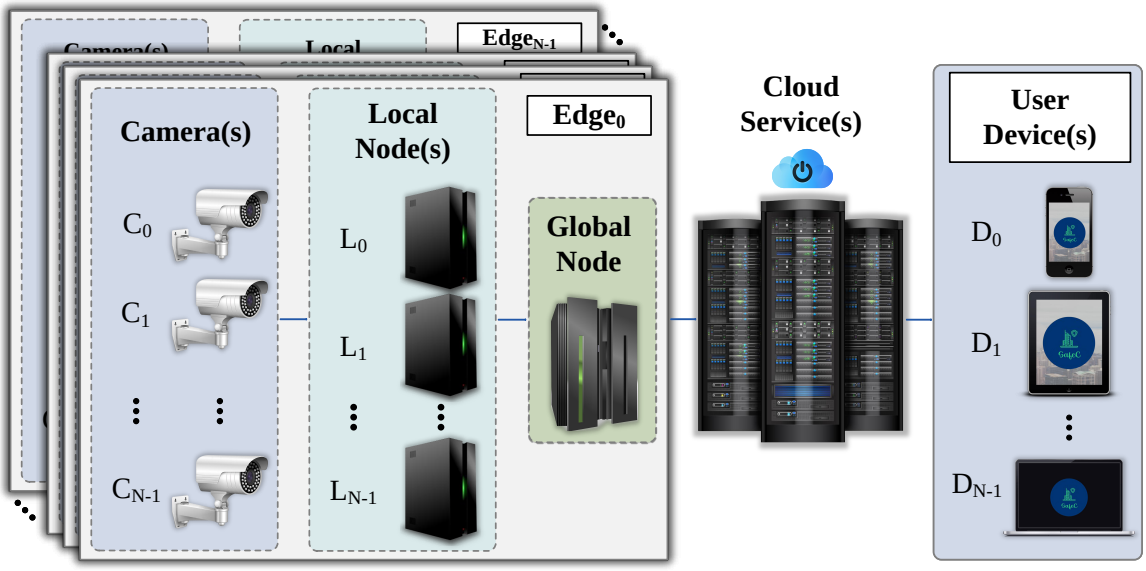

Figure 3.1: Conceptual overview of Ancilia.

Internet of Things (IoT) and uses localized servers to integrate with existing IoT camera ecosystems, facilitating processing on the edge. Current IoT methods often use cloud computing, which can introduce latency and privacy concerns, or they require custom sensors with high processing power. Ancilia offers a solution to utilize existing IoT sensors, minimizing the need for expensive infrastructure upgrades and reliance on cloud processing. Ancilia is device agnostic; As long as video from the camera can be accessed, Ancilia can provide intelligence. Shown in Fig. 3.1, Ancilia exists within three logical and physical segments: the edge, the cloud, and user devices. The edge uses a plethora of advanced artificial intelligence algorithms processing data received from cameras to facilitate intelligent security. Using a single workstation to perform edge processing, Ancilia can monitor up to 4 cameras in real-time at 30 FPS, or up to 8 cameras at 15 FPS, in scenarios with both medium and heavy crowd density. Ancilia performs high-level cognitive tasks (i.e. action recognition, anomaly detection) with $\sim 1\%$ deviation in accuracy from current State-of-the-Art (SotA).

Ancilia is designed from the ground up to respect the privacy of the people and communities being surveilled. Ancilia does not store any personally identifiable information in any databases and does not make use of invasive artificial intelligence

techniques such as facial recognition or gait detection. Ancilia strictly provides pose and locational information for high-level tasks (i.e. action recognition, anomaly detection), as opposed to identity information, which is common. Ancilla looks at what a person is doing, not who they are. This allows Ancilia to act as a buffer to help remove biases based on race, ethnicity, gender, age, and socio-economic factors, which can lead to a reduction in the unnecessary conflict between authorities and marginalized communities that has become increasingly problematic. After data is processed on edge and sent to the cloud for communication and service management with user devices. A mobile app allows user devices to receive data from the cloud, including alerts when potential security concerns arise.

In summary, this article has the following contributions:

- We present Ancilia, the first end-to-end scalable real-world intelligent video surveillance system capable of performing high-level cognitive tasks in real-time while achieving SotA accuracy.

- We analyze the ethical concerns of intelligent video surveillance, both from a privacy and fairness perspective, and illustrate how Ancilia's design is purpose-built to address them.

- We perform an end-to-end empirical evaluation of Ancilia using two high-level cognitive tasks directly related to intelligent surveillance, action recognition, and anomaly detection, investigating the trade-off in accuracy required to achieve real-time performance.

- We perform an exhaustive system-level evaluation of Ancilia's real-time performance and scalability across different classes of hardware and increasing scenario intensities, displaying how Ancilia is able to meet real-time intelligent security needs in different contexts.

3.2 Related Work

There has been a plethora of research regarding the use of artificial intelligence for video surveillance [75, 77, 78, 79]. [80] proposes the use of region proposal based optical flow to suppress background noise and a bidirectional Bayesian state transition strategy to model motion uncertainty to enhance spatio-temporal feature representations for the detection of salient objects in surveillance videos. [81] proposes the use of a person detector, tracking algorithm, and mask classifier for tracking pedestrians through surveillance video streams.

In [75], it is determined that in order to address the latency concerns of real-time video surveillance, a shift towards edge computing is needed. Nikouei et al. [82, 83, 84] explore the feasibility of using low-power edge devices to perform object detection and tracking in surveillance scenarios. They argue that in worst case 5 FPS is high enough throughput for tracking humans in surveillance applications, and as such computation can be pushed to the edge. However, their results show that even light weight convolutional neural networks can prove problematic for low-power devices, often reducing throughput below the 5 FPS threshold. [85] proposes a system using low-power embedded GPUs to perform detection, tracking, path prediction, pose estimation, and multi-camera re-identification in a surveillance environment, while placing a focus on real-time execution and the privacy of tracked pedestrians. [86] proposes a similar system, focusing solely on object detection, tracking, and multi-camera re-identification to increase throughput. [87] proposes using a combination of lightweight object detection models on the edge and more computationally expensive models in the cloud, splitting computation between the two to provide real-time video surveillance in a construction site environment. [88] proposes the use of background detection, vehicle detection, and kalman filter [89] based tracking for parking lot surveillance and determining lot occupancy. [90] proposes a system that uses object detection, person tracking, scene segmentation, and joint trajectory and activity prediction for pedes-

trians in a surveillance setting.

The future of intelligent surveillance is heading towards systems able to perform high-level cognitive tasks. A recent survey focusing on real-world video surveillance [75] asserts that while the domain of video surveillance is comprised of understanding stationary object, vehicles, individuals, and crowds, the ability to determine when anomalous events occur is paramount for intelligent surveillance systems. Other research has supported this assertion [77]. [91] utilizes the Infinite Hidden Markov Model and Bayesian Nonparametric Factor Analysis to find patterns in video streams and detect abnormal events. [92] proposes active learning and fuzzy aggregation to learn what constitutes an anomaly continually over time, adapting the scenarios not seen in standard datasets. [93] proposes a system to detect suspicious behaviors in a mall surveillance setting, using lightweight algorithms such as segmentation, blob fusion, and kalman filter based tracking [89]. AnomalyNet [94] is a recently proposed recurrent neutral network with adaptive iterative hard-thresholding and long short-term memory that works directly off pixel information to eliminate background noise, capture motion, and learn sparse representation and dictionary to perform anomaly detection in video surveillance.

3.3 Ethical Concerns

Video surveillance has always been associated with social and ethical concerns, whether in traditional form or more recent intelligent formats. Respecting citizens' privacy and autonomy while improving public safety and security are the most well-known and enduring ethical issues in this context [95, 96, 97, 98]. Developing a successful smart video surveillance solution that addresses the public safety problem and engages the community up to a certain level is only possible by considering these concerns.

There is rising attention among scholars to the issue of incorporating privacy concerns at the design level, referred to as "privacy by design" [99]. The source of

discrimination and privacy violation in many data-driven and AI-based systems, such as Smart video surveillance technology, is using Personal Identifiable Information (PII)[100, 101]. Using PII, such as actual footage of people's daily activities at any stage of the technology, can increase the risk of privacy violation. There is a long-lasting debate on the ethical challenges of using facial recognition technologies in different sectors and how using this technology can result in privacy violation[102, 103, 104, 105].

The approaches used to perform high-level cognitive tasks in intelligent video surveillance, such as action recognition and anomaly detection, can be grouped into two distinct categories based on the data used [106]. The first category directly utilizes pixel data. A common example is facial recognition [107], where algorithms look at images of people's faces to identify them. These algorithms can perform well with sufficient historical data, but are often seen as intrusive and increase the risk of identifying personal demographic information [104]. The second category only leverages processed information, such as pose data in the case of Ancilia, which tends to de-identify personal demographic information [108]. This is not a complete removal of PII, as some works have been able to identify individuals purely by gait [109] or silhouette [110], but it significantly reduces the risk to privacy compared to pixel-based approaches.

Similarly, avoiding facial recognition technologies does not guarantee the system is entirely privacy persevering. Storing images of pedestrians is another source of ethical violation. From the discrimination perspective, using any form of PII can contribute to the issue of marginalization in policing systems[111]. Therefore, an essential step in designing a non-discriminatory system is to ensure the system is not dependent on PII. This requires a specific approach toward the design of such technology in the choice of algorithm, the type of data used, and the storing of such data.

Ancilia addresses this by not storing any PII or sending any PII across the network.

Such data is destroyed after it is used. Ancilia utilizes pose-based methods for all high-level cognitive tasks, severely limiting the amount and quality of PII used by such algorithms. This allows such processing without any potential for gender, ethnicity, or class-based discrimination. As such, Ancilia is able to address many of the privacy concerns regarding intelligent video surveillance while also addressing the ethical issue of discrimination.

3.4 Ancilia Algorithmic Framework

The algorithmic core of Ancilia is separated into two conceptual systems: the local nodes containing the algorithmic pipeline of each camera and the global node that handles all processing that requires understanding of multiple camera perspectives. These two systems make up the algorithmic core of Ancilia and are the basis on which all higher understanding is achieved. A visual representation of this algorithmic core can be seen in Fig. 3.2.

3.4.1 Single Camera Vision Pipeline

As seen in Fig. 3.2, the local algorithmic pipeline starts when an image is extracted from the camera. The image is first run through an object detector to locate people, vehicles, animals, and other important objects in the scene. This is important not only because it acts as the basis for the rest of the algorithmic pipeline but also because it can be used for basic situational awareness. Sometimes, just the presence of a certain object in a scene is noteworthy, like a person in an unauthorized location, a bag left unattended, or the presence of a firearm. Ancilia uses YOLOv5 [112] for this purpose (however, it can be any detector). Please note that many objects of interest are not included in the default weights provided by the YOLOv5 authors. However, other works have trained the architecture for classes such as firearms [113, 114, 115], and custom weights can always be trained to match the target application. The locational coordinates of persons are sent to a tracker, where tracklets are created, matching

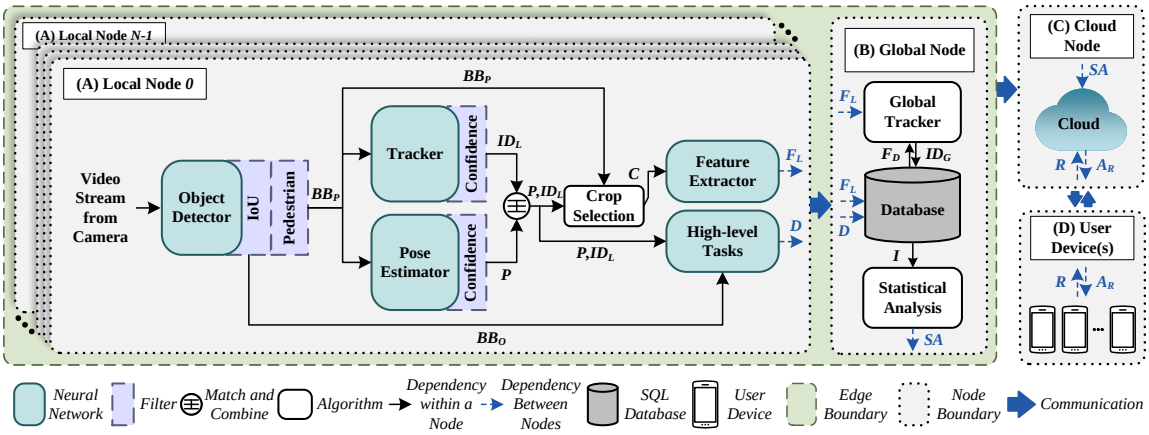

Figure 3.2: Ancilia algorithmic details. N local nodes are connected to a single global node on the edge. The final analyses are transferred to the cloud node to feed the application on the user device. Multiple edges may be connected to the could, though this figure only shows one edge for clarity. BB_P, BB_O, ID_L, P, C, F_L, D, F_D, ID_G, I, SA, R, and A_R refer to bounding boxes for pedestrians, bounding boxes of objects, local identities, poses, person crops that passed selection, features from the local node, data from the downstream tasks, features from the database, global identities, information from the database, completed statistical analysis, requests from users, and requested attributes respectively.

each person with their previous detections in prior images. Ancilia utilizes the version of ByteTrack [116] without frame similarity. In this configuration, ByteTrack does not perform feature extraction, which results in a notable reduction in computation. As shown in [116], locational similarity is sufficient for single camera tracking. The tracking allows for understanding how a person moves throughout a scene, which is vital for many surveillance applications. It also allows Ancilia to understand which poses belong to which persons over time, which is vital for many high-level tasks that provide much-needed situational awareness. Image crops of the people detected in the image are also sent to a human pose estimator, where two-dimensional pose skeletons are created. Ancilia uses HRNet [117] for extracting 2D skeletons. Using pose data for higher-level tasks has two major benefits over simply using raw pixel data. First, pose data is of much lower dimensionality than pixel data, making it much less computationally expensive and allowing the Ancilia to function in real-time. Second, pose data helps us remove the appearance-based PII information inherent in

pixel data, making it harder for high-level tasks to form unintended biases based on ethnicity, gender, age, or other identity-based metrics. Works such as [118, 119] try to identify subjects based on their poses, in a line of work called Gait Recognition, but as discussed in Sec. 3.3, pose-based approaches are shown to be more privacy preserving compared to their alternatives.

3.4.2 Multi-Camera Person Re-identification

While the tracker tracks people within a single camera, locational information cannot accurately re-identify a person across multiple cameras. For this, the same person crops that are sent to the human pose estimator are also sent to a person re-identification feature extractor, where an abstract feature representation is created for each person. Only one feature representation is created for each person during a single batch, and only when the quality of the representation can be assured, as poor quality representations are detrimental to accurate multi-camera person re-identification. Ancilia uses a feature representation filtering algorithm to verify two qualities for person crops. First, a person crop must contain a high-quality view of the person. To this end, the filter algorithm uses the 2D pose skeleton and verifies that at least 9 keypoints were detected with at least 60% confidence. The filter algorithm looks at the overlap (i.e. Intersection of Union) of the bounding boxes generated by the object detector. An individual's bound box must have an Intersection over Union (IoU) of no more than 0.1 with any other person. If those two conditions are met, the person crop is determined to be of high enough quality to produce an adequate feature representation. If more than one crop is deemed suitable for a single person during a 15 frame window, the one with the most confident pose is selected. The features created by the feature extractor are sent to the global node for multi-camera person re-identification. Ancilia uses OSNet [120] to extract feature representations.

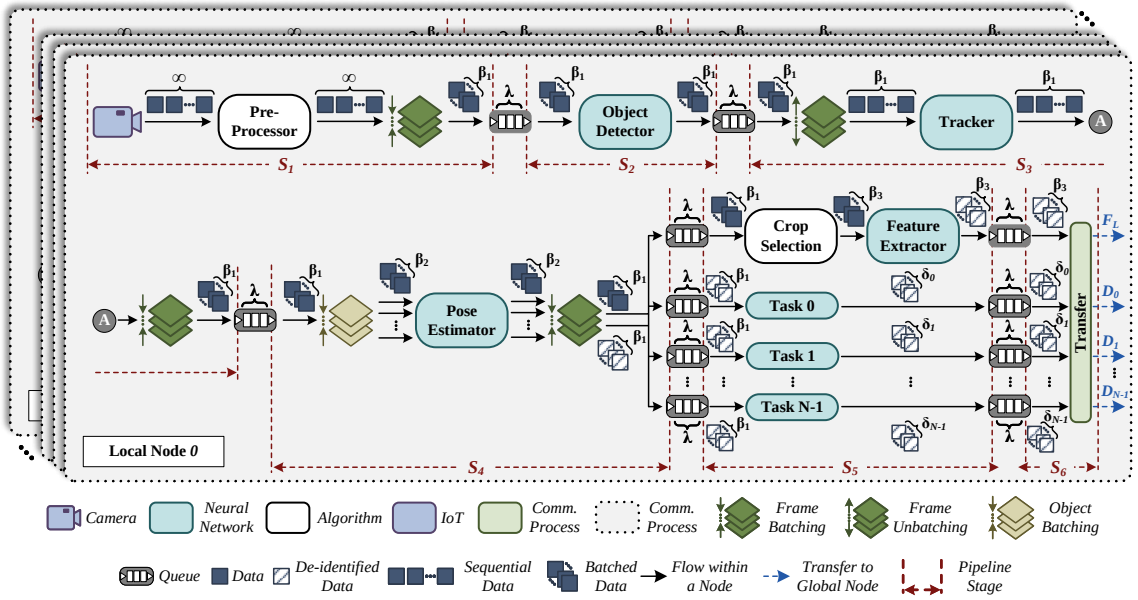

Figure 3.3: A detailed view of system design in Ancilia's local nodes. β and δ refer to different batch sizes. λ refers to the queue size. F_L and D represent local features and data received from downstream tasks respectively.

3.4.3 higher Level Tasks

To help preserve privacy from a system perspective, sensitive information is kept on the local machine by executing all high-level tasks on the local node. These tasks have access to the object, tracking, and pose data generated in the previous steps. Since the decision of which high-level tasks are needed is highly application dependent, we do not consider these tasks to be part of Ancilia's algorithmic core, but instead an extension to be customized based on intended use. In this paper, we use action recognition and anomaly detection as two common examples of high-level tasks that are highly relevant to intelligent surveillance. For action recognition, we choose PoseConv3D [3] and CTR-CGN [4], two state-of-the-art networks that can utilize the 2D human pose skeletons provided by Ancilia. For anomaly detection, we use GEPC [6] and MPED-RNN [7], which are based on 2D human pose skeletons.

There are many works that use pixel-based methods for these tasks that achieve superior accuracies than SotA posed-based methods, such as I3D [121], MVIT [122]

and Stargazer [123]. Argus [124] is a good example of a system that employs pixel-level information, with a subsequent evaluation conducted on a real-world surveillance dataset called Meva [125]. However, due to the privacy benefits discussed in Sec. 3.3 and the computational benefits of using lower-dimensional pose data, we have opted to stick with pose-based methods for Ancilia.

3.5 System Design

Beyond the algorithmic design, Ancilia can be analyzed from a system-level design and implementation perspective. The local node in particular has a complex system design, as seen in Fig. 3.3. The global node and cloud are much simpler, as shown in Fig. 3.2.

3.5.1 Parallelism

A key design objective of Ancilia is to achieve higher efficiency by balancing throughput and latency. Ancilia uses pipelining to take advantage of process parallelism, dividing tasks into six separate stages of a pipeline system ($S1, S2, ..., S6$). Each stage is implemented as a separate process, which executes concurrently with other processes as soon as it receives its required input. These stages communicate with each other using queues to utilize memory resources better and enable fast inter-process communication. While pipelining is a well-known technique for optimization, the overhead associated with its implementation means a balance needs to be found. Figure 3.3 shows a detailed view of the system design on the local node. Each pipeline stage is separated by a queue with a size limit of λ elements, preventing any potential overflow from uneven execution speed between pipeline stages. By default, Ancilia uses a λ value of 1. As is common, Ancilia offloads highly parallel tasks that rely on neural networks (i.e. object detection, pose estimation, feature extraction, and many high-level tasks) to Graphics Processing Units (GPUs) for execution.

3.5.2 Data Batching

Batching is another technique Ancilia implements to better utilize hardware resources. Generally, batching is able to greatly increase the throughput of a system at the cost of end-to-end latency. However, many high-level tasks (e.g. action recognition, anomaly detection) require multiple video frames worth of input data (often called a window) before they can start processing, so the latency that would be incurred by batching input frames is already inherent in these high-level task, as long as the frame batch and high-level task window are of the same size. In other words, if a high-level task needs X number of frames before it can start processing, having a batch size of X frames will ensure the task gets all the frames it needs simultaneously, incurring no additional latency for the task. If the window size of the high-level task is larger than the batch size multiple batches will be needed to be processed to receive output from the high-level task. Further, as frame batching ultimately increases the throughput, the end-to-end latency is decreased when compared processing each frame sequentially. While object detection works on entire frames, all other neural networks in Ancilia work off individual objects. These objects are batched together before being input to the network, greatly increasing hardware utilization. There can be multiple object batches within a single frame batch, based on how many of the relevant objects are detected in the video.

3.5.3 Local Node

3.5.3.1 S_1 - Preprocessing

Once the local node receives the video stream from the camera, the preprocessor is responsible for all basic image processing necessary before sending the frames through the algorithmic core. That includes any necessary resizing, frame dropping, and/or color channel reordering. Frame dropping is a dynamic machanism that ensures the framerate fed to the pipeline matches the throughput of the pipeline. For example, if

the frame source (i.e. camera) produces 60 FPS, but Ancilia can only run at 30 FPS, only every second frame from the source will be passed through preprocessing. After preprocessing, frames are batched in sequential segements of size β_1. Ancilia sets $\beta_1 = 15$. This is done to balance throughput and latency, as discussed in Sec. 3.6.4, as well as to more closely match the window size of the high-level tasks, requiring only two batches to complete before these tasks can produce an output. This is also suitable because most modern security and IoT cameras record video at either 30 or 60 FPS.

3.5.3.2 S_2 - Object Detection

The batched frames are sent to the object detector, which outputs a list of objects with class labels and bounding box coordinates. Bounding boxes for pedestrians are sent to the tracker, while bounding boxes for other objects are passed through the system for use in high-level tasks and statistical analysis. A crop of each pedestrian from the original frame is passed through to the pose estimator at later stages.

3.5.3.3 S_3 - Tracking

At the tracker, bounding boxes for pedestrians are unbatched to fit the tracker's sequential operation. The tracker groups the pedestrians and either matches them with previously seen pedestrians or assigns them a unique local ID. Afterwards, the pedestrians are once again batched by frame, back to the original batch size of $\beta_1 = 15$ frames, and sent to the pose estimator.

3.5.3.4 S_4 - Pose Estimation

At the pose estimator, the object batching is performed on the person crops, with a batch size of $\beta_2 = 32$. These batches are fed to the pose estimator, which outputs human pose skeletons for each person crop. Then the pedestrian bounding boxes, person crops, local IDs, and human pose skeletons are once again batched by frame and combined with the object bounding boxes from the object detector. Select data (ped-

estrian bounding boxes, person crops, local IDs, and pose skeletons) is sent to crop selection and feature extraction, while the de-identified data (pedestrian bounding boxes, object bounding boxes, local IDs, and pose skeletons) is sent to each high-level task as per their request.

3.5.3.5 S_5 - Feature Extraction and High-level Tasks

Before feature extraction, crop selection filters out low-quality person crops based on bounding box overlap and keypoint confidence, as described in Sec. 3.4. By default, crops with an IoU higher than 0.1 or with 9 or more keypoints with confidence below 60% are discarded. These thresholds can be adjusted to best suit the target application. Out of the remaining crops only a single crop with the highest overall keypoint confidence for each person is selected. The remaining crops are batched, with a dynamic size of β_3 based on the number of persons in the scene. Feature extractor receives the batch of β_3 crops. Once features are extracted, they are sent for transfer to the global node. Each high-level task receives data at the granularity of a frame batch with size β_1, and sends data to the global node at the granularity that task operates at $(\delta_0, \delta_1, ..., \delta_n)$. Only de-identified data is sent to the high-level tasks, keeping in line with the ethical concerns mentioned in Sec. 3.3 and Sec. 3.4Ėach high-level task has its own process and works in parallel with other tasks as well as with crop selection and feature extraction in stage 5 of the pipeline.

3.5.3.6 S_6 - Transfer

Communication is completely decoupled from the pipeline, so once the data is sent, the local node pipeline continues to function as normal without needing a response from the global node. Importantly, no identifiable information is ever sent to the global node, keeping in line with the privacy and ethical concerns mentioned in Sec. 3.3.

3.5.4 Global Node

All received data is stored in a relational database on the global node. The matching algorithm described in Sec. 3.4 compares the received features with existing features in the database over the period T and assigns a global ID based on the results. The default value for T is set to 1 hour, but this should be changed to suit the needs of the application. An assortment of algorithms performs statistical analysis using the relational database, as detailed in Sec. 3.4. The analysis is transmitted to the cloud node using APIs provided by the cloud service provider. By default, Ancilia uses Amazon Web Services, but this can be altered based on user/application needs. The cloud (e.g. Amazon Web Services (AWS)) receives analyzed data from the global node.

3.6 Experimental Results

3.6.1 Algorithmic Core

The algorithmic core of Ancilia consist of multiple algorithms, each of which works off of data generated by the previous algorithms. As these algorithms leverage imperfect neural networks, they generate noise that accumulates through the system. To understand the source of this noise, we must first look at the accuracy of each of these core algorithms in isolation. Table 3.1 shows the accuracies of the algorithmic core's four main tasks: object detection, pedestrian tracking, human pose estimation, and person re-identification. The table also shows the accuracies of the top SotA models in each task. These SotA methods are not suitable for intelligent surveillance applications, as their excessive computation and vast parameters make real-time execution impossible, but the comparison allows us to see the maximum potential allowable by current research and the accuracy loss incurred to keep Ancilia performing in real-time.

Object detection sees the biggest hit to accuracy, with a 16% drop from SotA. This

Table 3.1: Accuracy of Ancilia's Algorithmic Core networks in isolation. SotA Algorithms represent the highest performance currently achievable when computation and latency are not a concern.

Task	Method	Performance	Dataset
Ancilia's Algorithmic Core			
Object Detection	YOLOv5 [112]	49.0 (mAP)	COCO [126]
Tracking	ByteTrack [116]	77.8 (MOTA)	MOT20 [127]
Pose Estimation	HRNet [117]	75.1 (AP)	COCO [126]
Person ReID	OSNet [120]	88.6 (Top-1)	DukeMTMC [1]
State-of-the-Art Algorithms			
Object Detection	Internimage [128]	65.0 (mAP)	COCO [126]
Tracking	SOTMOT [129]	77.9 (MOTA)	MOT20 [127]
Pose Estimation	ViTPose [130]	81.1 (AP)	COCO [126]
Person ReID	Centeroids-ReID [131]	95.6 (Top-1)	DukeMTMC [1]

is intuitive, as YOLOv5 [112] is not only the largest model in the algorithmic core, but also the only one that operates on the raw camera stream. So while larger models are available and would be able to produce higher accuracy, even a slight increase in model size or computation would result in a noticeable decrease in throughput. Human pose estimation sees a decrease in accuracy for a similar reason, though much smaller in scale at only 6%. While HRNet [117] is not run on the raw camera stream, it is run individually for each person detected by the object detector. As such, maintaining a small model size is preferable. Person re-identification sees a slightly larger drop in accuracy than human pose estimation at 7%. While this is partly due to using a lightweight model, OSNet [120], the SotA model for person reID is also lightweight. However, the SotA uses a centroid based retrieval method not suitable for pen-set reID, of which most surveillance scenarios are. Pedestrian tracking sees almost no drop in accuracy, approximately 0.1%. This stems from the comparative ease of tracking pedestrians in a single camera, where a simple, lightweight algorithm like

ByteTrack [116] see almost no performance difference from the top of the line SotA approaches.

3.6.2 High-level Tasks

To better understand how the noise generated by the algorithmic core effects overall performance, and thus how well Ancilia performs in the realm of real-world intelligent surveillance, we examine the performance of two high-level cognitive surveillance tasks when running on Ancilia. For Ancilia to be a benefit to intelligent surveillance tasks, we must ensure that excess false alarms or missed positive events do not occur. To assess this, we choose action recognition and anomaly detection, as these tasks can utilize the human pose information generated by the algorithmic core, resulting in faster and less biased inference. Since both these methods utilize temporal batches of human poses for each individual, these experiments will directly reflect the quality of the object detection, tracking, re-identification, and pose estimation data generated by Ancilia.

3.6.2.1 High-level Task - Action Recognition

We select two state-of-the-art action recognition models, PoseConv3d [3] and CTR-GCN [4], and train them using data generated with Ancilia. For each model, we train and test with full (30 FPS) and half (15 FPS) throughput on NTU60-XSub [2]. Both models use a window size of 30 and are trained for 24 epochs using Stochastic Gradient Descent (SGD) with a momentum of 0.9 and Cosine Annealing scheduling. PoseConv3d and CTR-GCN have weight decay of $3e^{-4}$ and $5e^{-4}$ and an initial learning rate of 0.4 and 0.2, respectively.

The results of these experiments can be seen in Tab. 3.2. We report the Top-1 and Top-5 accuracy and compare the results using data generated by Ancilia to the original data available through the PYSKL toolbox [132]. We can see that Ancilia is able to provide data of comparable quality to the original; action recognition as a high-

Table 3.2: Top-1 and Top-5 accuracies on NTU60-XSub [2] in full and half throughput modes for PoseConv3D [3] and CTR-GCN [4].

Model	Data	FPS	Top-1 (%)	Top-5 (%)
PoseConv3D [3]	[132]	15	91.96	99.47
		30	92.76	99.57
	Ours	15	88.79	98.82
		30	91.99	99.28
CTR-GCN [4]	[132]	15	86.36	98.46
		30	83.07	98.26
	Ours	15	81.58	97.52
		30	80.44	97.2

level task in Ancilia sees around 1% drop in accuracy compared to the original data using PoseConv3D [3] at full throughput, and around 3% at half throughput. Using CTR-GCN [4], Ancilia sees a 2.5% drop at full throughput and a 4.8% drop at half throughput, compared to the original data. From this we can infer that PoseConv3D is more robust to noise than CTR-GCN, however both performed reasonably well with data generated from Ancilia, demonstrating its efficacy for intelligent surveillance applications.

Figure 3.4: Throughput of Ancilia with respect to number of nodes across different crowd densities. Hardware details can be seen in Tab. 3.4.

Another interesting observation is that CTR-GCN [4] actually performed noticeably better at half throughput than at full throughput. This means that CTR-GCN is more suited to taking advantage of the higher temporal window allowed when using half throughput. This is something to consider when choosing an action recognition model when a real-time throughput of 30 FPS cannot be guaranteed.

3.6.2.2 High-level Task - Anomaly Detection

Using the ShanghaiTech dataset [5] we train two state-of-the-art anomaly detection models, GEPC [6] and MPED-RNN [7], using both data generated by Ancilia and the data provided by the original authors. The same training strategy from Sec. 3.6.2.1 is

used, with both models trained in full (20 FPS) and half (10 FPS) modes. GEPC is trained for 25 epochs with a window size of 30 and stride of 20 using Adam optimizer with a learning rate of 1e-4, weight decay of 1e-5, and batch size of 512. MPED-RNN is trained with an input window size of 30, a reconstruction window of 12, and a prediction window of 6. The model is trained for 5 epochs using the Adam optimizer with a learning rate of $1e^{-3}$ and a batch size of 265.

Table 3.3: AUC ROC, AUC PR, and EER on ShanghaiTech dataset [5] in full and half throughput modes for GEPC [6] and MPED-RNN [7].

Model	Data	FPS	AUC ROC	AUC PR	EER
GEPC [6]	[6]	10	0.6906	0.5951	0.35
		20	0.7372	0.6427	0.31
	Ours	10	0.6888	0.5905	0.35
		20	0.7223	0.6023	0.32
MPED-RNN [7]	[7]	10	0.6645	0.5733	0.37
		20	0.7023	0.5869	0.36
	Ours	10	0.6685	0.5661	0.37
		20	0.6679	0.5487	0.37

The results of this experiment can be seen in Tab. 3.3. In line with current practices, we report Area Under the Receiver Operating Characteristic Cure (AUC ROC), Area Under the Precision-recall Curve (AUC PR), and the Equal Error Rate (EER). With GEPC, we can see that Ancilia more than measures up to the task, with only a 1.5% drop in AUC ROC at full throughput and less than a 0.2% drop in AUC ROC at half throughput. AUC PR shows a more substantial drop of 4% at full throughput, but goes down to less than 0.5% at half throughput. Equal Error Rates are almost identical, seeing almost no change (less than 0.01) when using Ancilia. MPED-RNN, which displayed lower overall accuracy in all regards to begin with, sees a more significant drop in AUC ROC at full throughput, losing 3.5%. However, at half throughput the AUC ROC actually increases when using Ancilia, though only by 0.5%. The AUC PR results mirror that of GEPC, dropping 3.8% at full throughput and 0.7% at half throughput. The Equal Error Rates are once again nearly identical.

Being able to perform a high-level task such as anomaly detection while maintaining accuracies so close to current SotA in research, demonstrates Ancilia's ability to produce quality data, suitable for intelligent surveillance applications.

3.6.3 Real-time System Performance

Algorithmic accuracy is vital for ensuring the information provided by high-level cognitive tasks is beneficial for surveillance applications. However, Ancilia's ability to perform in real-time is equally important. We conduct a series of experiments, evaluating the runtime performance of Ancilia on different hardware, with different scenario intensities, and for increasing number of local nodes per hardware device. We focus on the performance of the local node, as the global node is completely decoupled from the algorithmic pipeline and has no noticeable effect on throughput or latency.

We choose three different hardware configurations for these experiments: a high-end server, a lower-end server, and a high-end workstation, as seen in Tab. 3.4. For our scenarios, we use the DukeMTMC-video dataset [1] and pick three scenes with different crowd densities: normal density, heavy density, and extreme density. The distribution of detection density in each scenario, as well as their effect on throughput, can be seen in Fig. 3.5. Note that what is considered "normal density" will change based on application environment, which is why we report on such a wide range. Each video lasts for 32k frames, with 7k frames warm-up and cool-down. We test using 1, 2, 4, 6, and 8 local nodes on a single system, showing how throughput and latency scale in such cases. Each experiment is conducted three times, the throughput and latency averaged across runs. The results of these experiments can be seen in Tab. 3.5 and Fig. 3.4.

Under normal crowd density, Server A and Workstation are both able to achieve over 50 FPS with up to four local nodes, with an end-to-end latency of 1.60 and 1.58 seconds respectively. This is well above FPS required by action recognition and

Table 3.4: System configurations. Stats are per CPU/GPU of the listed type.

Name	Processor			GPU		
	Model	Cores	Clock Speed	Model	CUDA Cores	VRAM
Server A	2× EPYC 7513	32	2.6 GHz	4× V100	5120	32 GB
Server B	2× Xeon E5-2640 v4	10	2.4 GHz	2× Titan V	5120	12 GB
Workstation	Threadripper Pro 3975WX	32	3.50 GHz	3× A6000	10752	48 GB

Table 3.5: Average throughput and latency. Data collected using the Workstation with varying local node counts.

Crowd Density	Nodes	Server A		Server B		Workstation	
		FPS	Latency (s)	FPS	Latency (s)	FPS	Latency (s)
Normal (70 detections per second)	1	82.31	1.17	52.45	1.52	96.88	0.87
	2	77.59	1.15	39.50	2.05	84.57	1.00
	4	53.40	1.60	-	-	56.27	1.58
	6	33.43	1.99	-	-	36.40	2.27
	8	23.64	2.05	-	-	26.60	2.84
Heavy (216 detections per second)	1	57.47	1.80	38.97	2.62	67.25	1.53
	2	50.05	2.07	32.85	3.16	58.95	1.77
	4	32.09	3.45	-	-	33.99	3.98
	6	18.51	4.17	-	-	21.08	6.89
	8	13.35	5.87	-	-	15.48	9.54
Extreme (744 detections per second)	1	19.84	5.29	14.67	7.37	19.00	5.73
	2	20.76	5.09	16.56	6.54	18.45	5.81
	4	10.95	11.64	-	-	10.29	12.49
	6	6.25	20.70	-	-	5.93	21.87
	8	4.53	28.48	-	-	4.18	31.19

anomaly detection algorithms at full throughput, and the latency is low enough to be suitable for most surveillance applications where the main concern is to notify authorities in time for appropriate response. Both Server A and Workstation are able to handle 6 local nodes in the normal scenario while maintaining above 30 FPS. Workstation is able to maintain above 26 FPS while running all 8 local nodes, while Server A drops to just below 24 FPS at 8 local nodes. Server B is able to achieve over 50 FPS with a single node but falls just short of 40 FPS while handling two nodes simultaneously. Due to having only two GPUs with limited VRAM, Server B was unable to run 4 or more nodes concurrently.

Heavy crowd density proves more challenging, with both Server A and Workstation only able to achieve above 30 FPS with up to 4 nodes. The end-to-end latency is also longer than it was under normal crowd density, with all systems seeing between a 50% to 100% increase in most cases, and up to a 230% increase at the mose extreme. Server

A and Workstation are able to mainatin above 15 FPS at 6 and 8 nodes respectively, while Server A drops to just above 13 FPS at 8 nodes. Server B behaves similarly to how it did with normal crowd density, still able to maintain above 30 FPS for up to 2 nodes, though with slightly low throughput. Assuming only half throughput was needed for high-level tasks, Server B would still be suitable for running up to two nodes.

With the extreme crowd density scenario, Ancilia begins to struggle. None of the systems are able to achieve above 30 FPS even with a single camera, putting full throughput action recognition out of reach. Server A is able to achieve above 20 FPS with 2 nodes (but notably not with 1) and Workstation fall short even with 1 node. Both Server A and Workstation can maintain above 10 FPS at 4 nodes, but both drop to around 6 and 4 FPS at 6 and 8 nodes, respectively. [133] argues that 5 FPS is suitable for tacking pedestrians, and while that is true, high-level tasks that rely on detailed human motion, such as action recognition and anomaly detection, often struggle for accuracy when running below 10 FPS. Another issue is with the increased latency. Running 6+ nodes, Server A and Workstation have latencies over 20 seconds, which is suitable for many surveillance applications, but might be too much for those that require sharper response times. Combined with the low throughput, it becomes difficult to recommend running more than 4 nodes on a single system with Ancilia when operating under extreme crowd density, expect for applications where low throughput and high latency are not as much of a concern. Server B is unable to achieve 30 FPS, but does stay around 15 FPS for both 1 and 2 nodes, making it suitable for half throughput in action recognition and anomaly detection.

Interestingly, with extreme crowd density we start to see unusual behavior with both Servers having worse performance with a singe node than they do with 2 nodes. This is likely caused by the abundance of CPU resources available to them with their

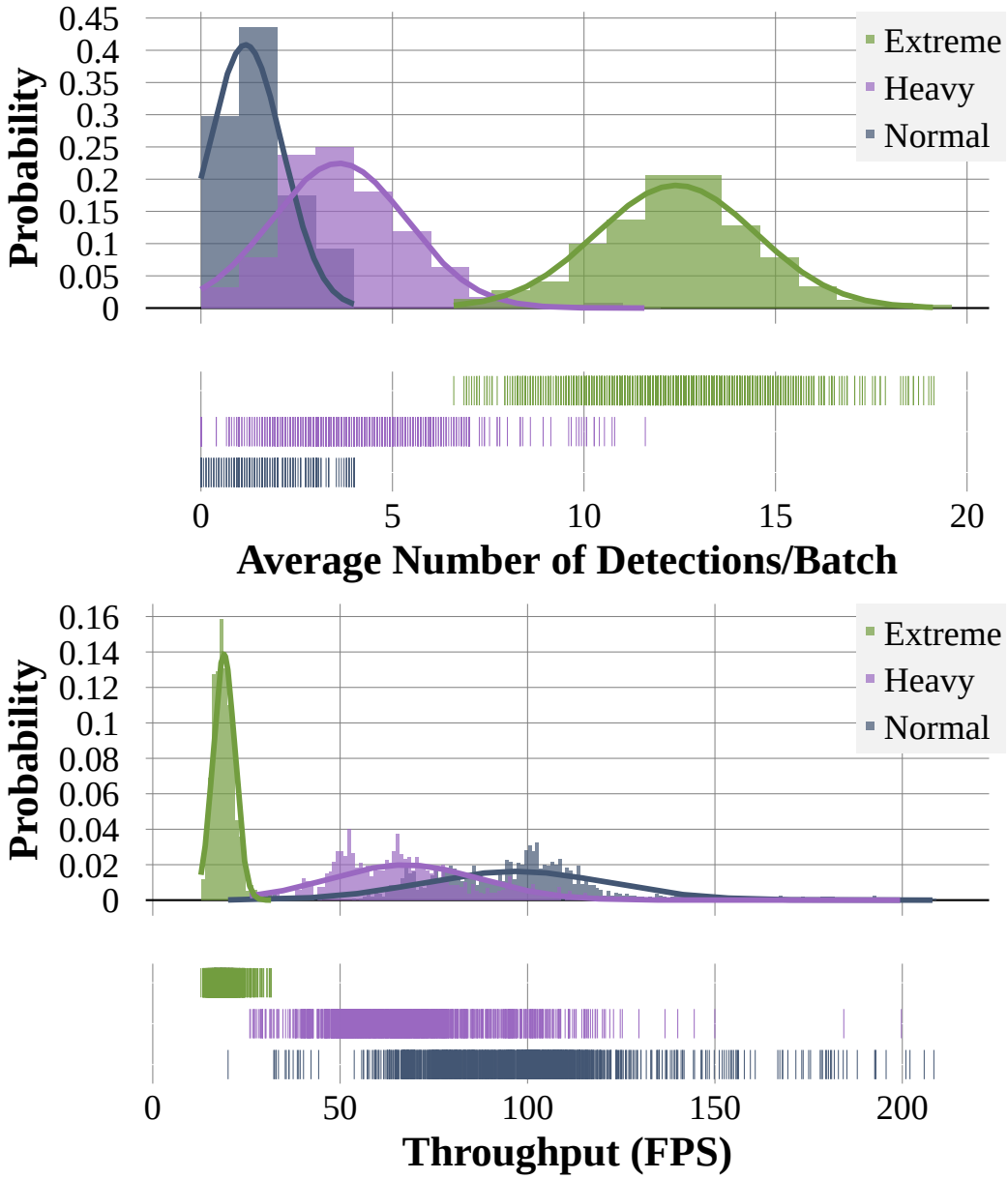

Figure 3.5: Distribution of detections for different crowd densities and its effect on throughput. Data collected using the Workstation with a single local node.

dual CPU configuration and a single node being unable to fully utilize them. As such, the behavior of both servers in the extreme crowd density scenario does not start to match the expected behavior and mimic the other systems until multiple nodes are being run simultaneously. This behavior is not too concerning, considering it does not make sense to purchase such a high-end server class machine for only running a single local node, when a more latency focused workstation would be both cheaper and more effective.

Overall, Ancilia is able to meet the needs of high-level cognitive tasks while still achieving performance suitable for real-time intelligent surveillance applications. Exact performance is dependent on both the hardware used and the intensity of the scene, but these results show that even for the most extreme of scenarios, Ancilia can be used to provide intelligent assistance to surveillance applications.

3.6.4 Effect of Batch Size on Real-time Performance

To understand the effect of batch size on end-to-end latency and throughput, we test using a single node on Workstation but varying the batch size. The results of this can be seen in Fig. 3.6. As expected, both latency and throughput increase with batch size across all densities. The jump in throughput from a batch size of 1 to a batch size of 10 is the most dramatic, with diminishing returns using larger batch sizes, while increases in latency tend to be more proportional. However, due to the high-level tasks needing 30 frames, the end-to-end latency is not directly representative of the latency of performing high-level tasks.

Overall, a balance needs to be struck between throughput, end-to-end latency, and batch size. Too high of an end-to-end latency will effect the speed at which detected objects of interest raise alarms, while a lower throughput can affect high-level task accuracy, as seen in Sec. 3.6.2. Likewise, having too small of a batch size means more batches need to be processed before high-level tasks can operate. A batch size of 15 strikes this balance well, with less than a second of end-to-end latency of 0.87

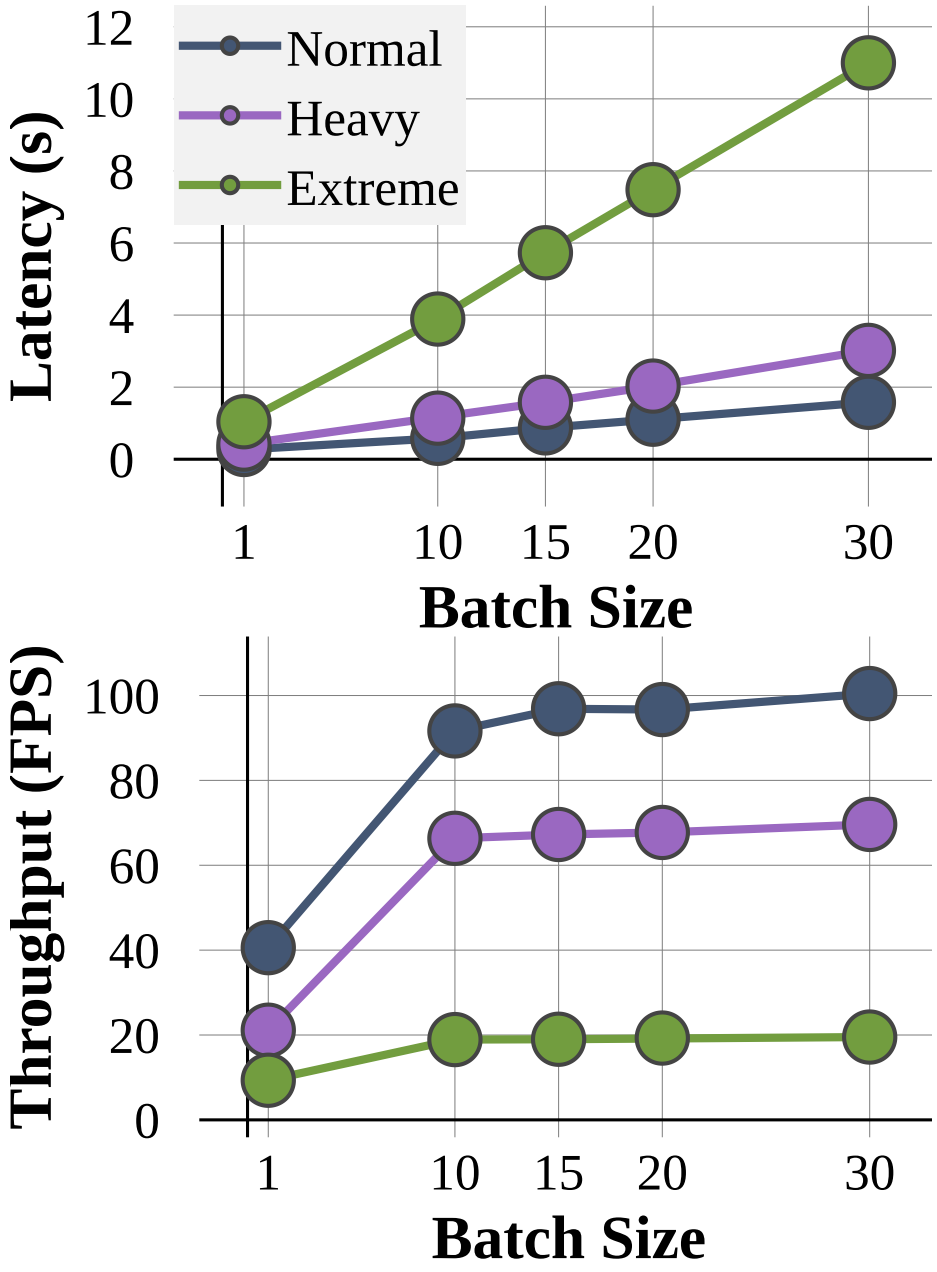

Figure 3.6: Throughput and latency trends with respect to batch size across different crowd densities. Data collected using Workstation with a single local node.

Table 3.6: Effect of different batch sizes on throughput and latency.

Crowd Density	Batch Size	FPS	Latency (s)
Normal	1	40.58	0.27
	10	91.66	0.58
	15	96.88	0.87
	20	96.69	1.11
	30	100.46	1.58
Heavy	1	21.13	0.42
	10	66.35	1.14
	15	67.25	1.58
	20	67.73	2.03
	30	69.62	3.02
Extreme	1	9.34	1.04
	10	18.92	3.89
	15	19.00	5.73
	20	19.28	7.48
	30	19.52	11.00

seconds and a throughput of 96.88 FPS in normal density, and only needing to process two batches for high-level tasks. This proves similar for heavy and extreme crowd densities as well, though the throughput is higher and latency is lower, as expected.

3.7 Conclusion

In this article we presented Ancilia, an end-to-end scalable intelligent video surveillance system for the Artificial Intelligence of Things. Through empirical evaluation, Ancilia has demonstrated its ability to bring state-of-the-art artificial intelligence to real-world surveillance applications. Ancilia performs high-level cognitive tasks (i.e. action recognition and anomaly detection) in real-time, all while respecting ethical and privacy concerns common to surveillance applications.

CHAPTER 4: Efficient and Scalable High-Resolution Networks for Real-Time Multi-Person 2D Human Pose Estimation

4.1 Introduction

Two-dimensional human pose estimation is a common task used in many popular smart applications and has made substantial progress in recent years. There are two primary approaches to 2D human pose estimation. The first is a top-down approach, where cropped images of humans are provided and the network uses those cropped images to produce human keypoints. Top-down approaches rely on object detectors to provide initial human crops, thus they often come with relatively higher computation cost, and are not truly end-to-end. The second is a bottom-up approach, where a network works off the original image and produces human keypoints for all people in the image. While these methods often do not quite reach the accuracy that is possible with state-of-the-art (SotA) top-down approaches, they come with relatively lower model size and computational overhead. Even so, SotA bottom-up approaches are still quite large and computationally expensive. The current SotA [8] having 63.8 million parameters and requiring 154.3 billion floating-point operations.

Many emerging Internet-of-Things (IoT) applications require lightweight real-time multi-person pose estimation at the edge, next to the cameras. This is more pronounced in a broad range of smart and connected applications with demands for continuous human activity analysis and behavioral monitoring. Few examples are video surveillance, patient monitoring, and public safety [85, 134, 135]. These applications demand agile but highly accurate human pose estimation that can run next to the cameras on the IoT edge devices. Despite this, there has been a dearth of attention towards developing lightweight bottom-up methods capable of real-time ex-

ecution under constrained computational resources. To address the gap, there is a need for a family of lightweight real-time human pose estimation models that achieves accuracy comparable to SotA.

In this paper, we present EfficientHRNet[1], a family of lightweight scalable networks for high-resolution and efficient real-time bottom-up multi-person pose estimation. EfficientHRNet unifies the principles of SotA EfficientNet [136] and HRNet [21], and presents a new formulation that enables near SotA human pose estimation while being more computationally efficient than all other bottom-up methods. Similar to HRNet, EfficientHRNet uses multiple resolutions of features to generate keypoints, but in a much more efficient manner. At the same time, it uses EfficientNet as a backbone and adapts its scaling methodology to be better suited for human pose estimation. To enable lightweight real-time execution, EfficientHRNet further expands the EfficientNet formulation to not only scale below the baseline, but also jointly scale down the input resolution, High-Resolution Network, and Heatmap Prediction Network. Through this, we create a family of networks that can address the entire domain of real-time 2D human pose estimation while being flexible towards accuracy and computation requirements of an application.

We evaluate accuracy on the COCO dataset [126] and real-time performance on the Nvidia NX Xavier. Figure. 4.1 demonstrates how our models provide equivalent or higher accuracy at lower computational costs than their direct peers. When comparing to SotA bottom-up models, baseline EfficientNet competes in accuracy while requiring much less computation, resulting in faster inference. Compared to HRNet[2] [21], EfficientHRNet achieves 0.4% higher accuracy while reducing computation requirements by 34%. When comparing to HigherHRNet [8] and PersonLab [137], EfficientHRNet sees between a 1.7% to 5.1% decrease in accuracy, while re-

[1]The source code of EfficientHRNet has been provided here: `https://github.com/TeCSAR-UNCC/EfficientHRNet`

[2]Bottom-up implementation reported in [8]

ducing computation requirements by an impressive 83% to 93%. This results in a 3.4x FPS increase over HigherHRNet. Even when comparing to models designed specifically for lightweight execution, such as Lightweight OpenPose [138], a scaled-down EfficientHRNet is able to achieve 10.1% higher accuracy while further reducing computation by 15%, maintaining similar FPS while requiring $\frac{1}{3}$ the power. In addition, the scaled-down backbone models have been evaluated in isolation on ImageNet. The results demonstrate competitive accuracies while achieving greater efficiency than their peers.

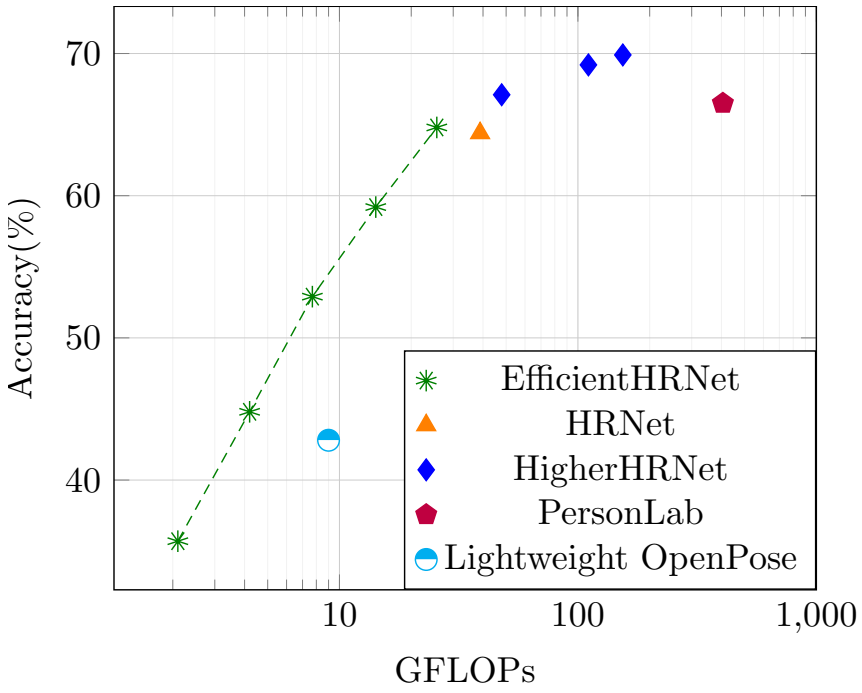

Figure 4.1: Comparison of computational complexity and accuracy between bottom-up human pose estimation methods measured on COCO val dataset. X-axis is logarithmic in scale.

4.2 Related Work

4.2.1 Top-down Methods

Top-down methods rely on first identifying all the persons in an image using a detector, and then detecting keypoints for a single person within a defined bounding

box. These single person [139, 140, 141, 142, 143, 144, 145] and multi-person [146, 147, 148, 149, 150] pose estimation methods often generate person bounding boxes using object detector [151, 152, 153, 154]. Regional Multi-Person Pose Estimation [149] adds symmetric spatial transformer network on top of single person pose estimator stacked hourglass network [143] to get high-quality regions from inaccurate bounding boxes, then detects poses using parametric non-maximum suppression.

4.2.2 Bottom-up Methods

Bottom-up methods [20, 29, 137, 155, 156, 157, 158, 159] detect identity-free keypoints in an image and group them into persons using various keypoints grouping techniques. Methods like [29] and [158] perform grouping by integer linear program and non-maximum suppression. This allows for faster inference times compared to top-down methods with almost similar accuracies. Other methods further improve upon prediction time by using greedy grouping techniques, along with other optimizations, as seen in [20, 137, 155, 156, 157]. For instance, OpenPose [20, 155] is a multi-stage network where one branch detects keypoints in the form of heatmaps, while the other branch generates Part Affinity Fields that are used to associate keypoints with each other. Grouping is done by calculating the line integral between all keypoints and grouping the pair that has the highest integral. Lightweight OpenPose [138] replaces larger backbone with MobileNet [160] to achieve real-time performance with fewer parameters and FLOPs while compromising on accuracy. PifPaf [156] uses Part *Intensity* Fields to detect body parts and Part *Associative* Fields for associating parts with each other to form human poses. In [157], a stacked hourglass network [143] is used both for predicting heatmaps and grouping keypoints. Grouping is done by assigning each keypoint with an embedding, called a tag, and then associating those keypoints based on the L_2 distance between the tag vectors. In this paper, we mainly focus on a highly accurate, end-to-end multi-person pose estimation method as in [157].

4.2.3 Top-down vs Bottom-up

While both top-down and bottom-up approaches can be applied to the domain of multi-person pose estimation, the way they function is inherently different. While bottom-up methods are designed specifically for end-to-end multi-person pose estimation, most top-down approaches require multiple instances and the use of external detectors, and are generally not end-to-end in nature. This makes direct quantitative comparisons between these two approaches impractical. As such, this paper focuses primarily on the domain of bottom-up multi-person pose estimation.

4.2.4 Multi-scale High-Resolution Networks

Feature pyramid networks augmented with multi-scale representations are widely adopted for complex and necessary computer vision applications like segmentation and pose estimation [161, 162, 163, 164, 165]. Recovering high-resolution feature maps using techniques like upsampling, dilated convolution, and deconvolution are also widely popular for object detection [163], semantic segmentation [166, 167, 168, 169, 170, 171] and pose estimation [29, 143, 158, 164, 165, 172, 173]. Moreover, there are several works that focus on generating high-resolution feature maps directly [8, 21, 117, 174, 175, 176, 177]. HRNet [21, 177] proposes to maintain high-resolution feature maps throughout the entire network. HRNet consists of multiple branches with different resolutions across multiple stages. With multi-scale fusion, HRNet is able to generate high resolution feature maps and has found its application in object detection, semantic segmentation, and pose estimation [21, 117, 177], achieving remarkable accuracy. Recently, DSPNet [178] is proposed for lightweight single-person pose estimation. EfficientNet [136] based, it has a pyramid architecture using lightweight up-sampling unit and achieves high accuracy, becoming the SotA top-down approach. Following HRNet, HigherHRNet for multi-person pose estimation [8] is proposed which uses HRNet as base network to generate high resolution feature maps,

and further adds a deconvolution module to predict accurate, high-quality heatmaps. HigherHRNet achieves SotA accuracy on the COCO dataset [126], surpassing all existing bottom-up methods. In this paper, we adopt the principles of HigherHRNet [8] for generating high-resolution feature maps with multi-scale fusion for predicting high quality heatmaps.

4.2.5 Model Scaling

Previous works on bottom-up pose estimation [8, 20, 21, 143, 155, 157] often rely on large backbone networks, like ResNet [179] or VGGNet [180], or large input resolutions and multi-scale training for achieving SotA accuracy. Recent works [8, 21] show that increasing the channel dimension of otherwise identical models can further improve accuracy. EfficientNet [136] and RegNet [181] show that by jointly scaling network width, depth, and input resolution, better efficiency for image classification can be achieved compared to previous SotA networks using much larger models. More recently, EfficientNet's lite models remove elements, such as squeeze and excite and swish layers, to make the network more hardware friendly. Inspired by EfficientNet, EfficientDet [182] proposes a compound scaling method for object detection along with efficient multi-scale feature fusion. We observe that there is a lack of an efficient scaling method for multi-person pose estimation, especially for embedded devices. Lightweight pose estimation models which are scalable and comparatively accurate are needed for computer vision applications which focus on real-time performance. Our proposed compound scaling, also inspired by EfficientNet, is a method that jointly scales the width, depth, and input resolution of EfficientHRNet, as well as the repetition within the *high-resolution modules*, explained in Section 4.3. In addition, this compound scaling allows our EfficientNet backbone to scale below the baseline B0, creating even lighter weight models.

4.2.6 Real-Time Pose Estimation

While most work in the field focuses on accuracy in isolation, some recent works have been developed that shift the focus more to real-time inference. In [183], focus is placed on real-time execution using a densely connected residual module and high-resolution feature maps, similar to [21], for accurate and lightweight single person pose estimation able to achieve real-time performance with an impressive 39 FPS on an Nvidia 1080TI. In [138], OpenPose [155] is modified to use a MobileNet [160] backbone and fewer refinement stages, creating a multi-person bottom-up model that achieves 28 FPS using the Intel OpenVINO Toolkit [184] on an Intel NUC 6i7KYB. Nvidia has also been focusing on real-time inference, releasing trt_pose [185], a single person pose estimation model optimized with TensorRT and DeepStream [186], achieving up to 251 FPS on the Nvidia Jetson Xavier [187].

4.3 EfficientHRNet

We have developed a family of lightweight, scalable networks for real-time multi-person human pose estimation called EfficientHRNet. This section gives an overview of EfficientHRNet and introduces the formulation for the compound scaling of EfficientHRNet's sub-networks.

4.3.1 Network Architecture and Formulation

EfficientHRNet, shown in Figure. 4.2, comprises of three sub-networks: (1) Backbone Network, (2) High-Resolution Network, and (3) Heatmap Prediction Network.

4.3.1.1 Backbone Network

The first stage of EfficientHRNet is the backbone, consisting of a modified of EfficientNet [136] altered to scale below the baseline, as discussed in Section 4.3.2. The backbone outputs four different resolution feature maps of decreasing resolutions $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$, and $\frac{1}{32}$ the size of the input image. These feature maps are passed into the main body of the network, called the High-Resolution Network.

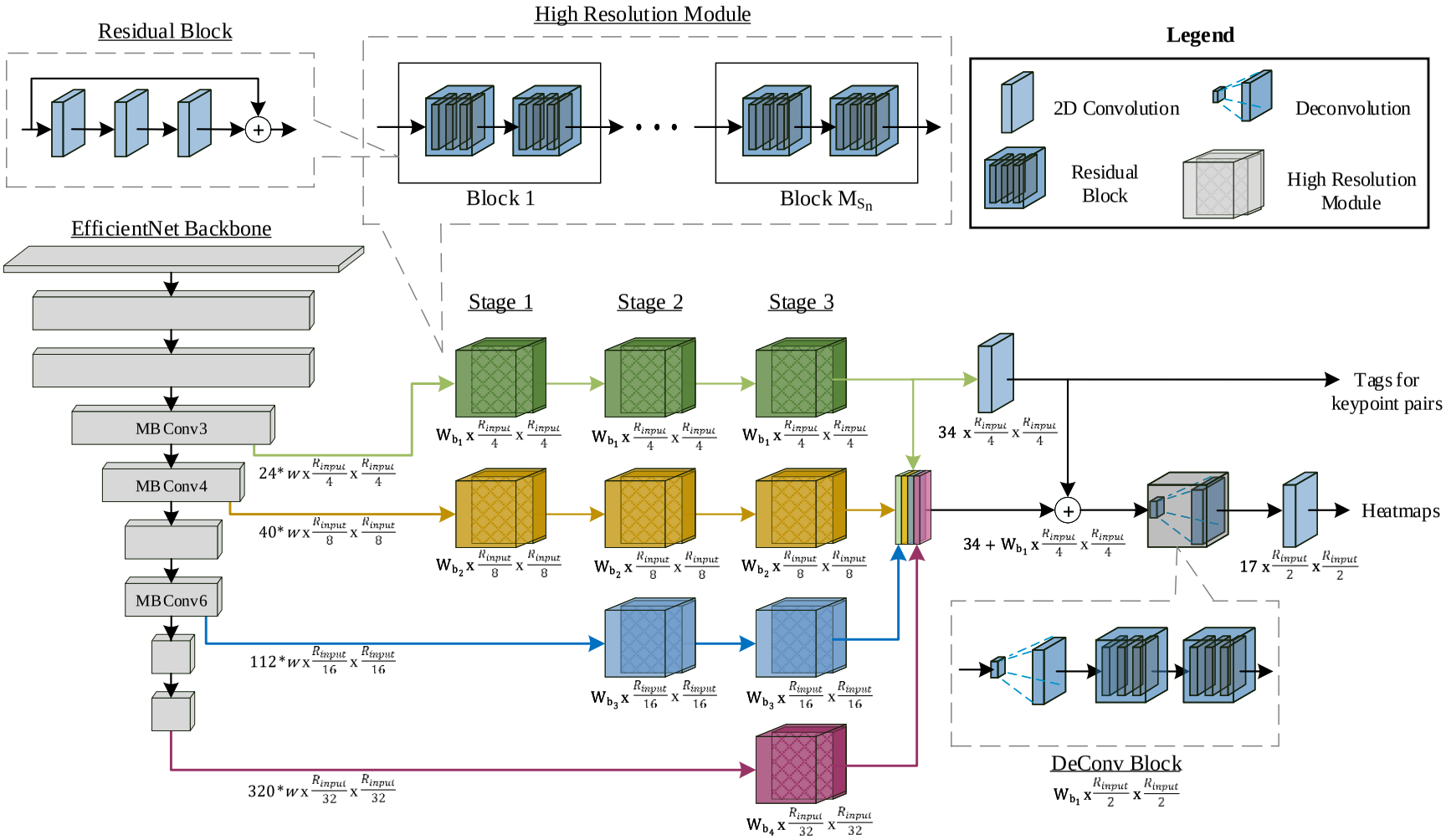

Figure 4.2: A detailed illustration of the EfficientHRNet architecture. Consisting of a backbone EfficientNet, a High-Resolution Network with three stages and four branches (denoted by different colors), and a Heatmap Prediction Network. EfficinetHRNet is completely scalable, allowing network complexity to be customized for target applications.

4.3.1.2 High-Resolution Network

The High-Resolution Network is inspired by HRNet [21, 177] and HigherHRNet [8]. Borrowing the principles of these higher resolution networks brings two major advantages:

1. By maintaining multiple high-resolution feature representations throughout the network, heatmaps with a higher degree of spatial precision are generated.

2. Repeated multi-scale fusions allow for high-resolution feature representations to inform lower-resolution representations, and vice versa, resulting in robust multi-resolution feature representations that are ideal for multi-person pose estimation.

Figure 4.2 presents a detailed architecture illustration of EfficientHRNet. It shows the three sub-networks: the Backbone Network, the High-Resolution Network, and the Heatmap Prediction Network. It also provides equations showing how the network scales the input resolution R_{input} and the width of feature maps W_{b_n}, which will be further explained in Section 4.3.2.

The High-Resolution Network has three stages s_1, s_2, and s_3 containing four parallel branches b_1, b_2, b_3, and b_4 of different resolutions. The first stage s_1 starts with two branches b_1 and b_2, with each consecutive stage adding an additional branch, until all four branches are present in s_3. These four branches each consist of *high resolution modules* with a width of W_{b_n}. Each branch b_n contains feature representations of decreasing resolutions that mirror the resolutions output by the Backbone Network, as shown in Figure. 4.2 and the following expression:

$$W_{b_n} \times \frac{R_{input}}{2^n + 1} \tag{4.1}$$

For instance, stage 2 (s_2) has three branches of resolutions $\frac{1}{4}$, $\frac{1}{8}$, and $\frac{1}{16}$ of the original

input image resolution and a width W_{b_n}. Moreover, each *high resolution module* is made up of a number of blocks, M_{s_n}, each containing two *residual blocks*, of which each perform three convolution operations with a residual connection.

4.3.1.3 Heatmap Prediction Network

The Heatmap Prediction Network is used to generate human keypoint predictions. In order to predict more accurate heatmaps, a *DeConv block* is added on top of the High-Resolution Network, as proposed in [8]. Transposed convolution is used to generate high quality feature maps which are $\frac{1}{2}$ the original input resolution. The input to the *DeConv block* is the concatenation of the feature maps and predicted heatmaps from the High-Resolution Network, as shown below:

$$34 + W_{b_1} \times \frac{R_{input}}{4} \times \frac{R_{input}}{4} \tag{4.2}$$

Two *residual blocks* are added after the deconvolution to refine the up-sampled feature maps. After the *DeConv block*, 1x1 convolution is used to predict heatmaps and tagmaps in a similar fashion to [157], the feature map size of each shown below:

$$\begin{aligned} T_{size} &= 34 \times \frac{R_{input}}{4} \times \frac{R_{input}}{4} \\ H_{size} &= 17 \times \frac{R_{input}}{2} \times \frac{R_{input}}{2} \end{aligned} \tag{4.3}$$

The grouping process clusters keypoints into multiple persons by grouping keypoints whose tags have minimum L_2 distance. Like [8], the High-Resolution Network is scale-aware and uses multi-resolution supervision for heatmaps during training to allow the network to learn with more precision, even for small-scale persons. From the ground truth, heatmaps for different resolutions are generated to match the predicted keypoints of different scales. Thus, the final heatmaps loss is the sum of mean squared errors for all resolutions. However, as high resolutions tagmaps do not converge well,

tagmaps are trained on a resolution $\frac{1}{4}$ of the original input resolution, as in [157].

4.3.2 Compound Scaling Method

This section details the compound scaling methodology, which jointly scales all parts of EfficientHRNet, as seen in Figure. 4.2 and Table 4.1. The aim of EfficientHRNet is to provide a family of models optimized for both accuracy and efficiency, which can be scaled to meet a diverse set of memory and compute constraints.

Table 4.1: Efficient scaling configs for EfficientHRNet

Model	Input Size (R_{input})	Backbone Network	Width per Branch (W_{b_1}, W_{b_2}, W_{b_3}, W_{b_4})	Blocks per Stage (M_{s_2}, M_{s_3}, M_{s_4})	Tags (T_{size})	Heatmaps (H_{size})
H_0 ($\phi = 0$)	512	$B0$	32, 64, 128, 256	1, 4, 3	128	256
H_{-1} (ϕ = -1)	480	B_{-1}	26, 52, 103, 206	1, 3, 3	120	240
H_{-2} (ϕ = -2)	448	B_{-2}	21, 42, 83, 166	1, 2, 3	112	224
H_{-3} (ϕ = -3)	416	B_{-3}	17, 34, 67, 133	1, 1, 3	104	208
H_{-4} (ϕ = -4)	384	B_{-4}	14, 27, 54, 107	1, 1, 2	96	192

Previous works on bottom-up human pose estimation and semantic segmentation mostly scale the base network by using bigger backbone networks like ResNet [179] and VGGNet [180], using large input image sizes, or using multi-scale training to achieve high accuracies. However, these methods rely on scaling only a single dimension, which has limited effectiveness. Recent works [136, 181] show notable performance on image classification by jointly scaling the width, depth, and input image resolution. Inspired by EfficientNet, EfficientDet [182] proposes a similar compound scaling method for object detection, which jointly scales the backbone network, multi-scale feature network, and the object detector network. We propose a heuristic-based compound scaling methodology for computer vision applications, specifically bottom-up human pose estimation and semantic segmentation, using EfficientHRNet. Based on [136], EfficientHRNet's methodology uses a scaling coefficient ϕ to jointly scale the Backbone Network, the High-Resolution Network, and Task-Specific Head. More precisely, the EfficientNet backbone is scaled below the baseline and the rest of EfficientHRNet is scaled down in order to maintain near SotA accuracy while creating

lightweight and flexible networks.

4.3.2.1 Backbone Network.

The same width and depth scaling coefficients are maintained as in EfficientNet [136]. In order to meet the demands of running models on constrained devices, a new formulation for scaling EfficientNet below the baseline and into a more compact model is provided.

Starting with the baseline EfficientNet-B0 scaling coefficients:

$$\begin{aligned} depth : d &= 1.2^{\phi} \\ width : w &= 1.1^{\phi} \\ resolution : r &= 1.15^{\phi} \end{aligned} \tag{4.4}$$

ϕ, i.e. ϕ = -1, -2, -3, -4, is inverted to calculate the scaling multipliers for the compact EfficientNet models, which is symbolized as B_{-1}, B_{-2}, B_{-3} and B_{-4} respectively. As an example, in order to take the baseline resolution, 224, and scale it down for our B_{-1} model, we would take r, from (4.4), with $\phi = -1$. This would result in a resolution scaling coefficient of 1.15^{-1}, i.e. 0.87, leaving a scaled resolution size of $ceil(224 * 0.87) = 195$. This pattern repeats for B_{-2} through B_{-4}, and can be seen in Table 4.2. We train these compact EfficientNet models (B_{-1} to B_{-4}) on ImageNet and use the resulting models for the Backbone Network in EfficientHRNet.

4.3.2.2 High-Resolution Network.

The High-Resolution Network has three stages and four branches with four different feature map sizes. Each branch n also has a different width W_{b_n} and our baseline H_0 model has a width of 32, 64, 128, and 256 for each branch respectively. We selectively pick a width scaling factor of 1.25 and scale down the width using the following

equation:

$$W_{b_n} = (n \cdot 32) \cdot (1.25)^{\phi} \tag{4.5}$$

where n is a particular branch number and ϕ is the compound scaling coefficient.

Furthermore, within each stage, each *high resolution module* has multiple *blocks* M_{s_n} which repeat a number of times, as seen in Table 4.1. In our baseline EfficientHRNet H_0 model, *blocks* within each stage repeat 1, 4, and 3 times respectively. We found that the number of repetitions in stage 3 had the largest impact on accuracy. Therefore, the number of repetitions within a *high resolution module* M_{s_2} decreases linearly as the models are scaled down, starting with stage 2 until reaching a single repetition and then moving on to stage 3, as shown in Table 4.1.

4.3.2.3 Heatmap Prediction Network

The *DeConv block* is scaled in the same manner as the width of the High Resolution Network (4.5). The Heatmap Prediction Network outputs tags and heatmaps whose width remains fixed across all models.

4.3.2.4 Input Image Resolution.

The EfficientNet layers downsample the original input image resolution by 32 times. Thus, the input resolution of EfficientHRNet must be dividable by 32, and is linearly scaled down as shown in the following equation:

$$R_{input} = 512 + 32 \cdot \phi \tag{4.6}$$

The final result of this compound scaling methodology on EfficientHRNet H_0 to H_{-4} can be seen in Table 4.1.

4.4 Experimental Results

This section evaluates our method for scaling EfficientNet below the baseline through classification on the popular ImageNet [188] and CIFAR-100 [189] datasets. Then, an exhaustive evaluation of five different EfficientHRNet models is conducted on the challenging COCO [126] dataset and compared to SotA methods. Additional, metrics on real-time inference are reported using the Nvidia Jetson NX Xavier and compared to SotA lightwieght approaches. Finally, a qualitative evaluation of EfficientHRNet is presented, illustrating both where the models excel and where they fall short.

4.4.1 Classification for Compact EfficientNet

4.4.1.1 Dataset

ImageNet [188] has been a long time standard benchmark for object classification and detection thanks to its annual contest, the ImageNet Large Scale Visual Recognition Challenge, that debuted in 2010. The challenge uses a subset of the full dataset with over a million images spread out over 1000 object classes. For training, validating, and testing purposes, the trimmed ImageNet is divided into three sets: 800k images will be used for training the network, 150k will be used for validation after each epoch, and 50k will be used for testing the fully trained model. CIFAR-100 [189] consists of 100 object classes each with 500 images for training, and 100 for testing. This relatively small dataset helps illuminate our lightweight models, which start to struggle with the larger ImageNet as ϕ decreases, designed for resource constrained devices that might not need to classify as many object classes.

4.4.1.2 Training

We use random rotation, random scale, and random aspect ratio to crop the input images to the desired resolutions based on the current EfficientNet model. Color jitter was also used to randomly change the brightness, contrast, saturation, and hue of the RGB channels using principle component analysis [190]. The images are then

normalized using per channel mean and standard deviation. Each model was trained using Stochastic Gradient Descent with Momentum [191] and a weight decay of $1e-4$. The weights were initialized using the Xavier algorithm [192] and underwent five warm-up epochs with a learning rate of $1e-4$ that increased linearly until it reached 0.05. The networks were then trained for an additional 195 epochs and followed the step decay learning rate scheduler [193] that reduces the learning rate by a factor of 10 every 30 epochs.

4.4.1.3 Testing

Compact EfficientNet models were tested for accuracy based on their respective test sets. For a fair comparison, the number of ImageNet test samples were reduced to 10,000 to match the test set of CIFAR-100, where the batch size is 1. These results can be seen in Table 4.2.

Table 4.2: Compact EfficientNet performance on ImageNet and CIFAR-100 datasets.

	Input		ImageNet		CIFAR-100	
Model	size	FLOPs	Params	Top-1	Params	Top-1
B0	224	0.4B	5.3M	75	4.1M	81.9
B_{-1}	195	0.3B	4.5M	73.8	3.5M	81.4
B_{-2}	170	0.2B	3.4M	71.3	2.5M	79.8
B_{-3}	145	0.1B	2.8M	68.5	1.9M	78.2
B_{-4}	128	0.05B	1.3M	65.6	1.3M	74.3

4.4.1.4 Results on ImageNet and CIFAR-100

Looking at B_{-1} there is a 15% reduction in parameters and 25% reduction in operations, yet an accuracy drop of only 1.2% and 0.5% on ImageNet and CIFAR-100 respectively. More impressively, B_{-2} sees a 35-40% reduction in parameters and a 50% reduction in operations, yet only a 3.7% and 2.1% drop in accuracy. This minor accuracy loss is negligible compared to the massive reduction in model size and computation, allowing for much faster inference and deployment on low-power and resource constrained devices. In the most extreme, B_{-4} shows a parameter reduction

of 68-75% and a 87.5% decrease in operations while having an accuracy drop of 9.4% and 7.6% on ImageNet and CIFAR-100. While the accuracy drop is a bit more significant here, the massive reduction in computation allows for much more flexibility when it comes to deployment in systems where a lightweight approach is needed. This gives us a solid foundation on which to build EfficientHRNet.

4.4.2 2D Human Pose Estimation for EfficientHRNet

4.4.2.1 Dataset

COCO [126] consists of over 200k images with 250k person instances, each annotated with 17 keypoints. COCO is divided into three sets, *train*, *val*, and *test*, which have 57k, 5k, and 40k images respectively. Additionaly, *test-dev* is a subset of *test* with 20k images and is used for fair comparison with other works, where possible. COCO evaluation meterics use mean average precision (AP) and are detailed on the COCO website[3].

4.4.2.2 Training

We use random rotation, random scale, and random translation for data augmentation. Following [8], we generate two ground truth heatmaps of different sizes, $\frac{1}{2}$ and $\frac{1}{4}$ of the original input size respectively. Each EfficientHRNet model is trained using Adam optimizer [194] and weight decay of $1e-4$. While we saw little difference in accuracy between using Adam and SGD with momentum in our initial testing, Adam was selected for its higher speed of convergence and overall effect on training time. All models from H_0 to H_{-4} are trained for a total of 300 epochs with a base learning rate of $1e-3$, decreasing to $1e-4$ and $1e-5$ at 200^{th} and 260^{th} epochs respectively. To maintain balance between heatmap loss and grouping loss, we weight the losses at 1 and $1e-3$ respectively.

[3] `http://cocodataset.org/#keypoints-eval`

4.4.2.3 Testing

Models are tested using both single scale and multi-scale heatmaps, as is common. Following [157], the output detection heatmaps across different scales are averaged and the tags concatenated into higher dimension tags, making the models considerably more scale-invariant.

Table 4.3: Comparisons with SotA bottom-up methods on COCO2017 test-dev dataset. Numbers for HRNet come from a bottom-up approach outlined in [8].

Method	Backbone	Input size	# Params	FLOPs	AP
w/o multi-scale test					
OpenPose	-	-	25.94M	160B	61.8
Hourglass	Hourglass	512	277.8M	206.9B	56.6
PersonLab	ResNet-152	1401	68.7M	405.5B	66.5
PifPaf	ResNet-152	-	-	-	66.7
HRNet	HRNet-W32	512	28.5M	38.9B	64.1
HigherHRNet	HRNet-W32	512	28.6M	47.9B	66.4
HigherHRNet	HRNet-W48	640	63.8M	154.3B	68.4
H_0	$B0$	512	23.3M	25.6B	64.0
H_{-1}	B_{-1}	480	16M	14.2B	59.1
H_{-2}	B_{-2}	448	10.3M	7.7B	52.8
H_{-3}	B_{-3}	416	6.9M	4.2B	44.5
H_{-4}	B_{-4}	384	3.7M	2.1B	35.5
w/ multi-scale test					
Hourglass	Hourglass	512	277.8M	206.9B	63.0
Hourglass	Hourglass	512	277.8M	206.9B	65.5
PersonLab	ResNet-152	1401	68.7M	405.5B	68.7
HigherHRNet	HRNet-W48	640	63.8M	154.3B	70.5
H_0	$B0$	512	23.3M	25.6B	67.1
H_{-1}	B_{-1}	480	16M	14.2B	62.3
H_{-2}	B_{-2}	448	10.3M	7.7B	55.0
H_{-3}	B_{-3}	416	6.9M	4.2B	45.5
H_{-4}	B_{-4}	384	3.7M	2.1B	39.7

4.4.2.4 Results on COCO2017 *test-dev*

Table 4.3 compares EfficientHRNet with other works on COCO *test-dev* set. As explained in Section 4.2.3, top-down methods, such as [21, 149, 178], are inherently not end-to-end. As such, we limit our comparisons to the domain of bottom-up multi-

person human pose estimation. The baseline H_0 model with single-scale testing serves as an efficient and accurate model for bottom-up methods as it is almost comparable to HRNet [8] in accuracy, losing by only 0.1%, while having 18% less parameter and 34% fewer FLOPs. H_0 outperforms Hourglass [143] in both single-scale and multi-scale testing by 7.4% and 1.6% respectively, with H_0 remarkably having about $\frac{1}{10}$ the model size and number of FLOPs as Hourglass. The highest performing of all models on COCO *test-dev*, HigherHRNet [8], beats H_0 in accuracy by 4.4%, but at the cost of nearly triple the model size and more than 6x the computation. In all cases where H_0 loses in accuracy, it more than makes up for it in a reduction in parameters and operations. Additionally, our H_{-1} model, with only 16M parameters and 14.2B FLOPs, outperforms both OpenPose [20, 155] and Hourglass [143], demonstrating EfficientHRNet's efficiency and suitability for low-power and resource constrained devices.

As EfficientHRNet is scaled down using the compound scaling method mentioned in Section 4.3.2, we see somewhat minor drops in accuracy with significant drops parameters and FLOPs as compared to the baseline H_0 model. H_{-1} has 31.3% less parameters and 44.5% less FLOPs as compared to H_0 while only being 4.9% less accurate. Similarly, our lightest model H_{-4} is 84% smaller and has 91.7% less FLOPs, with a less than 45% drop in accuracy. Interestingly, EfficientHRNet is the only bottom-up pose estimator that is able to provide such lightweight models while still having accuracies that are comparable to SotA bottom-up methods, as illustrated by both Table 4.3 and Figure. 4.1. These results nicely show the validity of our approach to scalability and efficiency in EfficientHRNet.

4.4.2.5 Results on COCO2017 *val*

We report EfficientHRNet accuracy on COCO *val*, noting the number of parameters and FLOPs, and compare it with other bottom-up methods. In addition, to accurately assess suitability for real-time performance on embedded devices, we inference our

models as well as one of our closest competitors, HigherHRnet [8], on the Jetson NX Xavier, first converting the models to ONNX and then inferencing in TensorRT. FPS results for Lightweight OpenPose are on an Intel NUC 6i7KYB as reported in [138]. Looking at Table 4.4, we can see that PersonLab is a very large network. With nearly 3x as many parameters and 16x as many operations as our most complex model H_0 it is too large to even run on the NX Xavier, despite an improvement of less than 2% accuracy. Still, the baseline H_0 model outperforms HRNet [21] with 0.4% more accuracy, 18% fewer parameters and 34% fewer FLOPs. H_{-2} and H_{-3} models outperform Lightweight OpenPose [138] in accuracy while having fewer FLOPs. H_{-4} has the worst accuracy of any model in Table 4.4. However, it boasts both the smallest model size and fewest number of operations, seeing an over 75% reduction from its lightest weight competitor. When looking at FPS, HigherHRNet becomes much less desirable, being the only model unable to achieve at least 20 FPS. H_0 is 3.4x faster while only being 2.3% less accurate. Comparing to Lightweight OpenPose, H_0 is 22% more accurate while only being 2 FPS slower. Scaling down to H_{-3} reduces EfficientHRNet's accuracy lead by to only 2%, but increases throughput to be 1.3x greater than Lightweight OpenPose. Our smallest model H_{-4} achieves an impressive 50 FPS, but at a substantial cost in accuracy. Note that we see an unusually high FPS drop in H_{-1}. This is due to H_{-1}'s input resolution and intermediate feature map sizes resulting in memory tiles that map poorly to the NX Xavier's Tensor Core architecture. In the following subsection, we provide further analysis and comparison regarding real-time execution on NX Xavier.

4.4.3 Real-Time Execution Analysis on Edge

Since real-time inference is highly dependent on the hardware utilized, we must account for more than just accuracy and throughput in our comparisons. To best compare accuracy and efficiency across differing platforms, we adopt the Accuracy•Efficiency (Æ) metric from [85]. Æ is simply the product of accuracy (measured in AP) and

Table 4.4: Comparisons with bottom-up methods on COCO2017 val dataset. Metrics and accuracy for HRNet come from a bottom-up approach outlined in [8] (FPS not reported). Lightweight OpenPose numbers were reported on the Intel NUC 6i7KYB. All other FPS results were preformed on the Nvidia Jetson NX Xavier [9].

Model	Input size	AP	# Params	FLOPs	FPS
PersonLab	1401	66.5	68.7M	405.5B	-
HRNet	512	64.4	28.5M	38.9B	-
HigherHRNet	512	67.1	28.6M	47.9B	6.68
Lightweight OpenPose	368	42.8	4.1M	9.0B	26
H_0 ($\phi = 0$)	512	64.8	23.3M	25.6B	22.95
H_{-1} (ϕ = -1)	480	59.2	16M	14.2B	20.43
H_{-2} (ϕ = -2)	448	52.9	10.3M	7.7B	24.53
H_{-3} (ϕ = -3)	416	44.8	6.9M	4.2B	33.78
H_{-4} (ϕ = -4)	384	35.7	3.7M	2.1B	50.96

Table 4.5: Æ comparisons with lightweight bottom-up approaches. Lightweight OpenPose reported on Intel NUC 6i7KYB (45W). All others Nvidia Jetson NX Xavier (15W).

Model	AP	FPS	Efficiency	Æ
HigherHRNet	67.1	6.68	0.445	29.850
Lightweight OpenPose	42.8	26	0.578	24.738
H_0 ($\phi = 0$)	64.8	22.95	1.530	99.144
H_{-1} (ϕ = -1)	59.2	20.43	1.362	80.630
H_{-2} (ϕ = -2)	52.9	24.53	1.635	86.492
H_{-3} (ϕ = -3)	44.8	33.78	2.252	100.89
H_{-4} (ϕ = -4)	35.7	50.96	3.397	121.273

efficiency (measured in FPS per Watt). Table 4.5 shows how EfficientHRNet compares when taking this into account. Note that Lightweight OpenPose reports results on Intel NUC 6i7KYB, which has a TDP of 45 Watts [138], while all other methods were measured on the Jetson NX Xavier with a maximum power draw of 15 Watts. We use these power numbers across all approaches in an attempt to create as fair a comparison as possible. EfficientHRNet greatly outperforms the competition in terms of Æ, with all models achieving an Æ score of over 80 while Lightweight OpenPose and HigherHRNet achieve scores of 25 and 30 respectively.

In terms of Æ, EfficientHRNet outperforms the competition between 3x to 5x. This is largely due to the poor throughput of HigherHRNet and the relatively higher power

NUC that Lightweight OpenPose reports on. HigherHRNet excels in accuracy and Lightweight OpenPose excels in FPS and model size, while EfficientHRNet is more equally balanced between accuracy, model size, throughput, and power consumption. This gives EfficientHRNet a leg up in terms of low-power, real-time inference, making its scalable models the new SotA for lightweight bottom-up human pose estimation for real-time edge applications.

Figure 4.3: Qualitative results for EfficientHRNet models on COCO2017 test. Left to right: simple, medium and complex examples.

4.4.4 Qualitative Analysis

To further demonstrate how EfficientHRNet models perform in relation to one another, we present qualitative results on COCO. Figure. 4.3 shows simple, medium, and complex examples for EfficientHRNet models H_0 to H_{-4}. We see that H_0 can accurately detect all but the most distant and occluded individuals. This accuracy is functionally identical to SotA models but is able to inference in real-time, making it immensely valuable for applications that require high accuracy but need to run in real-time or on low-power devices. Looking at H_{-1} we see that keypoints are accurately detected, but in medium and complex scenarios keypoint groupings become confused. Here, the confusion is minor enough that it can be filtered out with additional post processing, meaning that applications that require predicting complex scenarios on devices that can not fit H_0 can use H_{-1} with slightly decreased accuracy. For the medium scenario, there is also missed detection from the distant person occluded by the left-most surfboard, though such missed detections are relatively uncommon throughout the dataset. However, for simple scenarios there is little to no difference when compared with H_0. These qualities make H_{-1} a compelling model when using a device without enough memory resources for H_0 and when a minor amount of error is acceptable, or for applications that will only deal with simple scenarios. H_{-2} looks a lot like H_{-1}, but the confusion is worse, with multiple keypoint grouping being detected for a single person, and this even extends to simple scenarios. This would again require additional post processing, depending on the application. H_{-3} and H_{-4} follow the same pattern, with confusion continuing to get worse. Again, we see that actual keypoint detections themselves are fairly accurate, and the greatly reduced model size and computational complexity open up a wide range of additional devices capable of real-time performance. This makes the smaller models extremely compelling for real-time applications that can afford a certain amount of error but require the use of highly resource constrained devices, particularly in the case of simple

scenarios. This analysis helps visualize the relationship between detected keypoints and model size, and shows the overall affect on accurate human pose prediction as we move towards smaller models. This further validates EfficientHRNet as a family of high accuracy and efficient models capable of real-time 2D human pose estimation for a variety of embedded and resource constrained devices.

4.5 Conclusions

In this paper, we have presented EfficientHRNet, a family of scalable networks for high-resolution and efficient bottom-up multi-person pose estimation made for real-time execution on low-power edge devices. EfficientHRNet unifies the principles of state-of-the-art EfficientNet [136] and HRNet [21] to create a network architecture for lightweight real-time human pose estimation, and proposes a new compound scaling method that jointly scales down the input resolution, backbone network, and high-resolution feature network. EfficientHRNet is not only more efficient than all other bottom-up human pose estimation methods, but it can maintain accuracy competitive with state-of-the-art models on the challenging COCO dataset. Remarkably, EfficientHRNet can achieve this near state-of-the-art accuracy with fewer parameters and less computational complexity than other bottom-up multi-person pose estimation networks, all while being able to achieve 23 FPS on an Nvidia Jetson NX Xavier.

CHAPTER 5: Real-Time Online Unsupervised Domain Adaptation for Real-World Person Re-identification

5.1 Introduction

Person re-identification (ReID) is the task of matching a person in an image with other instances of that person in other images, either from the same camera or a different one. More specifically, it is associating a person's query with its match in a gallery of persons [22]. Person ReID is a common task in many real-world applications. Such applications include video surveillance (*e.g.* determining when unauthorized people are present in an area), public safety (*e.g.* understanding pedestrian motion to avoid accidents), and smart health (*e.g.* mobility assessment and fall detection for seniors needing assistance). Thus, achieving accurate and robust person ReID for any environment is an important research goal for the community.

Table 5.1: Challenges of Real-World Applications and if they are addressed in the UDA, OUDA, and R^{2}OUDA settings.
† Streaming data is simulated.

Real-World	UDA	OUDA	R^{2}OUDA (Ours)
Data from target domain is only available through a data stream.	✗	✓†	✓
Person crops are not provided and must be generated online.	✗	✗	✓
There is no guarantee that every identity will be available during training.	✗	✗	✓
The distribution of person crops must be determined online.	✗	✗	✓
Training time must be accounted for.	✗	✗	✓

Many methods have been developed for person ReID [195, 196, 197, 198], and many high quality datasets have been created for the task [1, 199, 200, 201, 202]. Deep learning approaches have been able to achieve incredible accuracies, nearly reaching saturation in some cases [203, 204, 205, 206]. However, person ReID is a highly context-specific task, and models trained on one dataset often fail to perform well on others [22]. Unsupervised Domain Adaptation (UDA) has been studied to

combat this domain shift [22, 201, 207, 208, 209, 210]. In UDA, initial training is performed on the labeled data of the source domain, and then inference is done in a different target domain. UDA methods generally achieve lower accuracies than State-of-the-Art (SotA) deep learning approaches that train directly on the target domain. However, recent approaches have begun to close that gap [211, 212, 213].

One common thread among these approaches is the reliance on having the entirety of the target domain available at training time. While this is convenient for research, many practical applications do not have unrestricted access to the entire target domain. Recently, [214] introduced the setting of Online Unsupervised Domain Adaptation (OUDA). OUDA specifies that data from the target domain can only be accessed through a data stream, bringing research more in line with real-world applications. OUDA adopts a batch-based relaxation [215] where different identities are separated among batches to simulate streaming data. OUDA also argues that confidentiality regulations make it such that many real-world applications can only store data for a limited amount of time, applying a restriction that image data cannot be stored beyond the batch in which it was collected.

Tab. 5.1 shows the challenges of real-world applications, and how UDA and OUDA fail to fully address them. Like UDA before it, OUDA uses hand-crafted person ReID datasets for the target domain. Not only is the data stream only simulated, but the provided person images were hand selected by the creators of the dataset. In a real-world system, person images need to be generated by the system itself, creating a layer of noise not present in hand-crafted datasets. Further, by using hand-crafted datasets, the distribution of person images is guaranteed to be suitable for training. Specifically, most person ReID dataset tend to have a fairly uniform distribution, having around the same number of person images for each identity [216]. However, in real-world applications, there is no guarantee that person images generated from streaming data will form a uniform distribution in identities. There is also no guarantee that

every identity in the dataset will be available for training. Additionally, in real-world applications, we often see multi-camera systems that rely on processing all this information in real-time. The UDA and OUDA settings do not address this.

To bring the field closer to the real-world, this paper proposes Real-World Real-Time Online Unsupervised Domain Adaptation (R^2OUDA), a setting designed to address the challenges found in real-world applications, as seen in Tab. 5.1. R^2OUDA defines four major considerations beyond the OUDA setting needed to develop systems for the real world. First, R^2OUDA considers that person images must be generated algorithmically from streaming data. Second, the distribution of data to be used in training must also be determined algorithmically. Third, R^2OUDA expands the batched-based relaxation [215] of online learning to use time segments, relating the conceptual mini-batch to the real-world notion of time inherent in streaming data. Fourth, R^2OUDA defines a time constraint such that the time spent training a single time segment cannot interfere with the training for subsequent time segments. The first two considerations address the noisy data inherent in real-world systems, while the last two considerations address the time-based streaming nature of data seen in real-time systems.

To address all aspects of the new R^2OUDA setting, this paper further proposes Real-World Real-Time Online Streaming Mutual Mean-Teaching (R^2MMT). R^2MMT is an end-to-end multi-camera system designed for real-world person ReID. Using object detection, pedestrian tracking, human pose estimation, and a novel approach for Subset Distribution Selection (SDS), R^2MMT is able to generate person crops directly from a data stream, filter them based on representation quality, and create a subset with a suitable distribution for real-time training. To show the viability of R^2MMT to meet the challenges of real-world applications, and to explore the breadth of the R^2OUDA setting, an exhaustive set of experiments were conducted on the popular and challenging DukeMTMC dataset [1]. Using R^2MMT, over 100 data

subsets were created and more than 3000 models were trained, capturing the trade-offs and limitations of real-world applications and the R^{2}OUDA setting. R^{2}MMT is a real-world system that can meet the demanding requirements of the proposed R^{2}OUDA setting, and is able to achieve over 73% Top-1 accuracy on DukeMTMC-reid, within 0.1% of comparable OUDA methods that cannot be directly applied for real-world applications.

To summarize, this paper's contributions are as follows:

- We define the setting of Real-World Real-Time Online Unsupervised Domain Adaptation, accounting for the challenges of real-world applications and bridging the gap between research and application.
- We propose Real-World Real-Time Online Streaming Mutal Mean-Teaching, a novel end-to-end multi-camera person ReID system designed to meet the challenges of R^{2}OUDA and real-world applications.
- We perform exhaustive experimentation, creating over 100 data subsets and training over 3000 models, to explore the breadth of the R^{2}OUDA setting and understand the trade-offs and limitations of real-world applications.

5.2 Related Work

The UDA setting for person ReID has been extensively explored by the research community [22, 210, 217, 218]. In general, there are two main categories of algorithms used to perform UDA for person ReID: style transfer methods and target domain clustering methods.

5.2.1 Style Transfer

Style transfer based methods generally use Generative Adversarial Networks (GANs) [219] to perform image-to-image translation [220], modifying images from the source domain to look like the target domain without affecting the context of the original images. [221] uses self-similarity and domain-dissimilarity to ensure transferred images

maintain cues to the original identity without matching to other identities in the target domain, while [222] introduces an online relation-consistency regularization term to ensure relations of the source domain are kept after transfer to the target domain. [208] separates transfers into factor-wise sub-transfers, across illumination, resolution, and camera view, to better fit the source images into the target domain. [207] uses a dual conditional GAN to transfer source domain images to multiple styles in the target domain, creating a multitude of training instances for each source identity. [201] uses a cycle consistent loss [223] with an emphasis on the foreground to better maintain identities between styles. [224] looks at domain shift as background shift and uses a GAN to remove backgrounds without damaging foregrounds, while a densely associated 2-stream network integrates identity related cues present in backgrounds.

5.2.2 Target Domain Clustering

Target domain clustering approaches focus on using clustering algorithms to group features of the target domain for use as labels to fine tune a neural network pretrained on the source domain [225]. This is usually done in an iterative fashion, where clustering is performed between training epochs to update the group labels as the model learns. [226] proposes using a dynamic graph matching framework to better handle large cross-camera variations. [227] introduces a self-similarity group to leverage part-based similarity to build clusters from different camera views. [216] utilizes a diversity regularization term to enforce a uniform distribution among the sizes of clusters. [228] introduces hybrid memory to dynamically generate instance-level supervisory signal for feature representation learning. [211] builds on [229], using two teacher models and their temporally averaged weights to produce soft pseudo labels for target domain clustering. [230] utilizes both target domain clustering and adversarial learning to create camera invariant features and improve target domain feature learning.

5.2.3 Online Unsupervised Domain Adaptation

While Online Unsupervised Domain Adaptation has been explored for other AI tasks [231, 232, 233, 234, 235, 236, 237], it was first defined for the field of person ReID in [214]. OUDA for Person ReID aims to create a practical online setting similar to that found in practical applications. OUDA builds upon the UDA setting by adding two considerations. First, data from the target domain is accessed via a data stream and not available all at once. Second, due to confidentiality concerns common in many countries, data from the target domain can only be stored for a limited time and only model parameters trained on that data may be persistent.

5.3 Proposed R^{2}OUDA Setting

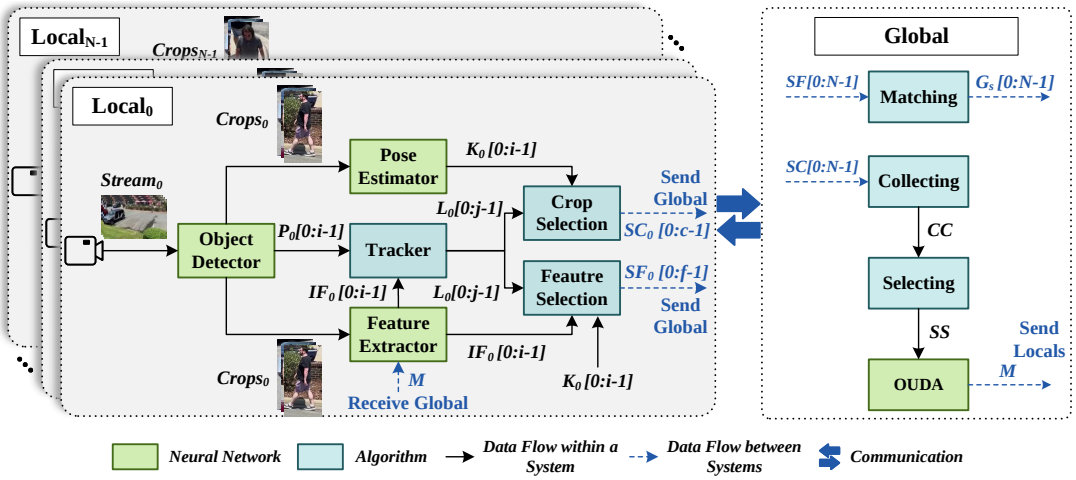

Figure 5.1: System view of Real-World Real-Time Online Streaming Mutual Mean-Teaching.

The proposed setting of Real-World Real-Time Online Unsupervised Domain Adaptation, building off OUDA [214], considers that we have access to a completely annotated source dataset D_S as well as partial access to an unlabeled target dataset D_T in the domain of our target application. In contrast to standard UDA, in both OUDA and R^{2}OUDA the data from D_T is only accessible as an online stream of data. Whereas both UDA and OUDA use person crops from hand crafted datasets,

R^{2}OUDA specifies that person crops from D_T must be generated algorithmically from the data stream. This reflects how data is gathered in the real world. Where hand selected crops from datasets are generally highly representative, crops generated from a data stream will have varying levels of quality. This introduces noise in D_T, both in quality and in the inevitable missed detections, which needs to be accounted for.

Additionally, hand crafted datasets choose person images to fit a distribution suitable for training. However, since crops in R^{2}OUDA are generated from streaming data, such a distribution can not be assumed. This leads to the second consideration of R^{2}OUDA, that the distribution of data to be used in training must be determined algorithmically. Instead of relying on a predefined set of person images, systems must generate their own data subset, determining its size and distribution appropriately. This also reflects the real-world, as it is rarely known beforehand the amount and distribution of person crops that will be collected by an application.

Continuing with the batched-based relaxation [215] of the online learning scenario proposed in [214], we further introduce a time constraint for R^{2}OUDA. First, instead of separating our "mini-batches" ("tasks" as defined in [214]) across identities, since R^{2}OUDA requires actual streaming data, the data stream is separated into discrete time segments. We consider that for a chosen time segment of length τ, the streaming data will be divided into equal, non-overlapping time segments of length τ whose combined contents are equivalent to the original data stream.

For R^{2}OUDA, we must account both for applications that run continuously (i.e. the total length of the data stream is infinite) and the fact that, in the real world, computation resources are not unlimited. This leads to the necessity of a time constraint, but one that is not simple to define. Training time is inherently linked to hardware, and there are many techniques to hide latency or increase throughput in system design. As such, we simply define the time constraint such that, for any time segment τ_i, the length of time spent training on data collected during τ_i must be such

to not interfere with the training for the data collected during τ_{i+1}. This is to prevent the training time deficit from increasing infinitely as i increases.

In summary, R^{2}OUDA introduces four new considerations to better match real-world applications:

- Person crops from the target domain must be generated algorithmically from a data stream.
- The selection and distribution of data to be used in training must be determined algorithmically.
- An expansion of the batch-based relaxation to use time segments, relating the conceptual mini-batch to the real-world notion of time inherent in streaming data.
- An additional time constraint such that the time spent training a single time segment cannot interfere with the training for any subsequent time segments.

5.4 Real-World Real-Time Online Streaming MMT

To address the challenges of R^{2}OUDA, we present Real-World Real-Time Online Streaming Mutual Mean Teaching, a novel multi-camera system for real-world person ReID. Similar to [85], R^{2}MMT is comprised of multiple Local Nodes and a single Global Node. Local nodes have access to the data stream directly from the cameras and are responsible for generating quality person images. The Global Node has access to all data generated by Local Nodes and is responsible for global ReID, subset distribution selection, and target domain training. An overview of R^{2}MMT can be seen in Fig. 5.1.

On the Local Node, YOLOv5 [112] is used as an object detector to find people in the video stream. Image crops are created for each person and sent to both a pose estimator (HRNet [117]) and a ReID feature extractor (ResNet-50 [179]). Coordinates for each person and features generated by the feature extractor are sent to a

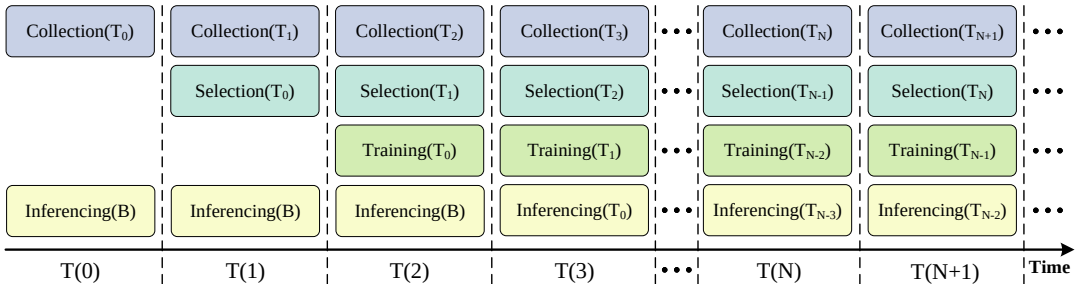

Figure 5.2: Illustration of computation overlap through time.

tracker [238] for local ReID. Afterward, feature and crop selection are performed to ensure that features and person crops sent to the Global Node for global ReID and crop collection are highly representative. This process utilizes person bounding box coordinates from the tracker to filter out any persons that have significant overlap (IoU >= 0.3) with other persons. This limits the number of crops used for training and features used for ReID containing multiple persons. The pose estimator is used to determine the quality of the features themselves. We reason that if a highly representative feature is present, then poses generated from the person crop should be of high confidence, while the number of keypoints present can help determine if there is significant occlusion or cutoff. Only crops and features with poses containing 15 or more keypoints (out of 17 total [126]) with at least 50% confidence are sent to the Global Node. This helps ensure that the quality of the crops used for training is similar to the quality of crops found in hand-crafted datasets.

On the Global Node, local identities and features are received from the Local Nodes and sent to a matching algorithm. This matching algorithm, as described in [85], performs global (i.e. multi-camera) ReID. Concurrently, person crops from all cameras are collected for a single time segment. Generally, far more features will be collected than can reasonably be used during training. For instance, when DukeMTMC-Video [1] is sampled every frame, the system produces over 4 million crops that pass feature selection. To reduce redundancy and computation, R^2MMT samples crops for selection once every 60 frames.

After all person crops from a single time segment are collected, the Subset Distribution Selection algorithm is used to create a subset that maintains a uniform distribution and number of crops suitable for training. R^{2}MMT uses an SDS algorithm based on the metric facility location problem [239]. We define that given a number of features in a metric space, we wish to find a subset of k features such that the minimum distance between any two features within the subset is maximized. However, this problem is known to be NP-hard [240], making it unsuitable for our real-world applications. R^{2}MMT instead uses a greedy implementation of the algorithm proven to be $\Omega(log\, k)$-competitive with the optimal solution while proving to be significantly faster, especially for larger sets of data [241]. For ease of readability, we adopt the nomenclature of K to mean the number of instances per identity. Therefore the total number of person crops in a subset k is equal to the number of identities in the dataset times K. To further reduce complexity, SDS is performed on the data from each camera individually, and their results are combined to form the complete subset. The SDS process helps ensure we have a uniform distribution of identities in our training data, similar to what is found in hand-crafted ReID datasets.

Once the training subset is created, domain adaptation is performed using Mutual-Mean Teaching (MMT) [211]. R^{2}MMT follows the training methodology described in [211], except that epochs and iterations are variable. Clustering is done using DBSCAN [242], as GPU acceleration allows it to perform much faster than CPU based approaches. Exact training parameters, both for pre-training on the source domain and domain transfer on the target domain, are as detailed in [211] unless otherwise noted.

Both SDS and training are time consuming, particularly when dealing with large amounts of data. To meet the time constraint of the R^{2}OUDA setting, R^{2}MMT utilizes a pipelined processing model, taking advantage of parallel computing resources while hiding the latency of the aforementioned tasks. An illustration of this pipelined

approach can be seen in Fig. 5.2. Crop collection, SDS, and training are separated into their own pipeline stages. This means that while a model collects data for the current time segment, SDS on that data will occur the following time segment, and the training for that subset will occur the time segment after that. More formally, during a single time segment T_N, a model trained on data from T_{N-3} is used to collect data from time segment T_N, while subset distribution selection is performed on data collected during T_{N-1} and another network is being trained on a subset created from data from T_{N-2}. All of these processes will finish before T_{N+1}. This means there will always be a latency of two time segments between collection and inference for a single time segment. However, due to the pipeline structure, training throughput remains at a rate of one time segment per time segment. This satisfies the time constraint of R^{2}OUDA.

5.5 Experimental Results

To explore the setting of R^{2}OUDA, we select the Market 1501 dataset [199] as the source domain and the DukeMTMC dataset [1] as the target domain. The DukeMTMC dataset is desirable as a target domain because it has both a video dataset (DukeMTMC-video) and a hand crafted person ReID dataset (DukeMTMC-reid), both in the same domain. The video dataset is required in order to satisfy the streaming data constraint of the R^{2}OUDA setting. The hand crafted ReID dataset brings two benefits. First, it allows us to directly observe the effect of noisy system generated crops compared hand selected person images when used for training. Second, testing on the ReID dataset allows direct comparison with works done in the UDA and OUDA space. As such, all our Top-1 accuracies are reported on the DukeMTMC-reid dataset. Similarly, we determining subset size, we treat the number of identities for both DukeMTMC-reid and DukeMTMC-video to be 702, as described in [1]. The number of person crops in a subset k is always equal to $k \times 702$.

For all experiments, R^{2}MMT is used to perform domain adaptation. Parameters

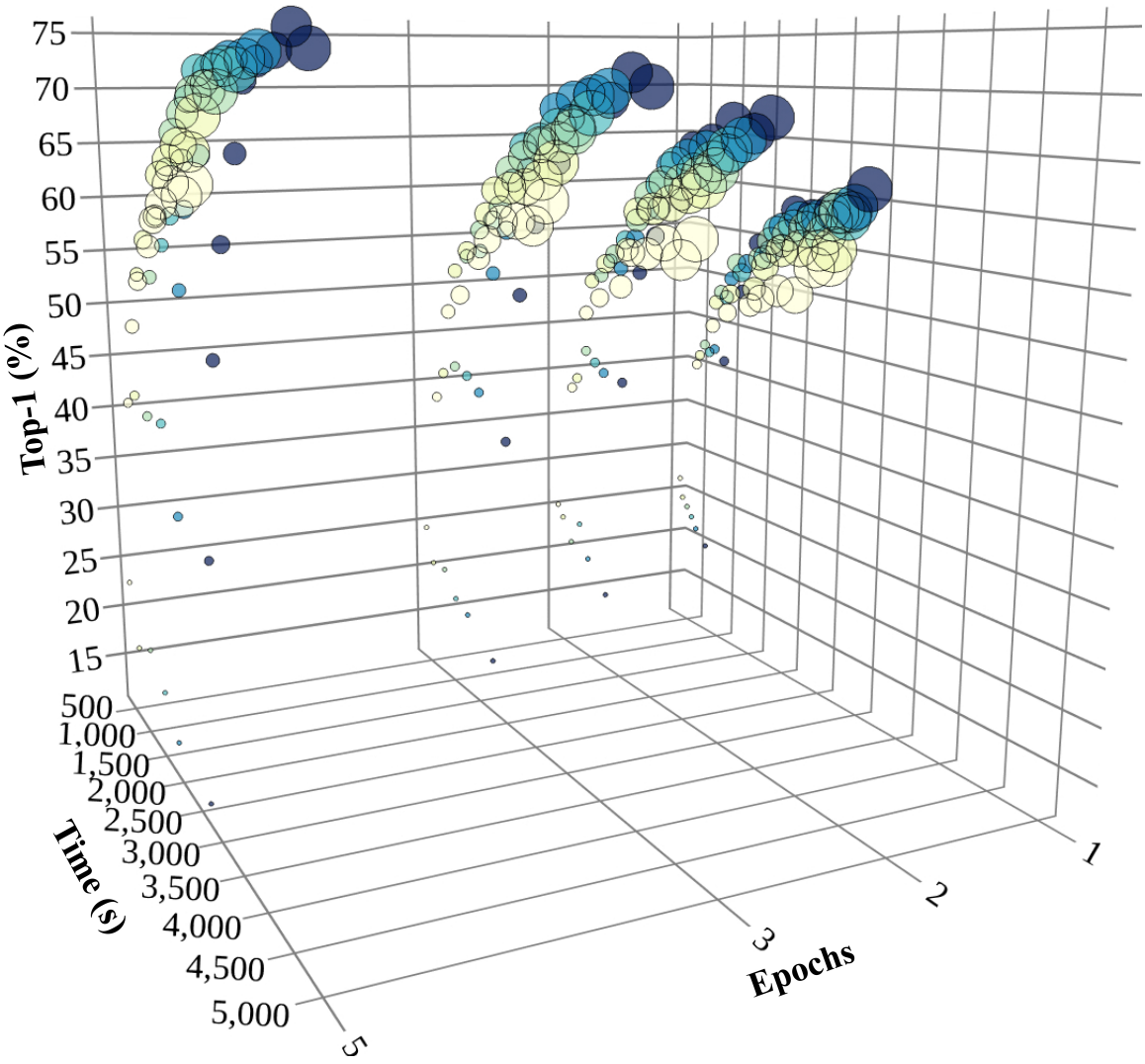

Figure 5.3: Results exploring SDS on the hand crafted DukeMTMC-reid dataset [1] plotted in three-dimensional space. Larger circles represent larger values of k.

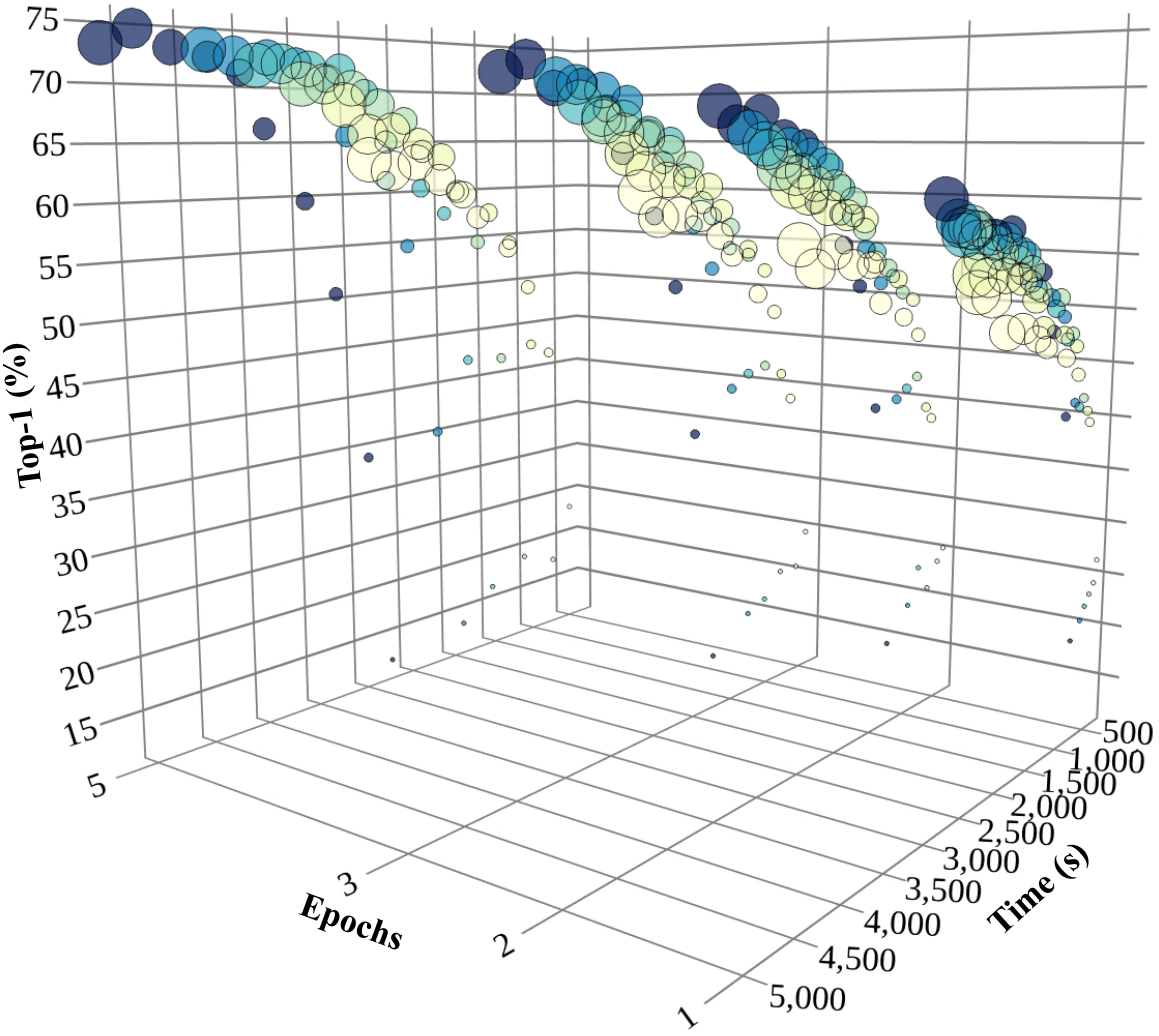

Figure 5.4: Results exploring SDS on the hand crafted DukeMTMC-reid dataset [1] plotted in three-dimensional space. Larger circles represent larger values of k.

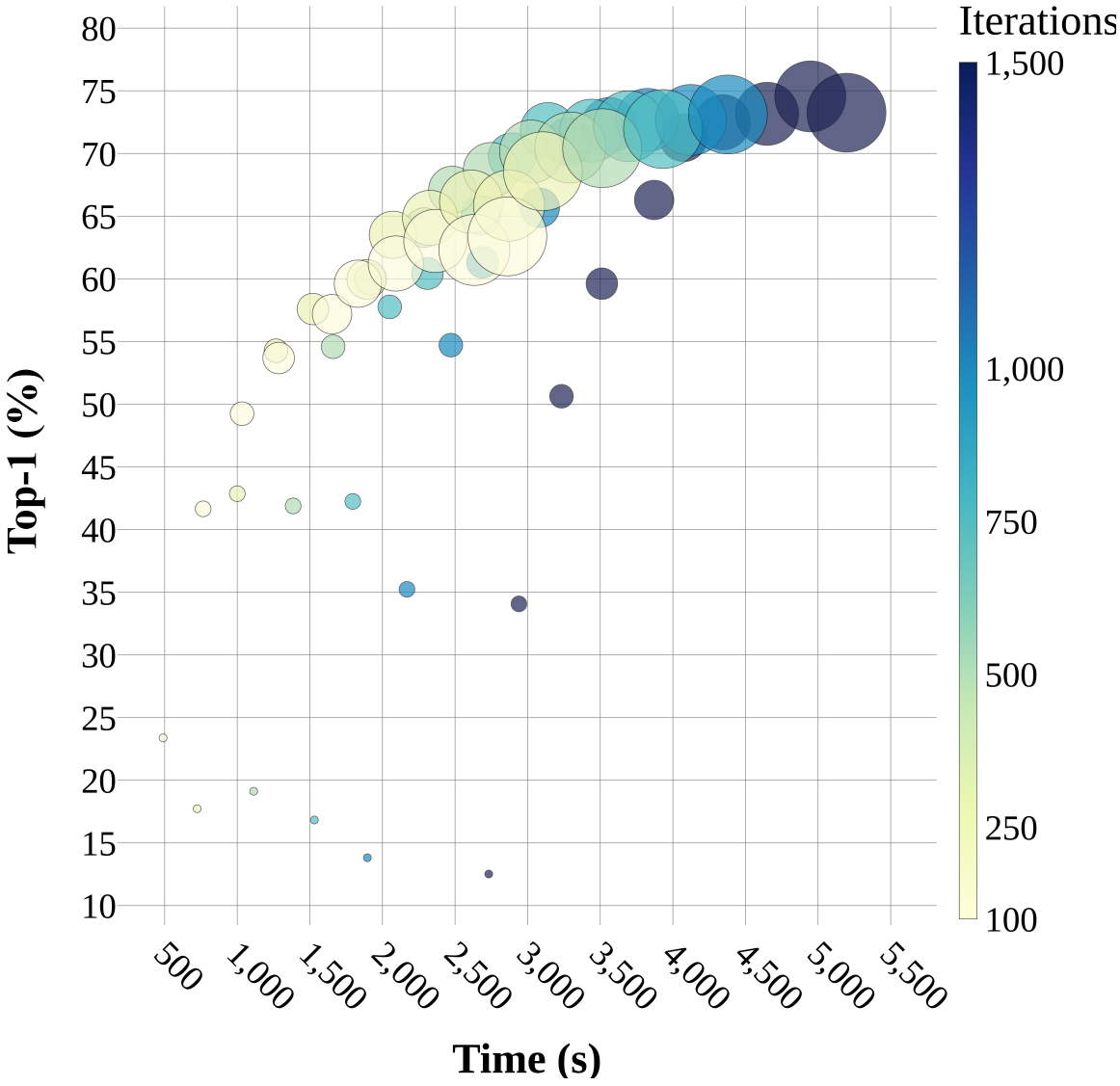

Figure 5.5: Results exploring SDS on the hand crafted DukeMTMC-reid dataset [1] showing a two-dimensional view when $E = 5$. Larger circles represent larger values of k.

in all experiments are the same as in [211], except where noted otherwise. All Local Nodes are run on a single server with two AMD EPYC 7513 CPUs, 256 GB of RAM, and three Nvidia V100 GPUs. The Global Node is run on a workstation with an AMD Threadripper Pro 3975WX CPU, 256 GB RAM, and three Nvidia RTX A6000 GPUs. All timing results presented in this section are using this Global Node.

5.5.1 Subset Distribution Selection

We first explore the effect of using our baseline Subset Distribution Selection algorithm for training on the DukeMTMC-reid dataset. By using hand selected person crops from the dataset, we remove the effect of noise generated by our system and single out the impact of our SDS algorithm and the reduction in amount of data on domain adaptation. We vary the number of person images per identity K, iterations per epoch I, and total epochs E as shown below. Note that using the entire DukeMTMC-reid dataset would be equivalent to $K = 25$.

$$\begin{aligned} K &\in [2, 4, 6, 8, 10, 12, 14, 16, 18, 20] \\ I &\in [100, 250, 500, 750, 1000, 1500] \\ E &\in [1, 2, 3, 5] \end{aligned} \tag{5.1}$$

These variable ranges lead to 240 training permutations, which is difficult to list in a single table. Instead, the results are plotted in a three-dimensional space and can be seen in Fig. 5.3, Fig. 5.4, and Fig. 5.5. *Training Time* and *Top-1* make up the x and y axes, *Epochs* are the z axis, *Iterations* are noted by color, and k is indicated by size, with bigger circles representing higher values of k. As the purpose of these experiments is to focus on the effects of our SDS algorithm, the system pipeline described in Sec. 5.4 is ignored and timing results count SDS and training sequentially. More detailed information on these experiments can be found in the

supplementary materials.

From these graphs, we can understand the general trend of the data. Intuitively, we see a fairly linear trend where more data generally results in higher Top-1 accuracy. Likewise, more iterations per epoch and more epochs also tend to result in higher accuracy. Interestingly, with lower values of k we see the reverse effect; more time spent training results in decreased accuracy, sometimes even below the pre-trained accuracy of 42.0%. In general, at least 6 person images per identity are needed to consistently learn, while we start to see diminishing returns at around 16 person images per identity. The top result occurs when $K = 20$, $I = 1500$, and $E = 5$, achieving a Top-1 accuracy of 74.55% with a training time of 82 minutes. This is only 3.5% less than what comparable algorithms are able to achieve in the UDA setting [211] and over 2% greater than the same algorithm in the OUDA setting [214]. When using the same hardware, R^{2}MMT is 2.6× faster than its UDA counterpart.

5.5.2 System Generated Data

As explained in Sec. 5.3, one of the requirements of the R^{2}OUDA setting is that person crops must be generated algorithmically from a data stream. As such, it is necessary to explore the effects of the noise this introduces. The structure of these experiments are exactly the same as in Sec. 5.5.1, except that instead of using DukeMTMC-reid, R^{2}MMT generates data from the DukeMTMC-video dataset. Similar to Sec. 5.5.1, we ignore the system pipeline and focus on the effects of the generated data. Based on the larger amount of data available in DukeMTMC-video, the ranges for our experimental variables are adjusted as shown below. Using all generated data would be equivalent to $K = 99$.

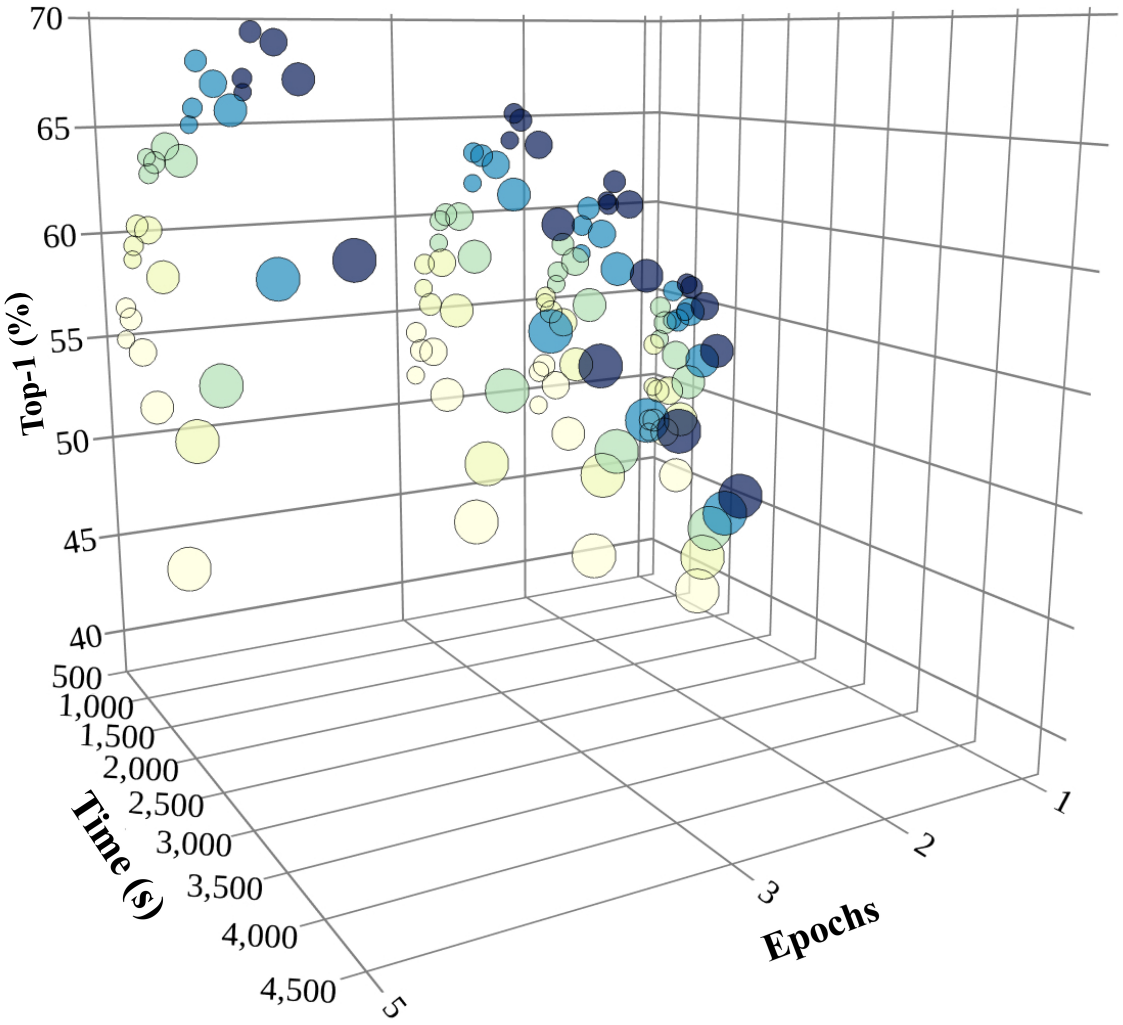

Figure 5.6: Results exploring the use of system generated data using DukeMTMC-video [1] plotted in three-dimensional space. Larger circles represent larger values of k.

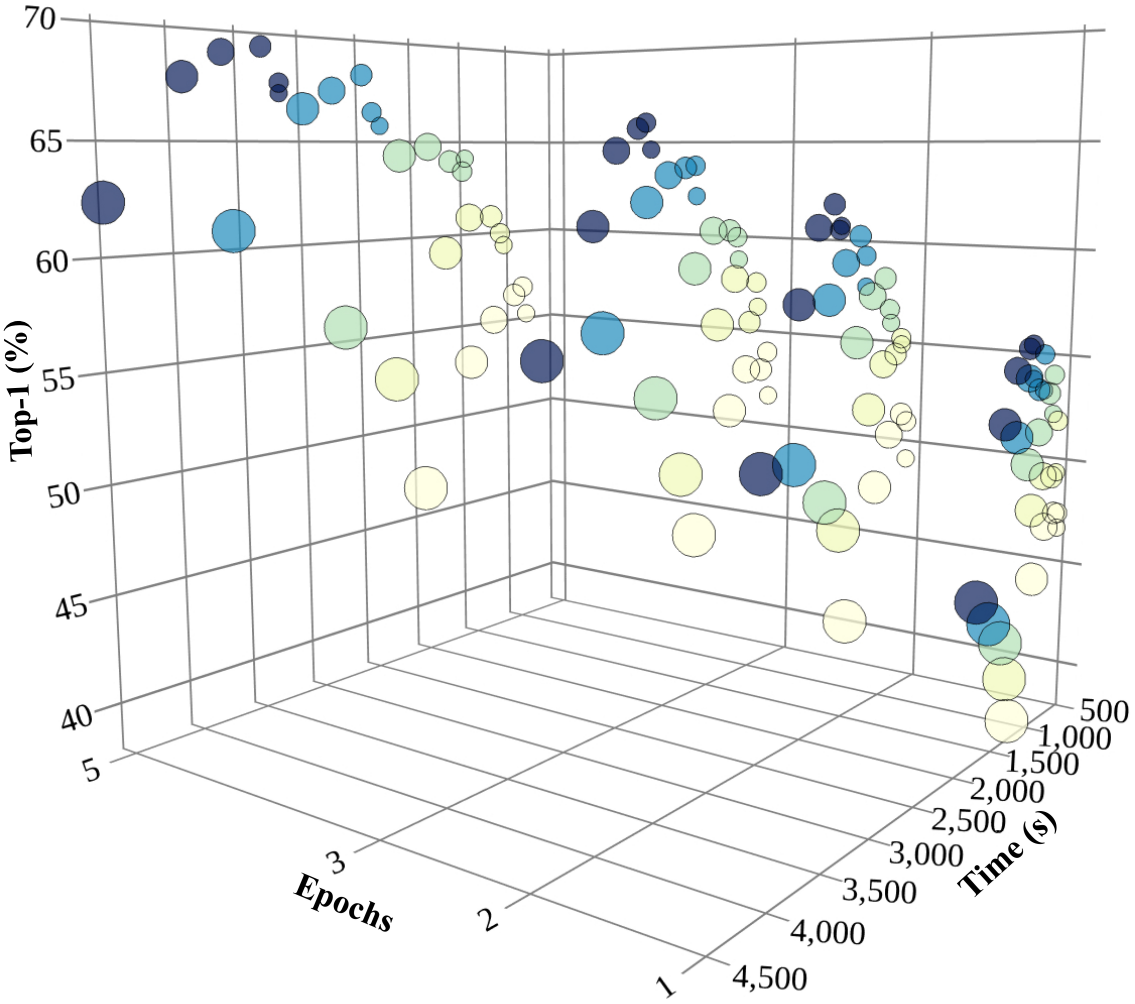

Figure 5.7: Results exploring the use of system generated data using DukeMTMC-video [1] plotted in three-dimensional space. Larger circles represent larger values of k.

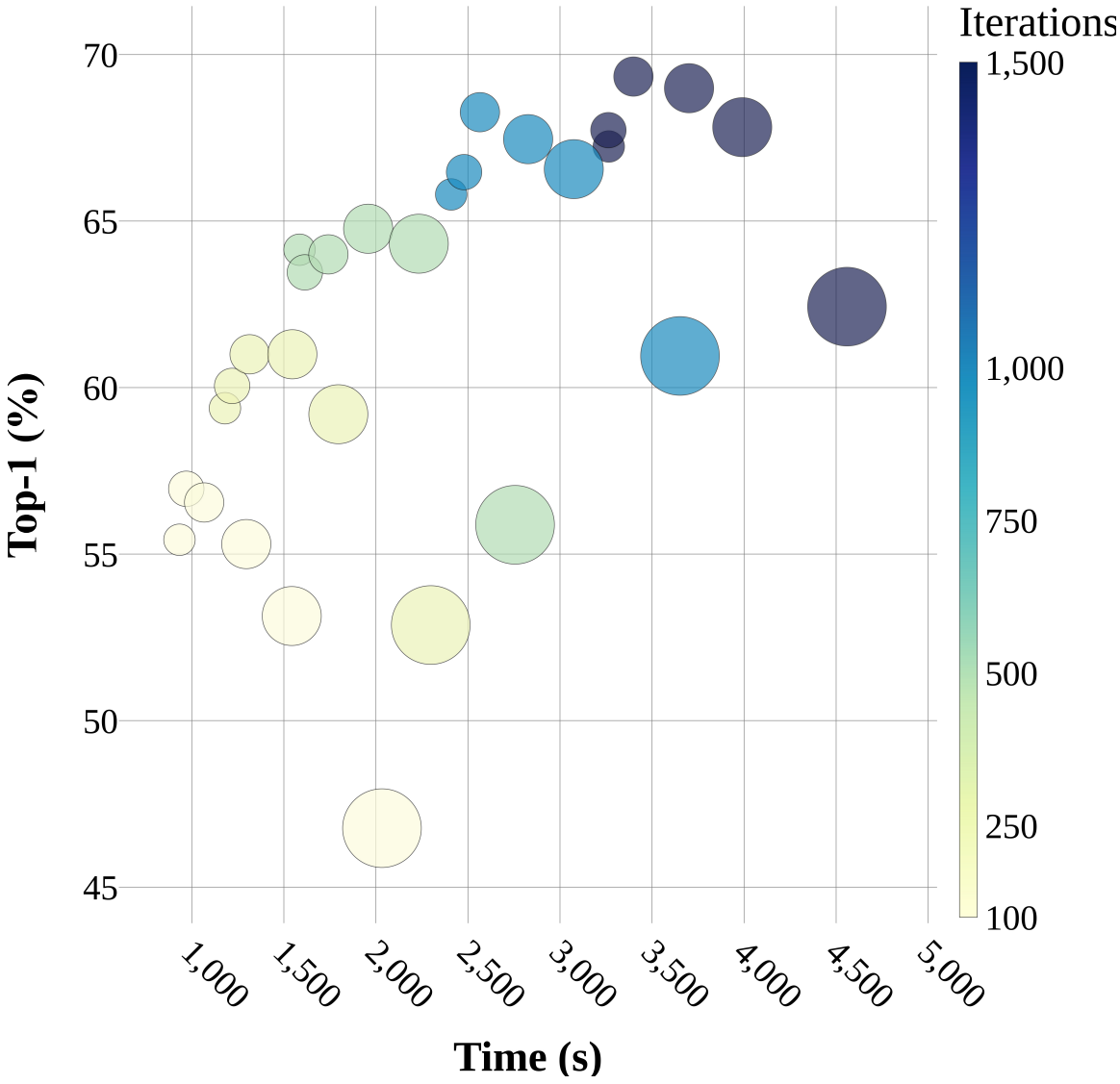

Figure 5.8: Results exploring the use of system generated data using DukeMTMC-video [1] showing a two-dimensional view when $E = 5$. Larger circles represent larger values of k.

$$\begin{aligned} K &\in [16, 18, 20, 25, 30, 40] \\ I &\in [100, 250, 500, 1000, 1500] \\ E &\in [1, 2, 3, 5] \end{aligned} \tag{5.2}$$

The results of this exploration can be seen in Fig. 5.6, Fig. 5.7, and Fig. 5.8, with more details available in the supplementary materials. Axes are identical to Fig. 5.3 and Fig. 5.5, with color and size representing iterations and k respectively. These graphs show a somewhat similar trend as in Sec. 5.5.1 with some interesting deviations. While the trend starts off with accuracy increasing as k gets larger, there is a sharp decrease in accuracy when k increases beyond a certain point. The scale of the decrease, as well as how early it occurs, lessens with both iterations and epochs. This is likely a byproduct of how many identities are present in DukeMTMC-video. While DukeMTMC only labels a total of 1404 identities, our system is able to detect far more. Increasing iterations has such a drastic effect here because it determines how many of and how often these identities are seen during an epoch. Further increasing iterations and epochs could help mitigate this, but would also increase overall training time. This, combined with the fact that more epochs and more iterations always result in higher accuracy, suggests that accuracy saturation has not been reached here, and the main limiting factor is training time. The highest accuracy achieved on this noisy data was a Top-1 of 69.34%, with $K = 20$, $I = 1500$, $E = 5$, and a total training time of just under 57 minutes. This is notably worse than both the 74.55% achieved in Sec. 5.5.1 and the 72.3% MMT achieves in the OUDA setting [214]. This demonstrates the extreme impact noisy data can have on unsupervised domain adaptation, and why the extra considerations of the R^2OUDA setting are a necessity when designing algorithms for real-world applications.

5.5.3 R^{2}MMT

Finally, we make the first attempt at addressing the R^{2}OUDA setting. An exhaustive set of experiments are conducted with R^{2}MMT, producing a fully functional, end-to-end system that meets all the requirements of the R^{2}OUDA setting. R^{2}MMT generates person crops from a stream of data, uses SDS to construct training subsets, operates on the notion of time segments, and must adhere to the strict time constraint outlined in Sec. 5.3. A successful implementation will conform to all of those standards while achieving the highest accuracy possible, ideally within range of what was seen in Sec. 5.5.1. One hour of DukeMTMC-video is used as the data stream, split into equal sized continuous segments of size τ. SDS is performed at each time segment on each camera individually, and k refers to the total number of person crops across all training subsets for the full hour. Two methods are used to determine the number of crops needed at each time segment. In the standard method, only data collected in a time segment may be used for training related to that time segment. The second method uses a form of memory, allowing the use of data from the current time segment and previous time segments still in memory. For these experiments, we assume a memory length of up to 60 minutes. Eq. (5.3) and Eq. (5.4) are used to calculate the number of person crops needed from each camera at each time segment, for the standard and memory based methods respectively.

$$k = \sum_{t=0}^{\frac{60}{\tau}-1} \sum_{i=1}^{8} P(C_i) P(C_i \cap \tau_t) \tag{5.3}$$

$$k = \sum_{t=0}^{\frac{60}{\tau}-1} \sum_{i=1}^{8} P(C_i) \sum_{\eta=0}^{t} P(C_i \cap \tau_\eta) \tag{5.4}$$

where k is the total number of person crops desired for the training subset over an hour of video stream, τ_t is a time segment of length τ minutes that begins at $\tau \times t$ minutes, C_i is the i^{th} camera, $P(C_i)$ is the percentage of total person crops received from C_I when compared to all cameras over an hour of video, and $P(C_i)P(C_i \cap \tau_t)$ is the percentage of person crops received during τ_t for C_i compared to all person crops received from C_i over an hour of video.

This ensures the number of person crops selected for a subset from each camera at each time segment is proportional to the number of person crops received. The variable ranges used in these experiments are shown below.

$$\begin{aligned} K &\in [18, 20, 25, 30, 40, 50] \\ I &\in [100, 250, 500, 750, 1000, 1500] \\ E &\in [1, 2, 3, 5] \\ \tau &\in [15, 20, 30] \\ t &\in \mathbb{Z} : \{0 \leq t \leq (\frac{60}{\tau} - 1)\} \end{aligned} \tag{5.5}$$

This creates over 2500 data points across the two methods, becoming difficult to visualize even in three dimensional space. Fig. 5.9 displays the distribution of training accuracies for each τ at each time segment. Out of the 864 configurations tested, more than half of them failed to consistently meet the time requirement of R^{2}OUDA and are not included in the statistics. Most notably, all configurations that used memory failed to consistently meet the time requirement when given a τ of 15. When memory is utilized, the time required for SDS greatly increases for successive time segments as more images accumulate. This limits how large k can be, restricting K to 20 or below when $\tau = 20$ and 30 or below when $\tau = 30$. Even without memory, the time constraint proves very limiting. Only when $\tau = 20$ is the entire range of K able to

be utilized. For a more fine grain look at all 2500+ data points in this experiment, please see the supplementary materials.

The data in general follows similar trends as seen in Sec. 5.5.1 and Sec. 5.5.2, but to more of an extreme. In addition to disqualifying several configurations off the bat, the segmented data stream and time constraint generally mean R^{2}MMT has less data to work with during any given training. Unlike in the previous experiments, the time constraint prevents the system from just throwing more data and more training at the problem. Instead, a balance must be found. We see an overall increase in top accuracies when τ increases, both in standard and memory configurations. Top accuracies also increase over time, with one notable exception. When $\tau = 15$, accuracy actually drops in the final time segment. This is due to the extremely low amount of data available in that particular time segment.

Another interesting observation can be made by looking at $\tau = 20$ both with and without memory. While the standard R^{2}MMT achieves higher overall top accuracies, the distribution is a lot more varied when compared to R^{2}MMT with memory. Many configurations actually lose accuracy, far more than when memory is present. This suggests that while memory is limiting, it may add stability to training over time. This is further demonstrated when $\tau = 30$. When memory is used the maximum accuracy is lower in the first time segment, being restricted to a lower value of K, but is higher in the second time segment due to the increased range of available data.

Fig. 5.10 shows the best configurations of R^{2}MMT, both with and without memory, for each τ. The overall highest accuracy is achieved with memory when $\tau = 30$, $K = 30$, $E = 5$, and $I = 500$, reaching an impressive 73.2% Top-1. Despite the much harsher requirements of the R^{2}OUDA setting, this is within 0.1% of the best possible accuracy using MMT in the OUDA setting [214]. However, with a τ of 30 it also has a latency of 60 minutes between collecting data and inferencing with a model trained on that data. This can be reduced to 30 minutes by changing τ to 15, but then

Table 5.2: Distribution of accuracies achieved on DukeMTMC [1] with R^{2}MMT.

τ	t	Min	Q_1	Q_2	Q_3	Max
R^{2}MMT						
15	0	35.28	40.93	43.76	47.80	52.29
	1	35.68	41.43	44.48	51.66	57.05
	2	30.92	37.75	41.97	52.74	59.92
	3	26.30	33.62	40.89	51.71	58.08
20	0	39.00	44.39	50.27	54.29	61.63
	1	33.75	42.42	55.39	61.15	68.76
	2	28.73	43.31	56.96	63.85	69.97
30	0	44.30	51.35	54.76	59.04	65.66
	1	43.22	53.91	58.71	65.04	72.08
R^{2}MMT with memory						
20	0	38.87	41.67	42.77	43.65	46.36
	1	38.42	45.20	48.03	49.87	54.26
	2	37.88	47.44	51.35	53.90	60.73
30	0	44.26	50.30	54.17	57.72	64.36
	1	47.58	58.39	62.17	67.00	73.21

accuracy drops to a disappointing 58.08%. Interestingly, with a τ of 15, accuracy actually drops in the final time segment. This is due to a limited number of persons present in the dataset during that time, leading to less data available for training and making the model less generalizable. A τ of 20 splits the difference, achieving a final Top-1 of 69.97% while reducing the inference latency to 40 minutes. This is within 4% of our best overall result, and reduces the delay by over 30%.

The strict time constraint disqualified many of the configurations in Sec. 5.5.3. However, if we ignore the time constraint for a moment we see accuracies reaching up to 76.53% when $\tau = 15$, $K = 40$, $E = 5$, and $I = 1500$ in a system with memory, putting it within 1.5% of MMT in the UDA setting [211]. With further optimization or more powerful hardware, R^{2}MMT might be able to achieve higher accuracies with decreased latency between collection and inference. This shows that there is a lot of room for improvement and growth in the R^{2}OUDA setting. Overall, it is clear that larger values of K and more training time lead to better results, but the time constraint limits both of these factors. For any practical implementation, a balance

must be found for that specific use case. The explorations in this paper can serve as a guideline for future works. The specific optimal ranges for each variable will shift with different target domains, but the overall trends and optimization techniques will be consistent.

5.6 Conclusion

This paper proposed the setting of R^{2}OUDA, to better represent the unique challenges of real-world applications. R^{2}MMT was introduced as the first attempt at a real-world, end-to-end system that can address all the demands of the R^{2}OUDA setting. An exhaustive set of experiments were conducted, using R^{2}MMT to create over 100 data subsets and train more than 3000 models, exploring the breadth of the R^{2}OUDA setting. While meeting the harsh requirements of R^{2}OUDA, including noisy data and time constraints, R^{2}MMT was able to achieve over 73% Top-1 accuracy, reaching within 0.1% of comparable SotA OUDA approaches without noisy data or a time constraint, that cannot be directly applied to real-world applications.

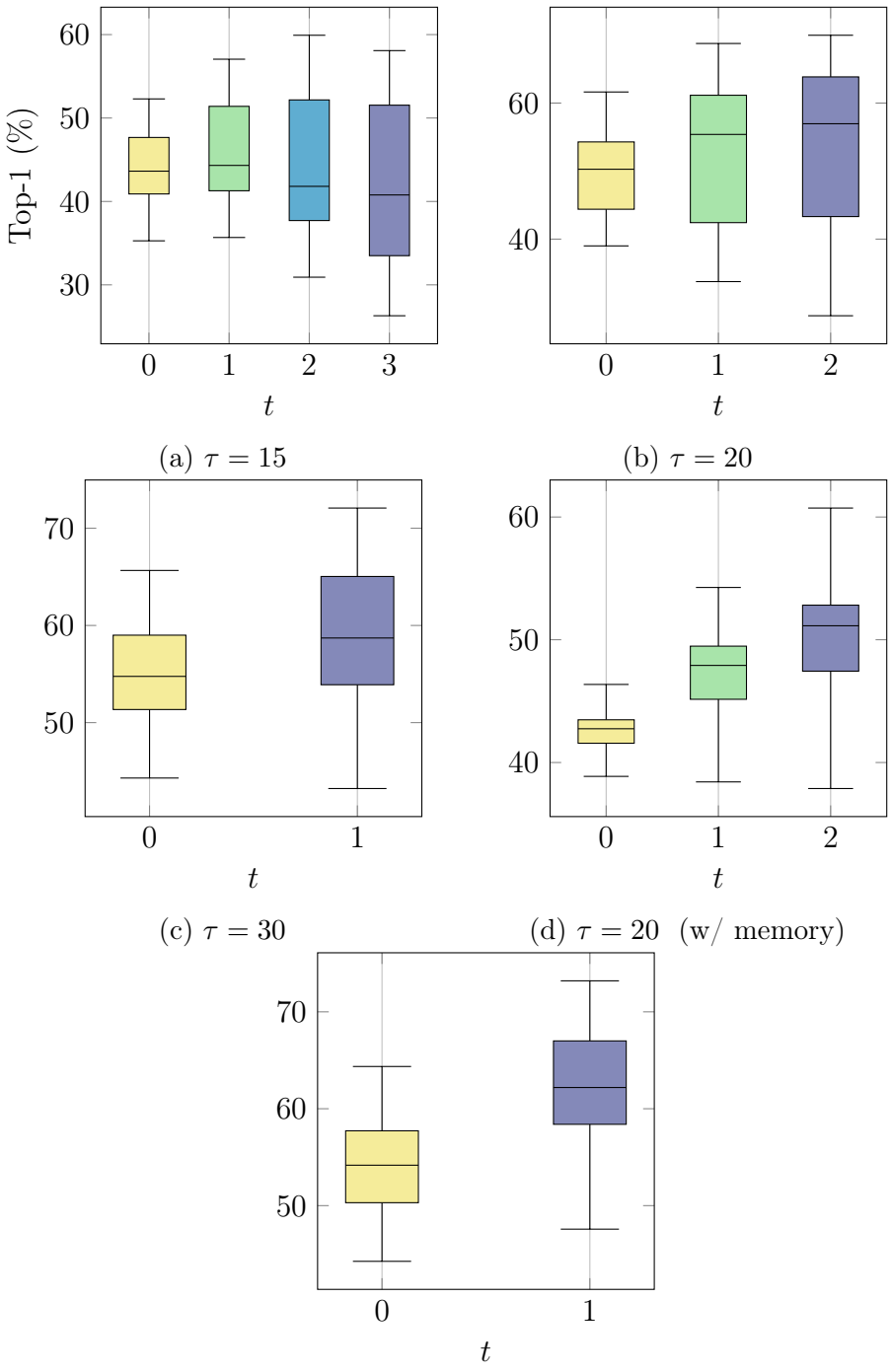

Figure 5.9: Distribution of accuracies achieved on DukeMTMC [1] with R^{2}MMT.

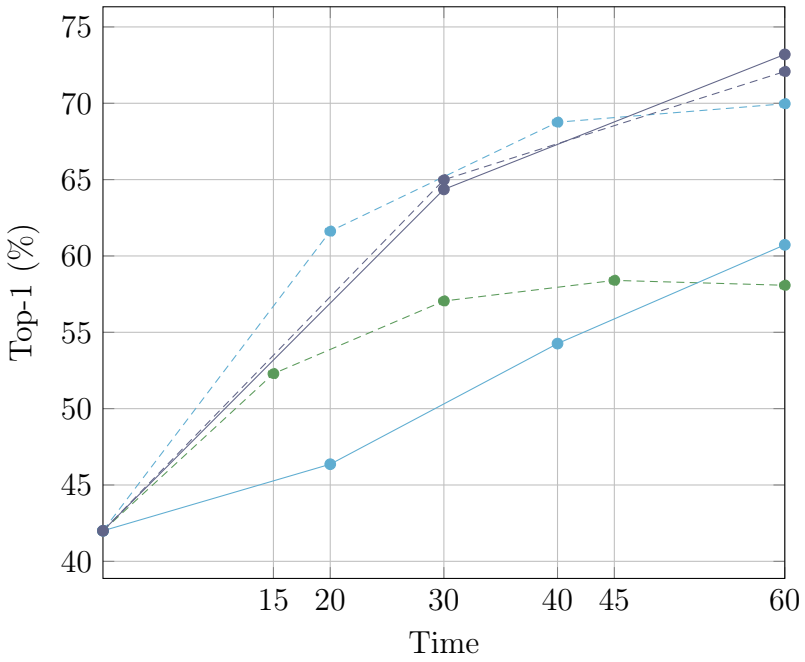

Figure 5.10: Best results for each system configuration. Dashed lines (- -) represent standard configurations. Solid lines (–) represent configurations with memory. Green, blue, and purple denote τ values of 15, 20, and 30 respectively.

CHAPTER 6: Conclusions

In this dissertation, we have explored cutting-edge artificial intelligence technology through four transformative creations: REVAMP^{2}T, Ancilia, EfficientHRNet, and R^{2}OUDA. These works encapsulate a spectrum of advancements within the fields of edge computing, intelligent video surveillance, lightweight human pose estimation, and real-world person re-identification. Each has made significant contributions to the landscape of the Artificial Intelligence of Things.

This research began with REVAMP^{2}T, an integrated end-to-end IoT system that signified a significant leap in decentralized edge cognitive intelligence for video-based situational awareness. With the introduction of the novel Accuracy•Efficiency (Æ) metric, REVAMP^{2}T set a new benchmark for real-time video analytics within IoT systems. Its exceptional performance, achieving up to a thirteen-fold improvement in Æcompared to pre-existing state-of-the-art solutions, is a testament to its efficacy in the world of edge computing. This achievement has profound implications for applications requiring swift and accurate decision-making, such as autonomous vehicles, industrial automation, and smart cities.

Second was Ancilia, a scalable intelligent video surveillance system tailored for the AIoT era. Ancilia's ability to bring cutting-edge artificial intelligence to the realm of video surveillance is a milestone in itself. It denotes significant progress, performing high-level cognitive tasks, including action recognition and anomaly detection, in real-time. Importantly, Ancilia achieves these feats while adhering to ethical and privacy considerations that are paramount in the realm of surveillance. Its potential applications span a wide spectrum, from enhancing public safety and security to enabling smarter and more responsive infrastructure management and protection.

In pursuit of efficiency and accuracy, we introduced EfficientHRNet, a family of scalable networks tailored for real-time, low-power execution of multi-person 2D human pose estimation on edge devices. By marrying the principles of EfficientNet [136] and HRNet [21], we engineered a lightweight yet powerful architecture. EfficientHRNet's ability to maintain competitive accuracy on challenging datasets while outperforming existing bottom-up pose estimation networks in terms of efficiency is unmatched. Its suitability for deployment on resource-constrained devices, achieving 23 FPS on an Nvidia Jetson NX Xavier, ushers in a new era for applications demanding real-time human pose estimation, such as sports analytics, health monitoring, and autonomous thread detection.

This research culminated in the introduction of R^{2}OUDA, a new paradigm that recognizes the unique challenges posed by real-world applications in online unsupervised domain adaptation for person re-identification. To address these challenges head-on, we conceived R^{2}MMT, a real-world, end-to-end system that navigates the demands of the exacting R^{2}OUDA setting. Through an exhaustive array of experiments, R^{2}MMT demonstrated its prowess by achieving over 73% Top-1 accuracy — a remarkable feat given the noisy data and time constraints inherent in real-world scenarios. This performance underscores the potential of R^{2}MMT and the broader R^{2}OUDA setting to revolutionize domains like healthcare, retail, public safety, transportation, video surveillance, and smart cities, where domain shifts are the norm rather than the exception.

In summary, this dissertation presents innovations with significant implications for the field of Human-Centric Computer Vision for the Artificial Intelligence of Things (AIoT). It has advanced the current state-of-the-art in edge computing, intelligent surveillance, efficient deep learning, and real-world domain adaptation. Looking forward, the groundwork laid in these fields is positioned to facilitate the emergence of intelligent, adaptive, and efficient technologies that will influence our interactions

with and utilization of the interconnected world. The chapters within this dissertation serve as critical milestones in the ongoing development of technology and its practical applications.

REFERENCES

[1] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *European Conference on Computer Vision workshop on Benchmarking Multi-Target Tracking*, 2016.

[2] J. Liu, A. Shahroudy, M. Perez, G. Wang, L.-Y. Duan, and A. C. Kot, "Ntu rgb+ d 120: A large-scale benchmark for 3d human activity understanding," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 10, pp. 2684–2701, 2019.

[3] H. Duan, Y. Zhao, K. Chen, D. Lin, and B. Dai, "Revisiting skeleton-based action recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2969–2978, 2022.

[4] Y. Chen, Z. Zhang, C. Yuan, B. Li, Y. Deng, and W. Hu, "Channel-wise topology refinement graph convolution for skeleton-based action recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 13359–13368, 2021.

[5] W. Liu, D. L. W. Luo, and S. Gao, "Future frame prediction for anomaly detection – a new baseline," in *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[6] A. Markovitz, G. Sharir, I. Friedman, L. Zelnik-Manor, and S. Avidan, "Graph embedded pose clustering for anomaly detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10539–10547, 2020.

[7] R. Morais, V. Le, T. Tran, B. Saha, M. Mansour, and S. Venkatesh, "Learning regularity in skeleton trajectories for anomaly detection in videos," in *Proceed-*

ings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 11996–12004, 2019.

[8] B. Cheng, B. Xiao, J. Wang, H. Shi, T. S. Huang, and L. Zhang, "Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation," 2019.

[9] "Jetson xavier nx developer kit," May 2020.

[10] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, *Fog Computing: A Platform for Internet of Things and Analytics*, pp. 169–186. Cham: Springer International Publishing, 2014.

[11] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, pp. 637–646, Oct 2016.

[12] M. Sapienza, E. Guardo, M. Cavallo, G. L. Torre, G. Leombruno, and O. Tomarchio, "Solving critical events through mobile edge computing: An approach for smart cities," in *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*, pp. 1–5, May 2016.

[13] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of mec in the internet of things," *IEEE Consumer Electronics Magazine*, vol. 5, pp. 84–91, Oct 2016.

[14] M. Chiang and T. Zhang, "Fog and iot: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–1, 2016.

[15] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, pp. 14–23, Oct 2009.

[16] "https://www.nytimes.com/2019/05/14/us/facial-recognition-ban-san-francisco.html."

[17] "https://www.wsj.com/articles/ai-surveillance-tools-scrutinized-by-european-regulators-11561562155."

[18] C. Alexandre, "The public safety implications of the itaewon tragedy," Dec 2022.

[19] N. Salahieh, J. Miller, and H. Yan, "As north carolinians regain power, investigators probe terrorism and threats against power substations across the us. one expert explains what needs to be done," Dec 2022.

[20] Z. Cao, T. Simon, S. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," *CoRR*, vol. abs/1611.08050, 2016.

[21] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," *CoRR*, vol. abs/1902.09212, 2019.

[22] M. Ye, J. Shen, G. Lin, T. Xiang, L. Shao, and S. C. H. Hoi, "Deep learning for person re-identification: A survey and outlook," 2020.

[23] M. Ditty, A. Karandikar, and D. Reed, "Nvidia xavier soc," Aug 2018.

[24] E. Ristani and C. Tomasi, "Features for multi-target multi-camera tracking and re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6036–6046, 2018.

[25] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *European Conference on Computer Vision workshop on Benchmarking Multi-Target Tracking*, 2016.

[26] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[27] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015.

[28] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[29] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler, and B. Schiele, "Deepcut: Joint subset partition and labeling for multi person pose estimation," *CoRR*, vol. abs/1511.06645, 2015.

[30] X. Zhang, H. Luo, X. Fan, W. Xiang, Y. Sun, Q. Xiao, W. Jiang, C. Zhang, and J. Sun, "Alignedreid: Surpassing human-level performance in person re-identification," *arXiv preprint arXiv:1711.08184*, 2017.

[31] E. Ristani and C. Tomasi, "Features for multi-target multi-camera tracking and re-identification," in *Conference on Computer Vision and Pattern Recognition*, 2018.

[32] K. Zhou, Y. Yang, A. Cavallaro, and T. Xiang, "Omni-scale feature learning for person re-identification," *arXiv preprint arXiv:1905.00953*, 2019.

[33] Y. Shen, H. Li, S. Yi, D. Chen, and X. Wang, "Person re-identification with deep similarity-guided graph neural network," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[34] S. Li, S. Bak, P. Carr, and X. Wang, "Diversity regularized spatiotemporal attention for video-based person re-identification," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 369–378, June 2018.

[35] M. Li, X. Zhu, and S. Gong, "Unsupervised tracklet person re-identification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2019.

[36] X. Zhu, X. Jing, X. You, X. Zhang, and T. Zhang, "Video-based person re-identification by simultaneously learning intra-video and inter-video distance metrics," *IEEE Transactions on Image Processing*, vol. 27, pp. 5683–5695, Nov 2018.

[37] T. Xiao, S. Li, B. Wang, L. Lin, and X. Wang, "Joint detection and identification feature learning for person search," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3376–3385, July 2017.

[38] W. Zhang, B. Ma, K. Liu, and R. Huang, "Video-based pedestrian re-identification by adaptive spatio-temporal appearance model," *IEEE Transactions on Image Processing*, vol. 26, pp. 2042–2054, April 2017.

[39] J. Dai, P. Zhang, D. Wang, H. Lu, and H. Wang, "Video person re-identification by temporal residual learning," *IEEE Transactions on Image Processing*, vol. 28, pp. 1366–1377, March 2019.

[40] Y. Zhang, Q. Zhong, L. Ma, D. Xie, and S. Pu, "Learning incremental triplet margin for person re-identification," *CoRR*, vol. abs/1812.06576, 2018.

[41] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *J. Mach. Learn. Res.*, vol. 10, pp. 207–244, June 2009.

[42] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *CoRR*, vol. abs/1703.07737, 2017.

[43] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 5, pp. 564–575, 2003.

[44] S.-K. Weng, C.-M. Kuo, and S.-K. Tu, "Video object tracking using adaptive kalman filter," *Journal of Visual Communication and Image Representation*, vol. 17, no. 6, pp. 1190–1208, 2006.

[45] X. Li, K. Wang, W. Wang, and Y. Li, "A multiple object tracking method using kalman filter," in *The 2010 IEEE international conference on information and automation*, pp. 1862–1866, IEEE, 2010.

[46] G. Ning, Z. Zhang, C. Huang, Z. He, X. Ren, and H. Wang, "Spatially supervised recurrent convolutional neural networks for visual object tracking," *CoRR*, vol. abs/1607.05781, 2016.

[47] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," *CoRR*, vol. abs/1701.01909, 2017.

[48] K. Koide, E. Menegatti, M. Carraro, M. Munaro, and J. Miura, "People tracking and re-identification by face recognition for rgb-d camera networks," in *2017 European Conference on Mobile Robots (ECMR)*, pp. 1–7, Sep. 2017.

[49] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[50] H. S. Dadi, G. K. M. Pillutla, and M. L. Makkena, "Face recognition and human tracking using gmm, hog and svm in surveillance videos," *Annals of Data Science*, vol. 5, pp. 157–179, Jun 2018.

[51] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 499–515, Springer International Publishing, 2016.

[52] "Surveillance solutions."

[53] M. Baharani, S. Mohan, and H. Tabkhi, "Real-time person re-identification at the edge: A mixed precision approach," in *Lecture Notes in Computer Science*, Springer International Publishing, 2019.

[54] P. Kulkarni, S. Mohan, S. Rogers, and H. Tabkhi, "Key-track: A lightweight scalable lstm-based pedestrian tracker for surveillance systems," in *Lecture Notes in Computer Science*, Springer International Publishing, 2019.

[55] E. A. Lee, B. Hartmann, J. Kubiatowicz, T. S. Rosing, J. Wawrzynek, D. Wessel, J. Rabaey, K. Pister, A. Sangiovanni-Vincentelli, S. A. Seshia, D. Blaauw, P. Dutta, K. Fu, C. Guestrin, B. Taskar, R. Jafari, D. Jones, V. Kumar, R. Mangharam, G. J. Pappas, R. M. Murray, and A. Rowe, "The swarm at the edge of the cloud," *IEEE Design Test*, vol. 31, pp. 8–20, June 2014.

[56] O. Vermesan, P. Friess, P. Guillemin, and S. Gusmeroli, *Internet of Things Strategic Research Agenda*. River Publishers, 2011.

[57] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, pp. 637–646, Oct 2016.

[58] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, (New York, NY, USA), pp. 13–16, ACM, 2012.

[59] J. Španhel, V. Bartl, R. Juránek, and A. Herout, "Vehicle re-identification and multi-camera tracking in challenging city-scale environment," in *Proc. CVPR Workshops*, 2019.

[60] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '14, (New York, NY, USA), pp. 68–81, ACM, 2014.

[61] Y. Lu, A. Chowdhery, and S. Kandula, "Visflow: A relational platform for efficient large-scale video analytics," tech. rep., June 2016.

[62] E. Ristani and C. Tomasi, "Tracking multiple people online and in real time," in *Asian Conference on Computer Vision*, pp. 444–459, Springer, 2014.

[63] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.

[64] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. G. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 2704–2713, 2018.

[65] P. Micikevicius, S. Narang, J. Alben, G. F. Diamos, E. Elsen, D. García, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, "Mixed precision training," *CoRR*, vol. abs/1710.03740, 2017.

[66] Z. Zheng, L. Zheng, and Y. Yang, "Unlabeled samples generated by gan improve the person re-identification baseline in vitro," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[67] W. Li, R. Zhao, T. Xiao, and X. Wang, "Deepreid: Deep filter pairing neural network for person re-identification," in *CVPR*, 2014.

[68] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable person re-identification: A benchmark," in *Computer Vision, IEEE International Conference on*, 2015.

[69] H. Moon and P. J. Phillips, "Computational and performance aspects of pca-based face-recognition algorithms," *Perception*, vol. 30, no. 3, pp. 303–321, 2001.

[70] P. Grother, R. J. Micheals, and P. J. Phillips, "Face recognition vendor test 2002 performance metrics," in *International Conference on Audio-and Video-based Biometric Person Authentication*, pp. 937–945, Springer, 2003.

[71] E. Ristani, F. Solera, R. S. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," *CoRR*, vol. abs/1609.01775, 2016.

[72] "Nvidia jetson nano system-on-module data sheet [preliminary]," May 2019. Rev. 0.7.

[73] Dusty-Nv, "Jetson agx xavier new era autonomous machines," May 2019.

[74] S. Feldstein and C. E. for International Peace, *The Global Expansion of AI Surveillance*. Carnegie Endowment for International Peace, 2019.

[75] M. R. Patrikar, Devashree R. Parate, "Anomaly detection using edge computing in video surveillance system: review," *International Journal of Multimedia Information Retrieval*, pp. 85–110, 2022.

[76] A. Danesh Pazho, G. Alinezhad Noghre, A. A. Purkayastha, J. Vempati, O. Martin, and H. Tabkhi, "A comprehensive survey of graph-based deep learn-

ing approaches for anomaly detection in complex distributed systems," *arXiv preprint arXiv:2206.04149*, 2022.

[77] X. Li and Z.-m. Cai, "Anomaly detection techniques in surveillance videos," in *2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pp. 54–59, 2016.

[78] K. Islam, "Person search: New paradigm of person re-identification: A survey and outlook of recent works," *Image and Vision Computing*, vol. 101, p. 103970, 2020.

[79] B. S. Shobha and R. Deepu, "A review on video based vehicle detection, recognition and tracking," in *2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS)*, pp. 183–186, 2018.

[80] J. Zhang, C. Xu, Z. Gao, J. J. P. C. Rodrigues, and V. H. C. de Albuquerque, "Industrial pervasive edge computing-based intelligence iot for surveillance saliency detection," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 5012–5020, 2021.

[81] G. T. Draughon, P. Sun, and J. P. Lynch, "Implementation of a computer vision framework for tracking and visualizing face mask usage in urban environments," in *2020 IEEE International Smart Cities Conference (ISC2)*, pp. 1–8, 2020.

[82] R. Xu, S. Y. Nikouei, Y. Chen, A. Polunchenko, S. Song, C. Deng, and T. R. Faughnan, "Real-time human objects tracking for smart surveillance at the edge," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, 2018.

[83] S. Y. Nikouei, Y. Chen, S. Song, R. Xu, B.-Y. Choi, and T. Faughnan, "Smart surveillance as an edge network service: From harr-cascade, svm to a lightweight

cnn," in *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, pp. 256–265, 2018.

[84] S. Y. Nikouei, Y. Chen, S. Song, B.-Y. Choi, and T. R. Faughnan, "Toward intelligent surveillance as an edge network service (isense) using lightweight detection and tracking algorithms," *IEEE Transactions on Services Computing*, vol. 14, no. 6, pp. 1624–1637, 2021.

[85] C. Neff, M. Mendieta, S. Mohan, M. Baharani, S. Rogers, and H. Tabkhi, "Revamp2t: Real-time edge video analytics for multicamera privacy-aware pedestrian tracking," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2591–2602, 2020.

[86] B. Gaikwad and A. Karmakar, "Smart surveillance system for real-time multi-person multi-camera tracking at the edge," in *Journal of Real-Time Image Processing*, vol. 18, 2021.

[87] Y. Zhao, Y. Yin, and G. Gui, "Lightweight deep learning based intelligent edge surveillance techniques," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 4, pp. 1146–1154, 2020.

[88] R. Ke, Y. Zhuang, Z. Pu, and Y. Wang, "A smart, efficient, and reliable parking surveillance system with edge artificial intelligence on iot devices," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 4962–4974, 2021.

[89] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[90] J. Liang, L. Jiang, J. C. Niebles, A. G. Hauptmann, and L. Fei-Fei, "Peeking into the future: Predicting future person activities and locations in videos,"

in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[91] V. Nguyen, D. Phung, D.-S. Pham, and S. Venkatesh, "Bayesian nonparametric approaches to abnormality detection in video surveillance," *Annals of Data Science*, vol. 2, pp. 21–41, 2015.

[92] R. Nawaratne, D. Alahakoon, D. De Silva, and X. Yu, "Spatiotemporal anomaly detection using deep learning for real-time video surveillance," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 393–402, 2020.

[93] R. Arroyo, J. J. Yebes, L. M. Bergasa, I. G. Daza, and J. Almazán, "Expert video-surveillance system for real-time detection of suspicious behaviors in shopping malls," *Expert Systems with Applications*, vol. 42, no. 21, pp. 7991–8005, 2015.

[94] J. T. Zhou, J. Du, H. Zhu, X. Peng, Y. Liu, and R. S. M. Goh, "Anomalynet: An anomaly detection network for video surveillance," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 10, pp. 2537–2550, 2019.

[95] J. Pierce, R. Y. Wong, and N. Merrill, "Sensor illumination: Exploring design qualities and ethical implications of smart cameras and image/video analytics," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, (New York, NY, USA), p. 1–19, Association for Computing Machinery, 2020.

[96] H. Nissenbaum, "Privacy as contextual integrity," *Wash. L. Rev.*, vol. 79, p. 119, 2004.

[97] Y. E. Appenzeller, P. S. Appelbaum, and M. Trachsel, "Ethical and practical issues in video surveillance of psychiatric units," *Psychiatric Services*, vol. 71, no. 5, pp. 480–486, 2020.

[98] F. Tariq, N. Kanwal, M. S. Ansari, A. Afzaal, M. N. Asghar, and M. J. Anjum, "Towards a privacy preserving surveillance approach for smart cities," in *3rd Smart Cities Symposium (SCS 2020)*, vol. 2020, pp. 450–455, 2020.

[99] W. Hartzog, *PrivacyÕs Blueprint: The Battle to Control the Design of New Technologies*. Harvard University Press, 2018.

[100] J. Daubert, A. Wiesmaier, and P. Kikiras, "A view on privacy & trust in iot," in *2015 IEEE International Conference on Communication Workshop (ICCW)*, pp. 2665–2670, IEEE, 2015.

[101] T. Speicher, M. Ali, G. Venkatadri, F. N. Ribeiro, G. Arvanitakis, F. Benevenuto, K. P. Gummadi, P. Loiseau, and A. Mislove, "Potential for discrimination in online targeted advertising," in *Conference on Fairness, Accountability and Transparency*, pp. 5–19, PMLR, 2018.

[102] I. D. Raji, T. Gebru, M. Mitchell, J. Buolamwini, J. Lee, and E. Denton, "Saving face: Investigating the ethical concerns of facial recognition auditing," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 145–151, 2020.

[103] N. Martinez-Martin, "What are important ethical implications of using facial recognition technology in health care?," *AMA journal of ethics*, vol. 21, no. 2, p. E180, 2019.

[104] L. Introna and H. Nissenbaum, "Facial recognition technology a survey of policy and implementation issues," 2010.

[105] E. Selinger and B. Leong, "The ethics of facial recognition technology," *Forthcoming in The Oxford Handbook of Digital Ethics ed. Carissa Véliz*, 2021.

[106] F. Angelini, Z. Fu, Y. Long, L. Shao, and S. M. Naqvi, "2d pose-based real-time human action recognition with occlusion-handling," *IEEE Transactions on Multimedia*, vol. 22, no. 6, pp. 1433–1446, 2019.

[107] J.-M. Guo, C.-C. Lin, M.-F. Wu, C.-H. Chang, and H. Lee, "Complexity reduced face detection using probability-based face mask prefiltering and pixel-based hierarchical-feature adaboosting," *IEEE Signal Processing Letters*, vol. 18, no. 8, pp. 447–450, 2011.

[108] T. Özyer, D. S. Ak, and R. Alhajj, "Human action recognition approaches with video datasets—a survey," *Knowledge-Based Systems*, vol. 222, p. 106995, 2021.

[109] R. Liao, S. Yu, W. An, and Y. Huang, "A model-based gait recognition method with body pose and human prior knowledge," *Pattern Recognition*, vol. 98, p. 107069, 2020.

[110] X. Jin, T. He, K. Zheng, Z. Yin, X. Shen, Z. Huang, R. Feng, J. Huang, Z. Chen, and X.-S. Hua, "Cloth-changing person re-identification from a single image with gait prediction and regularization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14278–14287, 2022.

[111] D. Leslie, "Understanding bias in facial recognition technologies," tech. rep., 2020.

[112] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, TaoXie, K. Michael, J. Fang, imyhxy, Lorna, C. Wong, Z. Yifu, A. V, D. Montes, Z. Wang, C. Fati, J. Nadar, Laughing, UnglvKitDe, tkianai, yxNONG, P. Skalski, A. Hogan, M. Strobel, M. Jain, L. Mammana, and xylieong, "ultralytics/yolov5: v6.2 - YOLOv5 Classification Models, Apple M1, Reproducibility, ClearML and Deci.ai integrations," Aug. 2022.

[113] A. Warsi, M. Abdullah, M. N. Husen, M. Yahya, S. Khan, and N. Jawaid, "Gun detection system using yolov3," in *2019 IEEE International Conference on Smart Instrumentation, Measurement and Application (ICSIMA)*, pp. 1–4, IEEE, 2019.

[114] S. Narejo, B. Pandey, D. Esenarro Vargas, C. Rodriguez, and M. R. Anjum, "Weapon detection using yolo v3 for smart surveillance system," *Mathematical Problems in Engineering*, vol. 2021, pp. 1–9, 2021.

[115] R. Garg and S. Singh, "Intelligent video surveillance based on yolo: A comparative study," in *2021 International Conference on Advances in Computing, Communication, and Control (ICAC3)*, pp. 1–6, IEEE, 2021.

[116] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "Bytetrack: Multi-object tracking by associating every detection box," 2022.

[117] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao, "Deep high-resolution representation learning for visual recognition," *TPAMI*, 2019.

[118] W. An, R. Liao, S. Yu, Y. Huang, and P. C. Yuen, "Improving gait recognition with 3d pose estimation," in *Biometric Recognition: 13th Chinese Conference, CCBR 2018, Urumqi, China, August 11-12, 2018, Proceedings 13*, pp. 137–147, Springer, 2018.

[119] M. M. Hasan and H. A. Mustafa, "Multi-level feature fusion for robust pose-based gait recognition using rnn," *Int. J. Comput. Sci. Inf. Secur.(IJCSIS)*, vol. 18, no. 1, 2020.

[120] K. Zhou, Y. Yang, A. Cavallaro, and T. Xiang, "Omni-scale feature learning

for person re-identification," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3701–3711, 2019.

[121] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308, 2017.

[122] H. Fan, B. Xiong, K. Mangalam, Y. Li, Z. Yan, J. Malik, and C. Feichtenhofer, "Multiscale vision transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6824–6835, 2021.

[123] J. Liang, H. Zhu, E. Zhang, and J. Zhang, "Stargazer: A transformer-based driver action detection system for intelligent transportation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3160–3167, 2022.

[124] W. Liu, G. Kang, P.-Y. Huang, X. Chang, Y. Qian, J. Liang, L. Gui, J. Wen, and P. Chen, "Argus: Efficient activity detection system for extended video analysis," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision Workshops*, pp. 126–133, 2020.

[125] K. Corona, K. Osterdahl, R. Collins, and A. Hoogs, "Meva: A large-scale multiview, multimodal video dataset for activity detection," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1060–1068, 2021.

[126] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 740–755, Springer International Publishing, 2014.

[127] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé, "Mot20: A benchmark for multi object tracking in crowded scenes," 2020.

[128] W. Wang, J. Dai, Z. Chen, Z. Huang, Z. Li, X. Zhu, X. Hu, T. Lu, L. Lu, H. Li, *et al.*, "Internimage: Exploring large-scale vision foundation models with deformable convolutions," *arXiv preprint arXiv:2211.05778*, 2022.

[129] L. Zheng, M. Tang, Y. Chen, G. Zhu, J. Wang, and H. Lu, "Improving multiple object tracking with single object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2453–2462, June 2021.

[130] Y. Xu, J. Zhang, Q. Zhang, and D. Tao, "ViTPose: Simple vision transformer baselines for human pose estimation," in *Advances in Neural Information Processing Systems* (A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, eds.), 2022.

[131] M. Wieczorek, B. Rychalska, and J. Dabrowski, "On the unreasonable effectiveness of centroids in image retrieval," in *Neural Information Processing: 28th International Conference, ICONIP 2021, Sanur, Bali, Indonesia, December 8–12, 2021, Proceedings, Part IV*, (Berlin, Heidelberg), p. 212–223, Springer-Verlag, 2021.

[132] H. Duan, J. Wang, K. Chen, and D. Lin, "Pyskl: Towards good practices for skeleton action recognition," 2022.

[133] L. Wang, D. Q. Huynh, and P. Koniusz, "A comparative review of recent kinect-based action recognition algorithms," *IEEE Transactions on Image Processing*, vol. 29, pp. 15–28, 2020.

[134] K. Chen, P. Gabriel, A. Alasfour, C. Gong, W. K. Doyle, O. Devinsky, D. Friedman, P. Dugan, L. Melloni, T. Thesen, D. Gonda, S. Sattar, S. Wang, and

V. Gilja, "Patient-specific pose estimation in clinical environments," *IEEE Journal of Translational Engineering in Health and Medicine*, vol. 6, pp. 1–11, 2018.

[135] Z. Fang and A. M. López, "Intention recognition of pedestrians and cyclists by 2d pose estimation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4773–4783, 2020.

[136] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *CoRR*, vol. abs/1905.11946, 2019.

[137] G. Papandreou, T. Zhu, L. Chen, S. Gidaris, J. Tompson, and K. Murphy, "Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model," *CoRR*, vol. abs/1803.08225, 2018.

[138] D. Osokin, "Real-time 2d multi-person pose estimation on CPU: lightweight openpose," *CoRR*, vol. abs/1811.12004, 2018.

[139] Y. Yang and D. Ramanan, "Articulated pose estimation with flexible mixtures-of-parts," in *CVPR 2011*, pp. 1385–1392, 2011.

[140] M. Dantone, J. Gall, C. Leistner, and L. Van Gool, "Human pose estimation using body parts dependent joint regressors," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3041–3048, 2013.

[141] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," *CoRR*, vol. abs/1312.4659, 2013.

[142] A. Jain, J. Tompson, M. Andriluka, G. Taylor, and C. Bregler, "Learning human pose estimation features with convolutional networks," 12 2013.

[143] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," *CoRR*, vol. abs/1603.06937, 2016.

[144] A. Bulat and G. Tzimiropoulos, "Human pose estimation via convolutional part heatmap regression," *CoRR*, vol. abs/1609.01743, 2016.

[145] S. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," *CoRR*, vol. abs/1602.00134, 2016.

[146] U. Iqbal and J. Gall, "Multi-person pose estimation with local joint-to-person associations," *CoRR*, vol. abs/1608.08526, 2016.

[147] G. Papandreou, T. Zhu, N. Kanazawa, A. Toshev, J. Tompson, C. Bregler, and K. P. Murphy, "Towards accurate multi-person pose estimation in the wild," *CoRR*, vol. abs/1701.01779, 2017.

[148] S. Huang, M. Gong, and D. Tao, "A coarse-fine network for keypoint localization," pp. 3047–3056, 10 2017.

[149] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu, "RMPE: Regional multi-person pose estimation," in *ICCV*, 2017.

[150] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun, "Cascaded pyramid network for multi-person pose estimation," *CoRR*, vol. abs/1711.07319, 2017.

[151] B. Cheng, Y. Wei, H. Shi, R. S. Feris, J. Xiong, and T. S. Huang, "Decoupled classification refinement: Hard false positive suppression for object detection," *CoRR*, vol. abs/1810.04002, 2018.

[152] B. Cheng, Y. Wei, H. Shi, R. S. Feris, J. Xiong, and T. S. Huang, "Revisiting RCNN: on awakening the classification power of faster RCNN," *CoRR*, vol. abs/1803.06799, 2018.

[153] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," *CoRR*, vol. abs/1612.03144, 2016.

[154] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[155] Z. Cao, G. Hidalgo, T. Simon, S. Wei, and Y. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *CoRR*, vol. abs/1812.08008, 2018.

[156] S. Kreiss, L. Bertoni, and A. Alahi, "Pifpaf: Composite fields for human pose estimation," *CoRR*, vol. abs/1903.06593, 2019.

[157] A. Newell and J. Deng, "Associative embedding: End-to-end learning for joint detection and grouping," *CoRR*, vol. abs/1611.05424, 2016.

[158] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "Deepercut: A deeper, stronger, and faster multi-person pose estimation model," *CoRR*, vol. abs/1605.03170, 2016.

[159] M. Kocabas, S. Karagoz, and E. Akbas, "Multiposenet: Fast multi-person pose estimation using pose residual network," *CoRR*, vol. abs/1807.04067, 2018.

[160] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[161] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *CoRR*, vol. abs/1606.00915, 2016.

[162] L. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, "Attention to scale: Scale-aware semantic image segmentation," *CoRR*, vol. abs/1511.03339, 2015.

[163] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," *CoRR*, vol. abs/1612.03144, 2016.

[164] W. Yang, S. Li, W. Ouyang, H. Li, and X. Wang, "Learning feature pyramids for human pose estimation," *CoRR*, vol. abs/1708.01101, 2017.

[165] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun, "Cascaded pyramid network for multi-person pose estimation," *CoRR*, vol. abs/1711.07319, 2017.

[166] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[167] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015.

[168] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," *CoRR*, vol. abs/1505.04366, 2015.

[169] Z. Zhang, X. Zhang, C. Peng, D. Cheng, and J. Sun, "Exfuse: Enhancing feature fusion for semantic segmentation," *CoRR*, vol. abs/1804.03821, 2018.

[170] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *CoRR*, vol. abs/1802.02611, 2018.

[171] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, "Unified perceptual parsing for scene understanding," *CoRR*, vol. abs/1807.10221, 2018.

[172] A. Bulat and G. Tzimiropoulos, "Binarized convolutional landmark localizers for human pose estimation and face alignment with limited resources," *CoRR*, vol. abs/1703.00862, 2017.

[173] L. Ke, M. Chang, H. Qi, and S. Lyu, "Multi-scale structure-aware network for human pose estimation," *CoRR*, vol. abs/1803.09894, 2018.

[174] S. Saxena and J. Verbeek, "Convolutional neural fabrics," *CoRR*, vol. abs/1606.02492, 2016.

[175] Y. Zhou, X. Hu, and B. Zhang, "Interlinked convolutional neural networks for face parsing," *CoRR*, vol. abs/1806.02479, 2018.

[176] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger, "Multi-scale dense convolutional networks for efficient prediction," *CoRR*, vol. abs/1703.09844, 2017.

[177] K. Sun, Y. Zhao, B. Jiang, T. Cheng, B. Xiao, D. Liu, Y. Mu, X. Wang, W. Liu, and J. Wang, "High-resolution representations for labeling pixels and regions," *CoRR*, vol. abs/1904.04514, 2019.

[178] X. Yan, Y. Huang, S. Chen, Z. Nan, J. Xin, and N. Zheng, "Dsp-net: Dense-to-sparse proposal generation approach for 3d object detection on point cloud," in *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2021.

[179] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[180] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[181] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," 2020.

[182] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," 2019.

[183] L. Yang, Y. Qin, and X. Zhang, "Lightweight densely connected residual network for human pose estimation," *Journal of Real-time Image Processing*, pp. 1–13, 2020.

[184] "Openvino toolkit.."

[185] John, "trt_pose."

[186] A. Saharan, "Creating a human pose estimation application with nvidia deepstream," Nov 2020.

[187] M. Ditty, A. Karandikar, and D. Reed, "Nvidia xavier soc," Aug 2018.

[188] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[189] A. Krizhevsky, "Learning multiple layers of features from tiny images," tech. rep., 2009.

[190] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, (Red Hook, NY, USA), p. 1097–1105, Curran Associates Inc., 2012.

[191] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016.

[192] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Y. W. Teh and M. Titterington,

eds.), vol. 9 of *Proceedings of Machine Learning Research*, (Chia Laguna Resort, Sardinia, Italy), pp. 249–256, PMLR, 13–15 May 2010.

[193] R. Ge, S. M. Kakade, R. Kidambi, and P. Netrapalli, "The step decay schedule: A near optimal, geometrically decaying learning rate procedure," *CoRR*, vol. abs/1904.12838, 2019.

[194] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[195] L. Zheng, H. Zhang, S. Sun, M. Chandraker, Y. Yang, and Q. Tian, "Person re-identification in the wild," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3346–3355, 2017.

[196] A. Hermans*, L. Beyer*, and B. Leibe, "In Defense of the Triplet Loss for Person Re-Identification," *arXiv preprint arXiv:1703.07737*, 2017.

[197] Y. Wang, Z. Chen, F. Wu, and G. Wang, "Person re-identification with cascaded pairwise convolutions," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1470–1478, 2018.

[198] Z. Zhong, L. Zheng, D. Cao, and S. Li, "Re-ranking person re-identification with k-reciprocal encoding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[199] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable person re-identification: A benchmark," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1116–1124, 2015.

[200] W. Li, R. Zhao, T. Xiao, and X. Wang, "Deepreid: Deep filter pairing neural network for person re-identification," in *CVPR*, 2014.

[201] L. Wei, S. Zhang, W. Gao, and Q. Tian, "Person transfer gan to bridge domain gap for person re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[202] L. Zheng, Z. Bie, Y. Sun, J. Wang, C. Su, S. Wang, and Q. Tian, "Mars: A video benchmark for large-scale person re-identification," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 868–884, Springer International Publishing, 2016.

[203] X. Ni and E. Rahtu, "Flipreid: Closing the gap between training and inference in person re-identification," in *2021 9th European Workshop on Visual Information Processing (EUVIP)*, pp. 1–6, 2021.

[204] M. Wieczorek, B. Rychalska, and J. Dabrowski, "On the unreasonable effectiveness of centroids in image retrieval," *ArXiv*, vol. abs/2104.13643, 2021.

[205] Z. Zhu, X. Jiang, F. Zheng, X. Guo, F. Huang, W. Zheng, and X. Sun, "Viewpoint-aware loss with angular regularization for person re-identification," 2019.

[206] G. Wang, J. Lai, P. Huang, and X. Xie, "Spatial-temporal person re-identification," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 8933–8940, Jul. 2019.

[207] Y. Chen, X. Zhu, and S. Gong, "Instance-guided context rendering for cross-domain person re-identification," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 232–242, 2019.

[208] J. Liu, Z.-J. Zha, D. Chen, R. Hong, and M. Wang, "Adaptive transfer network for cross-domain person re-identification," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7195–7204, 2019.

[209] H. Feng, M. Chen, J. Hu, D. Shen, H. Liu, and D. Cai, "Complementary pseudo labels for unsupervised domain adaptation on person re-identification," *IEEE Transactions on Image Processing*, vol. 30, pp. 2898–2907, 2021.

[210] L. Song, C. Wang, L. Zhang, B. Du, Q. Zhang, C. Huang, and X. Wang, "Unsupervised domain adaptive re-identification: Theory and practice," *Pattern Recogn.*, vol. 102, jun 2020.

[211] Y. Ge, D. Chen, and H. Li, "Mutual mean-teaching: Pseudo label refinery for unsupervised domain adaptation on person re-identification," in *International Conference on Learning Representations*, 2020.

[212] Y. Ge, F. Zhu, D. Chen, R. Zhao, and H. Li, "Self-paced contrastive learning with hybrid memory for domain adaptive object re-id," in *Advances in Neural Information Processing Systems*, 2020.

[213] K. Zeng, "Hierarchical clustering with hard-batch triplet loss for person re-identification," 2019.

[214] H. Rami, M. Ospici, and S. Lathuilière, "Online unsupervised domain adaptation for person re-identification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 3830–3839, June 2022.

[215] E. Fini, S. Lathuilière, E. Sangineto, M. Nabi, and E. Ricci, "Online continual learning under extreme memory constraints," in *Computer Vision – ECCV 2020* (A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, eds.), (Cham), pp. 720–735, Springer International Publishing, 2020.

[216] Y. Lin, X. Dong, L. Zheng, Y. Yan, and Y. Yang, "A bottom-up clustering approach to unsupervised person re-identification," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 8738–8745, Jul. 2019.

[217] Q. Leng, M. Ye, and Q. Tian, "A survey of open-world person re-identification," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 4, pp. 1092–1108, 2020.

[218] L. Zheng, Y. Yang, and A. Hauptmann, "Person re-identification: Past, present and future," *ArXiv*, vol. abs/1610.02984, 2016.

[219] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems* (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds.), vol. 27, Curran Associates, Inc., 2014.

[220] P. Isola, J.-Y. Zhu, T. Zhou, and A. Efros, "Image-to-image translation with conditional adversarial networks," in *2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5967–5976, 07 2017.

[221] W. Deng, L. Zheng, Q. Ye, G. Kang, Y. Yang, and J. Jiao, "Image-image domain adaptation with preserved self-similarity and domain-dissimilarity for person re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[222] Y. Ge, F. Zhu, D. Chen, R. Zhao, X. Wang, and H. Li, "Structured domain adaptation with online relation regularization for unsupervised person re-id," 2020.

[223] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2242–2251, 2017.

[224] Y. Huang, Q. Wu, J. Xu, and Y. Zhong, "Sbsgan: Suppression of inter-domain background shift for person re-identification," in *2019 IEEE/CVF International*

Conference on Computer Vision (ICCV), (Los Alamitos, CA, USA), pp. 9526–9535, IEEE Computer Society, nov 2019.

[225] H. Fan, L. Zheng, C. Yan, and Y. Yang, "Unsupervised person re-identification: Clustering and fine-tuning," vol. 14, oct 2018.

[226] M. Ye, J. Li, A. J. Ma, L. Zheng, and P. C. Yuen, "Dynamic graph co-matching for unsupervised video-based person re-identification," *IEEE Transactions on Image Processing*, vol. 28, no. 6, pp. 2976–2990, 2019.

[227] Y. Fu, Y. Wei, G. Wang, Y. Zhou, H. Shi, and T. S. Huang, "Self-similarity grouping: A simple unsupervised cross domain adaptation approach for person re-identification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[228] Y. Ge, F. Zhu, D. Chen, R. Zhao, and H. Li, "Self-paced contrastive learning with hybrid memory for domain adaptive object re-id," in *Advances in Neural Information Processing Systems*, 2020.

[229] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, (Red Hook, NY, USA), p. 1195–1204, Curran Associates Inc., 2017.

[230] G. Delorme, Y. Xu, S. Lathuiliere, R. Horaud, and X. Alameda-Pineda, "Canu-reid: A conditional adversarial network for unsupervised person re-identification," in *2020 25th International Conference on Pattern Recognition (ICPR)*, (Los Alamitos, CA, USA), pp. 4428–4435, IEEE Computer Society, jan 2021.

[231] N. Ewen and N. Khan, "Online unsupervised learning for domain shift in covid-19 ct scan datasets," in *2021 IEEE International Conference on Autonomous Systems (ICAS)*, pp. 1–5, 2021.

[232] Y. Ye, T. Pan, Q. Meng, J. Li, and H. T. Shen, "Online unsupervised domain adaptation via reducing inter- and intra-domain discrepancies," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2022.

[233] W. He, Y. Ye, Y. Li, T. Pan, and L. Lu, "Online cross-subject emotion recognition from ecg via unsupervised domain adaptation," in *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pp. 1001–1005, 2021.

[234] J. Moon, D. Das, and C. S. George Lee, "A multistage framework with mean subspace computation and recursive feedback for online unsupervised domain adaptation," *IEEE Transactions on Image Processing*, vol. 31, pp. 4622–4636, 2022.

[235] J. H. Moon, D. Das, and C. G. Lee, "Multi-step online unsupervised domain adaptation," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 41172–41576, 2020.

[236] Y. Kuznietsov, M. Proesmans, and L. V. Gool, "Towards unsupervised online domain adaptation for semantic segmentation," in *2022 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, pp. 261–271, 2022.

[237] J.-A. Termöhlen, M. Klingner, L. J. Brettin, N. M. Schmidt, and T. Fingscheidt, "Continual unsupervised domain adaptation for semantic segmentation by online frequency domain style transfer," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pp. 2881–2888, 2021.

[238] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 3645–3649, 2017.

[239] R. Rana and D. Garg, "Heuristic approaches for k-center problem," in *2009 IEEE International Advance Computing Conference*, pp. 332–335, 2009.

[240] N. Jali, N. Karamchandani, and S. Moharir, "Greedy kk-center from noisy distance samples," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 8, pp. 330–343, 2022.

[241] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, (USA), p. 1027–1035, Society for Industrial and Applied Mathematics, 2007.

[242] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *kdd*, vol. 96, pp. 226–231, 1996.

Appendix A: Extended Results for Subset Distribution Selection

As mentioned, we conducted over 3000 experiments when exploring R^{2}OUDA and testing R$_2$MMT. To present numerical results for each individual data point is impossible within the confines of a conference paper, which is why we instead used 3D representations to show trends in the data we collected. For transparency's sake, we use this appendix to provide detailed numerical results for each data point generated in our experiments. The amount of data here is massive, and we recommend only using it in combination with the figures and analysis provided in the main paper to better understand trends or to find values at individual data points. If these quantitative results are read in isolation, we fear the sheer amount of data itself will be obfuscating, which is counter to the point of providing it in the first place.

The following table contains the complete list of experiments conducted for Sec. 5.5.1. As can be seen here, we explored more values of K than we reported in Sec. 5.5.1. These additional data points helped us see the overall trend when we started our initial exploration and provided the values we went with for the other two experiments. However, we decided to leave them out for clarity, as Fig. 5.3, Fig. 5.4, and Fig. 5.5 were already quite crowded and the additional data points made the figures more difficult to understand. Even with the additional data points, the trends remain the same.

Table 1: Extended results for **Subset Distribution Selection**. Time is in format h:mm:ss.

					Time		
Epochs	**K**	**Iterations**	**mAP**	**Top-1**	**Training**	**SDS**	**Total**
1	2	100	13.90%	26.53%	0:01:43	0:03:59	0:05:42
1	2	250	12.04%	24.46%	0:02:30	0:03:59	0:06:29
1	2	500	11.47%	23.61%	0:03:49	0:03:59	0:07:48

Table 1: Extended results for **Subset Distribution Selection**. Time is in format h:mm:ss.

					Time		
Epochs	**K**	**Iterations**	**mAP**	**Top-1**	**Training**	**SDS**	**Total**
1	2	750	11.20%	22.67%	0:05:06	0:03:59	0:09:05
1	2	1000	10.92%	21.54%	0:06:24	0:03:59	0:10:23
1	2	1500	10.28%	20.11%	0:09:05	0:03:59	0:13:04
1	2	2000	9.62%	19.75%	0:11:35	0:03:59	0:15:34
1	4	100	24.77%	40.04%	0:01:50	0:08:03	0:09:53
1	4	250	26.02%	41.16%	0:02:37	0:08:03	0:10:40
1	4	500	27.07%	42.46%	0:03:54	0:08:03	0:11:57
1	4	750	27.22%	41.79%	0:05:12	0:08:03	0:13:15
1	4	1000	27.24%	42.28%	0:06:33	0:08:03	0:14:36
1	4	1500	27.11%	41.29%	0:09:07	0:08:03	0:17:10
1	4	2000	26.94%	41.34%	0:11:37	0:08:03	0:19:40
1	6	100	28.90%	44.75%	0:01:56	0:12:00	0:13:56
1	6	250	31.32%	47.35%	0:02:43	0:12:00	0:14:43
1	6	500	32.90%	48.56%	0:04:02	0:12:00	0:16:02
1	6	750	33.28%	48.16%	0:05:28	0:12:00	0:17:28
1	6	1000	34.47%	50.22%	0:06:39	0:12:00	0:18:39
1	6	1500	33.96%	49.10%	0:09:27	0:12:00	0:21:27
1	6	2000	34.18%	49.10%	0:11:59	0:12:00	0:23:59
1	8	100	30.03%	46.54%	0:02:03	0:15:38	0:17:41
1	8	250	32.75%	48.61%	0:02:51	0:15:38	0:18:29
1	8	500	35.63%	51.97%	0:04:10	0:15:38	0:19:48
1	8	750	35.89%	51.08%	0:05:29	0:15:38	0:21:07
1	8	1000	36.31%	52.11%	0:06:50	0:15:38	0:22:28

Table 1: Extended results for **Subset Distribution Selection**. Time is in format h:mm:ss.

					Time		
Epochs	**K**	**Iterations**	**mAP**	**Top-1**	**Training**	**SDS**	**Total**
1	8	1500	37.82%	54.40%	0:09:26	0:15:38	0:25:04
1	8	2000	36.79%	52.83%	0:12:03	0:15:38	0:27:41
1	10	100	32.06%	47.94%	0:02:10	0:21:14	0:23:24
1	10	250	34.10%	49.69%	0:03:04	0:21:14	0:24:18
1	10	500	37.30%	52.87%	0:04:16	0:21:14	0:25:30
1	10	750	37.22%	53.14%	0:05:34	0:21:14	0:26:48
1	10	1000	38.38%	53.82%	0:06:55	0:21:14	0:28:09
1	10	1500	39.39%	54.58%	0:09:31	0:21:14	0:30:45
1	10	2000	39.83%	55.07%	0:12:26	0:21:14	0:33:40
1	12	100	33.44%	48.83%	0:02:18	0:23:38	0:25:56
1	12	250	36.45%	52.20%	0:03:08	0:23:38	0:26:46
1	12	500	39.24%	54.85%	0:04:25	0:23:38	0:28:03
1	12	750	40.20%	56.06%	0:05:45	0:23:38	0:29:23
1	12	1000	40.24%	56.46%	0:07:04	0:23:38	0:30:42
1	12	1500	42.39%	58.30%	0:09:46	0:23:38	0:33:24
1	12	2000	42.82%	58.53%	0:12:40	0:23:38	0:36:18
1	14	100	34.32%	50.00%	0:02:26	0:27:24	0:29:50
1	14	250	37.84%	54.08%	0:03:12	0:27:24	0:30:36
1	14	500	40.03%	55.70%	0:04:33	0:27:24	0:31:57
1	14	750	40.86%	56.73%	0:05:53	0:27:24	0:33:17
1	14	1000	41.83%	57.63%	0:07:14	0:27:24	0:34:38
1	14	1500	42.90%	57.99%	0:09:57	0:27:24	0:37:21
1	14	2000	43.65%	58.21%	0:12:35	0:27:24	0:39:59

Table 1: Extended results for **Subset Distribution Selection**. Time is in format h:mm:ss.

					Time		
Epochs	**K**	**Iterations**	**mAP**	**Top-1**	**Training**	**SDS**	**Total**
1	16	100	34.31%	49.96%	0:02:32	0:31:23	0:33:55
1	16	250	38.47%	54.62%	0:03:21	0:31:23	0:34:44
1	16	500	40.98%	56.55%	0:04:42	0:31:23	0:36:05
1	16	750	41.66%	57.41%	0:06:06	0:31:23	0:37:29
1	16	1000	42.63%	57.85%	0:07:27	0:31:23	0:38:50
1	16	1500	43.58%	58.62%	0:10:07	0:31:23	0:41:30
1	16	2000	44.16%	58.48%	0:13:01	0:31:23	0:44:24
1	18	100	35.97%	53.05%	0:02:39	0:35:19	0:37:58
1	18	250	38.45%	54.71%	0:03:28	0:35:19	0:38:47
1	18	500	40.63%	56.33%	0:04:51	0:35:19	0:40:10
1	18	750	42.30%	57.76%	0:06:13	0:35:19	0:41:32
1	18	1000	42.95%	58.57%	0:07:30	0:35:19	0:42:49
1	18	1500	43.78%	58.84%	0:10:11	0:35:19	0:45:30
1	18	2000	44.61%	59.56%	0:13:00	0:35:19	0:48:19
1	20	100	36.57%	53.68%	0:02:48	0:38:28	0:41:16
1	20	250	39.44%	55.07%	0:03:38	0:38:28	0:42:06
1	20	500	42.59%	58.84%	0:05:00	0:38:28	0:43:28
1	20	750	43.07%	58.26%	0:06:22	0:38:28	0:44:50
1	20	1000	44.46%	59.34%	0:07:44	0:38:28	0:46:12
1	20	1500	45.28%	61.09%	0:10:27	0:38:28	0:48:55
1	20	2000	45.88%	60.50%	0:13:15	0:38:28	0:51:43
2	2	100	12.78%	25.40%	0:02:20	0:03:59	0:06:19
2	2	250	12.60%	24.28%	0:03:53	0:03:59	0:07:52

Table 1: Extended results for **Subset Distribution Selection**. Time is in format h:mm:ss.

					Time		
Epochs	**K**	**Iterations**	**mAP**	**Top-1**	**Training**	**SDS**	**Total**
2	2	500	11.83%	22.04%	0:06:31	0:03:59	0:10:30
2	2	750	11.76%	24.46%	0:09:03	0:03:59	0:13:02
2	2	1000	10.75%	21.18%	0:11:38	0:03:59	0:15:37
2	2	1500	9.72%	18.40%	0:16:51	0:03:59	0:20:50
2	2	2000	9.04%	17.32%	0:22:00	0:03:59	0:25:59
2	4	100	24.99%	38.87%	0:02:34	0:08:03	0:10:37
2	4	250	26.70%	40.08%	0:04:07	0:08:03	0:12:10
2	4	500	28.44%	43.27%	0:06:44	0:08:03	0:14:47
2	4	750	28.14%	42.37%	0:09:20	0:08:03	0:17:23
2	4	1000	27.29%	41.61%	0:11:51	0:08:03	0:19:54
2	4	1500	26.86%	41.29%	0:17:01	0:08:03	0:25:04
2	4	2000	24.07%	36.54%	0:22:19	0:08:03	0:30:22
2	6	100	31.02%	47.22%	0:02:46	0:12:00	0:14:46
2	6	250	34.21%	50.63%	0:04:22	0:12:00	0:16:22
2	6	500	36.02%	51.48%	0:07:04	0:12:00	0:19:04
2	6	750	36.70%	53.10%	0:09:33	0:12:00	0:21:33
2	6	1000	37.35%	52.60%	0:12:29	0:12:00	0:24:29
2	6	1500	36.97%	52.60%	0:17:25	0:12:00	0:29:25
2	6	2000	35.89%	50.81%	0:23:11	0:12:00	0:35:11
2	8	100	33.09%	49.15%	0:02:59	0:15:38	0:18:37
2	8	250	36.72%	52.74%	0:04:36	0:15:38	0:20:14
2	8	500	38.41%	53.90%	0:07:10	0:15:38	0:22:48
2	8	750	39.49%	55.48%	0:09:55	0:15:38	0:25:33

Table 1: Extended results for **Subset Distribution Selection**. Time is in format h:mm:ss.

					Time		
Epochs	**K**	**Iterations**	**mAP**	**Top-1**	**Training**	**SDS**	**Total**
2	8	1000	40.65%	55.79%	0:12:30	0:15:38	0:28:08
2	8	1500	41.43%	56.37%	0:17:38	0:15:38	0:33:16
2	8	2000	41.16%	56.37%	0:22:50	0:15:38	0:38:28
2	10	100	34.66%	50.81%	0:03:13	0:21:14	0:24:27
2	10	250	39.01%	54.58%	0:04:47	0:21:14	0:26:01
2	10	500	41.53%	57.59%	0:07:24	0:21:14	0:28:38
2	10	750	42.62%	58.84%	0:10:01	0:21:14	0:31:15
2	10	1000	43.87%	59.07%	0:12:50	0:21:14	0:34:04
2	10	1500	44.93%	60.19%	0:18:15	0:21:14	0:39:29
2	10	2000	44.32%	59.20%	0:23:48	0:21:14	0:45:02
2	12	100	37.52%	54.35%	0:03:26	0:23:38	0:27:04
2	12	250	42.07%	58.44%	0:05:01	0:23:38	0:28:39
2	12	500	44.08%	60.14%	0:07:42	0:23:38	0:31:20
2	12	750	45.39%	61.27%	0:10:26	0:23:38	0:34:04
2	12	1000	47.12%	63.11%	0:13:02	0:23:38	0:36:40
2	12	1500	49.42%	65.13%	0:18:22	0:23:38	0:42:00
2	12	2000	49.27%	64.54%	0:23:56	0:23:38	0:47:34
2	14	100	38.59%	54.58%	0:03:41	0:27:24	0:31:05
2	14	250	43.04%	58.93%	0:05:16	0:27:24	0:32:40
2	14	500	45.64%	61.54%	0:07:59	0:27:24	0:35:23
2	14	750	47.42%	63.42%	0:10:38	0:27:24	0:38:02
2	14	1000	48.64%	64.23%	0:13:18	0:27:24	0:40:42
2	14	1500	50.40%	65.75%	0:18:48	0:27:24	0:46:12

Table 1: Extended results for **Subset Distribution Selection**. Time is in format h:mm:ss.

					Time		
Epochs	**K**	**Iterations**	**mAP**	**Top-1**	**Training**	**SDS**	**Total**
2	14	2000	51.25%	66.74%	0:24:04	0:27:24	0:51:28
2	16	100	39.51%	55.92%	0:03:56	0:31:23	0:35:19
2	16	250	43.45%	59.69%	0:05:32	0:31:23	0:36:55
2	16	500	46.16%	61.76%	0:08:13	0:31:23	0:39:36
2	16	750	47.87%	62.84%	0:10:58	0:31:23	0:42:21
2	16	1000	49.44%	64.95%	0:13:37	0:31:23	0:45:00
2	16	1500	51.94%	67.55%	0:19:19	0:31:23	0:50:42
2	16	2000	52.42%	67.91%	0:24:29	0:31:23	0:55:52
2	18	100	38.78%	54.67%	0:04:08	0:35:19	0:39:27
2	18	250	44.36%	60.86%	0:05:50	0:35:19	0:41:09
2	18	500	46.84%	62.52%	0:08:32	0:35:19	0:43:51
2	18	750	48.46%	63.91%	0:11:12	0:35:19	0:46:31
2	18	1000	49.54%	64.63%	0:14:04	0:35:19	0:49:23
2	18	1500	51.13%	66.56%	0:19:44	0:35:19	0:55:03
2	18	2000	53.03%	68.18%	0:25:05	0:35:19	1:00:24
2	20	100	40.16%	56.82%	0:04:23	0:38:28	0:42:51
2	20	250	45.13%	61.67%	0:06:02	0:38:28	0:44:30
2	20	500	47.38%	63.15%	0:08:41	0:38:28	0:47:09
2	20	750	49.15%	65.08%	0:11:30	0:38:28	0:49:58
2	20	1000	51.05%	65.98%	0:14:24	0:38:28	0:52:52
2	20	1500	52.60%	67.95%	0:19:52	0:38:28	0:58:20
2	20	2000	53.49%	68.67%	0:25:27	0:38:28	1:03:55
3	2	100	12.64%	24.82%	0:02:56	0:03:59	0:06:55

Table 1: Extended results for **Subset Distribution Selection**. Time is in format h:mm:ss.

					Time		
Epochs	**K**	**Iterations**	**mAP**	**Top-1**	**Training**	**SDS**	**Total**
3	2	250	10.99%	21.50%	0:05:19	0:03:59	0:09:18
3	2	500	11.53%	21.59%	0:09:15	0:03:59	0:13:14
3	2	750	10.41%	19.34%	0:13:03	0:03:59	0:17:02
3	2	1000	9.56%	18.54%	0:17:00	0:03:59	0:20:59
3	2	1500	8.20%	15.75%	0:24:59	0:03:59	0:28:58
3	2	2000	6.54%	12.39%	0:32:43	0:03:59	0:36:42
3	4	100	25.75%	39.32%	0:03:16	0:08:03	0:11:19
3	4	250	27.58%	42.06%	0:05:37	0:08:03	0:13:40
3	4	500	28.15%	43.22%	0:09:33	0:08:03	0:17:36
3	4	750	28.12%	42.77%	0:13:27	0:08:03	0:21:30
3	4	1000	27.01%	41.65%	0:17:14	0:08:03	0:25:17
3	4	1500	24.78%	38.11%	0:25:12	0:08:03	0:33:15
3	4	2000	22.60%	35.95%	0:32:47	0:08:03	0:40:50
3	6	100	32.41%	48.43%	0:03:35	0:12:00	0:15:35
3	6	250	36.09%	52.65%	0:06:00	0:12:00	0:18:00
3	6	500	38.25%	54.35%	0:09:49	0:12:00	0:21:49
3	6	750	38.77%	55.16%	0:14:01	0:12:00	0:26:01
3	6	1000	37.71%	53.41%	0:17:51	0:12:00	0:29:51
3	6	1500	36.97%	52.11%	0:25:26	0:12:00	0:37:26
3	6	2000	36.71%	52.06%	0:33:18	0:12:00	0:45:18
3	8	100	34.64%	50.45%	0:03:55	0:15:38	0:19:33
3	8	250	39.17%	54.89%	0:06:17	0:15:38	0:21:55
3	8	500	41.42%	57.14%	0:10:16	0:15:38	0:25:54

Table 1: Extended results for **Subset Distribution Selection**. Time is in format h:mm:ss.

					Time		
Epochs	**K**	**Iterations**	**mAP**	**Top-1**	**Training**	**SDS**	**Total**
3	8	750	42.92%	58.30%	0:14:12	0:15:38	0:29:50
3	8	1000	42.75%	57.63%	0:18:11	0:15:38	0:33:49
3	8	1500	43.37%	58.66%	0:26:03	0:15:38	0:41:41
3	8	2000	42.31%	57.18%	0:34:03	0:15:38	0:49:41
3	10	100	37.13%	54.49%	0:04:15	0:21:14	0:25:29
3	10	250	41.66%	58.75%	0:06:38	0:21:14	0:27:52
3	10	500	43.71%	59.61%	0:10:49	0:21:14	0:32:03
3	10	750	45.86%	62.16%	0:14:29	0:21:14	0:35:43
3	10	1000	47.41%	63.33%	0:18:45	0:21:14	0:39:59
3	10	1500	48.24%	64.18%	0:26:28	0:21:14	0:47:42
3	10	2000	47.58%	63.02%	0:34:18	0:21:14	0:55:32
3	12	100	39.68%	56.42%	0:04:35	0:23:38	0:28:13
3	12	250	43.99%	61.04%	0:06:58	0:23:38	0:30:36
3	12	500	47.47%	63.06%	0:11:02	0:23:38	0:34:40
3	12	750	49.42%	65.26%	0:14:53	0:23:38	0:38:31
3	12	1000	50.61%	65.80%	0:19:23	0:23:38	0:43:01
3	12	1500	52.86%	67.95%	0:26:55	0:23:38	0:50:33
3	12	2000	52.58%	67.50%	0:35:29	0:23:38	0:59:07
3	14	100	41.80%	58.39%	0:04:56	0:27:24	0:32:20
3	14	250	45.58%	61.49%	0:07:24	0:27:24	0:34:48
3	14	500	49.04%	64.36%	0:11:19	0:27:24	0:38:43
3	14	750	50.85%	66.02%	0:15:25	0:27:24	0:42:49
3	14	1000	53.22%	68.63%	0:19:27	0:27:24	0:46:51

Table 1: Extended results for **Subset Distribution Selection**. Time is in format h:mm:ss.

					Time		
Epochs	**K**	**Iterations**	**mAP**	**Top-1**	**Training**	**SDS**	**Total**
3	14	1500	54.59%	69.93%	0:27:28	0:27:24	0:54:52
3	14	2000	55.35%	70.15%	0:35:29	0:27:24	1:02:53
3	16	100	42.28%	58.66%	0:05:16	0:31:23	0:36:39
3	16	250	46.17%	61.80%	0:07:46	0:31:23	0:39:09
3	16	500	50.15%	65.57%	0:11:46	0:31:23	0:43:09
3	16	750	52.02%	67.19%	0:16:02	0:31:23	0:47:25
3	16	1000	53.75%	69.48%	0:19:57	0:31:23	0:51:20
3	16	1500	55.07%	69.52%	0:28:10	0:31:23	0:59:33
3	16	2000	56.08%	70.65%	0:36:32	0:31:23	1:07:55
3	18	100	42.39%	58.53%	0:05:37	0:35:19	0:40:56
3	18	250	47.08%	62.61%	0:08:03	0:35:19	0:43:22
3	18	500	50.32%	65.93%	0:12:09	0:35:19	0:47:28
3	18	750	52.22%	67.32%	0:16:09	0:35:19	0:51:28
3	18	1000	54.55%	69.88%	0:20:32	0:35:19	0:55:51
3	18	1500	56.06%	71.68%	0:28:45	0:35:19	1:04:04
3	18	2000	56.91%	71.41%	0:37:16	0:35:19	1:12:35
3	20	100	43.10%	60.86%	0:05:58	0:38:28	0:44:26
3	20	250	48.12%	64.18%	0:08:27	0:38:28	0:46:55
3	20	500	51.18%	66.88%	0:12:40	0:38:28	0:51:08
3	20	750	53.41%	68.45%	0:16:50	0:38:28	0:55:18
3	20	1000	54.64%	70.24%	0:20:46	0:38:28	0:59:14
3	20	1500	56.12%	70.65%	0:29:33	0:38:28	1:08:01
3	20	2000	57.24%	72.49%	0:37:55	0:38:28	1:16:23

Table 1: Extended results for **Subset Distribution Selection**. Time is in format h:mm:ss.

					Time		
Epochs	**K**	**Iterations**	**mAP**	**Top-1**	**Training**	**SDS**	**Total**
5	2	100	12.35%	23.38%	0:04:10	0:03:59	0:08:09
5	2	250	9.21%	17.73%	0:08:05	0:03:59	0:12:04
5	2	500	9.83%	19.12%	0:14:34	0:03:59	0:18:33
5	2	750	8.20%	16.83%	0:21:32	0:03:59	0:25:31
5	2	1000	7.57%	13.82%	0:27:37	0:03:59	0:31:36
5	2	1500	6.76%	12.52%	0:41:33	0:03:59	0:45:32
5	2	2000	4.84%	10.55%	0:54:25	0:03:59	0:58:24
5	4	100	26.89%	41.65%	0:04:42	0:08:03	0:12:45
5	4	250	28.01%	42.86%	0:08:37	0:08:03	0:16:40
5	4	500	27.71%	41.88%	0:15:02	0:08:03	0:23:05
5	4	750	27.31%	42.24%	0:21:53	0:08:03	0:29:56
5	4	1000	23.02%	35.23%	0:28:06	0:08:03	0:36:09
5	4	1500	21.17%	34.07%	0:40:57	0:08:03	0:49:00
5	4	2000	19.45%	30.30%	0:54:15	0:08:03	1:02:18
5	6	100	33.78%	49.24%	0:05:13	0:12:00	0:17:13
5	6	250	38.35%	54.26%	0:09:07	0:12:00	0:21:07
5	6	500	39.25%	54.58%	0:15:40	0:12:00	0:27:40
5	6	750	40.58%	57.76%	0:22:10	0:12:00	0:34:10
5	6	1000	39.12%	54.71%	0:29:11	0:12:00	0:41:11
5	6	1500	36.21%	50.63%	0:41:53	0:12:00	0:53:53
5	6	2000	36.21%	51.21%	0:55:48	0:12:00	1:07:48
5	8	100	37.59%	53.68%	0:05:47	0:15:38	0:21:25
5	8	250	42.05%	57.59%	0:09:43	0:15:38	0:25:21

Table 1: Extended results for **Subset Distribution Selection**. Time is in format h:mm:ss.

					Time		
Epochs	**K**	**Iterations**	**mAP**	**Top-1**	**Training**	**SDS**	**Total**
5	8	500	44.57%	59.96%	0:16:17	0:15:38	0:31:55
5	8	750	45.06%	60.41%	0:22:52	0:15:38	0:38:30
5	8	1000	45.28%	61.31%	0:29:11	0:15:38	0:44:49
5	8	1500	44.67%	59.61%	0:42:53	0:15:38	0:58:31
5	8	2000	42.76%	58.17%	0:56:21	0:15:38	1:11:59
5	10	100	40.43%	57.18%	0:06:19	0:21:14	0:27:33
5	10	250	44.53%	59.96%	0:10:17	0:21:14	0:31:31
5	10	500	48.25%	64.09%	0:16:53	0:21:14	0:38:07
5	10	750	49.67%	65.04%	0:23:26	0:21:14	0:44:40
5	10	1000	50.52%	65.66%	0:30:09	0:21:14	0:51:23
5	10	1500	50.53%	66.29%	0:43:17	0:21:14	1:04:31
5	10	2000	49.57%	65.22%	0:56:34	0:21:14	1:17:48
5	12	100	42.86%	59.61%	0:06:53	0:23:38	0:30:31
5	12	250	47.98%	63.51%	0:10:54	0:23:38	0:34:32
5	12	500	52.11%	67.10%	0:17:43	0:23:38	0:41:21
5	12	750	54.04%	69.75%	0:24:34	0:23:38	0:48:12
5	12	1000	55.71%	70.92%	0:30:59	0:23:38	0:54:37
5	12	1500	56.27%	71.23%	0:44:15	0:23:38	1:07:53
5	12	2000	56.75%	71.86%	0:58:29	0:23:38	1:22:07
5	14	100	44.68%	61.22%	0:07:28	0:27:24	0:34:52
5	14	250	49.46%	64.86%	0:11:24	0:27:24	0:38:48
5	14	500	53.83%	68.67%	0:18:23	0:27:24	0:45:47
5	14	750	56.28%	71.86%	0:24:56	0:27:24	0:52:20

Table 1: Extended results for **Subset Distribution Selection**. Time is in format h:mm:ss.

					Time		
Epochs	**K**	**Iterations**	**mAP**	**Top-1**	**Training**	**SDS**	**Total**
5	14	1000	57.56%	72.26%	0:31:58	0:27:24	0:59:22
5	14	1500	58.01%	72.49%	0:45:00	0:27:24	1:12:24
5	14	2000	58.59%	73.83%	0:59:21	0:27:24	1:26:45
5	16	100	45.96%	63.02%	0:08:01	0:31:23	0:39:24
5	16	250	50.40%	66.16%	0:12:07	0:31:23	0:43:30
5	16	500	54.93%	70.15%	0:19:01	0:31:23	0:50:24
5	16	750	56.81%	71.81%	0:25:56	0:31:23	0:57:19
5	16	1000	58.29%	72.67%	0:32:22	0:31:23	1:03:45
5	16	1500	59.71%	73.16%	0:46:06	0:31:23	1:17:29
5	16	2000	60.34%	74.33%	1:00:38	0:31:23	1:32:01
5	18	100	45.40%	62.30%	0:08:34	0:35:19	0:43:53
5	18	250	50.85%	65.84%	0:12:33	0:35:19	0:47:52
5	18	500	54.87%	70.47%	0:19:35	0:35:19	0:54:54
5	18	750	57.62%	72.17%	0:26:19	0:35:19	1:01:38
5	18	1000	58.38%	72.67%	0:33:25	0:35:19	1:08:44
5	18	1500	60.44%	74.55%	0:47:08	0:35:19	1:22:27
5	18	2000	60.69%	75.18%	1:01:39	0:35:19	1:36:58
5	20	100	46.20%	63.38%	0:09:12	0:38:28	0:47:40
5	20	250	51.77%	68.58%	0:13:17	0:38:28	0:51:45
5	20	500	55.64%	70.42%	0:20:05	0:38:28	0:58:33
5	20	750	57.36%	71.95%	0:27:04	0:38:28	1:05:32
5	20	1000	58.96%	73.11%	0:34:31	0:38:28	1:12:59
5	20	1500	59.63%	73.25%	0:48:08	0:38:28	1:26:36

Table 1: Extended results for **Subset Distribution Selection**. Time is in format h:mm:ss.

					Time		
Epochs	**K**	**Iterations**	**mAP**	**Top-1**	**Training**	**SDS**	**Total**
5	20	2000	60.05%	73.97%	1:01:52	0:38:28	1:40:20

Appendix B: Extended Results for System Generated Data

The following table contains the complete list of experiments conducted for Sec. 5.5.2 of the main paper. One thing to notice is that all 750 iteration data points are missing from this section. Quite frankly, this was left out of our experiments on accident. However, it makes no difference in identifying the trends in the data. Since we were able to learn what we set out to understand with these experiments, we did not feel the need to go back and retrain those networks. Considering the amount of training needed for the experiments to follow, we decided our time and resources were better spent training the networks for Sec. 5.5.2.

Table 2: Extended results for **System Generated Data**. Time is in format h:mm:ss.

					Time		
Epochs	**K**	**Iterations**	**mAP**	**Top-1**	**Training**	**SDS**	**Total**
1	16	100	31.57%	46.86%	0:02:35	0:07:23	0:09:58
1	16	250	34.04%	49.60%	0:03:21	0:07:23	0:10:44
1	16	500	37.00%	52.47%	0:04:44	0:07:23	0:12:07
1	16	1000	38.71%	53.73%	0:07:26	0:07:23	0:14:49
1	16	1500	39.87%	54.40%	0:10:11	0:07:23	0:17:34
1	18	100	31.63%	47.58%	0:02:42	0:07:17	0:09:59
1	18	250	35.76%	52.06%	0:03:31	0:07:17	0:10:48
1	18	500	38.27%	54.31%	0:04:53	0:07:17	0:12:10
1	18	1000	39.79%	55.39%	0:07:38	0:07:17	0:14:55
1	18	1500	40.55%	55.97%	0:10:35	0:07:17	0:17:52
1	20	100	31.16%	47.67%	0:02:51	0:08:13	0:11:04
1	20	250	34.02%	49.46%	0:03:42	0:08:13	0:11:55
1	20	500	37.14%	53.50%	0:05:03	0:08:13	0:13:16

Table 2: Extended results for **System Generated Data**. Time is in format h:mm:ss.

					Time		
Epochs	**K**	**Iterations**	**mAP**	**Top-1**	**Training**	**SDS**	**Total**
1	20	1000	38.48%	53.82%	0:07:50	0:08:13	0:16:03
1	20	1500	40.14%	55.83%	0:10:35	0:08:13	0:18:48
1	25	100	30.22%	47.22%	0:03:16	0:10:14	0:13:30
1	25	250	32.40%	49.69%	0:04:05	0:10:14	0:14:19
1	25	500	34.91%	51.84%	0:05:30	0:10:14	0:15:44
1	25	1000	37.55%	54.49%	0:08:25	0:10:14	0:18:39
1	25	1500	38.25%	54.98%	0:11:23	0:10:14	0:21:37
1	30	100	28.80%	44.97%	0:03:45	0:12:24	0:16:09
1	30	250	31.09%	48.29%	0:04:33	0:12:24	0:16:57
1	30	500	33.79%	50.54%	0:06:03	0:12:24	0:18:27
1	30	1000	35.26%	51.97%	0:08:53	0:12:24	0:21:17
1	30	1500	36.10%	52.74%	0:11:50	0:12:24	0:24:14
1	40	100	22.92%	38.82%	0:04:41	0:16:32	0:21:13
1	40	250	24.31%	40.93%	0:05:39	0:16:32	0:22:11
1	40	500	25.57%	42.77%	0:07:01	0:16:32	0:23:33
1	40	1000	27.02%	43.99%	0:10:00	0:16:32	0:26:32
1	40	1500	27.34%	45.29%	0:12:59	0:16:32	0:29:31
2	16	100	34.10%	49.42%	0:04:00	0:07:23	0:11:23
2	16	250	38.92%	55.21%	0:05:39	0:07:23	0:13:02
2	16	500	41.36%	56.37%	0:08:14	0:07:23	0:15:37
2	16	1000	44.03%	58.30%	0:13:57	0:07:23	0:21:20
2	16	1500	46.24%	61.22%	0:19:18	0:07:23	0:26:41
2	18	100	35.22%	51.30%	0:04:14	0:07:17	0:11:31

Table 2: Extended results for **System Generated Data**. Time is in format h:mm:ss.

					Time		
Epochs	**K**	**Iterations**	**mAP**	**Top-1**	**Training**	**SDS**	**Total**
2	18	250	39.21%	55.52%	0:05:53	0:07:17	0:13:10
2	18	500	41.60%	57.05%	0:08:43	0:07:17	0:16:00
2	18	1000	44.64%	59.74%	0:14:09	0:07:17	0:21:26
2	18	1500	46.47%	61.04%	0:19:46	0:07:17	0:27:03
2	20	100	34.94%	51.75%	0:04:33	0:08:13	0:12:46
2	20	250	38.82%	54.80%	0:06:12	0:08:13	0:14:25
2	20	500	42.45%	58.57%	0:08:56	0:08:13	0:17:09
2	20	1000	45.06%	60.68%	0:14:34	0:08:13	0:22:47
2	20	1500	46.75%	62.21%	0:19:59	0:08:13	0:28:12
2	25	100	34.08%	50.90%	0:05:18	0:10:14	0:15:32
2	25	250	38.12%	54.44%	0:06:58	0:10:14	0:17:12
2	25	500	41.47%	57.81%	0:09:41	0:10:14	0:19:55
2	25	1000	43.87%	59.52%	0:15:26	0:10:14	0:25:40
2	25	1500	45.48%	61.22%	0:21:00	0:10:14	0:31:14
2	30	100	31.76%	48.52%	0:06:05	0:12:24	0:18:29
2	30	250	36.01%	52.42%	0:07:49	0:12:24	0:20:13
2	30	500	38.79%	55.75%	0:10:42	0:12:24	0:23:06
2	30	1000	41.34%	57.94%	0:16:32	0:12:24	0:28:56
2	30	1500	42.30%	57.94%	0:22:20	0:12:24	0:34:44
2	40	100	25.78%	42.50%	0:07:51	0:16:32	0:24:23
2	40	250	29.64%	47.08%	0:09:39	0:16:32	0:26:11
2	40	500	31.39%	48.65%	0:12:34	0:16:32	0:29:06
2	40	1000	34.23%	50.85%	0:18:40	0:16:32	0:35:12

Table 2: Extended results for **System Generated Data**. Time is in format h:mm:ss.

					Time		
Epochs	**K**	**Iterations**	**mAP**	**Top-1**	**Training**	**SDS**	**Total**
2	40	1500	33.77%	50.90%	0:24:44	0:16:32	0:41:16
3	16	100	35.75%	51.97%	0:05:23	0:07:23	0:12:46
3	16	250	41.17%	56.69%	0:07:52	0:07:23	0:15:15
3	16	500	43.89%	59.20%	0:11:53	0:07:23	0:19:16
3	16	1000	47.18%	62.43%	0:20:09	0:07:23	0:27:32
3	16	1500	49.12%	64.63%	0:28:19	0:07:23	0:35:42
3	18	100	37.82%	54.26%	0:05:48	0:07:17	0:13:05
3	18	250	41.58%	57.94%	0:08:18	0:07:17	0:15:35
3	18	500	44.84%	60.32%	0:12:22	0:07:17	0:19:39
3	18	1000	48.47%	63.87%	0:20:30	0:07:17	0:27:47
3	18	1500	50.46%	65.84%	0:29:18	0:07:17	0:36:35
3	20	100	36.60%	53.41%	0:06:11	0:08:13	0:14:24
3	20	250	40.81%	55.97%	0:08:42	0:08:13	0:16:55
3	20	500	45.17%	60.68%	0:12:55	0:08:13	0:21:08
3	20	1000	48.91%	63.78%	0:21:24	0:08:13	0:29:37
3	20	1500	50.87%	65.57%	0:29:45	0:08:13	0:37:58
3	25	100	36.63%	53.59%	0:07:18	0:10:14	0:17:32
3	25	250	41.40%	58.26%	0:09:47	0:10:14	0:20:01
3	25	500	45.24%	60.73%	0:14:07	0:10:14	0:24:21
3	25	1000	47.86%	63.46%	0:22:28	0:10:14	0:32:42
3	25	1500	48.33%	64.59%	0:31:16	0:10:14	0:41:30
3	30	100	35.02%	51.66%	0:08:27	0:12:24	0:20:51
3	30	250	39.05%	56.10%	0:11:02	0:12:24	0:23:26

Table 2: Extended results for **System Generated Data**. Time is in format h:mm:ss.

					Time		
Epochs	**K**	**Iterations**	**mAP**	**Top-1**	**Training**	**SDS**	**Total**
3	30	500	43.18%	58.98%	0:15:24	0:12:24	0:27:48
3	30	1000	45.76%	62.25%	0:24:04	0:12:24	0:36:28
3	30	1500	45.60%	61.31%	0:32:50	0:12:24	0:45:14
3	40	100	27.93%	45.87%	0:11:04	0:16:32	0:27:36
3	40	250	31.16%	49.10%	0:13:39	0:16:32	0:30:11
3	40	500	35.86%	53.10%	0:18:17	0:16:32	0:34:49
3	40	1000	40.09%	56.55%	0:27:13	0:16:32	0:43:45
3	40	1500	39.37%	55.75%	0:36:28	0:16:32	0:53:00
5	16	100	39.30%	55.43%	0:08:10	0:07:23	0:15:33
5	16	250	44.10%	59.38%	0:12:17	0:07:23	0:19:40
5	16	500	48.11%	64.14%	0:19:02	0:07:23	0:26:25
5	16	1000	50.75%	65.80%	0:32:46	0:07:23	0:40:09
5	16	1500	52.13%	67.24%	0:47:02	0:07:23	0:54:25
5	18	100	40.26%	56.96%	0:08:53	0:07:17	0:16:10
5	18	250	45.30%	60.05%	0:13:02	0:07:17	0:20:19
5	18	500	48.45%	63.46%	0:19:37	0:07:17	0:26:54
5	18	1000	51.11%	66.47%	0:34:02	0:07:17	0:41:19
5	18	1500	53.01%	67.73%	0:47:06	0:07:17	0:54:23
5	20	100	40.09%	56.55%	0:09:34	0:08:13	0:17:47
5	20	250	44.73%	61.00%	0:13:41	0:08:13	0:21:54
5	20	500	48.92%	64.00%	0:20:49	0:08:13	0:29:02
5	20	1000	53.03%	68.27%	0:34:32	0:08:13	0:42:45
5	20	1500	54.55%	69.34%	0:48:26	0:08:13	0:56:39

Table 2: Extended results for **System Generated Data**. Time is in format h:mm:ss.

					Time		
Epochs	**K**	**Iterations**	**mAP**	**Top-1**	**Training**	**SDS**	**Total**
5	25	100	39.16%	55.30%	0:11:22	0:10:14	0:21:36
5	25	250	44.90%	61.00%	0:15:32	0:10:14	0:25:46
5	25	500	49.02%	64.77%	0:22:24	0:10:14	0:32:38
5	25	1000	52.48%	67.46%	0:36:53	0:10:14	0:47:07
5	25	1500	55.03%	68.99%	0:51:27	0:10:14	1:01:41
5	30	100	36.79%	53.14%	0:13:19	0:12:24	0:25:43
5	30	250	42.55%	59.20%	0:17:32	0:12:24	0:29:56
5	30	500	47.81%	64.32%	0:24:49	0:12:24	0:37:13
5	30	1000	51.41%	66.56%	0:38:51	0:12:24	0:51:15
5	30	1500	52.54%	67.82%	0:54:06	0:12:24	1:06:30
5	40	100	29.58%	46.77%	0:17:21	0:16:32	0:33:53
5	40	250	36.14%	52.87%	0:21:46	0:16:32	0:38:18
5	40	500	39.39%	55.88%	0:29:24	0:16:32	0:45:56
5	40	1000	44.26%	60.95%	0:44:20	0:16:32	1:00:52
5	40	1500	46.12%	62.43%	0:59:27	0:16:32	1:15:59

Appendix C: Extended Results for R^{2}MMT

The following tables contain the complete list of experiments conducted for Sec. 5.5.3 of the main paper.

Standard Configuration

Here we see the results of R^{2}MMT in the standard configuration. Training time does not include SDS time. Due to the pipeline structure of R^{2}MMT, both the training time and the SDS time must be less than τ to be viable in the real world. Some experiments were completed with $\tau = 10$. However, due to their terrible accuracy, inability to meet the real-world time constraint of R^{2}OUDA, and the resources needed to train that many extra models, it was dropped from our testing configurations. We have included them here for anyone curious to see what a smaller time segment might look like, even if the picture is incomplete. Absolute highest accuracy in this test, ignoring the time constraint, is 74.96% Top-1. However, it takes around an hour of training time for each time segment and is thus not valid for real-world deployment.

Table 3: Extended results for **R^{2}MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	18	15	0	100	21.60%	36.40%	0:01:52
1	18	15	1	100	22.09%	35.68%	0:02:11
1	18	15	2	100	18.33%	30.92%	0:01:56
1	18	15	3	100	15.41%	26.93%	0:01:51
1	18	15	0	250	21.84%	35.28%	0:02:42
1	18	15	1	250	23.93%	38.11%	0:03:00
1	18	15	2	250	21.81%	34.25%	0:02:45
1	18	15	3	250	15.35%	26.30%	0:02:41
1	18	15	0	500	23.18%	38.20%	0:04:05

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	18	15	1	500	24.91%	38.29%	0:04:26
1	18	15	2	500	21.86%	34.20%	0:04:06
1	18	15	3	500	15.31%	26.84%	0:04:04
1	18	15	0	750	23.28%	37.88%	0:05:26
1	18	15	1	750	26.25%	40.57%	0:05:54
1	18	15	2	750	24.74%	38.96%	0:05:29
1	18	15	3	750	20.55%	34.43%	0:05:27
1	18	15	0	1000	23.60%	38.55%	0:06:48
1	18	15	1	1000	25.09%	38.29%	0:07:14
1	18	15	2	1000	22.19%	35.82%	0:06:54
1	18	15	3	1000	16.12%	28.28%	0:06:50
1	18	15	0	1500	24.37%	39.18%	0:09:30
1	18	15	1	1500	27.22%	41.43%	0:09:58
1	18	15	2	1500	24.91%	39.05%	0:09:39
1	18	15	3	1500	21.99%	36.13%	0:09:35
1	18	20	0	100	24.09%	39.00%	0:02:06
1	18	20	1	100	20.69%	33.75%	0:01:59
1	18	20	2	100	16.47%	28.73%	0:01:55
1	18	20	0	250	26.90%	42.10%	0:02:54
1	18	20	1	250	23.84%	37.03%	0:02:50
1	18	20	2	250	23.18%	35.86%	0:02:44
1	18	20	0	500	26.20%	40.66%	0:04:16
1	18	20	1	500	26.08%	39.86%	0:04:12
1	18	20	2	500	23.88%	36.62%	0:04:09

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	18	20	0	750	27.39%	41.97%	0:05:38
1	18	20	1	750	26.85%	39.90%	0:05:41
1	18	20	2	750	23.72%	36.22%	0:05:30
1	18	20	0	1000	28.06%	43.00%	0:07:00
1	18	20	1	1000	26.33%	39.72%	0:06:59
1	18	20	2	1000	21.50%	32.81%	0:06:55
1	18	20	0	1500	28.34%	42.73%	0:09:50
1	18	20	1	1500	28.80%	42.37%	0:09:45
1	18	20	2	1500	28.00%	41.43%	0:09:39
1	18	30	0	100	29.08%	45.06%	0:02:23
1	18	30	1	100	27.99%	43.22%	0:02:04
1	18	30	0	250	32.12%	48.07%	0:03:11
1	18	30	1	250	34.07%	49.87%	0:02:53
1	18	30	0	500	33.04%	49.42%	0:04:35
1	18	30	1	500	35.48%	50.76%	0:04:18
1	18	30	0	750	33.79%	49.64%	0:06:01
1	18	30	1	750	37.57%	53.23%	0:05:46
1	18	30	0	1000	34.23%	49.46%	0:07:20
1	18	30	1	1000	37.05%	52.20%	0:07:06
1	18	30	0	1500	34.65%	49.46%	0:10:08
1	18	30	1	1500	36.98%	51.75%	0:09:50
1	20	10	0	100	15.32%	27.11%	0:01:48
1	20	10	1	100	12.23%	22.26%	0:01:59
1	20	10	2	100	10.44%	20.20%	0:01:58

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	20	10	3	100	8.05%	16.56%	0:01:50
1	20	10	4	100	4.09%	10.64%	0:01:51
1	20	10	0	250	16.45%	30.39%	0:02:37
1	20	10	1	250	13.57%	24.69%	0:02:49
1	20	10	2	250	11.29%	20.92%	0:02:49
1	20	10	3	250	6.71%	14.81%	0:02:39
1	20	10	4	250	2.68%	7.85%	0:02:40
1	20	10	0	500	15.72%	28.46%	0:03:57
1	20	10	1	500	14.86%	25.76%	0:04:15
1	20	10	2	500	11.48%	20.47%	0:04:12
1	20	10	3	500	5.95%	13.69%	0:04:01
1	20	10	4	500	2.43%	6.78%	0:03:57
1	20	10	0	750	15.59%	28.59%	0:05:18
1	20	10	1	750	12.78%	22.44%	0:05:34
1	20	10	2	750	10.67%	19.97%	0:05:35
1	20	10	3	750	6.73%	15.22%	0:05:26
1	20	10	4	750	3.85%	10.37%	0:05:21
1	20	10	0	1000	16.22%	27.92%	0:06:40
1	20	10	1	1000	14.16%	24.91%	0:07:00
1	20	10	2	1000	10.59%	19.48%	0:06:58
1	20	10	3	1000	5.61%	13.46%	0:06:45
1	20	10	4	1000	2.23%	6.96%	0:06:44
1	20	10	0	1500	15.25%	27.51%	0:09:21
1	20	10	1	1500	11.83%	20.56%	0:09:45

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	20	10	2	1500	10.56%	19.52%	0:09:43
1	20	10	3	1500	5.23%	12.79%	0:09:31
1	20	10	4	1500	2.34%	6.60%	0:09:25
1	20	10	5	1500	0.54%	0.99%	0:09:28
1	20	10	0	2000	16.56%	28.73%	0:12:03
1	20	10	1	2000	11.75%	22.13%	0:12:32
1	20	10	2	2000	9.66%	18.00%	0:12:25
1	20	10	3	2000	5.08%	12.07%	0:12:10
1	20	10	4	2000	2.09%	6.33%	0:12:13
1	20	15	0	100	23.44%	38.96%	0:01:53
1	20	15	1	100	22.59%	36.09%	0:02:13
1	20	15	2	100	18.78%	31.51%	0:01:55
1	20	15	3	100	17.20%	30.88%	0:01:52
1	20	15	0	250	23.99%	38.24%	0:02:41
1	20	15	1	250	23.77%	36.54%	0:03:04
1	20	15	2	250	22.68%	35.64%	0:02:44
1	20	15	3	250	19.76%	32.50%	0:02:42
1	20	15	0	500	25.65%	39.90%	0:04:02
1	20	15	1	500	25.14%	39.27%	0:04:26
1	20	15	2	500	24.88%	37.61%	0:04:07
1	20	15	3	500	20.07%	32.99%	0:04:03
1	20	15	0	750	26.00%	41.20%	0:05:24
1	20	15	1	750	27.89%	42.46%	0:05:50
1	20	15	2	750	25.92%	39.05%	0:05:30

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	20	15	3	750	22.55%	36.40%	0:05:28
1	20	15	0	1000	26.40%	41.43%	0:06:46
1	20	15	1	1000	26.62%	39.99%	0:07:31
1	20	15	2	1000	24.62%	37.75%	0:06:52
1	20	15	3	1000	23.23%	37.34%	0:06:51
1	20	15	0	1500	27.33%	42.41%	0:09:29
1	20	15	1	1500	28.62%	42.59%	0:10:13
1	20	15	2	1500	25.90%	39.90%	0:09:37
1	20	15	3	1500	22.99%	36.62%	0:09:36
1	20	15	0	2000	26.77%	41.29%	0:12:13
1	20	15	1	2000	27.09%	40.57%	0:12:50
1	20	15	2	2000	23.84%	35.82%	0:12:21
1	20	15	3	2000	20.93%	33.17%	0:12:26
1	20	20	0	100	24.90%	39.50%	0:02:06
1	20	20	1	100	21.90%	35.64%	0:02:00
1	20	20	2	100	20.73%	33.39%	0:01:56
1	20	20	0	250	26.44%	41.34%	0:02:57
1	20	20	1	250	25.38%	39.59%	0:02:50
1	20	20	2	250	25.41%	38.64%	0:02:46
1	20	20	0	500	26.94%	41.88%	0:04:19
1	20	20	1	500	26.82%	39.77%	0:04:12
1	20	20	2	500	27.74%	41.38%	0:04:08
1	20	20	0	750	27.70%	42.10%	0:05:40
1	20	20	1	750	28.69%	42.59%	0:05:36

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	20	20	2	750	28.76%	41.70%	0:05:32
1	20	20	0	1000	27.26%	41.20%	0:07:06
1	20	20	1	1000	27.51%	40.35%	0:06:58
1	20	20	2	1000	26.00%	39.99%	0:06:55
1	20	20	0	1500	28.93%	43.72%	0:09:48
1	20	20	1	1500	29.16%	41.83%	0:09:45
1	20	20	2	1500	31.27%	45.56%	0:09:42
1	20	20	0	2000	28.76%	42.46%	0:12:34
1	20	20	1	2000	28.42%	41.38%	0:12:30
1	20	20	2	2000	27.89%	40.89%	0:12:28
1	20	30	0	100	29.26%	44.30%	0:02:27
1	20	30	1	100	28.99%	43.99%	0:02:05
1	20	30	0	250	32.19%	47.62%	0:03:16
1	20	30	1	250	33.11%	48.74%	0:02:54
1	20	30	0	500	34.09%	49.15%	0:04:40
1	20	30	1	500	36.81%	51.89%	0:04:20
1	20	30	0	750	34.60%	48.83%	0:06:04
1	20	30	1	750	38.37%	52.33%	0:05:48
1	20	30	0	1000	35.54%	49.96%	0:07:29
1	20	30	1	1000	39.85%	53.95%	0:07:05
1	20	30	0	1500	36.38%	50.94%	0:10:22
1	20	30	1	1500	40.49%	54.62%	0:09:50
1	20	30	0	2000	36.32%	49.78%	0:12:59
1	20	30	1	2000	39.51%	53.73%	0:12:41

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	25	15	0	100	25.13%	41.20%	0:01:57
1	25	15	1	100	26.65%	41.74%	0:02:24
1	25	15	2	100	23.31%	37.61%	0:01:59
1	25	15	3	100	22.16%	36.40%	0:01:55
1	25	15	0	250	27.38%	43.31%	0:02:46
1	25	15	1	250	30.43%	46.63%	0:03:13
1	25	15	2	250	29.46%	44.61%	0:02:48
1	25	15	3	250	29.49%	45.38%	0:02:46
1	25	15	0	500	29.62%	46.23%	0:04:08
1	25	15	1	500	33.71%	49.60%	0:04:38
1	25	15	2	500	33.44%	48.74%	0:04:12
1	25	15	3	500	30.54%	45.56%	0:04:08
1	25	15	0	750	30.78%	47.53%	0:05:29
1	25	15	1	750	34.31%	49.91%	0:06:08
1	25	15	2	750	34.49%	49.78%	0:05:36
1	25	15	3	750	33.05%	48.83%	0:05:32
1	25	15	0	1000	30.82%	47.35%	0:06:52
1	25	15	1	1000	34.12%	49.91%	0:07:31
1	25	15	2	1000	35.90%	51.35%	0:06:57
1	25	15	3	1000	34.08%	48.47%	0:06:59
1	25	15	0	1500	31.32%	46.95%	0:09:39
1	25	15	1	1500	35.47%	50.63%	0:10:15
1	25	15	2	1500	36.79%	51.97%	0:09:43
1	25	15	3	1500	35.71%	51.48%	0:09:46

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	25	20	0	100	28.81%	44.66%	0:02:16
1	25	20	1	100	29.49%	44.48%	0:02:18
1	25	20	2	100	28.19%	44.52%	0:02:00
1	25	20	0	250	31.07%	45.87%	0:03:05
1	25	20	1	250	34.53%	49.69%	0:03:07
1	25	20	2	250	35.04%	50.09%	0:02:51
1	25	20	0	500	32.38%	47.22%	0:04:31
1	25	20	1	500	37.08%	51.97%	0:04:32
1	25	20	2	500	40.42%	55.88%	0:04:14
1	25	20	0	750	33.16%	48.56%	0:05:51
1	25	20	1	750	38.22%	52.69%	0:05:56
1	25	20	2	750	40.57%	55.16%	0:05:37
1	25	20	0	1000	33.26%	48.88%	0:07:14
1	25	20	1	1000	38.19%	52.60%	0:07:18
1	25	20	2	1000	41.45%	56.64%	0:07:00
1	25	20	0	1500	33.55%	49.24%	0:09:58
1	25	20	1	1500	38.58%	52.83%	0:10:21
1	25	20	2	1500	41.85%	56.78%	0:09:45
1	25	30	0	100	33.82%	50.40%	0:02:41
1	25	30	1	100	33.06%	50.00%	0:02:12
1	25	30	0	250	36.14%	51.75%	0:03:30
1	25	30	1	250	39.28%	55.79%	0:03:02
1	25	30	0	500	38.65%	54.40%	0:04:54
1	25	30	1	500	42.86%	58.71%	0:04:25

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	25	30	0	750	39.81%	55.70%	0:06:28
1	25	30	1	750	43.57%	59.74%	0:05:55
1	25	30	0	1000	39.96%	55.57%	0:07:45
1	25	30	1	1000	45.45%	60.64%	0:07:15
1	25	30	0	1500	40.83%	56.51%	0:10:35
1	25	30	1	1500	45.78%	60.91%	0:10:02
1	30	15	0	100	26.40%	43.13%	0:02:01
1	30	15	1	100	28.72%	44.48%	0:02:33
1	30	15	2	100	25.38%	39.27%	0:02:03
1	30	15	3	100	23.81%	38.15%	0:01:57
1	30	15	0	250	28.49%	43.94%	0:02:49
1	30	15	1	250	33.04%	49.51%	0:03:24
1	30	15	2	250	32.66%	49.01%	0:02:51
1	30	15	3	250	32.11%	48.11%	0:02:46
1	30	15	0	500	31.14%	46.95%	0:04:11
1	30	15	1	500	35.54%	50.99%	0:04:48
1	30	15	2	500	36.66%	52.74%	0:04:15
1	30	15	3	500	36.07%	52.24%	0:04:10
1	30	15	0	750	31.52%	47.80%	0:05:33
1	30	15	1	750	37.21%	52.60%	0:06:19
1	30	15	2	750	38.86%	54.04%	0:05:40
1	30	15	3	750	38.52%	54.53%	0:05:33
1	30	15	0	1000	32.43%	48.70%	0:06:54
1	30	15	1	1000	37.86%	53.19%	0:07:39

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	30	15	2	1000	39.98%	56.33%	0:07:06
1	30	15	3	1000	38.10%	54.35%	0:06:57
1	30	15	0	1500	32.57%	47.62%	0:09:38
1	30	15	1	1500	37.44%	51.97%	0:10:30
1	30	15	2	1500	39.73%	55.03%	0:09:49
1	30	15	3	1500	38.34%	55.03%	0:09:45
1	30	20	0	100	30.48%	46.81%	0:02:24
1	30	20	1	100	32.30%	49.01%	0:02:26
1	30	20	2	100	31.63%	48.16%	0:02:03
1	30	20	0	250	32.21%	48.56%	0:03:13
1	30	20	1	250	36.86%	53.05%	0:03:15
1	30	20	2	250	38.52%	55.34%	0:02:53
1	30	20	0	500	34.40%	50.36%	0:04:36
1	30	20	1	500	40.65%	56.55%	0:04:39
1	30	20	2	500	42.83%	58.35%	0:04:16
1	30	20	0	750	35.03%	51.30%	0:06:02
1	30	20	1	750	41.38%	57.36%	0:06:04
1	30	20	2	750	43.98%	60.10%	0:05:45
1	30	20	0	1000	35.75%	51.71%	0:07:21
1	30	20	1	1000	42.31%	57.27%	0:07:27
1	30	20	2	1000	44.05%	59.16%	0:07:03
1	30	20	0	1500	36.11%	52.11%	0:10:07
1	30	20	1	1500	42.60%	57.72%	0:10:20
1	30	20	2	1500	46.01%	61.40%	0:09:50

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	30	30	0	100	33.48%	50.58%	0:02:58
1	30	30	1	100	35.22%	52.60%	0:02:19
1	30	30	0	250	36.70%	53.55%	0:03:47
1	30	30	1	250	40.43%	57.90%	0:03:08
1	30	30	0	500	38.79%	54.76%	0:05:14
1	30	30	1	500	43.80%	60.10%	0:04:31
1	30	30	0	750	39.94%	56.24%	0:06:42
1	30	30	1	750	45.82%	61.18%	0:06:05
1	30	30	0	1000	40.66%	57.27%	0:08:12
1	30	30	1	1000	47.14%	62.30%	0:07:22
1	30	30	0	1500	41.67%	57.45%	0:10:55
1	30	30	1	1500	48.23%	63.24%	0:10:13
1	40	15	0	100	28.47%	44.57%	0:02:08
1	40	15	1	100	32.91%	49.51%	0:02:55
1	40	15	2	100	33.16%	50.81%	0:02:10
1	40	15	3	100	31.92%	48.07%	0:02:01
1	40	15	0	250	31.51%	47.94%	0:02:57
1	40	15	1	250	36.58%	53.73%	0:03:46
1	40	15	2	250	38.55%	54.62%	0:02:58
1	40	15	3	250	37.68%	54.31%	0:02:52
1	40	15	0	500	33.62%	49.37%	0:04:19
1	40	15	1	500	39.57%	56.91%	0:05:10
1	40	15	2	500	41.90%	57.90%	0:04:23
1	40	15	3	500	42.58%	58.84%	0:04:16

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	40	15	0	750	34.68%	50.99%	0:05:40
1	40	15	1	750	40.98%	57.09%	0:06:37
1	40	15	2	750	43.47%	59.38%	0:05:46
1	40	15	3	750	43.74%	59.65%	0:05:41
1	40	15	0	1000	35.31%	52.06%	0:07:05
1	40	15	1	1000	41.63%	57.14%	0:08:10
1	40	15	2	1000	43.91%	60.10%	0:07:10
1	40	15	3	1000	44.43%	60.95%	0:07:00
1	40	15	0	1500	35.83%	52.47%	0:09:53
1	40	15	1	1500	43.20%	59.16%	0:10:55
1	40	15	2	1500	45.41%	59.92%	0:09:55
1	40	15	3	1500	45.99%	61.85%	0:09:47
1	40	20	0	100	33.24%	49.37%	0:02:39
1	40	20	1	100	35.17%	51.39%	0:02:41
1	40	20	2	100	35.75%	53.41%	0:02:11
1	40	20	0	250	35.41%	52.06%	0:03:30
1	40	20	1	250	39.99%	55.75%	0:03:32
1	40	20	2	250	40.80%	57.72%	0:02:59
1	40	20	0	500	36.80%	53.19%	0:04:58
1	40	20	1	500	44.01%	60.19%	0:04:56
1	40	20	2	500	45.35%	61.31%	0:04:23
1	40	20	0	750	37.81%	53.95%	0:06:19
1	40	20	1	750	44.74%	60.32%	0:06:20
1	40	20	2	750	47.22%	63.38%	0:05:46

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	40	20	0	1000	38.77%	55.07%	0:07:47
1	40	20	1	1000	46.16%	61.49%	0:07:50
1	40	20	2	1000	48.66%	64.68%	0:07:10
1	40	20	0	1500	39.66%	55.97%	0:10:29
1	40	20	1	1500	46.73%	61.18%	0:10:35
1	40	20	2	1500	49.68%	65.17%	0:09:56
1	40	30	0	100	32.16%	49.01%	0:03:31
1	40	30	1	100	34.42%	50.90%	0:02:33
1	40	30	0	250	36.20%	53.01%	0:04:22
1	40	30	1	250	41.36%	57.81%	0:03:21
1	40	30	0	500	38.54%	55.21%	0:05:49
1	40	30	1	500	45.71%	60.64%	0:04:49
1	40	30	0	750	40.14%	56.46%	0:07:18
1	40	30	1	750	47.09%	62.43%	0:06:12
1	40	30	0	1000	40.89%	57.00%	0:08:44
1	40	30	1	1000	48.44%	63.11%	0:07:38
1	40	30	0	1500	41.92%	57.99%	0:11:33
1	40	30	1	1500	49.88%	64.68%	0:10:21
1	50	15	0	100	29.53%	45.74%	0:02:15
1	50	15	1	100	32.76%	48.79%	0:03:18
1	50	15	2	100	33.63%	51.44%	0:02:15
1	50	15	3	100	33.18%	50.13%	0:02:06
1	50	15	0	250	32.96%	49.64%	0:03:05
1	50	15	1	250	38.28%	54.98%	0:04:08

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	50	15	2	250	40.09%	56.78%	0:03:06
1	50	15	3	250	38.78%	55.79%	0:02:55
1	50	15	0	500	34.63%	50.22%	0:04:27
1	50	15	1	500	40.47%	56.55%	0:05:42
1	50	15	2	500	43.66%	59.25%	0:04:29
1	50	15	3	500	42.52%	59.43%	0:04:19
1	50	15	0	750	36.44%	52.78%	0:05:51
1	50	15	1	750	44.05%	60.05%	0:07:04
1	50	15	2	750	46.90%	62.30%	0:05:50
1	50	15	3	750	44.73%	60.77%	0:05:42
1	50	15	0	1000	36.54%	52.65%	0:07:11
1	50	15	1	1000	43.41%	58.93%	0:08:31
1	50	15	2	1000	47.39%	63.02%	0:07:15
1	50	15	3	1000	45.53%	61.98%	0:07:19
1	50	15	0	1500	37.03%	53.05%	0:09:56
1	50	15	1	1500	44.76%	59.56%	0:11:28
1	50	15	2	1500	47.98%	63.11%	0:10:01
1	50	15	3	1500	46.67%	62.03%	0:09:59
1	50	20	0	100	32.19%	48.70%	0:03:00
1	50	20	1	100	35.04%	51.93%	0:03:00
1	50	20	2	100	35.86%	53.46%	0:02:18
1	50	20	0	250	34.82%	51.53%	0:03:49
1	50	20	1	250	40.78%	57.68%	0:03:50
1	50	20	2	250	42.14%	59.02%	0:03:07

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	50	20	0	500	37.01%	52.69%	0:05:14
1	50	20	1	500	45.06%	60.73%	0:05:18
1	50	20	2	500	46.69%	62.75%	0:04:30
1	50	20	0	750	38.30%	54.58%	0:06:39
1	50	20	1	750	45.18%	60.23%	0:06:43
1	50	20	2	750	47.59%	63.60%	0:05:55
1	50	20	0	1000	38.21%	54.31%	0:08:03
1	50	20	1	1000	46.53%	61.80%	0:08:15
1	50	20	2	1000	48.88%	64.72%	0:07:18
1	50	20	0	1500	40.06%	56.42%	0:10:56
1	50	20	1	1500	48.83%	63.82%	0:11:01
1	50	20	2	1500	50.99%	66.65%	0:10:09
1	50	30	0	100	31.99%	48.92%	0:04:10
1	50	30	1	100	36.87%	53.32%	0:02:46
1	50	30	0	250	36.17%	52.56%	0:04:59
1	50	30	1	250	41.30%	57.72%	0:03:35
1	50	30	0	500	38.76%	54.85%	0:06:31
1	50	30	1	500	46.30%	61.45%	0:05:01
1	50	30	0	750	40.00%	56.06%	0:07:55
1	50	30	1	750	48.16%	62.48%	0:06:29
1	50	30	0	1000	41.09%	57.59%	0:09:35
1	50	30	1	1000	50.06%	64.95%	0:07:52
1	50	30	0	1500	41.84%	57.09%	0:12:29
1	50	30	1	1500	51.69%	66.11%	0:10:42

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	18	15	0	100	24.87%	40.08%	0:02:36
2	18	15	1	100	26.40%	41.92%	0:03:10
2	18	15	2	100	23.72%	37.48%	0:02:39
2	18	15	3	100	19.56%	33.62%	0:02:32
2	18	15	0	250	25.61%	40.22%	0:04:15
2	18	15	1	250	27.31%	42.32%	0:04:49
2	18	15	2	250	24.98%	39.59%	0:04:17
2	18	15	3	250	16.43%	28.41%	0:04:12
2	18	15	0	500	26.84%	41.20%	0:07:00
2	18	15	1	500	29.79%	43.94%	0:07:36
2	18	15	2	500	26.00%	39.90%	0:07:03
2	18	15	3	500	21.40%	35.86%	0:06:57
2	18	15	0	750	26.51%	41.70%	0:09:51
2	18	15	1	750	28.10%	41.92%	0:10:29
2	18	15	2	750	25.05%	39.81%	0:09:48
2	18	15	3	750	22.43%	36.45%	0:09:43
2	18	15	0	1000	27.36%	42.15%	0:12:25
2	18	15	1	1000	26.83%	39.41%	0:13:15
2	18	15	2	1000	23.69%	37.07%	0:12:36
2	18	15	3	1000	16.40%	28.14%	0:12:45
2	18	15	0	1500	26.11%	39.36%	0:17:50
2	18	15	1	1500	26.68%	39.72%	0:18:49
2	18	15	2	1500	22.48%	34.38%	0:18:05
2	18	15	3	1500	15.72%	26.39%	0:17:58

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	18	20	0	100	26.57%	41.29%	0:03:01
2	18	20	1	100	22.88%	36.31%	0:02:48
2	18	20	2	100	23.26%	36.89%	0:02:39
2	18	20	0	250	28.19%	43.49%	0:04:43
2	18	20	1	250	26.29%	38.87%	0:04:26
2	18	20	2	250	24.65%	38.06%	0:04:18
2	18	20	0	500	29.33%	43.99%	0:07:27
2	18	20	1	500	28.83%	42.46%	0:07:15
2	18	20	2	500	26.59%	39.63%	0:07:07
2	18	20	0	750	29.73%	44.25%	0:10:06
2	18	20	1	750	29.65%	43.31%	0:09:58
2	18	20	2	750	29.05%	43.63%	0:09:52
2	18	20	0	1000	30.53%	44.93%	0:12:54
2	18	20	1	1000	30.52%	43.85%	0:12:43
2	18	20	2	1000	28.45%	41.97%	0:12:39
2	18	20	0	1500	29.54%	43.85%	0:18:20
2	18	20	1	1500	27.58%	39.81%	0:18:21
2	18	20	2	1500	24.83%	37.43%	0:18:12
2	18	30	0	100	31.28%	47.04%	0:03:36
2	18	30	1	100	31.25%	46.77%	0:02:56
2	18	30	0	250	34.05%	49.82%	0:05:14
2	18	30	1	250	35.48%	50.81%	0:04:38
2	18	30	0	500	35.68%	51.44%	0:08:02
2	18	30	1	500	38.15%	52.78%	0:07:23

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	18	30	0	750	36.55%	51.89%	0:10:48
2	18	30	1	750	39.47%	54.62%	0:10:07
2	18	30	0	1000	36.10%	50.67%	0:13:31
2	18	30	1	1000	39.73%	54.13%	0:12:59
2	18	30	0	1500	36.80%	51.39%	0:19:06
2	18	30	1	1500	38.76%	53.19%	0:18:33
2	20	10	0	100	16.24%	29.17%	0:02:30
2	20	10	1	100	13.81%	25.04%	0:02:49
2	20	10	2	100	10.78%	19.88%	0:02:47
2	20	10	3	100	6.20%	13.46%	0:02:32
2	20	10	4	100	2.22%	5.92%	0:02:30
2	20	10	0	250	16.98%	28.73%	0:04:07
2	20	10	1	250	13.56%	24.91%	0:04:28
2	20	10	2	250	10.59%	19.57%	0:04:26
2	20	10	3	250	4.79%	11.85%	0:04:10
2	20	10	4	250	3.30%	9.07%	0:04:05
2	20	10	0	500	16.73%	29.40%	0:06:49
2	20	10	1	500	13.08%	23.34%	0:07:28
2	20	10	2	500	10.29%	19.17%	0:07:15
2	20	10	3	500	4.01%	11.31%	0:06:51
2	20	10	4	500	1.43%	4.58%	0:06:51
2	20	10	0	750	16.83%	28.50%	0:09:32
2	20	10	1	750	10.81%	19.03%	0:10:06
2	20	10	2	750	6.75%	13.46%	0:09:58

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	20	10	3	750	0.80%	2.42%	0:09:36
2	20	10	4	750	0.32%	0.40%	0:09:35
2	20	10	5	750	0.23%	0.40%	0:09:33
2	20	10	0	1000	17.91%	30.39%	0:12:17
2	20	10	1	1000	13.57%	24.15%	0:12:47
2	20	10	2	1000	10.41%	19.17%	0:12:42
2	20	10	3	1000	3.54%	9.65%	0:12:21
2	20	10	4	1000	0.95%	2.47%	0:12:18
2	20	10	5	1000	0.21%	0.22%	0:12:12
2	20	10	0	1500	15.46%	25.85%	0:17:39
2	20	10	1	1500	10.66%	20.11%	0:18:25
2	20	10	2	1500	7.42%	14.86%	0:18:14
2	20	10	3	1500	2.19%	7.27%	0:17:50
2	20	10	4	1500	0.46%	1.30%	0:17:40
2	20	10	5	1500	0.19%	0.18%	0:17:47
2	20	10	0	2000	14.91%	25.90%	0:23:04
2	20	10	1	2000	9.41%	18.00%	0:24:03
2	20	10	2	2000	6.39%	13.55%	0:23:37
2	20	10	3	2000	1.13%	2.69%	0:23:18
2	20	10	4	2000	0.27%	0.40%	0:23:11
2	20	10	5	2000	0.13%	0.04%	0:23:16
2	20	15	0	100	25.19%	39.27%	0:02:38
2	20	15	1	100	22.46%	36.04%	0:03:17
2	20	15	2	100	22.20%	35.73%	0:02:41

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	20	15	3	100	21.27%	35.64%	0:02:34
2	20	15	0	250	26.63%	41.43%	0:04:16
2	20	15	1	250	25.86%	39.63%	0:04:57
2	20	15	2	250	22.81%	35.23%	0:04:19
2	20	15	3	250	19.18%	32.09%	0:04:13
2	20	15	0	500	28.16%	43.49%	0:07:00
2	20	15	1	500	28.05%	42.32%	0:07:43
2	20	15	2	500	27.28%	41.25%	0:07:05
2	20	15	3	500	22.67%	37.30%	0:07:04
2	20	15	0	750	28.76%	43.09%	0:09:42
2	20	15	1	750	30.10%	44.57%	0:10:31
2	20	15	2	750	28.46%	42.77%	0:09:54
2	20	15	3	750	26.42%	41.34%	0:09:45
2	20	15	0	1000	29.73%	45.92%	0:12:32
2	20	15	1	1000	29.37%	43.90%	0:13:21
2	20	15	2	1000	27.83%	41.97%	0:12:36
2	20	15	3	1000	25.68%	40.66%	0:12:31
2	20	15	0	1500	28.23%	42.28%	0:17:50
2	20	15	1	1500	28.23%	41.29%	0:18:57
2	20	15	2	1500	23.47%	36.18%	0:17:57
2	20	15	3	1500	19.27%	31.78%	0:17:59
2	20	15	0	2000	27.54%	41.74%	0:23:15
2	20	15	1	2000	28.93%	41.92%	0:24:31
2	20	15	2	2000	22.99%	35.91%	0:23:46

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	20	15	3	2000	19.41%	32.41%	0:23:31
2	20	20	0	100	26.78%	41.43%	0:03:07
2	20	20	1	100	24.63%	38.42%	0:02:51
2	20	20	2	100	23.52%	36.13%	0:02:42
2	20	20	0	250	28.94%	43.63%	0:04:44
2	20	20	1	250	28.59%	43.00%	0:04:30
2	20	20	2	250	29.38%	42.24%	0:04:21
2	20	20	0	500	29.21%	42.86%	0:07:32
2	20	20	1	500	28.93%	42.06%	0:07:17
2	20	20	2	500	27.70%	40.31%	0:07:06
2	20	20	0	750	29.80%	44.39%	0:10:15
2	20	20	1	750	30.76%	44.08%	0:10:03
2	20	20	2	750	31.13%	45.83%	0:09:55
2	20	20	0	1000	30.36%	44.79%	0:12:57
2	20	20	1	1000	29.02%	42.01%	0:12:48
2	20	20	2	1000	30.61%	44.34%	0:12:42
2	20	20	0	1500	29.48%	43.13%	0:18:23
2	20	20	1	1500	28.12%	40.04%	0:18:23
2	20	20	2	1500	25.96%	38.51%	0:18:07
2	20	20	0	2000	29.23%	42.59%	0:23:54
2	20	20	1	2000	26.76%	38.87%	0:24:35
2	20	20	2	2000	24.80%	37.30%	0:23:46
2	20	30	0	100	32.53%	49.51%	0:03:42
2	20	30	1	100	33.46%	50.13%	0:03:01

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	20	30	0	250	35.30%	50.58%	0:05:26
2	20	30	1	250	38.02%	53.46%	0:04:41
2	20	30	0	500	37.15%	51.97%	0:08:09
2	20	30	1	500	41.25%	56.19%	0:07:28
2	20	30	0	750	37.51%	51.30%	0:10:58
2	20	30	1	750	41.55%	55.92%	0:10:16
2	20	30	0	1000	38.44%	52.60%	0:13:42
2	20	30	1	1000	42.34%	56.42%	0:13:07
2	20	30	0	1500	38.41%	51.80%	0:19:14
2	20	30	1	1500	41.03%	54.08%	0:18:39
2	20	30	0	2000	37.29%	50.45%	0:25:04
2	20	30	1	2000	38.82%	50.94%	0:24:18
2	25	15	0	100	27.26%	42.91%	0:02:45
2	25	15	1	100	28.32%	43.63%	0:03:35
2	25	15	2	100	27.62%	40.80%	0:02:49
2	25	15	3	100	26.92%	40.89%	0:02:40
2	25	15	0	250	30.61%	47.85%	0:04:24
2	25	15	1	250	33.79%	50.72%	0:05:15
2	25	15	2	250	34.26%	49.42%	0:04:27
2	25	15	3	250	32.17%	47.53%	0:04:20
2	25	15	0	500	32.44%	48.79%	0:07:06
2	25	15	1	500	35.77%	51.17%	0:08:03
2	25	15	2	500	37.01%	51.80%	0:07:12
2	25	15	3	500	35.82%	51.93%	0:07:10

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	25	15	0	750	33.94%	50.76%	0:09:49
2	25	15	1	750	38.60%	54.35%	0:10:51
2	25	15	2	750	40.01%	54.76%	0:10:13
2	25	15	3	750	37.65%	53.50%	0:09:53
2	25	15	0	1000	33.48%	49.64%	0:12:32
2	25	15	1	1000	37.61%	53.05%	0:13:46
2	25	15	2	1000	38.64%	53.82%	0:12:46
2	25	15	3	1000	37.09%	52.74%	0:12:39
2	25	15	0	1500	34.16%	50.22%	0:17:59
2	25	15	1	1500	38.03%	53.23%	0:19:17
2	25	15	2	1500	38.41%	52.96%	0:18:20
2	25	15	3	1500	33.88%	48.74%	0:18:18
2	25	20	0	100	30.47%	45.78%	0:03:23
2	25	20	1	100	32.19%	48.11%	0:03:24
2	25	20	2	100	32.05%	47.31%	0:02:49
2	25	20	0	250	33.51%	49.91%	0:05:00
2	25	20	1	250	38.11%	53.05%	0:05:04
2	25	20	2	250	40.52%	56.96%	0:04:28
2	25	20	0	500	35.44%	51.26%	0:07:53
2	25	20	1	500	41.61%	56.37%	0:07:50
2	25	20	2	500	44.68%	60.59%	0:07:14
2	25	20	0	750	35.59%	50.85%	0:10:32
2	25	20	1	750	40.88%	55.70%	0:10:37
2	25	20	2	750	42.85%	58.12%	0:10:00

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	25	20	0	1000	35.92%	51.03%	0:13:15
2	25	20	1	1000	42.04%	55.92%	0:13:44
2	25	20	2	1000	44.07%	59.52%	0:12:54
2	25	20	0	1500	34.98%	49.87%	0:18:50
2	25	20	1	1500	41.14%	55.39%	0:19:12
2	25	20	2	1500	42.04%	56.33%	0:18:23
2	25	30	0	100	36.85%	53.64%	0:04:09
2	25	30	1	100	37.96%	55.30%	0:03:12
2	25	30	0	250	40.06%	55.97%	0:05:57
2	25	30	1	250	43.97%	60.37%	0:04:53
2	25	30	0	500	41.49%	57.14%	0:08:36
2	25	30	1	500	46.04%	61.45%	0:07:39
2	25	30	0	750	42.80%	58.66%	0:11:31
2	25	30	1	750	48.56%	63.91%	0:10:27
2	25	30	0	1000	43.90%	58.84%	0:14:23
2	25	30	1	1000	49.86%	64.59%	0:13:17
2	25	30	0	1500	44.57%	59.43%	0:19:55
2	25	30	1	1500	50.94%	65.62%	0:18:56
2	30	15	0	100	29.38%	45.06%	0:02:52
2	30	15	1	100	32.72%	48.88%	0:03:53
2	30	15	2	100	32.02%	48.34%	0:02:54
2	30	15	3	100	30.63%	46.05%	0:02:43
2	30	15	0	250	32.05%	48.29%	0:04:31
2	30	15	1	250	36.87%	53.14%	0:05:35

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	30	15	2	250	37.97%	54.22%	0:04:35
2	30	15	3	250	37.57%	54.04%	0:04:27
2	30	15	0	500	34.08%	50.31%	0:07:22
2	30	15	1	500	39.83%	54.85%	0:08:25
2	30	15	2	500	40.77%	56.33%	0:07:20
2	30	15	3	500	39.54%	55.70%	0:07:10
2	30	15	0	750	35.23%	51.62%	0:09:58
2	30	15	1	750	39.32%	54.49%	0:11:16
2	30	15	2	750	41.59%	56.91%	0:10:06
2	30	15	3	750	39.81%	55.97%	0:09:54
2	30	15	0	1000	36.11%	52.29%	0:12:48
2	30	15	1	1000	41.78%	57.05%	0:14:04
2	30	15	2	1000	43.79%	58.39%	0:12:50
2	30	15	3	1000	41.65%	58.08%	0:12:47
2	30	15	0	1500	35.22%	50.40%	0:18:04
2	30	15	1	1500	39.91%	54.13%	0:19:47
2	30	15	2	1500	41.35%	56.37%	0:18:25
2	30	15	3	1500	37.92%	53.64%	0:18:20
2	30	20	0	100	32.58%	48.88%	0:03:36
2	30	20	1	100	36.23%	52.60%	0:03:39
2	30	20	2	100	35.71%	52.65%	0:02:55
2	30	20	0	250	35.17%	52.38%	0:05:18
2	30	20	1	250	41.69%	57.54%	0:05:18
2	30	20	2	250	43.95%	59.61%	0:04:36

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	30	20	0	500	37.17%	53.41%	0:08:00
2	30	20	1	500	44.44%	60.05%	0:08:10
2	30	20	2	500	47.44%	62.97%	0:07:24
2	30	20	0	750	37.79%	53.64%	0:10:53
2	30	20	1	750	46.37%	61.13%	0:11:00
2	30	20	2	750	48.96%	64.59%	0:10:07
2	30	20	0	1000	38.77%	54.53%	0:13:34
2	30	20	1	1000	46.88%	61.36%	0:13:45
2	30	20	2	1000	49.85%	65.22%	0:12:57
2	30	20	0	1500	37.87%	53.50%	0:19:16
2	30	20	1	1500	45.94%	60.41%	0:19:26
2	30	20	2	1500	49.92%	65.57%	0:18:29
2	30	30	0	100	36.33%	53.73%	0:04:39
2	30	30	1	100	39.18%	56.96%	0:03:25
2	30	30	0	250	40.66%	57.54%	0:06:25
2	30	30	1	250	45.68%	62.25%	0:05:06
2	30	30	0	500	42.90%	58.98%	0:09:12
2	30	30	1	500	48.68%	64.59%	0:07:51
2	30	30	0	750	44.43%	60.68%	0:12:13
2	30	30	1	750	50.75%	66.29%	0:10:40
2	30	30	0	1000	44.72%	60.19%	0:15:06
2	30	30	1	1000	51.72%	67.32%	0:13:31
2	30	30	0	1500	46.91%	62.21%	0:20:34
2	30	30	1	1500	53.29%	68.04%	0:19:05

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	40	15	0	100	32.25%	47.62%	0:03:07
2	40	15	1	100	36.61%	53.82%	0:04:31
2	40	15	2	100	36.68%	53.46%	0:03:07
2	40	15	3	100	34.99%	52.47%	0:02:52
2	40	15	0	250	34.77%	50.36%	0:04:44
2	40	15	1	250	41.83%	58.26%	0:06:13
2	40	15	2	250	42.77%	58.93%	0:04:46
2	40	15	3	250	42.38%	58.66%	0:04:33
2	40	15	0	500	36.44%	53.19%	0:07:29
2	40	15	1	500	43.43%	59.29%	0:09:05
2	40	15	2	500	46.09%	62.12%	0:07:33
2	40	15	3	500	45.71%	61.89%	0:07:20
2	40	15	0	750	38.45%	54.62%	0:10:14
2	40	15	1	750	44.29%	60.23%	0:11:55
2	40	15	2	750	47.57%	63.15%	0:10:21
2	40	15	3	750	47.31%	63.20%	0:10:06
2	40	15	0	1000	38.67%	54.44%	0:13:01
2	40	15	1	1000	45.34%	60.55%	0:14:50
2	40	15	2	1000	48.14%	62.66%	0:13:11
2	40	15	3	1000	47.06%	62.21%	0:12:58
2	40	15	0	1500	39.37%	55.12%	0:18:26
2	40	15	1	1500	46.14%	61.22%	0:20:36
2	40	15	2	1500	48.68%	63.55%	0:18:57
2	40	15	3	1500	47.96%	64.45%	0:18:31

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	40	20	0	100	34.43%	50.99%	0:04:07
2	40	20	1	100	39.70%	56.60%	0:04:07
2	40	20	2	100	39.72%	57.18%	0:03:08
2	40	20	0	250	37.85%	54.26%	0:05:49
2	40	20	1	250	46.06%	62.21%	0:05:51
2	40	20	2	250	47.13%	63.42%	0:04:49
2	40	20	0	500	40.01%	56.33%	0:08:36
2	40	20	1	500	48.67%	63.91%	0:08:40
2	40	20	2	500	50.29%	65.66%	0:07:44
2	40	20	0	750	40.98%	57.23%	0:11:30
2	40	20	1	750	50.03%	64.45%	0:11:29
2	40	20	2	750	51.45%	66.61%	0:10:20
2	40	20	0	1000	42.12%	58.08%	0:14:11
2	40	20	1	1000	51.20%	65.39%	0:14:22
2	40	20	2	1000	52.19%	67.06%	0:13:18
2	40	20	0	1500	42.89%	58.89%	0:19:53
2	40	20	1	1500	51.52%	64.99%	0:20:03
2	40	20	2	1500	53.36%	67.91%	0:19:03
2	40	30	0	100	37.21%	53.90%	0:05:39
2	40	30	1	100	40.11%	56.78%	0:03:49
2	40	30	0	250	40.52%	56.91%	0:07:26
2	40	30	1	250	47.81%	63.55%	0:05:31
2	40	30	0	500	43.07%	59.11%	0:10:20
2	40	30	1	500	51.33%	66.47%	0:08:20

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	40	30	0	750	44.97%	61.62%	0:13:12
2	40	30	1	750	53.22%	68.22%	0:11:10
2	40	30	0	1000	46.02%	62.12%	0:16:04
2	40	30	1	1000	54.40%	68.85%	0:14:09
2	40	30	0	1500	47.97%	63.96%	0:21:57
2	40	30	1	1500	55.90%	69.66%	0:19:37
2	50	15	0	100	32.88%	49.10%	0:03:22
2	50	15	1	100	38.19%	55.66%	0:05:13
2	50	15	2	100	39.58%	57.18%	0:03:20
2	50	15	3	100	38.09%	56.06%	0:03:00
2	50	15	0	250	35.74%	52.06%	0:05:03
2	50	15	1	250	43.45%	59.83%	0:07:01
2	50	15	2	250	44.65%	60.64%	0:05:03
2	50	15	3	250	42.43%	59.34%	0:04:42
2	50	15	0	500	38.88%	55.43%	0:07:42
2	50	15	1	500	46.39%	62.43%	0:09:53
2	50	15	2	500	49.67%	65.80%	0:07:46
2	50	15	3	500	46.30%	62.34%	0:07:27
2	50	15	0	750	39.74%	56.42%	0:10:33
2	50	15	1	750	47.56%	63.29%	0:12:47
2	50	15	2	750	51.05%	66.61%	0:10:32
2	50	15	3	750	47.26%	62.93%	0:10:14
2	50	15	0	1000	39.94%	55.88%	0:13:15
2	50	15	1	1000	48.46%	64.00%	0:15:49

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	50	15	2	1000	51.68%	66.88%	0:13:22
2	50	15	3	1000	47.34%	62.61%	0:13:00
2	50	15	0	1500	40.59%	56.42%	0:18:51
2	50	15	1	1500	47.91%	62.34%	0:21:44
2	50	15	2	1500	51.78%	66.88%	0:19:25
2	50	15	3	1500	48.57%	63.87%	0:18:42
2	50	20	0	100	34.81%	52.42%	0:04:41
2	50	20	1	100	40.27%	57.23%	0:04:40
2	50	20	2	100	41.54%	58.80%	0:03:21
2	50	20	0	250	38.44%	55.07%	0:06:27
2	50	20	1	250	46.10%	62.12%	0:06:23
2	50	20	2	250	47.78%	64.09%	0:05:03
2	50	20	0	500	40.35%	56.42%	0:09:15
2	50	20	1	500	49.61%	64.95%	0:09:15
2	50	20	2	500	51.39%	67.24%	0:07:50
2	50	20	0	750	41.73%	58.08%	0:12:07
2	50	20	1	750	51.04%	65.62%	0:12:05
2	50	20	2	750	52.16%	67.73%	0:10:51
2	50	20	0	1000	42.54%	58.35%	0:14:54
2	50	20	1	1000	51.72%	66.07%	0:14:56
2	50	20	2	1000	53.63%	67.77%	0:13:45
2	50	20	0	1500	43.36%	59.16%	0:20:40
2	50	20	1	1500	52.80%	66.79%	0:20:47
2	50	20	2	1500	54.90%	69.30%	0:19:01

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	50	30	0	100	35.93%	52.96%	0:06:48
2	50	30	1	100	42.12%	59.38%	0:04:16
2	50	30	0	250	40.29%	56.91%	0:08:33
2	50	30	1	250	48.37%	63.73%	0:05:56
2	50	30	0	500	43.57%	60.01%	0:11:32
2	50	30	1	500	53.16%	67.46%	0:08:42
2	50	30	0	750	45.63%	60.86%	0:14:45
2	50	30	1	750	54.93%	68.99%	0:11:39
2	50	30	0	1000	46.70%	62.48%	0:17:42
2	50	30	1	1000	56.20%	70.69%	0:14:20
2	50	30	0	1500	47.87%	63.33%	0:23:21
2	50	30	1	1500	57.40%	71.50%	0:20:06
3	18	15	0	100	24.06%	37.66%	0:03:18
3	18	15	1	100	24.96%	39.41%	0:04:08
3	18	15	2	100	23.32%	36.71%	0:03:23
3	18	15	3	100	18.63%	32.23%	0:03:14
3	18	15	0	250	26.61%	40.08%	0:05:51
3	18	15	1	250	29.36%	43.63%	0:06:39
3	18	15	2	250	27.04%	41.25%	0:05:56
3	18	15	3	250	20.89%	33.75%	0:05:44
3	18	15	0	500	28.28%	42.50%	0:09:53
3	18	15	1	500	30.21%	43.90%	0:10:49
3	18	15	2	500	26.02%	40.53%	0:09:58
3	18	15	3	500	19.48%	32.76%	0:09:52

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	18	15	0	750	27.43%	42.10%	0:13:54
3	18	15	1	750	28.16%	41.38%	0:15:14
3	18	15	2	750	25.71%	39.68%	0:14:07
3	18	15	3	750	18.75%	31.96%	0:14:00
3	18	15	0	1000	26.58%	40.44%	0:18:06
3	18	15	1	1000	27.20%	40.08%	0:19:11
3	18	15	2	1000	22.66%	35.19%	0:18:16
3	18	15	3	1000	17.65%	29.49%	0:18:14
3	18	15	0	1500	25.15%	38.06%	0:26:08
3	18	15	1	1500	23.84%	35.91%	0:27:46
3	18	15	2	1500	15.66%	26.30%	0:26:19
3	18	15	3	1500	8.79%	16.92%	0:26:14
3	18	20	0	100	27.60%	43.18%	0:03:56
3	18	20	1	100	25.18%	39.00%	0:03:36
3	18	20	2	100	24.31%	36.76%	0:03:25
3	18	20	0	250	29.61%	45.20%	0:06:24
3	18	20	1	250	27.86%	41.52%	0:06:04
3	18	20	2	250	25.83%	38.42%	0:05:53
3	18	20	0	500	29.81%	43.67%	0:10:33
3	18	20	1	500	28.83%	42.32%	0:10:16
3	18	20	2	500	30.15%	44.88%	0:10:01
3	18	20	0	750	29.84%	43.67%	0:14:41
3	18	20	1	750	29.62%	42.32%	0:14:26
3	18	20	2	750	29.29%	43.09%	0:14:16

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	18	20	0	1000	30.28%	44.57%	0:18:39
3	18	20	1	1000	29.37%	42.06%	0:18:30
3	18	20	2	1000	22.95%	34.47%	0:18:22
3	18	20	0	1500	29.93%	43.94%	0:26:56
3	18	20	1	1500	24.25%	36.13%	0:27:08
3	18	20	2	1500	21.22%	33.80%	0:26:30
3	18	30	0	100	32.26%	47.80%	0:04:46
3	18	30	1	100	32.78%	48.11%	0:03:50
3	18	30	0	250	35.61%	51.12%	0:07:16
3	18	30	1	250	38.46%	54.17%	0:06:20
3	18	30	0	500	36.68%	51.71%	0:11:34
3	18	30	1	500	39.13%	54.22%	0:10:31
3	18	30	0	750	37.11%	51.75%	0:15:32
3	18	30	1	750	39.69%	53.82%	0:14:44
3	18	30	0	1000	37.85%	51.93%	0:19:46
3	18	30	1	1000	38.94%	52.87%	0:18:46
3	18	30	0	1500	36.86%	50.85%	0:28:07
3	18	30	1	1500	38.73%	53.50%	0:27:11
3	20	10	0	100	17.25%	30.97%	0:03:10
3	20	10	1	100	15.06%	26.03%	0:03:40
3	20	10	2	100	13.02%	23.07%	0:03:37
3	20	10	3	100	8.35%	17.01%	0:03:12
3	20	10	4	100	5.09%	12.03%	0:03:10
3	20	10	0	250	16.44%	28.28%	0:05:37

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	20	10	1	250	13.55%	23.83%	0:06:07
3	20	10	2	250	11.23%	20.60%	0:06:04
3	20	10	3	250	3.64%	10.95%	0:05:39
3	20	10	4	250	2.19%	7.23%	0:05:35
3	20	10	5	250	0.41%	0.99%	0:05:27
3	20	10	0	500	16.31%	28.41%	0:09:43
3	20	10	1	500	12.44%	22.22%	0:10:21
3	20	10	2	500	8.71%	17.06%	0:10:20
3	20	10	3	500	2.71%	7.36%	0:09:44
3	20	10	4	500	0.40%	0.67%	0:09:39
3	20	10	5	500	0.23%	0.27%	0:09:40
3	20	10	0	750	15.37%	26.80%	0:13:47
3	20	10	1	750	9.56%	18.13%	0:14:30
3	20	10	2	750	6.89%	13.78%	0:14:26
3	20	10	3	750	1.72%	4.35%	0:13:51
3	20	10	4	750	0.27%	0.31%	0:13:48
3	20	10	5	750	0.18%	0.27%	0:13:39
3	20	10	0	1000	14.76%	26.03%	0:17:52
3	20	10	1	1000	9.33%	17.64%	0:18:42
3	20	10	2	1000	6.74%	13.87%	0:18:34
3	20	10	3	1000	1.94%	5.92%	0:18:00
3	20	10	4	1000	0.34%	0.36%	0:17:47
3	20	10	5	1000	0.17%	0.18%	0:17:46
3	20	10	0	1500	14.98%	25.90%	0:25:53

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	20	10	1	1500	9.96%	18.04%	0:27:00
3	20	10	2	1500	5.99%	12.84%	0:26:54
3	20	10	3	1500	1.25%	3.37%	0:26:18
3	20	10	4	1500	0.21%	0.22%	0:25:53
3	20	10	5	1500	0.14%	0.13%	0:26:10
3	20	10	0	2000	12.38%	21.86%	0:34:08
3	20	10	1	2000	6.23%	12.75%	0:35:57
3	20	10	2	2000	1.95%	5.43%	0:35:13
3	20	10	3	2000	0.19%	0.04%	0:34:30
3	20	10	4	2000	0.13%	0.09%	0:34:05
3	20	10	5	2000	0.11%	0.00%	0:34:24
3	20	15	0	100	26.02%	40.93%	0:03:22
3	20	15	1	100	25.03%	39.63%	0:04:18
3	20	15	2	100	23.58%	36.89%	0:03:26
3	20	15	3	100	19.16%	31.42%	0:03:15
3	20	15	0	250	28.60%	44.03%	0:05:51
3	20	15	1	250	28.84%	43.40%	0:06:52
3	20	15	2	250	26.79%	40.62%	0:05:53
3	20	15	3	250	24.57%	38.78%	0:05:45
3	20	15	0	500	29.04%	44.79%	0:09:53
3	20	15	1	500	30.50%	44.17%	0:11:02
3	20	15	2	500	28.85%	43.81%	0:10:00
3	20	15	3	500	26.66%	42.55%	0:09:52
3	20	15	0	750	29.42%	44.39%	0:13:58

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	20	15	1	750	31.15%	45.69%	0:15:11
3	20	15	2	750	26.66%	40.22%	0:14:11
3	20	15	3	750	23.41%	37.66%	0:14:00
3	20	15	0	1000	29.51%	44.25%	0:18:04
3	20	15	1	1000	28.36%	42.01%	0:19:20
3	20	15	2	1000	23.78%	35.86%	0:18:18
3	20	15	3	1000	20.85%	33.71%	0:18:14
3	20	15	0	1500	26.06%	39.77%	0:26:12
3	20	15	1	1500	24.68%	37.12%	0:27:48
3	20	15	2	1500	14.59%	25.22%	0:26:15
3	20	15	3	1500	8.24%	15.84%	0:26:53
3	20	15	0	2000	25.05%	38.87%	0:34:25
3	20	15	1	2000	24.33%	36.49%	0:36:06
3	20	15	2	2000	15.71%	26.03%	0:34:44
3	20	15	3	2000	10.36%	18.81%	0:34:38
3	20	20	0	100	28.10%	42.86%	0:04:09
3	20	20	1	100	25.68%	39.45%	0:03:41
3	20	20	2	100	24.85%	38.29%	0:03:27
3	20	20	0	250	29.60%	44.17%	0:06:34
3	20	20	1	250	31.11%	46.05%	0:06:11
3	20	20	2	250	32.93%	47.40%	0:05:58
3	20	20	0	500	30.84%	45.29%	0:10:39
3	20	20	1	500	30.57%	43.85%	0:10:17
3	20	20	2	500	30.00%	43.54%	0:10:07

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	20	20	0	750	30.69%	45.20%	0:14:51
3	20	20	1	750	29.02%	41.20%	0:14:26
3	20	20	2	750	31.04%	45.24%	0:14:11
3	20	20	0	1000	29.96%	44.03%	0:19:01
3	20	20	1	1000	28.18%	40.71%	0:18:59
3	20	20	2	1000	28.53%	41.47%	0:18:35
3	20	20	0	1500	28.92%	42.24%	0:27:14
3	20	20	1	1500	26.57%	37.88%	0:26:54
3	20	20	2	1500	22.53%	33.89%	0:26:37
3	20	20	0	2000	27.60%	40.26%	0:35:28
3	20	20	1	2000	21.78%	31.01%	0:35:20
3	20	20	2	2000	17.48%	27.56%	0:35:04
3	20	30	0	100	34.12%	49.01%	0:05:01
3	20	30	1	100	35.82%	51.12%	0:03:56
3	20	30	0	250	36.99%	52.47%	0:07:32
3	20	30	1	250	39.83%	54.80%	0:06:24
3	20	30	0	500	38.76%	53.82%	0:11:39
3	20	30	1	500	42.72%	57.68%	0:10:36
3	20	30	0	750	39.61%	54.85%	0:15:48
3	20	30	1	750	42.98%	57.63%	0:14:49
3	20	30	0	1000	39.30%	53.32%	0:20:09
3	20	30	1	1000	41.87%	55.30%	0:19:08
3	20	30	0	1500	38.55%	51.48%	0:28:22
3	20	30	1	1500	40.75%	54.31%	0:27:12

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	20	30	0	2000	37.02%	49.78%	0:36:43
3	20	30	1	2000	36.21%	48.38%	0:35:46
3	25	15	0	100	29.42%	44.88%	0:03:35
3	25	15	1	100	31.96%	48.43%	0:04:48
3	25	15	2	100	31.57%	46.36%	0:03:36
3	25	15	3	100	29.45%	43.49%	0:03:25
3	25	15	0	250	31.76%	47.89%	0:06:03
3	25	15	1	250	35.51%	51.30%	0:07:16
3	25	15	2	250	36.24%	51.57%	0:06:09
3	25	15	3	250	35.10%	51.71%	0:05:52
3	25	15	0	500	34.34%	50.31%	0:10:05
3	25	15	1	500	37.29%	53.82%	0:11:27
3	25	15	2	500	39.07%	54.94%	0:10:14
3	25	15	3	500	37.02%	54.17%	0:10:10
3	25	15	0	750	34.46%	50.76%	0:14:20
3	25	15	1	750	38.29%	54.35%	0:15:42
3	25	15	2	750	39.35%	54.71%	0:14:26
3	25	15	3	750	35.63%	51.44%	0:14:09
3	25	15	0	1000	33.94%	50.31%	0:18:14
3	25	15	1	1000	38.99%	55.66%	0:20:01
3	25	15	2	1000	37.64%	52.96%	0:18:37
3	25	15	3	1000	31.95%	47.08%	0:18:25
3	25	15	0	1500	33.03%	48.16%	0:26:26
3	25	15	1	1500	36.97%	52.87%	0:28:26

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	25	15	2	1500	34.49%	49.60%	0:26:58
3	25	15	3	1500	28.30%	43.22%	0:26:40
3	25	20	0	100	31.99%	49.10%	0:04:30
3	25	20	1	100	36.27%	52.06%	0:04:28
3	25	20	2	100	36.55%	52.47%	0:03:39
3	25	20	0	250	33.94%	49.60%	0:06:56
3	25	20	1	250	39.19%	54.85%	0:06:59
3	25	20	2	250	42.03%	57.23%	0:06:08
3	25	20	0	500	35.31%	50.45%	0:11:02
3	25	20	1	500	41.80%	56.64%	0:11:10
3	25	20	2	500	44.79%	60.28%	0:10:23
3	25	20	0	750	36.13%	52.02%	0:15:11
3	25	20	1	750	42.12%	55.83%	0:15:23
3	25	20	2	750	46.45%	62.12%	0:14:28
3	25	20	0	1000	35.35%	50.45%	0:19:31
3	25	20	1	1000	40.98%	55.25%	0:19:38
3	25	20	2	1000	40.69%	55.61%	0:18:40
3	25	20	0	1500	35.67%	49.78%	0:27:31
3	25	20	1	1500	41.06%	55.12%	0:28:17
3	25	20	2	1500	39.60%	54.17%	0:26:56
3	25	30	0	100	39.01%	54.80%	0:05:41
3	25	30	1	100	40.99%	57.90%	0:04:13
3	25	30	0	250	41.61%	56.91%	0:08:11
3	25	30	1	250	46.55%	62.97%	0:06:44

Table 3: Extended results for **R^{2}MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	25	30	0	500	44.02%	58.93%	0:12:24
3	25	30	1	500	50.38%	65.44%	0:10:56
3	25	30	0	750	45.60%	60.64%	0:17:01
3	25	30	1	750	52.96%	68.04%	0:15:08
3	25	30	0	1000	46.28%	61.18%	0:20:48
3	25	30	1	1000	52.58%	67.28%	0:19:16
3	25	30	0	1500	46.13%	61.49%	0:29:15
3	25	30	1	1500	52.61%	67.15%	0:27:37
3	30	15	0	100	30.54%	46.01%	0:03:47
3	30	15	1	100	35.40%	51.66%	0:05:15
3	30	15	2	100	34.29%	50.94%	0:03:45
3	30	15	3	100	33.66%	50.31%	0:03:30
3	30	15	0	250	34.10%	50.09%	0:06:13
3	30	15	1	250	39.49%	55.75%	0:07:47
3	30	15	2	250	41.39%	57.90%	0:06:16
3	30	15	3	250	40.67%	57.81%	0:06:01
3	30	15	0	500	35.35%	51.30%	0:10:19
3	30	15	1	500	41.17%	56.78%	0:11:57
3	30	15	2	500	43.26%	58.84%	0:10:26
3	30	15	3	500	40.96%	57.59%	0:10:12
3	30	15	0	750	36.74%	51.84%	0:14:36
3	30	15	1	750	41.86%	57.50%	0:16:18
3	30	15	2	750	43.94%	59.11%	0:14:33
3	30	15	3	750	41.66%	58.17%	0:14:18

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	30	15	0	1000	36.75%	51.84%	0:18:38
3	30	15	1	1000	43.25%	58.12%	0:20:22
3	30	15	2	1000	45.25%	60.50%	0:18:39
3	30	15	3	1000	43.27%	59.52%	0:18:29
3	30	15	0	1500	36.68%	52.15%	0:26:58
3	30	15	1	1500	41.14%	56.06%	0:29:09
3	30	15	2	1500	42.80%	57.76%	0:26:53
3	30	15	3	1500	36.24%	50.76%	0:26:53
3	30	20	0	100	34.28%	51.17%	0:04:51
3	30	20	1	100	38.62%	55.39%	0:04:50
3	30	20	2	100	39.14%	55.43%	0:03:47
3	30	20	0	250	36.85%	53.01%	0:07:19
3	30	20	1	250	44.87%	60.23%	0:07:22
3	30	20	2	250	46.72%	62.75%	0:06:18
3	30	20	0	500	39.53%	55.52%	0:11:26
3	30	20	1	500	47.13%	62.75%	0:11:38
3	30	20	2	500	49.63%	65.22%	0:10:36
3	30	20	0	750	40.17%	56.46%	0:15:56
3	30	20	1	750	49.12%	63.33%	0:15:46
3	30	20	2	750	52.24%	67.46%	0:14:37
3	30	20	0	1000	40.45%	56.10%	0:20:05
3	30	20	1	1000	49.29%	64.23%	0:20:03
3	30	20	2	1000	51.77%	65.98%	0:18:43
3	30	20	0	1500	39.87%	55.12%	0:28:24

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	30	20	1	1500	48.06%	62.70%	0:28:24
3	30	20	2	1500	50.53%	65.84%	0:27:04
3	30	30	0	100	38.87%	55.66%	0:06:23
3	30	30	1	100	41.57%	57.72%	0:04:33
3	30	30	0	250	43.55%	60.14%	0:08:56
3	30	30	1	250	48.46%	64.18%	0:07:02
3	30	30	0	500	45.32%	61.45%	0:13:13
3	30	30	1	500	51.53%	67.28%	0:11:17
3	30	30	0	750	47.55%	62.07%	0:17:45
3	30	30	1	750	53.16%	67.91%	0:15:22
3	30	30	0	1000	48.30%	64.09%	0:22:07
3	30	30	1	1000	54.89%	69.17%	0:19:36
3	30	30	0	1500	48.97%	64.00%	0:30:56
3	30	30	1	1500	56.20%	70.38%	0:28:04
3	40	15	0	100	33.19%	49.01%	0:04:08
3	40	15	1	100	38.74%	56.28%	0:06:07
3	40	15	2	100	39.13%	56.01%	0:04:04
3	40	15	3	100	37.33%	54.08%	0:03:41
3	40	15	0	250	36.82%	53.23%	0:06:37
3	40	15	1	250	44.08%	60.86%	0:08:41
3	40	15	2	250	46.22%	61.76%	0:06:31
3	40	15	3	250	44.96%	61.13%	0:06:11
3	40	15	0	500	38.74%	55.48%	0:10:44
3	40	15	1	500	46.47%	62.16%	0:13:05

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	40	15	2	500	48.63%	64.36%	0:10:42
3	40	15	3	500	47.38%	63.96%	0:10:22
3	40	15	0	750	40.40%	56.69%	0:14:58
3	40	15	1	750	47.70%	62.97%	0:17:13
3	40	15	2	750	50.03%	65.35%	0:14:48
3	40	15	3	750	48.78%	64.68%	0:14:29
3	40	15	0	1000	40.83%	56.96%	0:19:02
3	40	15	1	1000	47.62%	62.07%	0:21:26
3	40	15	2	1000	50.07%	65.26%	0:19:14
3	40	15	3	1000	47.70%	64.14%	0:18:44
3	40	15	0	1500	41.26%	57.14%	0:27:15
3	40	15	1	1500	48.75%	64.05%	0:30:11
3	40	15	2	1500	50.65%	65.26%	0:27:27
3	40	15	3	1500	45.97%	60.95%	0:27:11
3	40	20	0	100	36.77%	54.08%	0:05:43
3	40	20	1	100	42.50%	58.98%	0:05:34
3	40	20	2	100	43.04%	59.65%	0:04:06
3	40	20	0	250	40.61%	57.81%	0:08:19
3	40	20	1	250	48.81%	64.72%	0:08:04
3	40	20	2	250	49.16%	65.04%	0:06:34
3	40	20	0	500	42.70%	59.20%	0:12:27
3	40	20	1	500	51.14%	66.34%	0:12:31
3	40	20	2	500	52.33%	67.32%	0:10:44
3	40	20	0	750	43.73%	59.78%	0:16:43

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	40	20	1	750	52.95%	67.19%	0:16:38
3	40	20	2	750	53.59%	67.95%	0:14:57
3	40	20	0	1000	43.79%	59.25%	0:20:59
3	40	20	1	1000	53.42%	67.82%	0:20:56
3	40	20	2	1000	54.08%	68.22%	0:19:08
3	40	20	0	1500	44.15%	59.52%	0:29:27
3	40	20	1	1500	53.56%	67.59%	0:29:31
3	40	20	2	1500	53.49%	67.32%	0:27:18
3	40	30	0	100	38.67%	56.33%	0:08:10
3	40	30	1	100	43.86%	60.86%	0:05:06
3	40	30	0	250	43.27%	59.47%	0:10:49
3	40	30	1	250	50.16%	65.98%	0:07:40
3	40	30	0	500	46.66%	62.97%	0:15:14
3	40	30	1	500	53.76%	69.34%	0:11:50
3	40	30	0	750	48.74%	64.59%	0:19:56
3	40	30	1	750	56.35%	70.83%	0:16:07
3	40	30	0	1000	49.41%	64.99%	0:24:11
3	40	30	1	1000	57.43%	72.08%	0:20:20
3	40	30	0	1500	51.42%	65.62%	0:33:02
3	40	30	1	1500	59.22%	72.35%	0:28:34
3	50	15	0	100	34.60%	50.58%	0:04:28
3	50	15	1	100	41.12%	57.76%	0:07:10
3	50	15	2	100	42.90%	60.10%	0:04:23
3	50	15	3	100	41.12%	58.98%	0:03:55

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	50	15	0	250	37.79%	54.08%	0:06:59
3	50	15	1	250	45.66%	61.89%	0:09:48
3	50	15	2	250	48.41%	64.27%	0:06:52
3	50	15	3	250	45.29%	61.67%	0:06:23
3	50	15	0	500	41.08%	57.05%	0:11:11
3	50	15	1	500	48.26%	63.91%	0:14:09
3	50	15	2	500	52.19%	67.37%	0:11:09
3	50	15	3	500	48.47%	64.68%	0:10:34
3	50	15	0	750	41.98%	57.36%	0:15:25
3	50	15	1	750	49.77%	64.23%	0:18:53
3	50	15	2	750	52.46%	66.88%	0:15:10
3	50	15	3	750	47.63%	62.66%	0:14:49
3	50	15	0	1000	42.63%	58.80%	0:19:32
3	50	15	1	1000	50.48%	65.66%	0:22:55
3	50	15	2	1000	53.51%	68.58%	0:19:24
3	50	15	3	1000	46.94%	62.61%	0:18:57
3	50	15	0	1500	42.52%	58.12%	0:27:48
3	50	15	1	1500	50.36%	65.08%	0:31:31
3	50	15	2	1500	54.05%	68.40%	0:27:33
3	50	15	3	1500	44.49%	61.04%	0:27:18
3	50	20	0	100	37.42%	54.13%	0:06:38
3	50	20	1	100	43.08%	60.55%	0:06:20
3	50	20	2	100	44.24%	60.77%	0:04:26
3	50	20	0	250	40.52%	57.00%	0:09:13

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	50	20	1	250	49.04%	64.54%	0:08:53
3	50	20	2	250	50.38%	66.11%	0:06:54
3	50	20	0	500	42.75%	59.29%	0:13:28
3	50	20	1	500	51.41%	65.75%	0:13:05
3	50	20	2	500	53.01%	68.00%	0:11:09
3	50	20	0	750	44.05%	60.28%	0:17:44
3	50	20	1	750	53.72%	68.27%	0:17:38
3	50	20	2	750	54.45%	68.90%	0:15:20
3	50	20	0	1000	45.12%	60.64%	0:22:09
3	50	20	1	1000	54.09%	68.49%	0:21:51
3	50	20	2	1000	55.24%	69.39%	0:19:39
3	50	20	0	1500	45.55%	61.36%	0:30:49
3	50	20	1	1500	55.62%	68.94%	0:30:21
3	50	20	2	1500	55.62%	69.25%	0:27:54
3	50	30	0	100	38.05%	55.30%	0:09:55
3	50	30	1	100	45.03%	61.49%	0:05:43
3	50	30	0	250	43.87%	59.96%	0:12:36
3	50	30	1	250	51.76%	67.32%	0:08:13
3	50	30	0	500	46.26%	62.43%	0:17:18
3	50	30	1	500	55.24%	70.11%	0:12:31
3	50	30	0	750	49.40%	64.54%	0:22:05
3	50	30	1	750	58.08%	71.99%	0:16:45
3	50	30	0	1000	49.77%	64.90%	0:26:45
3	50	30	1	1000	58.77%	72.35%	0:21:13

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	50	30	0	1500	51.41%	65.80%	0:36:03
3	50	30	1	1500	59.67%	72.85%	0:29:25
4	18	15	0	100	26.77%	41.34%	0:04:03
4	18	15	1	100	27.51%	42.28%	0:05:12
4	18	15	2	100	24.96%	38.87%	0:04:08
4	18	15	3	100	19.86%	32.72%	0:03:54
4	18	15	0	250	27.66%	42.24%	0:07:19
4	18	15	1	250	28.22%	41.65%	0:08:30
4	18	15	2	250	25.39%	38.73%	0:07:23
4	18	15	3	250	19.24%	32.14%	0:07:14
4	18	15	0	500	27.76%	41.97%	0:12:43
4	18	15	1	500	30.30%	44.08%	0:14:08
4	18	15	2	500	26.58%	40.44%	0:13:01
4	18	15	3	500	20.29%	33.53%	0:12:48
4	18	15	0	750	26.85%	39.68%	0:18:14
4	18	15	1	750	27.09%	39.81%	0:19:47
4	18	15	2	750	20.16%	32.59%	0:18:24
4	18	15	3	750	14.97%	25.04%	0:18:09
4	18	15	0	1000	26.43%	40.26%	0:23:44
4	18	15	1	1000	26.07%	39.41%	0:25:12
4	18	15	2	1000	17.04%	27.74%	0:24:01
4	18	15	3	1000	10.09%	17.95%	0:23:40
4	18	15	0	1500	23.45%	36.31%	0:34:37
4	18	15	1	1500	20.41%	31.64%	0:36:24

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
4	18	15	2	1500	7.99%	14.81%	0:34:32
4	18	15	3	1500	1.06%	2.87%	0:34:31
4	18	20	0	100	28.65%	43.90%	0:04:52
4	18	20	1	100	26.43%	39.68%	0:04:25
4	18	20	2	100	27.42%	40.93%	0:04:10
4	18	20	0	250	30.71%	45.87%	0:08:10
4	18	20	1	250	30.54%	44.84%	0:07:47
4	18	20	2	250	32.20%	47.49%	0:07:28
4	18	20	0	500	30.91%	45.56%	0:13:35
4	18	20	1	500	30.10%	43.67%	0:13:16
4	18	20	2	500	30.53%	44.75%	0:13:00
4	18	20	0	750	29.49%	43.18%	0:19:05
4	18	20	1	750	27.18%	38.64%	0:18:47
4	18	20	2	750	25.15%	38.15%	0:18:34
4	18	20	0	1000	30.38%	44.17%	0:24:37
4	18	20	1	1000	27.30%	39.05%	0:24:29
4	18	20	2	1000	22.41%	34.69%	0:23:59
4	18	20	0	1500	28.18%	40.98%	0:35:30
4	18	20	1	1500	22.28%	34.11%	0:35:25
4	18	20	2	1500	14.07%	23.38%	0:35:00
4	18	30	0	100	33.36%	48.97%	0:06:01
4	18	30	1	100	33.72%	49.28%	0:04:44
4	18	30	0	250	36.72%	52.06%	0:09:24
4	18	30	1	250	38.40%	53.68%	0:08:00

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
4	18	30	0	500	37.97%	53.05%	0:14:45
4	18	30	1	500	40.53%	55.30%	0:13:31
4	18	30	0	750	37.75%	52.87%	0:20:24
4	18	30	1	750	40.74%	56.46%	0:19:13
4	18	30	0	1000	38.57%	52.96%	0:25:47
4	18	30	1	1000	39.85%	55.12%	0:24:50
4	18	30	0	1500	36.02%	48.79%	0:37:02
4	18	30	1	1500	33.42%	46.54%	0:35:52
4	20	10	0	100	16.97%	28.59%	0:03:51
4	20	10	1	100	14.02%	24.15%	0:04:28
4	20	10	2	100	12.09%	21.54%	0:04:25
4	20	10	3	100	6.03%	14.23%	0:03:53
4	20	10	4	100	2.86%	8.44%	0:03:50
4	20	10	5	100	0.69%	1.84%	0:03:41
4	20	10	0	250	16.73%	27.92%	0:07:08
4	20	10	1	250	12.94%	22.31%	0:07:50
4	20	10	2	250	8.31%	15.44%	0:07:44
4	20	10	3	250	1.71%	4.22%	0:07:10
4	20	10	4	250	0.40%	0.63%	0:07:03
4	20	10	5	250	0.19%	0.31%	0:06:51
4	20	10	0	500	17.26%	29.22%	0:12:40
4	20	10	1	500	13.07%	23.20%	0:13:20
4	20	10	2	500	10.07%	18.58%	0:13:10
4	20	10	3	500	4.43%	11.58%	0:12:38

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
4	20	10	4	500	1.03%	2.69%	0:12:26
4	20	10	5	500	0.17%	0.09%	0:12:18
4	20	10	0	750	14.53%	25.67%	0:17:55
4	20	10	1	750	9.97%	19.43%	0:19:32
4	20	10	2	750	6.49%	13.55%	0:18:48
4	20	10	3	750	1.37%	3.99%	0:18:01
4	20	10	4	750	0.25%	0.22%	0:17:53
4	20	10	5	750	0.13%	0.13%	0:18:22
4	20	10	0	1000	13.69%	25.09%	0:23:26
4	20	10	1	1000	8.56%	16.88%	0:24:30
4	20	10	2	1000	5.68%	12.43%	0:24:24
4	20	10	3	1000	1.34%	4.35%	0:23:32
4	20	10	4	1000	0.17%	0.13%	0:23:21
4	20	10	5	1000	0.13%	0.09%	0:23:25
4	20	10	0	1500	13.90%	25.81%	0:34:14
4	20	10	1	1500	7.83%	15.22%	0:35:26
4	20	10	2	1500	3.35%	8.35%	0:35:40
4	20	10	3	1500	0.28%	0.18%	0:34:30
4	20	10	4	1500	0.19%	0.13%	0:34:30
4	20	10	5	1500	0.15%	0.09%	0:34:09
4	20	10	0	2000	12.43%	22.71%	0:45:16
4	20	10	1	2000	5.13%	10.77%	0:46:32
4	20	10	2	2000	1.58%	4.40%	0:46:30
4	20	10	3	2000	0.20%	0.18%	0:45:32

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
4	20	10	4	2000	0.17%	0.04%	0:46:25
4	20	10	5	2000	0.14%	0.22%	0:45:23
4	20	15	0	100	26.16%	41.02%	0:04:09
4	20	15	1	100	25.50%	40.26%	0:05:23
4	20	15	2	100	22.24%	34.52%	0:04:10
4	20	15	3	100	19.01%	31.51%	0:03:56
4	20	15	0	250	29.24%	45.24%	0:07:25
4	20	15	1	250	27.99%	42.37%	0:08:42
4	20	15	2	250	26.72%	41.20%	0:07:28
4	20	15	3	250	24.11%	39.18%	0:07:14
4	20	15	0	500	30.28%	46.10%	0:12:50
4	20	15	1	500	29.62%	44.43%	0:14:17
4	20	15	2	500	27.82%	41.56%	0:12:53
4	20	15	3	500	22.35%	36.94%	0:12:46
4	20	15	0	750	29.32%	44.08%	0:18:17
4	20	15	1	750	29.04%	43.13%	0:19:54
4	20	15	2	750	23.21%	36.85%	0:18:33
4	20	15	3	750	19.36%	32.18%	0:18:16
4	20	15	0	1000	28.25%	43.09%	0:23:44
4	20	15	1	1000	29.22%	42.50%	0:25:22
4	20	15	2	1000	23.12%	36.31%	0:24:01
4	20	15	3	1000	17.77%	30.12%	0:23:49
4	20	15	0	1500	25.56%	40.22%	0:34:25
4	20	15	1	1500	24.47%	36.31%	0:36:59

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
4	20	15	2	1500	14.49%	23.83%	0:34:50
4	20	15	3	1500	9.36%	17.41%	0:34:54
4	20	15	0	2000	24.05%	37.16%	0:45:04
4	20	15	1	2000	20.95%	32.63%	0:47:37
4	20	15	2	2000	9.78%	17.19%	0:45:28
4	20	15	3	2000	1.09%	3.23%	0:45:41
4	20	20	0	100	28.98%	44.75%	0:05:03
4	20	20	1	100	27.13%	40.93%	0:04:31
4	20	20	2	100	29.21%	43.13%	0:04:13
4	20	20	0	250	30.41%	45.78%	0:08:21
4	20	20	1	250	30.30%	44.93%	0:07:50
4	20	20	2	250	32.35%	46.63%	0:07:31
4	20	20	0	500	30.84%	45.42%	0:14:00
4	20	20	1	500	29.71%	42.37%	0:13:32
4	20	20	2	500	32.59%	45.96%	0:13:06
4	20	20	0	750	30.49%	44.43%	0:19:20
4	20	20	1	750	28.44%	40.62%	0:18:50
4	20	20	2	750	30.50%	45.15%	0:18:38
4	20	20	0	1000	29.60%	42.68%	0:24:57
4	20	20	1	1000	25.00%	35.95%	0:24:19
4	20	20	2	1000	24.02%	36.54%	0:23:55
4	20	20	0	1500	28.32%	41.52%	0:35:42
4	20	20	1	1500	22.96%	33.17%	0:35:28
4	20	20	2	1500	18.91%	29.08%	0:34:59

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
4	20	20	0	2000	27.41%	40.26%	0:47:15
4	20	20	1	2000	19.46%	29.22%	0:46:38
4	20	20	2	2000	15.12%	24.82%	0:46:08
4	20	30	0	100	34.98%	51.03%	0:06:15
4	20	30	1	100	37.31%	54.08%	0:04:49
4	20	30	0	250	37.49%	53.46%	0:09:39
4	20	30	1	250	41.91%	57.05%	0:08:14
4	20	30	0	500	38.66%	53.73%	0:15:11
4	20	30	1	500	42.18%	56.06%	0:13:42
4	20	30	0	750	40.27%	54.62%	0:20:44
4	20	30	1	750	44.35%	58.08%	0:19:29
4	20	30	0	1000	40.02%	53.77%	0:26:16
4	20	30	1	1000	42.48%	55.70%	0:24:47
4	20	30	0	1500	37.53%	51.48%	0:37:22
4	20	30	1	1500	36.20%	48.25%	0:36:03
4	20	30	0	2000	37.09%	49.69%	0:49:16
4	20	30	1	2000	37.41%	50.40%	0:47:03
5	18	15	0	100	26.49%	40.66%	0:04:46
5	18	15	1	100	26.72%	40.84%	0:06:08
5	18	15	2	100	23.90%	37.48%	0:04:50
5	18	15	3	100	17.65%	29.22%	0:04:35
5	18	15	0	250	28.90%	43.49%	0:08:52
5	18	15	1	250	30.37%	45.02%	0:10:24
5	18	15	2	250	26.78%	41.65%	0:08:58

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	18	15	3	250	20.19%	33.17%	0:08:44
5	18	15	0	500	28.33%	43.18%	0:15:44
5	18	15	1	500	27.72%	41.20%	0:17:19
5	18	15	2	500	22.84%	35.59%	0:15:49
5	18	15	3	500	15.33%	25.99%	0:15:34
5	18	15	0	750	26.78%	40.17%	0:22:40
5	18	15	1	750	26.36%	39.09%	0:24:15
5	18	15	2	750	21.36%	32.85%	0:22:33
5	18	15	3	750	13.88%	24.87%	0:22:29
5	18	15	0	1000	25.69%	38.51%	0:29:12
5	18	15	1	1000	24.54%	36.40%	0:31:22
5	18	15	2	1000	15.78%	25.85%	0:29:21
5	18	15	3	1000	9.17%	17.15%	0:29:14
5	18	15	0	1500	23.58%	36.36%	0:42:52
5	18	15	1	1500	19.63%	31.06%	0:46:11
5	18	15	2	1500	7.57%	14.50%	0:44:10
5	18	15	3	1500	1.30%	3.41%	0:42:54
5	18	20	0	100	28.20%	43.63%	0:05:50
5	18	20	1	100	26.08%	39.95%	0:05:14
5	18	20	2	100	26.95%	40.08%	0:04:53
5	18	20	0	250	30.59%	45.47%	0:09:56
5	18	20	1	250	28.95%	41.83%	0:09:24
5	18	20	2	250	29.55%	44.03%	0:09:01
5	18	20	0	500	30.67%	45.47%	0:16:47

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	18	20	1	500	31.74%	46.05%	0:16:15
5	18	20	2	500	32.79%	47.62%	0:15:57
5	18	20	0	750	31.18%	46.05%	0:23:38
5	18	20	1	750	29.48%	41.74%	0:23:40
5	18	20	2	750	28.59%	42.68%	0:22:55
5	18	20	0	1000	30.97%	45.92%	0:30:41
5	18	20	1	1000	26.35%	38.06%	0:29:58
5	18	20	2	1000	20.58%	33.03%	0:29:50
5	18	20	0	1500	26.50%	39.77%	0:44:15
5	18	20	1	1500	18.63%	28.23%	0:43:46
5	18	20	2	1500	11.52%	19.75%	0:43:22
5	18	30	0	100	33.71%	49.69%	0:07:09
5	18	30	1	100	34.02%	49.33%	0:05:35
5	18	30	0	250	37.78%	53.41%	0:11:18
5	18	30	1	250	40.62%	56.15%	0:09:42
5	18	30	0	500	39.30%	54.40%	0:18:13
5	18	30	1	500	42.73%	56.96%	0:16:41
5	18	30	0	750	38.34%	53.05%	0:25:16
5	18	30	1	750	39.52%	53.86%	0:23:35
5	18	30	0	1000	37.48%	51.26%	0:32:05
5	18	30	1	1000	37.93%	52.02%	0:30:30
5	18	30	0	1500	36.70%	49.73%	0:46:00
5	18	30	1	1500	33.63%	46.05%	0:44:18
5	20	10	0	100	15.66%	27.69%	0:04:33

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	20	10	1	100	11.28%	20.20%	0:05:18
5	20	10	2	100	7.36%	14.32%	0:05:19
5	20	10	3	100	2.21%	6.33%	0:04:35
5	20	10	4	100	0.87%	1.97%	0:04:29
5	20	10	5	100	0.47%	1.03%	0:04:16
5	20	10	0	250	17.36%	29.58%	0:08:36
5	20	10	1	250	13.92%	24.87%	0:09:28
5	20	10	2	250	9.43%	17.91%	0:09:24
5	20	10	3	250	2.15%	6.01%	0:08:38
5	20	10	4	250	0.34%	0.40%	0:08:38
5	20	10	5	250	0.20%	0.22%	0:08:18
5	20	10	0	500	15.35%	26.75%	0:15:23
5	20	10	1	500	10.27%	19.25%	0:16:24
5	20	10	2	500	7.65%	15.08%	0:16:22
5	20	10	3	500	2.31%	6.69%	0:15:31
5	20	10	4	500	0.42%	0.40%	0:15:22
5	20	10	5	500	0.21%	0.27%	0:15:08
5	20	10	0	750	14.49%	26.26%	0:22:18
5	20	10	1	750	9.45%	18.13%	0:23:24
5	20	10	2	750	7.17%	15.13%	0:23:22
5	20	10	3	750	2.64%	7.63%	0:22:17
5	20	10	4	750	0.31%	0.40%	0:22:08
5	20	10	0	1000	14.09%	24.69%	0:28:53
5	20	10	1	1000	7.13%	14.81%	0:30:27

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	20	10	2	1000	2.76%	7.45%	0:30:03
5	20	10	3	1000	0.17%	0.09%	0:29:04
5	20	10	4	1000	0.15%	0.04%	0:29:09
5	20	10	5	1000	0.12%	0.00%	0:28:57
5	20	10	0	1500	12.39%	22.58%	0:42:43
5	20	10	1	1500	6.04%	12.48%	0:44:15
5	20	10	2	1500	1.13%	2.74%	0:44:00
5	20	10	3	1500	0.16%	0.04%	0:43:00
5	20	10	4	1500	0.17%	0.00%	0:42:41
5	20	10	5	1500	0.13%	0.22%	0:42:48
5	20	10	0	2000	10.62%	20.15%	0:56:09
5	20	10	1	2000	3.65%	8.30%	0:58:01
5	20	10	2	2000	0.41%	0.31%	0:57:37
5	20	10	3	2000	0.16%	0.04%	0:56:32
5	20	10	4	2000	0.16%	0.04%	0:56:21
5	20	10	5	2000	0.13%	0.04%	0:56:22
5	20	15	0	100	26.00%	40.80%	0:04:53
5	20	15	1	100	24.71%	39.72%	0:06:26
5	20	15	2	100	23.04%	36.09%	0:04:57
5	20	15	3	100	18.69%	31.69%	0:04:41
5	20	15	0	250	28.42%	43.76%	0:08:59
5	20	15	1	250	27.83%	41.88%	0:10:38
5	20	15	2	250	24.67%	38.20%	0:09:08
5	20	15	3	250	22.26%	37.03%	0:08:49

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	20	15	0	500	29.49%	44.34%	0:15:42
5	20	15	1	500	30.92%	45.15%	0:17:30
5	20	15	2	500	25.04%	38.91%	0:15:58
5	20	15	3	500	22.19%	36.22%	0:15:41
5	20	15	0	750	29.00%	44.79%	0:22:37
5	20	15	1	750	29.51%	43.99%	0:24:36
5	20	15	2	750	22.61%	35.50%	0:22:42
5	20	15	3	750	18.62%	30.92%	0:22:39
5	20	15	0	1000	27.04%	42.15%	0:29:19
5	20	15	1	1000	26.36%	38.82%	0:31:33
5	20	15	2	1000	18.39%	28.86%	0:30:16
5	20	15	3	1000	14.20%	26.21%	0:29:36
5	20	15	0	1500	25.17%	38.96%	0:43:01
5	20	15	1	1500	22.39%	34.20%	0:46:04
5	20	15	2	1500	10.36%	19.12%	0:43:12
5	20	15	3	1500	2.41%	5.92%	0:43:04
5	20	15	0	2000	21.98%	34.65%	0:56:18
5	20	15	1	2000	20.00%	31.01%	0:59:37
5	20	15	2	2000	6.61%	13.78%	0:56:31
5	20	15	3	2000	1.77%	5.03%	0:56:32
5	20	20	0	100	28.64%	44.39%	0:06:03
5	20	20	1	100	28.22%	42.77%	0:05:25
5	20	20	2	100	27.33%	41.02%	0:04:57
5	20	20	0	250	31.11%	46.01%	0:10:05

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	20	20	1	250	30.72%	44.84%	0:09:30
5	20	20	2	250	29.87%	42.59%	0:09:10
5	20	20	0	500	30.77%	44.48%	0:17:02
5	20	20	1	500	29.05%	41.43%	0:16:27
5	20	20	2	500	31.85%	46.18%	0:16:05
5	20	20	0	750	29.99%	43.67%	0:23:50
5	20	20	1	750	27.66%	40.22%	0:23:22
5	20	20	2	750	27.64%	41.74%	0:22:56
5	20	20	0	1000	29.87%	42.86%	0:30:44
5	20	20	1	1000	24.39%	34.87%	0:30:12
5	20	20	2	1000	23.23%	34.25%	0:30:06
5	20	20	0	1500	27.44%	40.26%	0:44:28
5	20	20	1	1500	19.41%	28.82%	0:44:14
5	20	20	2	1500	18.05%	28.68%	0:44:28
5	20	20	0	2000	25.74%	37.97%	0:58:10
5	20	20	1	2000	17.42%	26.08%	0:58:09
5	20	20	2	2000	13.99%	22.58%	0:57:20
5	20	30	0	100	35.05%	50.54%	0:07:38
5	20	30	1	100	36.92%	53.50%	0:05:47
5	20	30	0	250	39.15%	54.67%	0:11:47
5	20	30	1	250	43.13%	59.38%	0:10:06
5	20	30	0	500	40.08%	54.44%	0:18:51
5	20	30	1	500	43.91%	58.53%	0:17:01
5	20	30	0	750	40.35%	54.58%	0:25:34

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	20	30	1	750	45.16%	60.59%	0:23:50
5	20	30	0	1000	40.98%	54.94%	0:32:39
5	20	30	1	1000	44.08%	58.12%	0:30:52
5	20	30	0	1500	39.10%	52.56%	0:46:32
5	20	30	1	1500	38.48%	51.08%	0:44:49
5	20	30	0	2000	37.38%	50.81%	1:01:31
5	20	30	1	2000	35.44%	47.17%	0:58:21
5	25	15	0	100	30.27%	45.87%	0:05:17
5	25	15	1	100	32.55%	49.19%	0:07:08
5	25	15	2	100	32.36%	47.17%	0:05:12
5	25	15	3	100	33.07%	47.76%	0:04:50
5	25	15	0	250	32.98%	48.43%	0:09:20
5	25	15	1	250	36.00%	52.20%	0:11:19
5	25	15	2	250	38.18%	53.73%	0:09:22
5	25	15	3	250	34.95%	50.58%	0:08:56
5	25	15	0	500	35.12%	51.44%	0:16:24
5	25	15	1	500	38.78%	55.25%	0:18:18
5	25	15	2	500	39.36%	54.98%	0:16:15
5	25	15	3	500	34.88%	50.45%	0:15:50
5	25	15	0	750	35.05%	51.12%	0:23:14
5	25	15	1	750	38.05%	53.86%	0:25:20
5	25	15	2	750	37.88%	53.95%	0:23:06
5	25	15	3	750	29.92%	45.33%	0:22:49
5	25	15	0	1000	33.25%	49.15%	0:30:07

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	25	15	1	1000	36.59%	52.20%	0:32:58
5	25	15	2	1000	32.90%	48.03%	0:29:55
5	25	15	3	1000	26.48%	41.97%	0:29:48
5	25	15	0	1500	32.65%	47.67%	0:43:52
5	25	15	1	1500	33.97%	49.46%	0:47:29
5	25	15	2	1500	31.34%	44.57%	0:43:54
5	25	15	3	1500	22.96%	36.62%	0:43:23
5	25	20	0	100	33.63%	49.73%	0:06:44
5	25	20	1	100	37.39%	53.23%	0:06:41
5	25	20	2	100	37.90%	53.05%	0:05:14
5	25	20	0	250	35.06%	50.27%	0:10:57
5	25	20	1	250	41.85%	56.51%	0:10:51
5	25	20	2	250	44.39%	59.43%	0:09:25
5	25	20	0	500	37.13%	52.56%	0:17:51
5	25	20	1	500	43.84%	58.44%	0:17:49
5	25	20	2	500	46.55%	61.13%	0:16:14
5	25	20	0	750	37.27%	52.69%	0:24:57
5	25	20	1	750	43.75%	58.39%	0:24:43
5	25	20	2	750	45.84%	61.80%	0:23:14
5	25	20	0	1000	36.81%	52.74%	0:31:56
5	25	20	1	1000	44.50%	58.84%	0:32:00
5	25	20	2	1000	46.93%	62.34%	0:30:08
5	25	20	0	1500	34.88%	49.37%	0:45:58
5	25	20	1	1500	40.80%	55.03%	0:45:43

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	25	20	2	1500	41.57%	56.87%	0:43:58
5	25	30	0	100	39.97%	56.19%	0:08:47
5	25	30	1	100	43.28%	60.05%	0:06:11
5	25	30	0	250	45.12%	61.49%	0:13:04
5	25	30	1	250	49.31%	65.39%	0:10:20
5	25	30	0	500	47.46%	62.61%	0:20:07
5	25	30	1	500	53.25%	68.09%	0:17:18
5	25	30	0	750	48.88%	64.23%	0:27:20
5	25	30	1	750	55.19%	70.29%	0:25:45
5	25	30	0	1000	50.00%	64.99%	0:34:22
5	25	30	1	1000	55.59%	69.97%	0:31:16
5	25	30	0	1500	49.72%	64.18%	0:48:37
5	25	30	1	1500	55.17%	68.67%	0:46:04
5	30	15	0	100	31.90%	47.44%	0:05:33
5	30	15	1	100	37.78%	54.67%	0:07:50
5	30	15	2	100	37.53%	53.90%	0:05:28
5	30	15	3	100	35.32%	50.63%	0:05:02
5	30	15	0	250	35.49%	51.71%	0:09:36
5	30	15	1	250	41.51%	56.96%	0:12:02
5	30	15	2	250	44.11%	59.92%	0:09:37
5	30	15	3	250	41.07%	56.96%	0:09:11
5	30	15	0	500	37.70%	53.50%	0:16:35
5	30	15	1	500	43.00%	58.08%	0:19:05
5	30	15	2	500	45.57%	60.37%	0:16:26

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	30	15	3	500	42.07%	58.48%	0:16:02
5	30	15	0	750	38.16%	53.90%	0:23:30
5	30	15	1	750	44.10%	58.71%	0:26:22
5	30	15	2	750	45.63%	61.04%	0:23:24
5	30	15	3	750	39.71%	56.06%	0:23:01
5	30	15	0	1000	37.05%	52.87%	0:30:30
5	30	15	1	1000	44.24%	58.98%	0:33:15
5	30	15	2	1000	42.56%	57.32%	0:30:22
5	30	15	3	1000	34.34%	50.49%	0:29:47
5	30	15	0	1500	36.76%	52.56%	0:44:20
5	30	15	1	1500	41.08%	56.55%	0:48:23
5	30	15	2	1500	39.24%	55.25%	0:44:26
5	30	15	3	1500	34.39%	50.18%	0:43:41
5	30	20	0	100	35.97%	52.47%	0:07:25
5	30	20	1	100	42.20%	58.66%	0:07:11
5	30	20	2	100	42.93%	59.25%	0:05:31
5	30	20	0	250	39.97%	56.37%	0:11:40
5	30	20	1	250	47.50%	62.97%	0:11:25
5	30	20	2	250	49.97%	65.48%	0:09:42
5	30	20	0	500	41.37%	56.60%	0:18:29
5	30	20	1	500	50.44%	65.44%	0:18:25
5	30	20	2	500	53.55%	68.36%	0:16:35
5	30	20	0	750	40.48%	56.15%	0:25:37
5	30	20	1	750	50.62%	64.95%	0:25:56

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	30	20	2	750	52.89%	67.15%	0:23:39
5	30	20	0	1000	41.32%	57.09%	0:32:52
5	30	20	1	1000	51.09%	64.95%	0:32:27
5	30	20	2	1000	53.02%	67.91%	0:30:29
5	30	20	0	1500	40.02%	54.94%	0:46:40
5	30	20	1	1500	48.27%	62.12%	0:46:33
5	30	20	2	1500	50.57%	65.22%	0:44:39
5	30	30	0	100	41.62%	57.63%	0:09:57
5	30	30	1	100	45.53%	61.62%	0:06:41
5	30	30	0	250	45.93%	62.52%	0:14:16
5	30	30	1	250	50.87%	66.74%	0:10:54
5	30	30	0	500	48.83%	64.86%	0:21:42
5	30	30	1	500	55.03%	70.11%	0:17:53
5	30	30	0	750	50.26%	65.66%	0:28:51
5	30	30	1	750	56.90%	71.27%	0:24:48
5	30	30	0	1000	51.46%	66.43%	0:35:49
5	30	30	1	1000	57.76%	71.63%	0:32:02
5	30	30	0	1500	52.29%	67.19%	0:51:01
5	30	30	1	1500	58.68%	72.58%	0:45:43
5	40	15	0	100	36.20%	52.11%	0:06:06
5	40	15	1	100	41.14%	58.08%	0:09:21
5	40	15	2	100	41.57%	58.17%	0:05:58
5	40	15	3	100	40.30%	56.46%	0:05:25
5	40	15	0	250	39.15%	56.01%	0:10:13

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	40	15	1	250	46.16%	61.76%	0:13:35
5	40	15	2	250	48.50%	64.18%	0:10:07
5	40	15	3	250	46.75%	62.93%	0:09:32
5	40	15	0	500	41.60%	57.76%	0:17:07
5	40	15	1	500	49.35%	65.26%	0:20:48
5	40	15	2	500	51.64%	66.43%	0:17:02
5	40	15	3	500	47.82%	63.11%	0:16:27
5	40	15	0	750	41.66%	57.36%	0:24:00
5	40	15	1	750	49.82%	64.72%	0:27:53
5	40	15	2	750	51.22%	65.57%	0:23:55
5	40	15	3	750	46.06%	61.58%	0:23:18
5	40	15	0	1000	41.83%	57.54%	0:30:59
5	40	15	1	1000	49.58%	64.27%	0:35:09
5	40	15	2	1000	51.69%	66.34%	0:30:59
5	40	15	3	1000	47.90%	64.14%	0:30:19
5	40	15	0	1500	41.79%	57.09%	0:45:10
5	40	15	1	1500	49.25%	63.64%	0:49:28
5	40	15	2	1500	50.10%	64.72%	0:45:09
5	40	15	3	1500	44.77%	60.59%	0:45:02
5	40	20	0	100	37.99%	54.40%	0:08:41
5	40	20	1	100	45.54%	62.21%	0:08:24
5	40	20	2	100	46.40%	63.11%	0:06:01
5	40	20	0	250	41.88%	58.57%	0:12:54
5	40	20	1	250	50.17%	64.95%	0:12:35

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	40	20	2	250	50.99%	66.47%	0:10:13
5	40	20	0	500	44.61%	60.55%	0:20:04
5	40	20	1	500	53.27%	67.59%	0:19:48
5	40	20	2	500	54.36%	69.08%	0:17:05
5	40	20	0	750	45.46%	61.49%	0:27:06
5	40	20	1	750	54.97%	68.40%	0:26:38
5	40	20	2	750	54.75%	68.67%	0:24:13
5	40	20	0	1000	45.74%	61.22%	0:34:10
5	40	20	1	1000	55.33%	68.81%	0:33:56
5	40	20	2	1000	55.07%	69.30%	0:31:07
5	40	20	0	1500	45.59%	59.92%	0:48:39
5	40	20	1	1500	55.10%	68.18%	0:48:11
5	40	20	2	1500	54.64%	68.31%	0:44:42
5	40	30	0	100	41.28%	58.26%	0:12:32
5	40	30	1	100	47.31%	63.96%	0:07:37
5	40	30	0	250	46.23%	63.02%	0:17:01
5	40	30	1	250	53.08%	68.40%	0:11:53
5	40	30	0	500	49.59%	65.22%	0:24:34
5	40	30	1	500	57.53%	71.23%	0:18:45
5	40	30	0	750	51.78%	67.64%	0:31:50
5	40	30	1	750	59.07%	73.29%	0:25:51
5	40	30	0	1000	53.43%	68.63%	0:39:22
5	40	30	1	1000	60.59%	73.65%	0:32:57
5	40	30	0	1500	54.85%	69.30%	0:54:23

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	40	30	1	1500	60.49%	73.56%	0:46:54
5	50	15	0	100	36.63%	53.68%	0:06:39
5	50	15	1	100	44.02%	61.76%	0:10:59
5	50	15	2	100	45.99%	63.06%	0:06:28
5	50	15	3	100	43.31%	60.10%	0:05:43
5	50	15	0	250	41.42%	57.59%	0:10:52
5	50	15	1	250	48.36%	64.14%	0:15:20
5	50	15	2	250	50.51%	66.47%	0:10:36
5	50	15	3	250	47.25%	62.79%	0:09:51
5	50	15	0	500	42.40%	57.85%	0:17:42
5	50	15	1	500	51.24%	66.07%	0:22:32
5	50	15	2	500	54.53%	69.17%	0:17:32
5	50	15	3	500	48.10%	63.42%	0:16:52
5	50	15	0	750	43.07%	59.07%	0:24:41
5	50	15	1	750	51.01%	65.84%	0:30:00
5	50	15	2	750	54.17%	68.49%	0:24:36
5	50	15	3	750	45.49%	60.55%	0:23:44
5	50	15	0	1000	43.44%	59.43%	0:32:10
5	50	15	1	1000	50.85%	64.86%	0:37:28
5	50	15	2	1000	54.40%	68.72%	0:31:21
5	50	15	3	1000	44.80%	60.32%	0:30:52
5	50	15	0	1500	44.33%	58.66%	0:45:33
5	50	15	1	1500	52.25%	66.92%	0:52:14
5	50	15	2	1500	54.45%	68.00%	0:45:16

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	50	15	3	1500	44.39%	60.59%	0:44:39
5	50	20	0	100	39.27%	55.97%	0:10:09
5	50	20	1	100	46.57%	62.88%	0:09:39
5	50	20	2	100	47.71%	64.36%	0:06:37
5	50	20	0	250	43.01%	58.80%	0:14:29
5	50	20	1	250	51.29%	66.79%	0:13:54
5	50	20	2	250	51.82%	67.41%	0:10:46
5	50	20	0	500	45.49%	61.62%	0:21:35
5	50	20	1	500	54.56%	68.76%	0:21:05
5	50	20	2	500	55.57%	69.97%	0:17:48
5	50	20	0	750	46.00%	61.71%	0:29:26
5	50	20	1	750	56.47%	70.51%	0:28:32
5	50	20	2	750	56.22%	70.06%	0:24:42
5	50	20	0	1000	46.73%	61.85%	0:36:12
5	50	20	1	1000	57.03%	70.42%	0:35:18
5	50	20	2	1000	56.01%	70.47%	0:31:35
5	50	20	0	1500	47.78%	62.88%	0:50:34
5	50	20	1	1500	57.73%	70.92%	0:49:39
5	50	20	2	1500	55.90%	69.79%	0:45:20
5	50	30	0	100	41.33%	57.68%	0:15:44
5	50	30	1	100	49.40%	65.84%	0:08:35
5	50	30	0	250	46.76%	63.38%	0:20:13
5	50	30	1	250	55.02%	70.06%	0:12:48
5	50	30	0	500	50.77%	65.66%	0:27:50

Table 3: Extended results for **R²MMT**, standard configuration. Time is in format h:mm:ss.

Epochs	K	τ	t	Iterations	mAP	Top-1	Training Time
5	50	30	1	500	59.08%	72.98%	0:19:55
5	50	30	0	750	51.33%	66.07%	0:35:49
5	50	30	1	750	60.14%	73.11%	0:27:04
5	50	30	0	1000	53.08%	68.04%	0:43:31
5	50	30	1	1000	61.30%	74.28%	0:33:55
5	50	30	0	1500	55.61%	69.88%	0:58:52
5	50	30	1	1500	62.64%	74.96%	0:48:17

Configuration with Memory

Here we see the results of R²MMT in the configuration with memory. Training time does not include SDS time. Due to the pipeline structure of R²MMT, both the training time and the SDS time must be less than τ to be viable in the real world. Similar to Chapter 6 there are some results for $\tau = 10$ included. Accuracies for the same K are generally higher here, at the expense of greater SDS times. The highest Top-1 here, ignoring the time constraint, is 76.53%. However, it is impossible to reach this within the time constraint in our system. While algorithmically possible, achieving this level of accuracy in the real-world would require further optimization of the R²MMT system. There is the possibility of just using more capable hardware to achieve this. However, considering the GPU server we use is already high-end, attempting to do so would quickly become cost prohibitive.

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	K	τ	t	Iterations	mAP	Top-1	Training Time
1	18	15	0	100	19.54%	33.21%	0:01:51

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	18	15	1	100	23.11%	36.94%	0:02:25
1	18	15	2	100	26.40%	39.95%	0:02:44
1	18	15	3	100	28.49%	43.27%	0:02:56
1	18	15	0	250	20.76%	34.11%	0:02:41
1	18	15	1	250	25.67%	39.63%	0:03:16
1	18	15	2	250	30.78%	44.88%	0:03:33
1	18	15	3	250	35.06%	49.33%	0:03:46
1	18	15	0	500	21.13%	33.80%	0:04:04
1	18	15	1	500	26.88%	40.89%	0:04:41
1	18	15	2	500	32.45%	46.14%	0:05:01
1	18	15	3	500	37.62%	51.57%	0:05:13
1	18	15	0	750	21.44%	34.96%	0:05:36
1	18	15	1	750	28.88%	43.00%	0:06:16
1	18	15	2	750	33.80%	48.07%	0:06:21
1	18	15	3	750	39.22%	53.10%	0:06:36
1	18	15	0	1000	22.76%	36.18%	0:06:50
1	18	15	1	1000	29.56%	43.13%	0:07:28
1	18	15	2	1000	35.28%	48.92%	0:07:48
1	18	15	3	1000	41.01%	54.17%	0:08:04
1	18	15	0	1500	21.58%	35.19%	0:09:31
1	18	15	1	1500	29.18%	42.55%	0:10:20
1	18	15	2	1500	34.60%	47.31%	0:10:37
1	18	15	3	1500	39.72%	52.92%	0:10:53
1	18	20	0	100	23.79%	40.44%	0:02:05

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	18	20	1	100	24.46%	38.42%	0:02:17
1	18	20	2	100	24.60%	37.88%	0:02:29
1	18	20	0	250	24.72%	39.72%	0:02:53
1	18	20	1	250	26.76%	40.26%	0:03:06
1	18	20	2	250	27.91%	41.56%	0:03:19
1	18	20	0	500	26.72%	41.29%	0:04:18
1	18	20	1	500	29.82%	43.72%	0:04:31
1	18	20	2	500	33.34%	46.72%	0:04:45
1	18	20	0	750	26.76%	41.88%	0:05:38
1	18	20	1	750	31.65%	45.60%	0:05:53
1	18	20	2	750	34.40%	47.44%	0:06:08
1	18	20	0	1000	26.96%	41.47%	0:06:58
1	18	20	1	1000	32.16%	46.32%	0:07:17
1	18	20	2	1000	35.32%	47.89%	0:07:32
1	18	20	0	1500	27.28%	42.01%	0:10:02
1	18	20	1	1500	31.98%	45.29%	0:10:08
1	18	20	2	1500	35.58%	47.40%	0:10:22
1	18	30	0	100	28.51%	44.25%	0:02:22
1	18	30	1	100	32.52%	47.58%	0:02:56
1	18	30	0	250	30.25%	46.23%	0:03:12
1	18	30	1	250	35.97%	51.35%	0:03:45
1	18	30	0	500	31.37%	46.32%	0:04:36
1	18	30	1	500	38.35%	52.92%	0:05:14
1	18	30	0	750	32.78%	47.85%	0:06:02

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	18	30	1	750	40.53%	55.66%	0:06:37
1	18	30	0	1000	33.15%	48.07%	0:07:28
1	18	30	1	1000	41.17%	55.75%	0:08:02
1	18	30	0	1500	33.73%	49.06%	0:10:07
1	18	30	1	1500	42.01%	56.19%	0:10:55
1	20	10	0	100	16.20%	29.04%	0:01:49
1	20	10	1	100	18.41%	30.83%	0:02:09
1	20	10	2	100	20.88%	32.09%	0:02:32
1	20	10	3	100	21.75%	33.26%	0:02:44
1	20	10	4	100	23.94%	36.40%	0:02:55
1	20	10	5	100	27.51%	41.07%	0:03:18
1	20	10	0	250	16.69%	29.08%	0:02:37
1	20	10	1	250	22.12%	35.41%	0:03:01
1	20	10	2	250	24.40%	36.40%	0:03:22
1	20	10	3	250	27.82%	40.98%	0:03:34
1	20	10	4	250	31.07%	44.48%	0:03:47
1	20	10	5	250	35.39%	49.37%	0:04:09
1	20	10	0	500	17.22%	29.89%	0:04:02
1	20	10	1	500	22.00%	34.29%	0:04:26
1	20	10	2	500	25.51%	37.93%	0:04:44
1	20	10	3	500	29.59%	42.24%	0:04:58
1	20	10	4	500	33.98%	46.68%	0:05:17
1	20	10	5	500	38.24%	51.08%	0:05:38
1	20	10	0	750	16.99%	29.58%	0:05:22

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	20	10	1	750	22.56%	35.28%	0:05:49
1	20	10	2	750	25.15%	37.07%	0:06:08
1	20	10	3	750	28.50%	40.22%	0:06:22
1	20	10	4	750	32.10%	43.81%	0:06:39
1	20	10	5	750	36.50%	48.88%	0:07:05
1	20	10	0	1000	17.19%	28.99%	0:06:41
1	20	10	1	1000	21.14%	33.26%	0:07:11
1	20	10	2	1000	23.70%	34.65%	0:07:37
1	20	10	3	1000	27.34%	38.73%	0:07:51
1	20	10	4	1000	31.88%	43.72%	0:08:06
1	20	10	5	1000	37.22%	49.60%	0:08:40
1	20	10	0	1500	17.88%	31.33%	0:09:24
1	20	10	1	1500	22.30%	34.07%	0:10:01
1	20	10	2	1500	26.26%	38.38%	0:10:27
1	20	10	3	1500	30.90%	42.68%	0:10:38
1	20	10	4	1500	35.39%	48.20%	0:10:54
1	20	10	5	1500	39.32%	51.57%	0:11:36
1	20	10	0	2000	16.98%	29.94%	0:12:07
1	20	10	1	2000	21.78%	33.12%	0:12:46
1	20	10	2	2000	24.93%	36.54%	0:13:13
1	20	10	3	2000	27.11%	37.21%	0:13:33
1	20	10	4	2000	31.77%	42.32%	0:14:00
1	20	10	5	2000	34.77%	45.29%	0:14:39
1	20	15	0	100	23.41%	38.64%	0:01:53

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	20	15	1	100	25.75%	40.53%	0:02:31
1	20	15	2	100	27.82%	42.46%	0:02:49
1	20	15	3	100	30.69%	45.60%	0:03:03
1	20	15	0	250	25.33%	40.17%	0:02:43
1	20	15	1	250	29.24%	44.08%	0:03:21
1	20	15	2	250	32.97%	47.80%	0:03:40
1	20	15	3	250	37.71%	53.68%	0:03:55
1	20	15	0	500	26.88%	42.28%	0:04:04
1	20	15	1	500	32.09%	47.13%	0:04:45
1	20	15	2	500	36.79%	52.20%	0:05:09
1	20	15	3	500	42.11%	57.81%	0:05:21
1	20	15	0	750	26.36%	40.89%	0:05:25
1	20	15	1	750	32.26%	46.99%	0:06:12
1	20	15	2	750	38.05%	53.01%	0:06:29
1	20	15	3	750	43.07%	58.12%	0:06:53
1	20	15	0	1000	27.10%	42.28%	0:06:46
1	20	15	1	1000	33.07%	48.61%	0:07:34
1	20	15	2	1000	38.49%	53.73%	0:08:00
1	20	15	3	1000	45.24%	60.73%	0:08:14
1	20	15	0	1500	27.74%	42.64%	0:09:33
1	20	15	1	1500	33.78%	47.80%	0:10:29
1	20	15	2	1500	39.43%	53.82%	0:10:47
1	20	15	3	1500	44.79%	59.25%	0:11:18
1	20	15	0	2000	27.27%	41.25%	0:12:10

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	20	15	1	2000	32.75%	46.50%	0:13:12
1	20	15	2	2000	38.98%	53.19%	0:13:40
1	20	15	3	2000	45.17%	59.11%	0:13:58
1	20	20	0	100	25.20%	40.66%	0:02:07
1	20	20	1	100	25.55%	39.23%	0:02:20
1	20	20	2	100	25.81%	39.63%	0:02:34
1	20	20	0	250	25.93%	40.62%	0:02:59
1	20	20	1	250	29.39%	43.54%	0:03:11
1	20	20	2	250	32.83%	46.59%	0:03:25
1	20	20	0	500	27.53%	42.19%	0:04:21
1	20	20	1	500	32.29%	46.68%	0:04:34
1	20	20	2	500	36.92%	50.31%	0:04:48
1	20	20	0	750	29.12%	43.81%	0:05:41
1	20	20	1	750	33.94%	47.22%	0:05:56
1	20	20	2	750	38.06%	50.94%	0:06:15
1	20	20	0	1000	28.49%	43.22%	0:07:05
1	20	20	1	1000	34.66%	48.29%	0:07:21
1	20	20	2	1000	38.63%	51.48%	0:07:45
1	20	20	0	1500	28.32%	42.77%	0:09:47
1	20	20	1	1500	34.73%	48.29%	0:10:23
1	20	20	2	1500	39.10%	51.97%	0:10:29
1	20	20	0	2000	28.06%	41.83%	0:12:33
1	20	20	1	2000	35.08%	47.98%	0:13:02
1	20	20	2	2000	38.92%	50.18%	0:13:18

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	20	30	0	100	32.06%	48.11%	0:02:27
1	20	30	1	100	35.09%	51.89%	0:03:02
1	20	30	0	250	33.58%	49.01%	0:03:17
1	20	30	1	250	39.62%	55.57%	0:03:54
1	20	30	0	500	35.52%	51.08%	0:04:41
1	20	30	1	500	42.95%	57.85%	0:05:23
1	20	30	0	750	36.94%	52.92%	0:06:03
1	20	30	1	750	45.37%	61.04%	0:06:53
1	20	30	0	1000	37.24%	52.24%	0:07:27
1	20	30	1	1000	44.98%	60.64%	0:08:16
1	20	30	0	1500	37.91%	53.01%	0:10:12
1	20	30	1	1500	45.96%	60.55%	0:11:16
1	20	30	0	2000	38.88%	54.08%	0:13:06
1	20	30	1	2000	46.48%	60.55%	0:14:13
1	25	15	0	100	26.26%	42.64%	0:01:57
1	25	15	1	100	31.51%	47.35%	0:02:46
1	25	15	2	100	33.21%	49.46%	0:03:11
1	25	15	3	100	35.86%	52.87%	0:03:32
1	25	15	0	250	27.56%	43.13%	0:02:46
1	25	15	1	250	33.97%	50.63%	0:03:36
1	25	15	2	250	39.84%	56.64%	0:04:03
1	25	15	3	250	43.90%	60.50%	0:04:23
1	25	15	0	500	30.04%	46.59%	0:04:07
1	25	15	1	500	37.43%	53.95%	0:05:03

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	25	15	2	500	43.63%	59.83%	0:05:30
1	25	15	3	500	48.71%	64.23%	0:05:51
1	25	15	0	750	29.45%	46.27%	0:05:29
1	25	15	1	750	38.89%	55.66%	0:06:27
1	25	15	2	750	44.88%	61.27%	0:06:58
1	25	15	3	750	50.74%	65.48%	0:07:21
1	25	15	0	1000	30.67%	47.31%	0:06:51
1	25	15	1	1000	38.53%	54.76%	0:07:51
1	25	15	2	1000	45.12%	60.73%	0:08:25
1	25	15	3	1000	50.40%	64.68%	0:08:50
1	25	15	0	1500	30.54%	47.26%	0:09:35
1	25	15	1	1500	40.18%	55.52%	0:10:43
1	25	15	2	1500	47.00%	61.85%	0:11:31
1	25	15	3	1500	53.20%	67.64%	0:11:40
1	25	20	0	100	27.22%	42.86%	0:02:16
1	25	20	1	100	33.56%	49.60%	0:03:04
1	25	20	2	100	36.60%	53.28%	0:03:30
1	25	20	0	250	29.55%	44.93%	0:03:06
1	25	20	1	250	37.13%	52.78%	0:03:55
1	25	20	2	250	41.79%	57.85%	0:04:23
1	25	20	0	500	32.26%	47.80%	0:04:27
1	25	20	1	500	40.32%	56.42%	0:05:22
1	25	20	2	500	45.44%	60.82%	0:05:55
1	25	20	0	750	32.63%	47.76%	0:05:48

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	25	20	1	750	40.96%	56.64%	0:06:48
1	25	20	2	750	47.38%	62.84%	0:07:21
1	25	20	0	1000	32.65%	48.11%	0:07:14
1	25	20	1	1000	41.48%	55.79%	0:08:15
1	25	20	2	1000	48.50%	63.06%	0:08:51
1	25	20	0	1500	33.32%	48.74%	0:09:56
1	25	20	1	1500	41.58%	55.48%	0:11:05
1	25	20	2	1500	48.50%	62.30%	0:11:44
1	25	30	0	100	32.55%	49.06%	0:02:41
1	25	30	1	100	36.10%	53.46%	0:03:30
1	25	30	0	250	34.92%	51.39%	0:03:31
1	25	30	1	250	40.00%	56.24%	0:04:23
1	25	30	0	500	36.66%	52.56%	0:04:56
1	25	30	1	500	44.89%	59.96%	0:05:53
1	25	30	0	750	38.30%	52.96%	0:06:20
1	25	30	1	750	46.74%	61.40%	0:07:20
1	25	30	0	1000	38.69%	53.86%	0:07:43
1	25	30	1	1000	47.12%	62.16%	0:08:49
1	25	30	0	1500	39.55%	55.07%	0:10:30
1	25	30	1	1500	49.43%	64.00%	0:11:47
1	30	15	0	100	27.14%	43.31%	0:01:59
1	30	15	1	100	31.85%	48.43%	0:03:01
1	30	15	2	100	35.44%	53.59%	0:03:34
1	30	15	3	100	37.50%	55.12%	0:03:59

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	30	15	0	250	28.53%	44.34%	0:02:50
1	30	15	1	250	36.33%	53.90%	0:03:54
1	30	15	2	250	41.74%	58.84%	0:04:27
1	30	15	3	250	45.16%	61.45%	0:04:54
1	30	15	0	500	31.05%	46.99%	0:04:12
1	30	15	1	500	39.58%	55.79%	0:05:19
1	30	15	2	500	45.82%	61.67%	0:05:58
1	30	15	3	500	50.55%	66.29%	0:06:26
1	30	15	0	750	31.72%	48.29%	0:05:35
1	30	15	1	750	40.19%	56.24%	0:06:47
1	30	15	2	750	47.36%	63.24%	0:07:25
1	30	15	3	750	52.15%	67.32%	0:07:57
1	30	15	0	1000	31.84%	47.98%	0:06:54
1	30	15	1	1000	40.57%	56.33%	0:08:21
1	30	15	2	1000	47.56%	62.48%	0:09:03
1	30	15	3	1000	53.53%	68.04%	0:09:24
1	30	15	0	1500	31.90%	47.76%	0:09:57
1	30	15	1	1500	40.92%	55.97%	0:11:12
1	30	15	2	1500	49.01%	64.14%	0:12:01
1	30	15	3	1500	54.27%	68.90%	0:12:27
1	30	20	0	100	29.61%	45.74%	0:02:24
1	30	20	1	100	34.79%	51.84%	0:03:24
1	30	20	2	100	35.66%	52.51%	0:04:00
1	30	20	0	250	32.07%	49.10%	0:03:12

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	30	20	1	250	39.55%	56.37%	0:04:16
1	30	20	2	250	43.56%	60.73%	0:04:54
1	30	20	0	500	33.98%	50.18%	0:04:36
1	30	20	1	500	42.23%	58.80%	0:05:44
1	30	20	2	500	47.36%	63.11%	0:06:26
1	30	20	0	750	34.92%	50.99%	0:05:58
1	30	20	1	750	43.65%	59.69%	0:07:11
1	30	20	2	750	50.43%	66.02%	0:07:57
1	30	20	0	1000	34.94%	51.30%	0:07:22
1	30	20	1	1000	44.43%	59.96%	0:08:40
1	30	20	2	1000	50.72%	65.80%	0:09:25
1	30	20	0	1500	35.53%	51.93%	0:10:22
1	30	20	1	1500	46.66%	61.71%	0:11:39
1	30	20	2	1500	52.64%	67.41%	0:12:31
1	30	30	0	100	33.27%	50.27%	0:02:55
1	30	30	1	100	36.36%	53.41%	0:04:00
1	30	30	0	250	37.94%	54.53%	0:03:47
1	30	30	1	250	42.38%	58.75%	0:04:55
1	30	30	0	500	38.70%	54.13%	0:05:13
1	30	30	1	500	45.91%	62.16%	0:06:26
1	30	30	0	750	40.48%	56.06%	0:06:40
1	30	30	1	750	47.91%	62.21%	0:07:56
1	30	30	0	1000	40.80%	55.66%	0:08:00
1	30	30	1	1000	48.55%	63.33%	0:09:29

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	30	30	0	1500	42.19%	57.59%	0:10:51
1	30	30	1	1500	51.08%	66.07%	0:12:32
1	40	15	0	100	28.61%	44.97%	0:02:08
1	40	15	1	100	33.51%	50.40%	0:03:37
1	40	15	2	100	35.77%	53.10%	0:04:26
1	40	15	3	100	36.42%	52.83%	0:05:06
1	40	15	0	250	31.27%	46.90%	0:03:00
1	40	15	1	250	38.38%	54.76%	0:04:31
1	40	15	2	250	42.54%	58.98%	0:05:24
1	40	15	3	250	45.66%	62.16%	0:06:02
1	40	15	0	500	33.80%	49.82%	0:04:19
1	40	15	1	500	42.51%	59.11%	0:05:59
1	40	15	2	500	47.24%	63.06%	0:06:55
1	40	15	3	500	51.45%	66.61%	0:07:43
1	40	15	0	750	35.13%	51.48%	0:05:40
1	40	15	1	750	44.51%	61.04%	0:07:30
1	40	15	2	750	50.53%	66.34%	0:08:29
1	40	15	3	750	54.45%	69.34%	0:09:15
1	40	15	0	1000	35.37%	52.69%	0:07:04
1	40	15	1	1000	44.77%	60.46%	0:09:02
1	40	15	2	1000	51.42%	66.07%	0:10:09
1	40	15	3	1000	55.67%	70.38%	0:10:51
1	40	15	0	1500	36.37%	53.14%	0:09:46
1	40	15	1	1500	45.90%	61.22%	0:11:58

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	40	15	2	1500	53.13%	68.40%	0:13:13
1	40	15	3	1500	57.40%	70.74%	0:14:09
1	40	20	0	100	31.95%	47.40%	0:02:41
1	40	20	1	100	35.61%	53.05%	0:04:12
1	40	20	2	100	35.67%	52.78%	0:05:05
1	40	20	0	250	34.88%	50.72%	0:03:30
1	40	20	1	250	40.32%	56.55%	0:05:05
1	40	20	2	250	42.92%	59.20%	0:06:01
1	40	20	0	500	37.07%	53.55%	0:04:59
1	40	20	1	500	45.45%	61.89%	0:06:38
1	40	20	2	500	51.11%	67.15%	0:07:38
1	40	20	0	750	37.76%	54.67%	0:06:18
1	40	20	1	750	46.08%	61.71%	0:08:14
1	40	20	2	750	50.77%	65.04%	0:09:13
1	40	20	0	1000	37.66%	53.95%	0:07:40
1	40	20	1	1000	47.20%	62.57%	0:09:50
1	40	20	2	1000	52.83%	67.37%	0:10:58
1	40	20	0	1500	38.70%	54.89%	0:10:29
1	40	20	1	1500	49.80%	65.22%	0:12:50
1	40	20	2	1500	55.30%	70.38%	0:14:20
1	40	30	0	100	33.46%	49.24%	0:03:29
1	40	30	1	100	34.46%	51.30%	0:05:04
1	40	30	0	250	36.92%	52.96%	0:04:22
1	40	30	1	250	39.44%	55.52%	0:06:02

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	40	30	0	500	39.32%	55.30%	0:05:46
1	40	30	1	500	44.62%	61.00%	0:07:44
1	40	30	0	750	40.57%	56.01%	0:07:15
1	40	30	1	750	47.81%	63.73%	0:09:21
1	40	30	0	1000	41.82%	57.41%	0:08:41
1	40	30	1	1000	49.35%	64.90%	0:10:55
1	40	30	0	1500	42.53%	58.44%	0:11:37
1	40	30	1	1500	50.90%	65.22%	0:14:20
1	50	15	0	100	30.52%	46.81%	0:02:15
1	50	15	1	100	33.62%	50.27%	0:04:18
1	50	15	0	250	33.45%	50.04%	0:03:05
1	50	15	1	250	38.27%	54.85%	0:05:15
1	50	15	0	500	35.51%	51.57%	0:04:28
1	50	15	1	500	42.97%	58.93%	0:06:49
1	50	15	0	750	36.73%	52.15%	0:05:49
1	50	15	1	750	44.83%	61.00%	0:08:20
1	50	15	0	1000	37.16%	53.59%	0:07:11
1	50	15	1	1000	45.97%	60.91%	0:09:51
1	50	15	0	1500	38.30%	55.21%	0:09:57
1	50	15	1	1500	47.08%	62.52%	0:13:04
1	50	20	0	100	32.79%	50.04%	0:02:59
1	50	20	1	100	34.92%	51.44%	0:05:02
1	50	20	0	250	35.84%	52.96%	0:03:48
1	50	20	1	250	39.62%	56.96%	0:05:59

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
1	50	20	0	500	37.95%	54.80%	0:05:14
1	50	20	1	500	44.08%	61.80%	0:07:38
1	50	20	0	750	38.78%	55.30%	0:06:42
1	50	20	1	750	45.82%	62.39%	0:09:16
1	50	20	0	1000	38.87%	55.16%	0:08:11
1	50	20	1	1000	46.99%	62.75%	0:11:00
1	50	20	0	1500	40.52%	56.55%	0:10:53
1	50	20	1	1500	49.00%	65.17%	0:14:04
1	50	30	0	100	32.80%	50.13%	0:04:05
1	50	30	0	250	35.78%	52.15%	0:04:56
1	50	30	0	500	38.63%	55.34%	0:06:25
1	50	30	0	750	40.09%	57.00%	0:07:55
1	50	30	0	1000	40.74%	57.09%	0:09:23
1	50	30	0	1500	42.56%	58.39%	0:12:34
2	18	15	0	100	22.07%	35.73%	0:02:35
2	18	15	1	100	27.44%	42.32%	0:03:39
2	18	15	2	100	32.95%	48.83%	0:04:09
2	18	15	3	100	36.67%	52.56%	0:04:31
2	18	15	0	250	22.40%	36.54%	0:04:19
2	18	15	1	250	27.53%	40.75%	0:05:23
2	18	15	2	250	33.76%	48.03%	0:05:51
2	18	15	3	250	40.64%	55.97%	0:06:15
2	18	15	0	500	23.42%	36.89%	0:06:58
2	18	15	1	500	30.05%	44.25%	0:08:08

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	18	15	2	500	36.48%	50.72%	0:08:48
2	18	15	3	500	42.44%	56.24%	0:09:08
2	18	15	0	750	24.69%	38.42%	0:09:42
2	18	15	1	750	31.68%	45.60%	0:10:56
2	18	15	2	750	38.90%	53.14%	0:11:30
2	18	15	3	750	45.18%	59.25%	0:11:56
2	18	15	0	1000	23.81%	36.89%	0:12:22
2	18	15	1	1000	30.03%	42.59%	0:13:52
2	18	15	2	1000	36.29%	49.10%	0:14:17
2	18	15	3	1000	42.90%	55.30%	0:14:49
2	18	15	0	1500	23.74%	37.07%	0:17:55
2	18	15	1	1500	30.08%	42.55%	0:19:28
2	18	15	2	1500	35.48%	47.13%	0:20:13
2	18	15	3	1500	41.81%	52.51%	0:20:36
2	18	20	0	100	24.48%	38.87%	0:03:02
2	18	20	1	100	26.44%	40.93%	0:03:21
2	18	20	2	100	29.18%	42.77%	0:03:46
2	18	20	0	250	26.93%	40.84%	0:04:43
2	18	20	1	250	30.41%	43.09%	0:05:06
2	18	20	2	250	33.79%	47.44%	0:05:27
2	18	20	0	500	28.35%	42.50%	0:07:28
2	18	20	1	500	34.81%	48.47%	0:07:48
2	18	20	2	500	38.37%	51.75%	0:08:14
2	18	20	0	750	28.94%	43.13%	0:10:06

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	18	20	1	750	34.55%	48.79%	0:10:36
2	18	20	2	750	37.52%	50.31%	0:11:07
2	18	20	0	1000	28.91%	43.49%	0:12:54
2	18	20	1	1000	35.64%	49.69%	0:13:28
2	18	20	2	1000	39.47%	51.75%	0:13:52
2	18	20	0	1500	28.91%	42.73%	0:18:40
2	18	20	1	1500	33.99%	47.49%	0:18:57
2	18	20	2	1500	38.34%	50.58%	0:19:36
2	18	30	0	100	30.70%	46.10%	0:03:34
2	18	30	1	100	35.66%	50.76%	0:04:31
2	18	30	0	250	32.46%	47.71%	0:05:14
2	18	30	1	250	40.84%	56.51%	0:06:17
2	18	30	0	500	33.85%	48.56%	0:08:00
2	18	30	1	500	42.34%	56.78%	0:09:06
2	18	30	0	750	34.97%	49.60%	0:10:46
2	18	30	1	750	44.06%	58.84%	0:12:05
2	18	30	0	1000	34.92%	49.24%	0:13:30
2	18	30	1	1000	44.13%	58.17%	0:14:50
2	18	30	0	1500	35.16%	49.24%	0:19:09
2	18	30	1	1500	43.94%	57.27%	0:20:33
2	20	10	0	100	16.49%	29.62%	0:02:30
2	20	10	1	100	21.07%	33.98%	0:03:08
2	20	10	2	100	24.56%	37.39%	0:03:49
2	20	10	3	100	26.41%	38.55%	0:04:12

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	20	10	4	100	30.57%	43.94%	0:04:37
2	20	10	5	100	35.76%	49.82%	0:05:14
2	20	10	0	250	17.61%	30.75%	0:04:07
2	20	10	1	250	22.89%	36.27%	0:04:49
2	20	10	2	250	25.97%	38.29%	0:05:29
2	20	10	3	250	29.82%	42.68%	0:05:55
2	20	10	4	250	34.16%	47.89%	0:06:15
2	20	10	5	250	39.15%	53.46%	0:06:58
2	20	10	0	500	17.67%	29.98%	0:06:51
2	20	10	1	500	21.45%	32.85%	0:07:40
2	20	10	2	500	24.06%	35.19%	0:08:19
2	20	10	3	500	28.23%	40.08%	0:08:45
2	20	10	4	500	33.76%	46.32%	0:09:06
2	20	10	5	500	39.32%	52.15%	0:09:54
2	20	10	0	750	17.25%	29.58%	0:09:34
2	20	10	1	750	21.16%	31.82%	0:10:27
2	20	10	2	750	25.15%	35.59%	0:11:07
2	20	10	3	750	30.23%	41.02%	0:11:40
2	20	10	4	750	35.26%	47.04%	0:12:00
2	20	10	5	750	40.72%	52.33%	0:12:58
2	20	10	0	1000	17.43%	29.58%	0:12:16
2	20	10	1	1000	21.43%	32.72%	0:13:18
2	20	10	2	1000	24.38%	33.98%	0:14:03
2	20	10	3	1000	29.26%	39.50%	0:14:26

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	20	10	4	1000	34.15%	44.84%	0:14:50
2	20	10	5	1000	38.70%	49.73%	0:15:46
2	20	10	0	1500	17.39%	28.86%	0:17:43
2	20	10	1	1500	20.58%	31.24%	0:19:15
2	20	10	2	1500	23.13%	33.17%	0:19:58
2	20	10	3	1500	25.69%	35.01%	0:20:15
2	20	10	4	1500	28.98%	37.93%	0:20:32
2	20	10	5	1500	33.29%	42.10%	0:21:34
2	20	10	0	2000	14.59%	25.31%	0:23:08
2	20	10	1	2000	17.38%	27.51%	0:24:16
2	20	10	2	2000	18.97%	27.78%	0:25:13
2	20	10	3	2000	20.80%	28.59%	0:25:53
2	20	10	4	2000	23.77%	32.23%	0:26:22
2	20	10	5	2000	26.55%	35.32%	0:27:35
2	20	15	0	100	26.25%	41.61%	0:02:39
2	20	15	1	100	29.38%	44.84%	0:03:48
2	20	15	2	100	34.37%	50.27%	0:04:22
2	20	15	3	100	38.07%	54.08%	0:04:49
2	20	15	0	250	27.54%	42.32%	0:04:16
2	20	15	1	250	32.11%	47.26%	0:05:34
2	20	15	2	250	38.47%	53.82%	0:06:04
2	20	15	3	250	45.16%	61.00%	0:06:32
2	20	15	0	500	28.83%	43.72%	0:07:04
2	20	15	1	500	33.80%	48.79%	0:08:19

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	20	15	2	500	40.50%	55.66%	0:08:57
2	20	15	3	500	47.48%	62.21%	0:09:28
2	20	15	0	750	29.37%	44.34%	0:09:45
2	20	15	1	750	34.04%	49.01%	0:11:09
2	20	15	2	750	41.15%	56.37%	0:11:55
2	20	15	3	750	48.13%	62.70%	0:12:33
2	20	15	0	1000	28.29%	43.67%	0:12:24
2	20	15	1	1000	32.95%	47.49%	0:14:04
2	20	15	2	1000	40.74%	54.71%	0:14:52
2	20	15	3	1000	47.36%	61.62%	0:15:11
2	20	15	0	1500	28.78%	43.49%	0:17:47
2	20	15	1	1500	32.79%	46.36%	0:19:37
2	20	15	2	1500	39.71%	53.19%	0:20:18
2	20	15	3	1500	46.76%	60.64%	0:21:03
2	20	15	0	2000	28.27%	42.59%	0:23:15
2	20	15	1	2000	33.30%	46.54%	0:25:07
2	20	15	2	2000	39.36%	52.42%	0:26:03
2	20	15	3	2000	45.72%	58.66%	0:26:48
2	20	20	0	100	27.58%	42.50%	0:03:06
2	20	20	1	100	28.89%	42.77%	0:03:28
2	20	20	2	100	30.73%	45.29%	0:03:56
2	20	20	0	250	28.15%	42.86%	0:04:45
2	20	20	1	250	33.15%	48.03%	0:05:08
2	20	20	2	250	38.20%	52.38%	0:05:37

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	20	20	0	500	29.53%	44.21%	0:07:31
2	20	20	1	500	36.55%	51.93%	0:07:52
2	20	20	2	500	42.28%	56.73%	0:08:34
2	20	20	0	750	29.57%	43.36%	0:10:15
2	20	20	1	750	36.74%	50.72%	0:10:44
2	20	20	2	750	42.16%	55.34%	0:11:13
2	20	20	0	1000	29.51%	43.99%	0:12:59
2	20	20	1	1000	36.96%	50.22%	0:13:32
2	20	20	2	1000	42.73%	55.70%	0:14:02
2	20	20	0	1500	28.88%	42.73%	0:18:26
2	20	20	1	1500	35.68%	47.80%	0:19:01
2	20	20	2	1500	40.70%	52.24%	0:19:47
2	20	20	0	2000	28.25%	41.02%	0:23:49
2	20	20	1	2000	35.57%	47.53%	0:24:36
2	20	20	2	2000	38.93%	50.22%	0:25:17
2	20	30	0	100	33.65%	48.70%	0:03:46
2	20	30	1	100	38.95%	54.89%	0:04:50
2	20	30	0	250	36.83%	52.02%	0:05:24
2	20	30	1	250	44.81%	60.73%	0:06:34
2	20	30	0	500	39.63%	55.34%	0:08:09
2	20	30	1	500	48.79%	64.32%	0:09:32
2	20	30	0	750	40.14%	55.12%	0:11:02
2	20	30	1	750	50.11%	65.13%	0:12:19
2	20	30	0	1000	40.55%	55.16%	0:14:01

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	20	30	1	1000	50.52%	64.72%	0:15:26
2	20	30	0	1500	40.94%	55.34%	0:19:16
2	20	30	1	1500	50.97%	65.31%	0:20:52
2	20	30	0	2000	41.17%	55.52%	0:24:49
2	20	30	1	2000	50.30%	64.05%	0:27:08
2	25	15	0	100	29.21%	45.69%	0:02:46
2	25	15	1	100	35.36%	51.03%	0:04:17
2	25	15	2	100	39.50%	55.97%	0:05:04
2	25	15	3	100	44.13%	60.50%	0:05:37
2	25	15	0	250	32.04%	48.79%	0:04:24
2	25	15	1	250	39.79%	56.51%	0:05:56
2	25	15	2	250	45.72%	61.98%	0:06:46
2	25	15	3	250	51.17%	67.01%	0:07:23
2	25	15	0	500	32.91%	48.79%	0:07:04
2	25	15	1	500	40.69%	56.87%	0:08:49
2	25	15	2	500	48.33%	64.05%	0:09:44
2	25	15	3	500	54.69%	69.17%	0:10:25
2	25	15	0	750	33.39%	50.58%	0:09:51
2	25	15	1	750	42.27%	58.26%	0:11:42
2	25	15	2	750	49.22%	64.54%	0:12:31
2	25	15	3	750	55.57%	69.39%	0:13:17
2	25	15	0	1000	33.77%	51.17%	0:12:32
2	25	15	1	1000	42.27%	57.94%	0:14:31
2	25	15	2	1000	49.72%	64.90%	0:15:27

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	25	15	3	1000	56.28%	69.61%	0:16:15
2	25	15	0	1500	33.30%	50.63%	0:17:59
2	25	15	1	1500	42.33%	57.32%	0:20:10
2	25	15	2	1500	50.18%	65.62%	0:21:17
2	25	15	3	1500	56.37%	69.17%	0:22:09
2	25	20	0	100	30.15%	46.95%	0:03:22
2	25	20	1	100	36.71%	53.90%	0:04:48
2	25	20	2	100	41.61%	58.30%	0:05:39
2	25	20	0	250	32.91%	48.11%	0:05:04
2	25	20	1	250	41.06%	57.99%	0:06:32
2	25	20	2	250	48.30%	63.42%	0:07:23
2	25	20	0	500	34.37%	50.18%	0:07:49
2	25	20	1	500	44.50%	60.41%	0:09:24
2	25	20	2	500	51.99%	67.55%	0:10:23
2	25	20	0	750	34.64%	50.49%	0:10:32
2	25	20	1	750	44.94%	59.96%	0:12:16
2	25	20	2	750	52.63%	67.19%	0:13:15
2	25	20	0	1000	35.19%	50.27%	0:13:15
2	25	20	1	1000	45.78%	59.87%	0:15:10
2	25	20	2	1000	53.99%	68.76%	0:16:16
2	25	20	0	1500	34.81%	49.55%	0:19:01
2	25	20	1	1500	45.04%	58.17%	0:20:58
2	25	20	2	1500	53.77%	66.65%	0:22:10
2	25	30	0	100	34.67%	50.49%	0:04:10

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	25	30	1	100	41.20%	58.53%	0:05:38
2	25	30	0	250	37.40%	52.60%	0:05:51
2	25	30	1	250	46.44%	61.76%	0:07:24
2	25	30	0	500	40.77%	56.46%	0:08:40
2	25	30	1	500	50.93%	66.20%	0:10:19
2	25	30	0	750	41.29%	56.60%	0:11:36
2	25	30	1	750	51.09%	65.17%	0:13:18
2	25	30	0	1000	42.69%	58.08%	0:14:14
2	25	30	1	1000	53.48%	67.06%	0:16:12
2	25	30	0	1500	44.01%	58.48%	0:19:50
2	25	30	1	1500	53.93%	67.10%	0:22:11
2	30	15	0	100	29.52%	46.32%	0:02:53
2	30	15	1	100	36.39%	53.77%	0:04:46
2	30	15	2	100	42.14%	60.73%	0:05:46
2	30	15	3	100	45.38%	62.48%	0:06:30
2	30	15	0	250	32.41%	48.88%	0:04:30
2	30	15	1	250	41.66%	58.30%	0:06:35
2	30	15	2	250	48.14%	63.96%	0:07:32
2	30	15	3	250	52.83%	68.04%	0:08:22
2	30	15	0	500	33.50%	50.04%	0:07:16
2	30	15	1	500	43.99%	60.05%	0:09:27
2	30	15	2	500	51.43%	66.29%	0:10:36
2	30	15	3	500	56.63%	71.32%	0:11:20
2	30	15	0	750	34.98%	51.97%	0:10:00

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	30	15	1	750	45.59%	61.09%	0:12:29
2	30	15	2	750	53.86%	68.36%	0:13:29
2	30	15	3	750	59.41%	71.68%	0:14:25
2	30	15	0	1000	35.56%	51.80%	0:12:44
2	30	15	1	1000	45.63%	61.13%	0:15:06
2	30	15	2	1000	54.15%	68.18%	0:16:28
2	30	15	3	1000	59.46%	71.86%	0:17:31
2	30	15	0	1500	36.42%	53.05%	0:18:07
2	30	15	1	1500	47.44%	61.76%	0:20:58
2	30	15	2	1500	55.15%	68.90%	0:22:21
2	30	15	3	1500	60.31%	72.62%	0:23:43
2	30	20	0	100	32.20%	49.96%	0:03:38
2	30	20	1	100	38.73%	55.70%	0:05:27
2	30	20	2	100	41.95%	58.84%	0:06:34
2	30	20	0	250	35.40%	52.33%	0:05:17
2	30	20	1	250	44.67%	61.58%	0:07:17
2	30	20	2	250	49.93%	65.66%	0:08:25
2	30	20	0	500	37.30%	54.22%	0:08:02
2	30	20	1	500	47.28%	63.33%	0:10:18
2	30	20	2	500	54.07%	69.03%	0:11:26
2	30	20	0	750	37.18%	54.08%	0:10:51
2	30	20	1	750	47.90%	63.87%	0:13:06
2	30	20	2	750	55.40%	70.15%	0:14:39
2	30	20	0	1000	38.06%	54.67%	0:13:34

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	30	20	1	1000	50.15%	64.86%	0:16:05
2	30	20	2	1000	58.27%	71.95%	0:17:26
2	30	20	0	1500	38.01%	54.22%	0:19:06
2	30	20	1	1500	50.70%	65.62%	0:21:57
2	30	20	2	1500	58.36%	71.63%	0:23:32
2	30	30	0	100	37.36%	54.22%	0:04:40
2	30	30	1	100	42.04%	58.35%	0:06:32
2	30	30	0	250	40.67%	56.78%	0:06:21
2	30	30	1	250	49.18%	65.04%	0:08:22
2	30	30	0	500	43.56%	58.89%	0:09:19
2	30	30	1	500	51.84%	67.10%	0:11:27
2	30	30	0	750	44.54%	59.69%	0:12:00
2	30	30	1	750	54.39%	68.76%	0:14:40
2	30	30	0	1000	45.72%	61.18%	0:14:51
2	30	30	1	1000	55.38%	69.43%	0:17:29
2	30	30	0	1500	46.78%	61.27%	0:20:53
2	30	30	1	1500	56.56%	70.51%	0:23:34
2	40	15	0	100	31.75%	48.29%	0:03:05
2	40	15	1	100	38.05%	56.06%	0:05:55
2	40	15	2	100	41.57%	58.53%	0:07:21
2	40	15	3	100	44.05%	60.46%	0:08:32
2	40	15	0	250	34.35%	50.27%	0:04:46
2	40	15	1	250	43.91%	60.14%	0:07:41
2	40	15	2	250	49.79%	65.71%	0:09:12

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	40	15	3	250	53.14%	68.36%	0:10:25
2	40	15	0	500	37.36%	53.50%	0:07:31
2	40	15	1	500	46.75%	62.61%	0:10:42
2	40	15	2	500	54.11%	69.57%	0:12:19
2	40	15	3	500	57.64%	71.95%	0:13:40
2	40	15	0	750	38.44%	54.67%	0:10:16
2	40	15	1	750	50.19%	66.02%	0:13:34
2	40	15	2	750	57.08%	71.27%	0:15:28
2	40	15	3	750	60.68%	73.88%	0:16:52
2	40	15	0	1000	39.22%	56.46%	0:12:55
2	40	15	1	1000	50.81%	65.75%	0:16:46
2	40	15	2	1000	57.31%	71.18%	0:18:36
2	40	15	3	1000	61.41%	74.01%	0:20:09
2	40	15	0	1500	39.78%	56.10%	0:19:15
2	40	15	1	1500	50.74%	65.17%	0:22:30
2	40	15	2	1500	57.94%	71.59%	0:24:56
2	40	15	3	1500	62.13%	74.24%	0:26:43
2	40	20	0	100	35.05%	52.60%	0:04:08
2	40	20	1	100	40.48%	58.12%	0:06:53
2	40	20	2	100	42.82%	60.05%	0:08:29
2	40	20	0	250	37.59%	53.90%	0:05:50
2	40	20	1	250	45.00%	61.27%	0:08:43
2	40	20	2	250	50.91%	67.01%	0:10:25
2	40	20	0	500	40.47%	56.42%	0:08:36

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	40	20	1	500	50.79%	67.15%	0:11:49
2	40	20	2	500	55.86%	70.83%	0:13:39
2	40	20	0	750	40.97%	57.18%	0:11:25
2	40	20	1	750	51.28%	66.02%	0:14:54
2	40	20	2	750	57.44%	71.72%	0:16:56
2	40	20	0	1000	42.24%	57.90%	0:14:16
2	40	20	1	1000	53.44%	68.45%	0:17:59
2	40	20	2	1000	59.45%	72.62%	0:20:08
2	40	20	0	1500	42.30%	58.26%	0:19:47
2	40	20	1	1500	54.11%	67.95%	0:24:13
2	40	20	2	1500	60.13%	72.94%	0:26:41
2	40	30	0	100	37.27%	54.13%	0:05:39
2	40	30	1	100	39.62%	56.46%	0:08:31
2	40	30	0	250	41.80%	58.39%	0:07:27
2	40	30	1	250	47.40%	62.84%	0:10:28
2	40	30	0	500	44.34%	60.01%	0:10:24
2	40	30	1	500	52.70%	67.59%	0:13:45
2	40	30	0	750	45.91%	61.49%	0:13:10
2	40	30	1	750	55.40%	69.39%	0:17:00
2	40	30	0	1000	47.54%	62.34%	0:16:06
2	40	30	1	1000	56.81%	70.51%	0:20:11
2	40	30	0	1500	48.94%	63.24%	0:21:58
2	40	30	1	1500	57.88%	71.36%	0:27:04
2	50	15	0	100	33.25%	49.82%	0:03:19

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	50	15	1	100	38.25%	54.94%	0:07:07
2	50	15	0	250	36.54%	53.46%	0:05:03
2	50	15	1	250	43.58%	60.68%	0:08:57
2	50	15	0	500	39.17%	55.75%	0:07:45
2	50	15	1	500	48.46%	64.50%	0:12:10
2	50	15	0	750	39.94%	56.28%	0:10:27
2	50	15	1	750	49.51%	64.81%	0:15:11
2	50	15	0	1000	41.19%	57.36%	0:13:13
2	50	15	1	1000	50.89%	66.34%	0:18:28
2	50	15	0	1500	42.06%	57.27%	0:18:48
2	50	15	1	1500	52.61%	67.86%	0:24:34
2	50	20	0	100	35.87%	53.28%	0:04:43
2	50	20	1	100	38.99%	55.75%	0:08:30
2	50	20	0	250	38.80%	55.43%	0:06:25
2	50	20	1	250	44.97%	60.77%	0:10:43
2	50	20	0	500	40.92%	56.96%	0:09:16
2	50	20	1	500	49.38%	64.95%	0:13:41
2	50	20	0	750	42.24%	58.21%	0:12:06
2	50	20	1	750	51.58%	67.24%	0:16:55
2	50	20	0	1000	43.67%	59.25%	0:14:54
2	50	20	1	1000	53.22%	67.55%	0:20:09
2	50	20	0	1500	44.40%	60.55%	0:20:36
2	50	20	1	1500	55.53%	69.70%	0:26:42
2	50	30	0	100	36.69%	53.46%	0:06:50

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
2	50	30	0	250	40.21%	56.55%	0:08:32
2	50	30	0	500	43.27%	59.78%	0:11:40
2	50	30	0	750	45.28%	61.09%	0:14:28
2	50	30	0	1000	46.39%	61.94%	0:17:33
2	50	30	0	1500	47.40%	62.48%	0:23:45
3	18	15	0	100	22.46%	36.00%	0:03:20
3	18	15	1	100	28.36%	42.55%	0:04:53
3	18	15	2	100	33.06%	47.85%	0:05:41
3	18	15	3	100	38.49%	53.90%	0:06:10
3	18	15	0	250	23.30%	36.49%	0:05:49
3	18	15	1	250	29.72%	44.39%	0:07:25
3	18	15	2	250	36.64%	50.85%	0:08:12
3	18	15	3	250	42.56%	57.27%	0:08:45
3	18	15	0	500	24.25%	38.20%	0:09:51
3	18	15	1	500	30.60%	43.94%	0:11:38
3	18	15	2	500	37.24%	50.54%	0:12:24
3	18	15	3	500	44.44%	58.12%	0:13:03
3	18	15	0	750	24.85%	38.42%	0:13:57
3	18	15	1	750	31.01%	44.30%	0:15:48
3	18	15	2	750	37.22%	50.27%	0:16:38
3	18	15	3	750	44.16%	56.64%	0:17:16
3	18	15	0	1000	24.69%	37.16%	0:18:08
3	18	15	1	1000	29.40%	41.20%	0:20:04
3	18	15	2	1000	34.38%	46.41%	0:20:58

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	18	15	3	1000	41.34%	52.92%	0:21:37
3	18	15	0	1500	23.47%	35.64%	0:26:25
3	18	15	1	1500	27.02%	38.73%	0:28:32
3	18	15	2	1500	30.87%	41.25%	0:29:22
3	18	15	3	1500	36.21%	45.87%	0:30:43
3	18	20	0	100	26.13%	40.57%	0:03:56
3	18	20	1	100	27.95%	42.46%	0:04:25
3	18	20	2	100	31.00%	45.56%	0:05:02
3	18	20	0	250	27.89%	42.55%	0:06:28
3	18	20	1	250	31.50%	45.11%	0:07:01
3	18	20	2	250	35.98%	49.37%	0:07:33
3	18	20	0	500	29.86%	44.66%	0:10:33
3	18	20	1	500	34.58%	48.07%	0:11:16
3	18	20	2	500	40.38%	52.38%	0:11:46
3	18	20	0	750	28.62%	42.86%	0:14:37
3	18	20	1	750	35.15%	48.70%	0:15:30
3	18	20	2	750	39.33%	51.35%	0:16:01
3	18	20	0	1000	29.17%	43.36%	0:18:49
3	18	20	1	1000	34.66%	47.31%	0:19:31
3	18	20	2	1000	39.86%	51.84%	0:20:13
3	18	20	0	1500	26.77%	39.41%	0:26:58
3	18	20	1	1500	32.14%	44.57%	0:28:00
3	18	20	2	1500	34.09%	46.01%	0:28:40
3	18	30	0	100	31.61%	47.04%	0:04:47

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	18	30	1	100	39.23%	54.98%	0:06:11
3	18	30	0	250	34.49%	50.13%	0:07:17
3	18	30	1	250	43.86%	60.01%	0:08:45
3	18	30	0	500	35.64%	50.40%	0:11:34
3	18	30	1	500	45.48%	60.73%	0:13:00
3	18	30	0	750	36.16%	50.58%	0:15:36
3	18	30	1	750	46.75%	61.36%	0:17:28
3	18	30	0	1000	36.67%	50.49%	0:19:42
3	18	30	1	1000	47.30%	61.13%	0:21:37
3	18	30	0	1500	35.01%	49.15%	0:28:05
3	18	30	1	1500	45.50%	59.43%	0:30:27
3	20	10	0	100	17.66%	29.71%	0:03:11
3	20	10	1	100	21.63%	34.47%	0:04:08
3	20	10	2	100	24.21%	36.45%	0:05:07
3	20	10	3	100	28.36%	41.70%	0:05:43
3	20	10	4	100	32.23%	46.05%	0:06:12
3	20	10	5	100	37.30%	51.89%	0:07:08
3	20	10	0	250	17.23%	29.22%	0:05:37
3	20	10	1	250	21.52%	33.30%	0:06:40
3	20	10	2	250	24.19%	35.05%	0:07:45
3	20	10	3	250	28.78%	40.53%	0:08:13
3	20	10	4	250	34.97%	48.07%	0:08:47
3	20	10	5	250	41.16%	55.75%	0:09:44
3	20	10	0	500	17.40%	28.99%	0:09:43

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	20	10	1	500	22.34%	33.66%	0:11:02
3	20	10	2	500	26.05%	37.30%	0:11:51
3	20	10	3	500	32.70%	44.61%	0:12:35
3	20	10	4	500	38.50%	50.36%	0:13:12
3	20	10	5	500	45.57%	57.90%	0:14:12
3	20	10	0	750	17.94%	29.76%	0:13:47
3	20	10	1	750	23.29%	35.14%	0:15:01
3	20	10	2	750	26.39%	36.89%	0:16:04
3	20	10	3	750	31.15%	41.61%	0:16:40
3	20	10	4	750	36.31%	47.17%	0:17:18
3	20	10	5	750	42.40%	53.73%	0:18:27
3	20	10	0	1000	17.60%	29.80%	0:17:56
3	20	10	1	1000	21.07%	31.82%	0:19:16
3	20	10	2	1000	24.87%	34.92%	0:20:18
3	20	10	3	1000	28.95%	39.59%	0:21:17
3	20	10	4	1000	32.19%	42.01%	0:21:38
3	20	10	5	1000	38.44%	49.24%	0:23:20
3	20	10	0	1500	13.89%	23.83%	0:26:09
3	20	10	1	1500	15.23%	24.69%	0:27:34
3	20	10	2	1500	16.93%	25.13%	0:28:51
3	20	10	3	1500	18.00%	24.60%	0:29:50
3	20	10	4	1500	19.12%	26.53%	0:30:29
3	20	10	5	1500	21.22%	28.64%	0:31:34
3	20	10	0	2000	13.38%	23.52%	0:34:12

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	20	10	1	2000	14.89%	24.19%	0:36:39
3	20	10	2	2000	16.14%	24.60%	0:37:22
3	20	10	3	2000	15.88%	23.16%	0:38:26
3	20	10	4	2000	17.07%	23.97%	0:38:54
3	20	10	5	2000	18.77%	25.94%	0:40:28
3	20	15	0	100	26.69%	42.46%	0:03:24
3	20	15	1	100	30.85%	46.27%	0:05:07
3	20	15	2	100	36.80%	52.29%	0:05:56
3	20	15	3	100	41.26%	57.94%	0:06:35
3	20	15	0	250	28.36%	44.39%	0:05:51
3	20	15	1	250	33.32%	49.24%	0:07:42
3	20	15	2	250	40.82%	56.64%	0:08:33
3	20	15	3	250	47.29%	62.79%	0:09:15
3	20	15	0	500	29.76%	45.06%	0:09:52
3	20	15	1	500	33.89%	48.38%	0:11:51
3	20	15	2	500	41.59%	56.33%	0:12:51
3	20	15	3	500	49.74%	63.64%	0:13:30
3	20	15	0	750	29.70%	44.88%	0:14:01
3	20	15	1	750	35.53%	50.18%	0:16:09
3	20	15	2	750	43.82%	58.12%	0:17:09
3	20	15	3	750	51.44%	65.80%	0:17:53
3	20	15	0	1000	29.05%	43.54%	0:18:20
3	20	15	1	1000	34.30%	48.38%	0:20:18
3	20	15	2	1000	40.42%	53.68%	0:21:36

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	20	15	3	1000	47.90%	61.22%	0:22:06
3	20	15	0	1500	26.99%	41.52%	0:26:16
3	20	15	1	1500	32.66%	46.10%	0:29:12
3	20	15	2	1500	38.59%	51.80%	0:30:00
3	20	15	3	1500	46.04%	58.98%	0:30:46
3	20	15	0	2000	26.27%	41.02%	0:34:22
3	20	15	1	2000	30.75%	44.70%	0:37:29
3	20	15	2	2000	34.87%	47.26%	0:38:29
3	20	15	3	2000	39.99%	51.93%	0:39:39
3	20	20	0	100	27.76%	42.01%	0:04:05
3	20	20	1	100	31.68%	46.32%	0:04:38
3	20	20	2	100	34.95%	49.37%	0:05:16
3	20	20	0	250	29.12%	43.99%	0:06:32
3	20	20	1	250	36.14%	51.08%	0:07:09
3	20	20	2	250	41.45%	54.98%	0:07:50
3	20	20	0	500	30.25%	44.70%	0:10:44
3	20	20	1	500	38.71%	52.42%	0:11:16
3	20	20	2	500	43.74%	57.18%	0:12:05
3	20	20	0	750	29.93%	44.03%	0:14:56
3	20	20	1	750	38.91%	52.83%	0:15:28
3	20	20	2	750	44.39%	57.54%	0:16:17
3	20	20	0	1000	28.65%	42.01%	0:18:55
3	20	20	1	1000	38.13%	51.26%	0:19:53
3	20	20	2	1000	44.45%	57.59%	0:20:35

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	20	20	0	1500	27.84%	41.16%	0:27:10
3	20	20	1	1500	35.45%	47.17%	0:28:36
3	20	20	2	1500	40.10%	51.62%	0:28:51
3	20	20	0	2000	26.22%	38.64%	0:35:18
3	20	20	1	2000	32.73%	44.08%	0:36:25
3	20	20	2	2000	36.89%	46.99%	0:37:23
3	20	30	0	100	35.23%	50.22%	0:05:01
3	20	30	1	100	42.72%	59.83%	0:06:33
3	20	30	0	250	38.83%	54.13%	0:07:29
3	20	30	1	250	47.97%	63.06%	0:09:13
3	20	30	0	500	42.30%	58.08%	0:11:39
3	20	30	1	500	51.93%	66.83%	0:13:40
3	20	30	0	750	41.82%	56.82%	0:15:52
3	20	30	1	750	52.90%	67.64%	0:17:45
3	20	30	0	1000	42.25%	57.18%	0:20:00
3	20	30	1	1000	53.23%	67.41%	0:22:11
3	20	30	0	1500	42.40%	56.33%	0:28:19
3	20	30	1	1500	53.89%	67.59%	0:30:54
3	20	30	0	2000	42.42%	56.46%	0:36:50
3	20	30	1	2000	52.80%	66.74%	0:39:48
3	25	15	0	100	30.43%	46.59%	0:03:34
3	25	15	1	100	38.37%	55.57%	0:05:47
3	25	15	2	100	43.72%	60.46%	0:06:55
3	25	15	3	100	48.14%	63.91%	0:07:49

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	25	15	0	250	32.44%	50.18%	0:06:01
3	25	15	1	250	41.76%	58.17%	0:08:20
3	25	15	2	250	48.86%	65.13%	0:09:30
3	25	15	3	250	54.33%	68.90%	0:10:24
3	25	15	0	500	33.84%	50.40%	0:10:06
3	25	15	1	500	43.61%	59.25%	0:12:34
3	25	15	2	500	51.64%	67.15%	0:13:49
3	25	15	3	500	58.27%	72.31%	0:14:43
3	25	15	0	750	33.61%	49.96%	0:14:10
3	25	15	1	750	43.54%	59.43%	0:16:46
3	25	15	2	750	52.33%	67.73%	0:18:11
3	25	15	3	750	58.88%	72.17%	0:19:10
3	25	15	0	1000	34.19%	50.63%	0:18:10
3	25	15	1	1000	44.15%	58.98%	0:21:13
3	25	15	2	1000	52.25%	67.41%	0:22:41
3	25	15	3	1000	58.52%	71.10%	0:24:01
3	25	15	0	1500	33.18%	49.33%	0:26:24
3	25	15	1	1500	41.80%	56.51%	0:29:43
3	25	15	2	1500	49.59%	63.78%	0:31:12
3	25	15	3	1500	56.05%	68.54%	0:33:07
3	25	20	0	100	31.23%	47.58%	0:04:28
3	25	20	1	100	39.86%	56.60%	0:06:36
3	25	20	2	100	44.65%	60.95%	0:07:44
3	25	20	0	250	34.39%	50.22%	0:06:55

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	25	20	1	250	43.96%	59.38%	0:09:08
3	25	20	2	250	51.12%	66.47%	0:10:23
3	25	20	0	500	35.79%	51.89%	0:11:02
3	25	20	1	500	47.62%	63.20%	0:13:30
3	25	20	2	500	55.37%	69.52%	0:14:51
3	25	20	0	750	35.90%	51.71%	0:15:29
3	25	20	1	750	47.81%	62.75%	0:17:44
3	25	20	2	750	56.61%	70.29%	0:19:18
3	25	20	0	1000	35.57%	50.31%	0:19:23
3	25	20	1	1000	47.25%	61.76%	0:22:09
3	25	20	2	1000	55.80%	68.67%	0:23:38
3	25	20	0	1500	34.64%	49.60%	0:27:42
3	25	20	1	1500	46.11%	60.68%	0:30:51
3	25	20	2	1500	54.65%	68.13%	0:33:08
3	25	30	0	100	36.51%	52.96%	0:05:40
3	25	30	1	100	43.31%	60.23%	0:07:45
3	25	30	0	250	40.75%	57.09%	0:08:11
3	25	30	1	250	50.35%	66.11%	0:10:24
3	25	30	0	500	42.09%	57.76%	0:12:28
3	25	30	1	500	53.15%	67.55%	0:14:48
3	25	30	0	750	44.41%	60.32%	0:16:34
3	25	30	1	750	55.02%	69.39%	0:19:14
3	25	30	0	1000	44.66%	59.92%	0:20:55
3	25	30	1	1000	55.87%	70.11%	0:24:02

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	25	30	0	1500	45.16%	59.87%	0:29:31
3	25	30	1	1500	57.40%	70.38%	0:32:51
3	30	15	0	100	30.42%	46.81%	0:03:43
3	30	15	1	100	39.10%	56.78%	0:06:34
3	30	15	2	100	44.92%	62.03%	0:07:57
3	30	15	3	100	48.45%	64.23%	0:09:04
3	30	15	0	250	33.98%	51.12%	0:06:15
3	30	15	1	250	43.27%	60.01%	0:09:08
3	30	15	2	250	51.20%	67.06%	0:10:39
3	30	15	3	250	56.23%	71.36%	0:11:45
3	30	15	0	500	35.65%	52.33%	0:10:21
3	30	15	1	500	46.22%	61.76%	0:13:25
3	30	15	2	500	54.60%	69.61%	0:15:01
3	30	15	3	500	59.74%	73.43%	0:16:18
3	30	15	0	750	35.88%	52.15%	0:14:23
3	30	15	1	750	46.33%	61.67%	0:17:46
3	30	15	2	750	55.85%	70.15%	0:19:28
3	30	15	3	750	61.58%	74.06%	0:20:50
3	30	15	0	1000	36.44%	52.24%	0:18:43
3	30	15	1	1000	48.56%	64.59%	0:22:15
3	30	15	2	1000	57.15%	70.65%	0:24:00
3	30	15	3	1000	61.92%	73.83%	0:25:30
3	30	15	0	1500	37.11%	52.65%	0:26:32
3	30	15	1	1500	48.69%	64.18%	0:30:46

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	30	15	2	1500	56.88%	70.24%	0:33:31
3	30	15	3	1500	61.82%	73.83%	0:34:32
3	30	20	0	100	33.15%	50.00%	0:04:51
3	30	20	1	100	41.91%	58.66%	0:07:30
3	30	20	2	100	46.84%	62.88%	0:09:05
3	30	20	0	250	37.52%	54.62%	0:07:21
3	30	20	1	250	48.35%	64.23%	0:10:13
3	30	20	2	250	54.59%	69.75%	0:11:43
3	30	20	0	500	37.41%	53.95%	0:11:29
3	30	20	1	500	50.08%	65.35%	0:14:35
3	30	20	2	500	57.90%	71.95%	0:16:22
3	30	20	0	750	39.34%	55.75%	0:15:41
3	30	20	1	750	51.62%	66.25%	0:19:09
3	30	20	2	750	59.32%	73.03%	0:20:49
3	30	20	0	1000	39.74%	56.37%	0:19:50
3	30	20	1	1000	52.64%	66.97%	0:23:19
3	30	20	2	1000	59.93%	72.31%	0:25:49
3	30	20	0	1500	39.32%	55.92%	0:28:04
3	30	20	1	1500	52.53%	66.56%	0:32:31
3	30	20	2	1500	59.48%	71.54%	0:35:10
3	30	30	0	100	38.85%	56.10%	0:06:24
3	30	30	1	100	45.20%	62.21%	0:09:04
3	30	30	0	250	43.58%	59.56%	0:08:54
3	30	30	1	250	51.88%	67.73%	0:11:47

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	30	30	0	500	45.84%	61.00%	0:13:15
3	30	30	1	500	55.10%	69.08%	0:16:17
3	30	30	0	750	47.85%	62.57%	0:17:42
3	30	30	1	750	57.56%	71.14%	0:20:55
3	30	30	0	1000	48.43%	62.70%	0:21:42
3	30	30	1	1000	59.29%	72.62%	0:25:28
3	30	30	0	1500	49.28%	63.64%	0:30:15
3	30	30	1	1500	60.16%	72.35%	0:34:37
3	40	15	0	100	33.87%	50.09%	0:04:06
3	40	15	1	100	41.03%	58.21%	0:08:05
3	40	15	2	100	45.79%	62.84%	0:10:19
3	40	15	3	100	48.87%	65.53%	0:11:58
3	40	15	0	250	37.10%	53.41%	0:06:32
3	40	15	1	250	46.80%	62.88%	0:10:47
3	40	15	2	250	53.25%	68.99%	0:13:03
3	40	15	3	250	57.70%	71.41%	0:14:46
3	40	15	0	500	38.66%	55.92%	0:10:40
3	40	15	1	500	49.86%	65.35%	0:15:12
3	40	15	2	500	56.78%	70.69%	0:17:45
3	40	15	3	500	61.00%	73.56%	0:19:34
3	40	15	0	750	41.04%	57.41%	0:14:42
3	40	15	1	750	52.30%	67.15%	0:19:44
3	40	15	2	750	59.07%	72.58%	0:22:26
3	40	15	3	750	63.49%	75.31%	0:24:28

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	40	15	0	1000	41.50%	57.54%	0:18:59
3	40	15	1	1000	53.42%	67.82%	0:24:11
3	40	15	2	1000	59.77%	72.62%	0:27:07
3	40	15	3	1000	63.90%	75.36%	0:29:37
3	40	15	0	1500	41.36%	57.41%	0:27:06
3	40	15	1	1500	53.83%	67.24%	0:33:08
3	40	15	2	1500	60.60%	72.85%	0:36:36
3	40	15	3	1500	63.96%	74.96%	0:39:02
3	40	20	0	100	36.03%	52.74%	0:05:38
3	40	20	1	100	43.01%	60.23%	0:09:34
3	40	20	2	100	46.19%	63.02%	0:11:55
3	40	20	0	250	40.10%	56.51%	0:08:07
3	40	20	1	250	49.11%	64.77%	0:12:18
3	40	20	2	250	54.41%	68.94%	0:14:48
3	40	20	0	500	41.75%	57.63%	0:12:18
3	40	20	1	500	53.17%	68.94%	0:16:58
3	40	20	2	500	57.93%	71.99%	0:19:37
3	40	20	0	750	42.95%	58.57%	0:16:33
3	40	20	1	750	54.67%	69.12%	0:21:31
3	40	20	2	750	60.58%	73.47%	0:24:33
3	40	20	0	1000	43.18%	58.93%	0:20:56
3	40	20	1	1000	54.27%	68.49%	0:26:17
3	40	20	2	1000	60.25%	73.34%	0:29:23
3	40	20	0	1500	44.27%	59.61%	0:29:04

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	40	20	1	1500	56.95%	70.06%	0:35:31
3	40	20	2	1500	62.28%	74.73%	0:39:08
3	40	30	0	100	39.16%	56.42%	0:07:50
3	40	30	1	100	42.12%	58.75%	0:11:56
3	40	30	0	250	44.42%	60.50%	0:10:26
3	40	30	1	250	51.24%	66.20%	0:14:49
3	40	30	0	500	46.43%	62.48%	0:14:47
3	40	30	1	500	55.13%	69.66%	0:19:40
3	40	30	0	750	48.96%	64.50%	0:19:16
3	40	30	1	750	58.31%	71.50%	0:24:47
3	40	30	0	1000	49.27%	63.51%	0:23:32
3	40	30	1	1000	59.95%	73.34%	0:29:29
3	40	30	0	1500	51.89%	65.35%	0:32:12
3	40	30	1	1500	61.28%	73.38%	0:39:44
3	50	15	0	100	35.71%	52.47%	0:04:25
3	50	15	1	100	41.13%	57.94%	0:09:57
3	50	15	0	250	38.64%	55.52%	0:06:55
3	50	15	1	250	47.60%	63.64%	0:12:41
3	50	15	0	500	41.41%	57.41%	0:11:04
3	50	15	1	500	50.82%	65.89%	0:17:27
3	50	15	0	750	42.46%	58.30%	0:15:20
3	50	15	1	750	53.20%	67.15%	0:22:03
3	50	15	0	1000	42.98%	58.35%	0:19:20
3	50	15	1	1000	54.31%	68.67%	0:26:48

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
3	50	15	0	1500	43.35%	58.84%	0:27:33
3	50	15	1	1500	54.02%	67.95%	0:36:04
3	50	20	0	100	37.33%	53.32%	0:06:27
3	50	20	1	100	41.48%	57.90%	0:11:52
3	50	20	0	250	40.46%	56.91%	0:09:01
3	50	20	1	250	48.74%	65.04%	0:14:49
3	50	20	0	500	43.96%	60.19%	0:13:12
3	50	20	1	500	53.29%	69.25%	0:19:51
3	50	20	0	750	43.98%	60.41%	0:17:45
3	50	20	1	750	55.05%	69.03%	0:24:33
3	50	20	0	1000	45.10%	61.04%	0:21:43
3	50	20	1	1000	56.35%	70.69%	0:29:21
3	50	20	0	1500	45.99%	60.82%	0:30:33
3	50	20	1	1500	56.42%	70.38%	0:39:46
3	50	30	0	100	38.49%	55.39%	0:09:27
3	50	30	0	250	43.21%	59.96%	0:12:06
3	50	30	0	500	46.75%	62.93%	0:16:47
3	50	30	0	750	48.26%	63.46%	0:20:58
3	50	30	0	1000	49.38%	64.77%	0:25:25
3	50	30	0	1500	50.26%	65.35%	0:34:18
4	18	15	0	100	23.45%	38.11%	0:04:04
4	18	15	1	100	30.21%	45.02%	0:06:05
4	18	15	2	100	36.92%	53.23%	0:07:05
4	18	15	3	100	41.05%	57.63%	0:07:48

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
4	18	15	0	250	24.62%	38.24%	0:07:19
4	18	15	1	250	30.17%	43.36%	0:09:24
4	18	15	2	250	38.44%	53.01%	0:10:23
4	18	15	3	250	44.97%	59.92%	0:11:12
4	18	15	0	500	25.98%	39.90%	0:12:46
4	18	15	1	500	33.08%	46.86%	0:15:03
4	18	15	2	500	40.71%	54.17%	0:16:03
4	18	15	3	500	48.34%	62.57%	0:16:58
4	18	15	0	750	25.55%	38.51%	0:18:14
4	18	15	1	750	30.61%	43.99%	0:20:46
4	18	15	2	750	37.96%	51.48%	0:21:46
4	18	15	3	750	45.01%	58.35%	0:22:32
4	18	15	0	1000	24.99%	37.43%	0:23:43
4	18	15	1	1000	29.54%	40.53%	0:26:15
4	18	15	2	1000	33.72%	45.38%	0:27:29
4	18	15	3	1000	40.56%	52.33%	0:28:27
4	18	15	0	1500	23.21%	36.45%	0:34:42
4	18	15	1	1500	26.90%	38.11%	0:37:37
4	18	15	2	1500	29.64%	40.13%	0:39:03
4	18	15	3	1500	34.15%	43.85%	0:40:13
4	18	20	0	100	26.32%	41.79%	0:04:54
4	18	20	1	100	28.69%	42.46%	0:05:32
4	18	20	2	100	32.14%	46.27%	0:06:19
4	18	20	0	250	28.59%	43.90%	0:08:09

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
4	18	20	1	250	34.01%	48.43%	0:08:49
4	18	20	2	250	39.19%	52.69%	0:09:38
4	18	20	0	500	29.79%	44.43%	0:13:40
4	18	20	1	500	36.33%	49.42%	0:14:27
4	18	20	2	500	41.60%	54.67%	0:15:23
4	18	20	0	750	28.44%	42.50%	0:19:14
4	18	20	1	750	34.85%	47.85%	0:19:56
4	18	20	2	750	40.28%	52.51%	0:20:45
4	18	20	0	1000	28.64%	43.72%	0:24:40
4	18	20	1	1000	33.12%	45.92%	0:25:34
4	18	20	2	1000	36.67%	48.56%	0:26:32
4	18	20	0	1500	25.75%	38.78%	0:35:32
4	18	20	1	1500	27.96%	39.18%	0:36:38
4	18	20	2	1500	30.88%	41.16%	0:38:09
4	18	30	0	100	32.45%	48.20%	0:05:59
4	18	30	1	100	39.85%	56.24%	0:07:45
4	18	30	0	250	34.91%	50.27%	0:09:16
4	18	30	1	250	44.96%	60.68%	0:11:12
4	18	30	0	500	37.01%	52.15%	0:14:56
4	18	30	1	500	47.78%	62.88%	0:16:57
4	18	30	0	750	37.20%	51.53%	0:20:21
4	18	30	1	750	48.60%	63.02%	0:22:44
4	18	30	0	1000	36.06%	50.36%	0:26:42
4	18	30	1	1000	47.89%	61.27%	0:28:34

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
4	18	30	0	1500	34.79%	49.01%	0:37:15
4	18	30	1	1500	46.35%	59.20%	0:39:58
4	20	10	0	100	16.83%	28.99%	0:03:51
4	20	10	1	100	20.34%	32.18%	0:05:09
4	20	10	2	100	24.41%	37.30%	0:06:25
4	20	10	3	100	29.23%	41.83%	0:07:12
4	20	10	4	100	33.59%	47.85%	0:07:50
4	20	10	5	100	38.98%	54.08%	0:09:05
4	20	10	0	250	17.26%	28.55%	0:07:06
4	20	10	1	250	22.02%	33.30%	0:08:32
4	20	10	2	250	25.66%	37.21%	0:09:47
4	20	10	3	250	32.06%	44.52%	0:10:36
4	20	10	4	250	36.87%	49.37%	0:11:17
4	20	10	5	250	44.20%	58.35%	0:12:35
4	20	10	0	500	16.67%	28.46%	0:12:31
4	20	10	1	500	19.60%	29.94%	0:14:03
4	20	10	2	500	22.61%	31.82%	0:15:33
4	20	10	3	500	26.46%	35.77%	0:16:18
4	20	10	4	500	31.68%	41.34%	0:16:55
4	20	10	5	500	37.13%	47.35%	0:18:27
4	20	10	0	750	16.81%	28.32%	0:18:02
4	20	10	1	750	20.01%	30.12%	0:19:39
4	20	10	2	750	23.37%	33.21%	0:21:03
4	20	10	3	750	26.16%	35.37%	0:21:50

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
4	20	10	4	750	30.98%	39.90%	0:22:48
4	20	10	5	750	37.41%	47.67%	0:24:15
4	20	10	0	1000	15.78%	27.33%	0:23:26
4	20	10	1	1000	17.98%	28.23%	0:25:32
4	20	10	2	1000	19.82%	29.13%	0:26:38
4	20	10	3	1000	21.04%	29.31%	0:27:32
4	20	10	4	1000	23.89%	31.91%	0:28:41
4	20	10	5	1000	28.14%	36.94%	0:30:17
4	20	10	0	1500	13.41%	24.28%	0:34:28
4	20	10	1	1500	13.91%	22.80%	0:36:19
4	20	10	2	1500	14.36%	21.99%	0:38:26
4	20	10	3	1500	13.75%	21.23%	0:38:54
4	20	10	4	1500	14.88%	22.26%	0:39:48
4	20	10	5	1500	16.42%	23.88%	0:41:44
4	20	10	0	2000	13.57%	23.52%	0:45:16
4	20	10	1	2000	13.20%	21.72%	0:47:20
4	20	10	2	2000	13.22%	21.14%	0:49:33
4	20	10	3	2000	11.98%	19.21%	0:50:34
4	20	10	4	2000	12.41%	18.99%	0:51:14
4	20	10	5	2000	13.97%	21.05%	0:53:15
4	20	15	0	100	26.76%	41.83%	0:04:08
4	20	15	1	100	31.60%	47.62%	0:06:26
4	20	15	2	100	37.68%	53.10%	0:07:34
4	20	15	3	100	43.62%	60.14%	0:08:21

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
4	20	15	0	250	28.11%	42.46%	0:07:24
4	20	15	1	250	33.33%	48.65%	0:09:46
4	20	15	2	250	40.78%	56.10%	0:11:18
4	20	15	3	250	47.68%	61.89%	0:11:47
4	20	15	0	500	29.27%	44.79%	0:12:51
4	20	15	1	500	35.01%	49.82%	0:15:23
4	20	15	2	500	43.13%	58.17%	0:16:51
4	20	15	3	500	50.66%	65.66%	0:17:27
4	20	15	0	750	28.73%	43.36%	0:18:14
4	20	15	1	750	33.98%	47.35%	0:21:15
4	20	15	2	750	42.55%	56.33%	0:22:17
4	20	15	3	750	49.76%	62.84%	0:23:26
4	20	15	0	1000	27.05%	41.65%	0:23:44
4	20	15	1	1000	32.72%	46.14%	0:26:48
4	20	15	2	1000	40.08%	54.53%	0:28:00
4	20	15	3	1000	47.87%	61.45%	0:29:09
4	20	15	0	1500	25.47%	39.09%	0:34:40
4	20	15	1	1500	29.69%	42.82%	0:38:38
4	20	15	2	1500	35.51%	47.80%	0:40:19
4	20	15	3	1500	41.40%	54.17%	0:40:41
4	20	15	0	2000	23.75%	37.39%	0:45:35
4	20	15	1	2000	26.15%	37.79%	0:48:59
4	20	15	2	2000	31.31%	43.09%	0:50:49
4	20	15	3	2000	34.79%	46.32%	0:52:11

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
4	20	20	0	100	27.84%	43.22%	0:05:04
4	20	20	1	100	32.58%	47.04%	0:05:44
4	20	20	2	100	36.26%	50.13%	0:06:40
4	20	20	0	250	29.06%	43.58%	0:08:21
4	20	20	1	250	36.01%	50.09%	0:09:07
4	20	20	2	250	40.33%	53.95%	0:09:59
4	20	20	0	500	30.33%	44.66%	0:13:50
4	20	20	1	500	39.44%	53.10%	0:14:44
4	20	20	2	500	45.56%	59.29%	0:15:36
4	20	20	0	750	30.30%	43.94%	0:19:36
4	20	20	1	750	38.44%	51.48%	0:20:19
4	20	20	2	750	44.51%	56.69%	0:21:10
4	20	20	0	1000	29.56%	42.91%	0:24:47
4	20	20	1	1000	38.18%	50.76%	0:25:51
4	20	20	2	1000	45.30%	57.63%	0:26:51
4	20	20	0	1500	27.48%	40.39%	0:35:44
4	20	20	1	1500	32.78%	43.94%	0:36:56
4	20	20	2	1500	37.73%	47.67%	0:38:04
4	20	20	0	2000	26.05%	39.05%	0:46:45
4	20	20	1	2000	31.06%	41.83%	0:48:23
4	20	20	2	2000	34.55%	44.21%	0:49:48
4	20	30	0	100	36.88%	52.74%	0:06:16
4	20	30	1	100	44.31%	60.55%	0:08:22
4	20	30	0	250	40.59%	56.28%	0:09:36

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
4	20	30	1	250	49.81%	65.39%	0:11:47
4	20	30	0	500	42.84%	58.66%	0:15:15
4	20	30	1	500	53.79%	68.85%	0:17:28
4	20	30	0	750	43.18%	58.80%	0:20:44
4	20	30	1	750	54.27%	68.85%	0:23:15
4	20	30	0	1000	43.67%	57.76%	0:26:12
4	20	30	1	1000	55.51%	69.48%	0:29:14
4	20	30	0	1500	43.45%	57.36%	0:37:27
4	20	30	1	1500	55.38%	69.12%	0:40:42
4	20	30	0	2000	41.61%	55.66%	0:48:50
4	20	30	1	2000	52.15%	65.35%	0:52:31
4	25	15	0	100	31.76%	48.70%	0:04:22
4	25	15	1	100	38.56%	54.85%	0:07:20
4	25	15	2	100	44.81%	61.18%	0:08:44
4	25	15	3	100	49.57%	65.66%	0:09:53
4	25	15	0	250	33.71%	50.99%	0:07:38
4	25	15	1	250	43.22%	60.28%	0:10:39
4	25	15	2	250	51.46%	67.19%	0:12:11
4	25	15	3	250	56.93%	71.27%	0:13:20
4	25	15	0	500	35.00%	52.47%	0:13:08
4	25	15	1	500	45.05%	60.95%	0:16:24
4	25	15	2	500	52.64%	67.73%	0:18:01
4	25	15	3	500	59.56%	72.85%	0:19:21
4	25	15	0	750	34.63%	51.12%	0:18:37

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
4	25	15	1	750	44.34%	59.25%	0:22:06
4	25	15	2	750	53.64%	67.50%	0:23:50
4	25	15	3	750	60.25%	72.08%	0:25:03
4	25	20	0	100	33.51%	49.91%	0:05:37
4	25	20	1	100	40.83%	56.51%	0:08:19
4	25	20	2	100	47.24%	62.43%	0:09:58
4	25	20	0	250	34.28%	50.09%	0:08:52
4	25	20	1	250	45.27%	60.32%	0:11:44
4	25	20	2	250	53.14%	68.00%	0:13:21
4	25	20	0	500	36.84%	53.28%	0:14:17
4	25	20	1	500	48.82%	63.42%	0:17:36
4	25	20	2	500	56.61%	70.47%	0:19:15
4	25	20	0	750	36.04%	50.85%	0:19:51
4	25	20	1	750	48.77%	63.29%	0:23:16
4	25	30	0	100	38.82%	55.30%	0:07:08
4	25	30	1	100	45.66%	61.49%	0:09:55
4	25	30	0	250	41.83%	57.59%	0:10:34
4	25	30	1	250	51.77%	66.79%	0:13:25
4	25	30	0	500	44.55%	59.87%	0:16:11
4	25	30	1	500	55.80%	70.15%	0:19:17
5	18	15	0	100	22.74%	35.95%	0:04:50
5	18	15	1	100	28.73%	42.77%	0:07:18
5	18	15	2	100	36.05%	51.21%	0:08:30
5	18	15	3	100	42.23%	57.76%	0:09:22

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	18	15	0	250	26.09%	39.23%	0:08:53
5	18	15	1	250	32.28%	46.05%	0:11:30
5	18	15	2	250	39.25%	54.08%	0:12:43
5	18	15	3	250	47.04%	61.54%	0:13:41
5	18	15	0	500	25.68%	39.09%	0:15:37
5	18	15	1	500	33.11%	46.54%	0:18:34
5	18	15	2	500	40.20%	54.44%	0:19:47
5	18	15	3	500	46.90%	59.78%	0:20:44
5	18	15	0	750	26.00%	39.90%	0:22:37
5	18	15	1	750	30.54%	44.52%	0:25:38
5	18	15	2	750	37.57%	50.18%	0:26:49
5	18	15	3	750	44.04%	57.00%	0:28:00
5	18	15	0	1000	24.35%	37.16%	0:29:13
5	18	15	1	1000	28.37%	40.39%	0:32:39
5	18	15	2	1000	31.92%	43.09%	0:34:09
5	18	15	3	1000	38.75%	49.64%	0:35:09
5	18	15	0	1500	22.60%	35.59%	0:42:58
5	18	15	1	1500	24.44%	35.46%	0:46:33
5	18	15	2	1500	27.54%	37.61%	0:48:13
5	18	15	3	1500	31.34%	41.29%	0:49:43
5	18	20	0	100	26.69%	40.31%	0:05:50
5	18	20	1	100	31.93%	46.81%	0:06:37
5	18	20	2	100	35.67%	50.81%	0:07:33
5	18	20	0	250	28.62%	42.82%	0:10:03

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	18	20	1	250	34.00%	48.11%	0:10:42
5	18	20	2	250	39.37%	52.96%	0:11:42
5	18	20	0	500	29.05%	43.09%	0:16:50
5	18	20	1	500	36.11%	50.04%	0:17:53
5	18	20	2	500	41.82%	54.85%	0:18:47
5	18	20	0	750	29.25%	43.76%	0:23:33
5	18	20	1	750	35.23%	48.56%	0:24:45
5	18	20	2	750	38.41%	50.58%	0:25:55
5	18	20	0	1000	27.23%	40.89%	0:30:32
5	18	20	1	1000	32.37%	44.57%	0:31:37
5	18	20	2	1000	35.40%	47.26%	0:32:53
5	18	20	0	1500	25.42%	38.73%	0:44:07
5	18	20	1	1500	26.35%	36.09%	0:45:41
5	18	20	2	1500	27.66%	37.75%	0:46:55
5	18	30	0	100	31.89%	46.81%	0:07:09
5	18	30	1	100	40.43%	56.33%	0:09:24
5	18	30	0	250	34.67%	50.63%	0:11:22
5	18	30	1	250	44.40%	60.10%	0:13:37
5	18	30	0	500	36.48%	52.29%	0:18:15
5	18	30	1	500	47.48%	61.85%	0:20:45
5	18	30	0	750	37.97%	53.28%	0:25:12
5	18	30	1	750	51.01%	64.72%	0:28:07
5	18	30	0	1000	36.85%	51.89%	0:32:18
5	18	30	1	1000	48.89%	62.30%	0:35:17

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	18	30	0	1500	34.83%	48.34%	0:45:59
5	18	30	1	1500	45.49%	58.17%	0:49:36
5	20	10	0	100	15.73%	27.83%	0:04:31
5	20	10	1	100	19.48%	30.66%	0:06:10
5	20	10	2	100	22.93%	34.56%	0:07:50
5	20	10	3	100	27.80%	39.81%	0:08:43
5	20	10	4	100	32.57%	45.87%	0:09:28
5	20	10	5	100	38.50%	52.69%	0:11:00
5	20	10	0	250	18.09%	30.75%	0:08:34
5	20	10	1	250	21.82%	33.80%	0:10:25
5	20	10	2	250	26.62%	38.33%	0:11:54
5	20	10	3	250	32.63%	45.02%	0:12:50
5	20	10	4	250	40.56%	53.10%	0:13:44
5	20	10	5	250	48.36%	62.34%	0:15:29
5	20	10	0	500	16.46%	28.01%	0:15:28
5	20	10	1	500	20.26%	30.88%	0:17:16
5	20	10	2	500	23.06%	32.09%	0:19:01
5	20	10	3	500	27.72%	37.07%	0:19:58
5	20	10	4	500	33.06%	43.45%	0:20:59
5	20	10	5	500	38.74%	49.69%	0:22:47
5	20	10	0	750	15.11%	26.71%	0:22:13
5	20	10	1	750	18.06%	28.55%	0:24:27
5	20	10	2	750	20.24%	28.77%	0:26:02
5	20	10	3	750	21.65%	30.39%	0:27:01

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	20	10	4	750	24.60%	33.98%	0:28:02
5	20	10	5	750	28.99%	37.48%	0:30:14
5	20	10	0	1000	14.12%	25.72%	0:29:00
5	20	10	1	1000	14.70%	23.88%	0:31:17
5	20	10	2	1000	15.19%	23.56%	0:33:20
5	20	10	3	1000	15.40%	22.89%	0:34:11
5	20	10	4	1000	15.99%	23.20%	0:35:05
5	20	10	5	1000	17.16%	25.04%	0:37:27
5	20	10	0	1500	12.34%	22.67%	0:42:22
5	20	10	1	1500	12.16%	20.60%	0:44:48
5	20	10	2	1500	11.18%	18.40%	0:47:01
5	20	10	3	1500	10.17%	16.83%	0:48:33
5	20	10	4	1500	10.54%	17.15%	0:49:42
5	20	10	5	1500	10.87%	18.09%	0:51:55
5	20	10	0	2000	10.75%	19.93%	0:56:12
5	20	10	1	2000	10.78%	19.03%	0:58:55
5	20	10	2	2000	9.29%	16.38%	1:01:38
5	20	10	3	2000	8.02%	14.72%	1:02:43
5	20	10	4	2000	7.42%	14.27%	1:03:45
5	20	10	5	2000	8.03%	14.77%	1:06:37
5	20	15	0	100	27.63%	42.82%	0:04:56
5	20	15	1	100	32.04%	47.67%	0:07:45
5	20	15	2	100	40.23%	55.70%	0:09:01
5	20	15	3	100	45.32%	61.13%	0:10:05

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	20	15	0	250	29.17%	44.52%	0:08:57
5	20	15	1	250	35.18%	50.45%	0:11:55
5	20	15	2	250	43.37%	59.16%	0:13:17
5	20	15	3	250	51.45%	66.02%	0:14:25
5	20	15	0	500	30.30%	44.79%	0:15:51
5	20	15	1	500	36.22%	50.94%	0:18:58
5	20	15	2	500	45.57%	60.32%	0:20:21
5	20	15	3	500	52.73%	66.47%	0:21:33
5	20	15	0	750	28.48%	43.27%	0:22:35
5	20	15	1	750	34.15%	48.65%	0:26:21
5	20	15	2	750	42.57%	56.73%	0:27:37
5	20	15	3	750	51.40%	65.71%	0:28:54
5	20	15	0	1000	27.91%	42.73%	0:30:16
5	20	15	1	1000	33.49%	47.71%	0:32:55
5	20	15	2	1000	41.39%	54.85%	0:34:41
5	20	15	3	1000	48.06%	61.22%	0:36:25
5	20	15	0	1500	26.33%	40.75%	0:42:58
5	20	15	1	1500	28.38%	41.83%	0:47:08
5	20	15	2	1500	34.21%	46.77%	0:49:21
5	20	15	3	1500	40.32%	52.29%	0:50:39
5	20	15	0	2000	23.06%	36.89%	0:56:27
5	20	15	1	2000	24.81%	36.71%	1:01:15
5	20	15	2	2000	27.65%	38.96%	1:03:37
5	20	15	3	2000	32.67%	43.40%	1:05:13

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	20	20	0	100	27.94%	43.45%	0:06:05
5	20	20	1	100	34.26%	48.88%	0:06:53
5	20	20	2	100	37.54%	51.48%	0:07:59
5	20	20	0	250	30.45%	45.33%	0:10:13
5	20	20	1	250	38.26%	53.10%	0:11:04
5	20	20	2	250	42.88%	56.64%	0:12:15
5	20	20	0	500	31.32%	46.36%	0:17:06
5	20	20	1	500	41.11%	54.26%	0:17:59
5	20	20	2	500	47.19%	60.73%	0:19:08
5	20	20	0	750	28.89%	42.91%	0:23:59
5	20	20	1	750	38.40%	51.93%	0:25:04
5	20	20	2	750	45.43%	58.30%	0:26:17
5	20	20	0	1000	28.60%	42.15%	0:31:29
5	20	20	1	1000	37.77%	50.63%	0:31:54
5	20	20	2	1000	43.69%	55.57%	0:33:32
5	20	20	0	1500	25.76%	37.97%	0:44:16
5	20	20	1	1500	32.76%	44.48%	0:45:52
5	20	20	2	1500	38.80%	49.64%	0:47:42
5	20	20	0	2000	24.29%	36.22%	0:58:14
5	20	20	1	2000	28.51%	38.78%	1:00:43
5	20	20	2	2000	30.93%	39.86%	1:01:59
5	20	30	0	100	38.58%	54.76%	0:07:40
5	20	30	1	100	46.21%	62.57%	0:10:02
5	20	30	0	250	41.22%	57.32%	0:11:42

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	20	30	1	250	51.63%	66.61%	0:14:23
5	20	30	0	500	43.08%	58.62%	0:18:42
5	20	30	1	500	55.01%	69.61%	0:21:33
5	20	30	0	750	44.10%	59.38%	0:25:40
5	20	30	1	750	55.22%	69.30%	0:29:08
5	20	30	0	1000	44.58%	60.37%	0:32:50
5	20	30	1	1000	56.11%	69.70%	0:36:08
5	20	30	0	1500	44.54%	58.80%	0:46:52
5	20	30	1	1500	56.71%	70.24%	0:50:51
5	20	30	0	2000	41.94%	55.57%	1:00:34
5	20	30	1	2000	52.78%	65.66%	1:04:50
5	25	15	0	100	30.44%	46.41%	0:05:11
5	25	15	1	100	39.65%	56.42%	0:08:51
5	25	15	2	100	45.97%	62.61%	0:10:43
5	25	15	3	100	52.61%	67.15%	0:12:02
5	25	15	0	250	33.54%	49.87%	0:09:17
5	25	15	1	250	43.36%	59.20%	0:13:05
5	25	15	2	250	51.43%	67.24%	0:14:49
5	25	15	3	250	57.65%	71.59%	0:16:21
5	25	15	0	500	35.26%	51.97%	0:16:04
5	25	15	1	500	44.96%	60.14%	0:20:10
5	25	15	2	500	53.21%	68.09%	0:22:07
5	25	15	3	500	59.27%	72.13%	0:23:52
5	25	15	0	750	33.65%	49.69%	0:22:51

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	25	15	1	750	45.35%	60.91%	0:27:20
5	25	15	2	750	54.00%	67.95%	0:29:52
5	25	15	3	750	60.22%	72.67%	0:31:01
5	25	15	0	1000	33.56%	49.51%	0:29:39
5	25	15	1	1000	42.97%	57.50%	0:34:26
5	25	15	2	1000	51.52%	65.48%	0:36:42
5	25	15	3	1000	58.85%	71.18%	0:38:32
5	25	15	0	1500	31.41%	46.23%	0:43:15
5	25	15	1	1500	41.53%	55.57%	0:48:44
5	25	15	2	1500	51.72%	64.36%	0:51:11
5	25	15	3	1500	58.88%	70.20%	0:53:18
5	25	20	0	100	33.18%	49.33%	0:06:42
5	25	20	1	100	42.02%	58.57%	0:10:06
5	25	20	2	100	47.86%	63.91%	0:12:00
5	25	20	0	250	35.58%	51.71%	0:10:50
5	25	20	1	250	47.63%	63.51%	0:14:22
5	25	20	2	250	55.02%	69.34%	0:16:23
5	25	20	0	500	36.99%	53.14%	0:17:40
5	25	20	1	500	49.82%	64.63%	0:21:37
5	25	20	2	500	57.79%	71.23%	0:23:43
5	25	20	0	750	36.61%	52.15%	0:24:39
5	25	20	1	750	50.40%	64.95%	0:29:16
5	25	20	2	750	58.12%	71.14%	0:31:13
5	25	20	0	1000	35.73%	50.45%	0:31:26

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	25	20	1	1000	49.72%	64.59%	0:35:57
5	25	20	2	1000	57.44%	69.93%	0:38:35
5	25	20	0	1500	34.37%	48.16%	0:45:11
5	25	20	1	1500	46.30%	60.19%	0:50:37
5	25	20	2	1500	55.03%	67.15%	0:53:31
5	25	30	0	100	39.38%	56.15%	0:08:39
5	25	30	1	100	47.08%	63.24%	0:12:02
5	25	30	0	250	43.22%	58.93%	0:12:52
5	25	30	1	250	53.09%	68.00%	0:16:25
5	25	30	0	500	46.05%	61.36%	0:19:54
5	25	30	1	500	57.30%	71.18%	0:23:40
5	25	30	0	750	47.01%	61.71%	0:26:48
5	25	30	1	750	58.10%	71.10%	0:31:12
5	25	30	0	1000	46.46%	61.40%	0:34:00
5	25	30	1	1000	59.47%	72.08%	0:38:31
5	25	30	0	1500	46.59%	60.95%	0:47:56
5	25	30	1	1500	59.03%	71.41%	0:54:07
5	30	15	0	100	31.45%	46.99%	0:05:29
5	30	15	1	100	42.25%	59.34%	0:09:59
5	30	15	2	100	48.65%	64.68%	0:12:17
5	30	15	3	100	52.78%	68.36%	0:14:12
5	30	15	0	250	36.02%	51.84%	0:09:31
5	30	15	1	250	47.68%	63.46%	0:14:19
5	30	15	2	250	55.92%	71.05%	0:16:44

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	30	15	3	250	60.82%	74.42%	0:18:32
5	30	15	0	500	37.49%	53.50%	0:16:29
5	30	15	1	500	50.16%	65.57%	0:21:24
5	30	15	2	500	59.02%	72.67%	0:24:06
5	30	15	3	500	63.17%	75.18%	0:26:08
5	30	15	0	750	37.09%	52.96%	0:23:08
5	30	15	1	750	50.86%	66.11%	0:29:04
5	30	15	2	750	59.84%	71.86%	0:31:39
5	30	15	3	750	63.34%	74.82%	0:33:54
5	30	15	0	1000	37.01%	52.02%	0:29:56
5	30	15	1	1000	50.25%	65.62%	0:35:46
5	30	15	2	1000	58.52%	71.63%	0:38:59
5	30	15	3	1000	62.46%	73.16%	0:41:26
5	30	15	0	1500	36.54%	51.39%	0:43:32
5	30	15	1	1500	48.62%	63.73%	0:50:45
5	30	15	2	1500	57.30%	69.75%	0:53:39
5	30	15	3	1500	62.35%	73.29%	0:56:36
5	30	20	0	100	36.37%	52.96%	0:07:18
5	30	20	1	100	45.57%	61.98%	0:11:42
5	30	20	2	100	51.67%	67.41%	0:14:05
5	30	20	0	250	38.75%	55.43%	0:11:27
5	30	20	1	250	50.66%	66.02%	0:16:01
5	30	20	2	250	57.21%	71.68%	0:18:38
5	30	20	0	500	39.96%	56.33%	0:18:24

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	30	20	1	500	53.56%	68.36%	0:23:20
5	30	20	2	500	60.07%	73.25%	0:26:09
5	30	20	0	750	39.83%	56.37%	0:25:15
5	30	20	1	750	53.82%	67.91%	0:30:41
5	30	20	2	750	60.73%	72.76%	0:33:40
5	30	20	0	1000	40.13%	55.92%	0:32:19
5	30	20	1	1000	54.48%	67.46%	0:38:01
5	30	20	2	1000	61.74%	73.43%	0:41:18
5	30	20	0	1500	39.33%	55.16%	0:45:59
5	30	20	1	1500	52.87%	66.70%	0:52:34
5	30	20	2	1500	60.67%	72.44%	0:56:34
5	30	30	0	100	42.37%	59.56%	0:09:55
5	30	30	1	100	49.37%	65.62%	0:14:06
5	30	30	0	250	45.93%	62.48%	0:14:01
5	30	30	1	250	54.66%	69.84%	0:18:40
5	30	30	0	500	49.07%	64.36%	0:21:18
5	30	30	1	500	59.28%	73.20%	0:26:13
5	30	30	0	750	50.87%	65.35%	0:28:12
5	30	30	1	750	61.66%	73.97%	0:33:49
5	30	30	0	1000	51.49%	66.25%	0:35:53
5	30	30	1	1000	62.34%	74.10%	0:41:28
5	30	30	0	1500	51.54%	65.80%	0:49:28
5	30	30	1	1500	62.59%	74.51%	0:56:45
5	40	15	0	100	34.74%	50.76%	0:06:01

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	40	15	1	100	44.73%	61.89%	0:12:36
5	40	15	2	100	50.24%	65.93%	0:16:04
5	40	15	3	100	53.80%	68.58%	0:18:44
5	40	15	0	250	38.37%	54.08%	0:10:08
5	40	15	1	250	50.56%	66.16%	0:17:03
5	40	15	2	250	56.75%	70.51%	0:20:41
5	40	15	3	250	61.00%	74.06%	0:23:23
5	40	15	0	500	41.56%	57.27%	0:16:59
5	40	15	1	500	53.98%	69.03%	0:24:38
5	40	15	2	500	60.74%	74.06%	0:28:29
5	40	15	3	500	63.93%	75.85%	0:31:32
5	40	15	0	750	41.62%	58.03%	0:23:49
5	40	15	1	750	54.55%	68.63%	0:31:53
5	40	15	2	750	61.45%	74.51%	0:36:25
5	40	15	3	750	64.84%	76.30%	0:39:40
5	40	15	0	1000	42.26%	57.90%	0:30:43
5	40	15	1	1000	54.11%	67.59%	0:39:39
5	40	15	2	1000	61.05%	72.80%	0:44:13
5	40	15	3	1000	63.94%	74.87%	0:47:49
5	40	15	0	1500	41.40%	56.24%	0:44:46
5	40	15	1	1500	54.89%	68.85%	0:54:46
5	40	15	2	1500	61.39%	72.94%	1:00:00
5	40	15	3	1500	65.24%	76.53%	1:04:13
5	40	20	0	100	38.66%	55.39%	0:08:35

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	40	20	1	100	47.33%	64.14%	0:15:04
5	40	20	2	100	52.00%	68.00%	0:18:47
5	40	20	0	250	42.44%	59.16%	0:12:45
5	40	20	1	250	52.39%	67.46%	0:19:33
5	40	20	2	250	57.82%	71.50%	0:23:31
5	40	20	0	500	44.25%	60.10%	0:19:40
5	40	20	1	500	56.45%	70.74%	0:27:12
5	40	20	2	500	62.63%	74.42%	0:31:28
5	40	20	0	750	44.86%	60.41%	0:26:38
5	40	20	1	750	57.27%	70.38%	0:34:50
5	40	20	2	750	62.90%	75.40%	0:39:43
5	40	20	0	1000	45.03%	59.87%	0:33:33
5	40	20	1	1000	57.90%	70.51%	0:43:05
5	40	20	2	1000	63.26%	75.00%	0:47:47
5	40	20	0	1500	45.10%	59.83%	0:47:46
5	40	20	1	1500	58.10%	70.69%	0:59:00
5	40	20	2	1500	63.03%	74.46%	1:04:50
5	40	30	0	100	42.00%	59.61%	0:12:20
5	40	30	1	100	47.78%	64.45%	0:18:46
5	40	30	0	250	47.80%	63.78%	0:16:33
5	40	30	1	250	55.34%	70.02%	0:23:36
5	40	30	0	500	51.19%	66.56%	0:23:49
5	40	30	1	500	60.25%	72.94%	0:31:30
5	40	30	0	750	52.67%	67.10%	0:31:04

Table 4: Extended results for **R²MMT**, configuration with memory. Time is in format h:mm:ss.

Epochs	**K**	τ	t	**Iterations**	**mAP**	**Top-1**	**Training Time**
5	40	30	1	750	62.07%	74.73%	0:39:55
5	40	30	0	1000	53.83%	68.04%	0:38:07
5	40	30	1	1000	63.33%	74.69%	0:48:32
5	40	30	0	1500	55.07%	69.43%	0:53:21
5	40	30	1	1500	64.78%	75.90%	1:04:23

.0.0.1 Subset Distribution Selection

Here we see the time it takes to perform our SDS algorithm for each camera for each time segment in Sec. 5.5.3. Even though we use a greedy implementation, SDS time scales poorly with the amount of data provided. Any K that has an SDS time greater than τ for any camera for any time segment does not meet the time constraint for RW-OUDA for that τ.

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
8	1	15	0	0:00:06	8	1	15	0	0:00:06
8	1	15	1	0:03:46	8	1	15	1	0:04:58
8	1	15	2	0:00:03	8	1	15	2	0:05:58
8	1	15	3	0:00:02	8	1	15	3	0:06:48
8	2	15	0	0:00:01	8	2	15	0	0:00:01
8	2	15	1	0:00:07	8	2	15	1	0:00:15
8	2	15	2	0:00:05	8	2	15	2	0:00:36
8	2	15	3	0:00:05	8	2	15	3	0:01:05

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
8	3	15	0	0:00:00	8	3	15	0	0:00:00
8	3	15	1	0:00:01	8	3	15	1	0:00:02
8	3	15	2	0:00:00	8	3	15	2	0:00:04
8	3	15	3	0:00:00	8	3	15	3	0:00:06
8	4	15	0	0:00:01	8	4	15	0	0:00:01
8	4	15	1	0:00:03	8	4	15	1	0:00:08
8	4	15	2	0:00:01	8	4	15	2	0:00:17
8	4	15	3	0:00:00	8	4	15	3	0:00:22
8	5	15	0	0:00:02	8	5	15	0	0:00:02
8	5	15	1	0:00:02	8	5	15	1	0:00:08
8	5	15	2	0:00:01	8	5	15	2	0:00:15
8	5	15	3	0:00:01	8	5	15	3	0:00:25
8	6	15	0	0:00:04	8	6	15	0	0:00:04
8	6	15	1	0:00:10	8	6	15	1	0:00:27
8	6	15	2	0:00:06	8	6	15	2	0:00:54
8	6	15	3	0:00:04	8	6	15	3	0:01:23
8	7	15	0	0:00:01	8	7	15	0	0:00:01
8	7	15	1	0:00:02	8	7	15	1	0:00:06
8	7	15	2	0:00:01	8	7	15	2	0:00:10
8	7	15	3	0:00:00	8	7	15	3	0:00:13
8	8	15	0	0:00:19	8	8	15	0	0:00:19
8	8	15	1	0:00:14	8	8	15	1	0:00:59
8	8	15	2	0:00:08	8	8	15	2	0:01:44

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
8	8	15	3	0:00:01	8	8	15	3	0:02:00
8	1	20	0	0:01:52	8	1	20	0	0:01:52
8	2	20	0	0:00:03	8	2	20	0	0:00:03
8	2	20	1	0:00:13	8	2	20	1	0:00:29
8	2	20	2	0:00:09	8	2	20	2	0:01:05
8	3	20	0	0:00:01	8	3	20	0	0:00:01
8	3	20	1	0:00:01	8	3	20	1	0:00:03
8	3	20	2	0:00:00	8	3	20	2	0:00:06
8	4	20	0	0:00:03	8	4	20	0	0:00:03
8	4	20	1	0:00:04	8	4	20	1	0:00:14
8	4	20	2	0:00:01	8	4	20	2	0:00:22
8	5	20	0	0:00:03	8	5	20	0	0:00:03
8	5	20	1	0:00:03	8	5	20	1	0:00:13
8	5	20	2	0:00:02	8	5	20	2	0:00:25
8	6	20	0	0:00:09	8	6	20	0	0:00:09
8	6	20	1	0:00:13	8	6	20	1	0:00:42
8	6	20	2	0:00:09	8	6	20	2	0:01:23
8	7	20	0	0:00:02	8	7	20	0	0:00:02
8	7	20	1	0:00:02	8	7	20	1	0:00:08
8	7	20	2	0:00:01	8	7	20	2	0:00:13
8	8	20	0	0:00:33	8	8	20	0	0:00:33
8	8	20	1	0:00:17	8	8	20	1	0:01:30
8	8	20	2	0:00:02	8	8	20	2	0:02:00

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
8	1	30	0	0:04:58	8	1	30	0	0:04:59
8	1	30	1	0:00:10	8	1	30	1	0:06:49
8	2	30	0	0:00:15	8	2	30	0	0:00:15
8	2	30	1	0:00:21	8	2	30	1	0:01:05
8	3	30	0	0:00:02	8	3	30	0	0:00:02
8	3	30	1	0:00:01	8	3	30	1	0:00:06
8	4	30	0	0:00:08	8	4	30	0	0:00:08
8	4	30	1	0:00:03	8	4	30	1	0:00:22
8	5	30	0	0:00:08	8	5	30	0	0:00:08
8	5	30	1	0:00:04	8	5	30	1	0:00:25
8	6	30	0	0:00:27	8	6	30	0	0:00:27
8	6	30	1	0:00:19	8	6	30	1	0:01:23
8	7	30	0	0:00:06	8	7	30	0	0:00:06
8	7	30	1	0:00:02	8	7	30	1	0:00:13
8	8	30	0	0:01:00	8	8	30	0	0:01:00
8	8	30	1	0:00:13	8	8	30	1	0:02:00
10	1	15	0	0:00:07	10	1	15	0	0:00:07
10	1	15	1	0:04:44	10	1	15	1	0:06:13
10	1	15	2	0:00:04	10	1	15	2	0:07:29
10	1	15	3	0:00:03	10	1	15	3	0:08:31
10	2	15	0	0:00:02	10	2	15	0	0:00:02
10	2	15	1	0:00:09	10	2	15	1	0:00:18
10	2	15	2	0:00:06	10	2	15	2	0:00:44

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
10	2	15	3	0:00:07	10	2	15	3	0:01:21
10	3	15	0	0:00:00	10	3	15	0	0:00:00
10	3	15	1	0:00:01	10	3	15	1	0:00:03
10	3	15	2	0:00:00	10	3	15	2	0:00:05
10	3	15	3	0:00:00	10	3	15	3	0:00:07
10	4	15	0	0:00:01	10	4	15	0	0:00:01
10	4	15	1	0:00:04	10	4	15	1	0:00:10
10	4	15	2	0:00:02	10	4	15	2	0:00:21
10	4	15	3	0:00:01	10	4	15	3	0:00:28
10	5	15	0	0:00:02	10	5	15	0	0:00:02
10	5	15	1	0:00:03	10	5	15	1	0:00:10
10	5	15	2	0:00:01	10	5	15	2	0:00:19
10	5	15	3	0:00:02	10	5	15	3	0:00:31
10	6	15	0	0:00:05	10	6	15	0	0:00:05
10	6	15	1	0:00:13	10	6	15	1	0:00:33
10	6	15	2	0:00:07	10	6	15	2	0:01:08
10	6	15	3	0:00:05	10	6	15	3	0:01:43
10	7	15	0	0:00:01	10	7	15	0	0:00:01
10	7	15	1	0:00:02	10	7	15	1	0:00:07
10	7	15	2	0:00:01	10	7	15	2	0:00:12
10	7	15	3	0:00:00	10	7	15	3	0:00:17
10	8	15	0	0:00:23	10	8	15	0	0:00:23
10	8	15	1	0:00:17	10	8	15	1	0:01:14

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
10	8	15	2	0:00:10	10	8	15	2	0:02:11
10	8	15	3	0:00:01	10	8	15	3	0:02:30
10	1	20	0	0:02:21	10	1	20	0	0:02:21
10	1	20	2	0:00:05	10	2	20	0	0:00:04
10	2	20	0	0:00:04	10	2	20	1	0:00:36
10	2	20	1	0:00:16	10	2	20	2	0:01:21
10	2	20	2	0:00:11	10	3	20	0	0:00:01
10	3	20	0	0:00:01	10	3	20	1	0:00:04
10	3	20	1	0:00:01	10	3	20	2	0:00:07
10	3	20	2	0:00:00	10	4	20	0	0:00:04
10	4	20	0	0:00:04	10	4	20	1	0:00:18
10	4	20	1	0:00:05	10	4	20	2	0:00:28
10	4	20	2	0:00:01	10	5	20	0	0:00:04
10	5	20	0	0:00:04	10	5	20	1	0:00:16
10	5	20	1	0:00:04	10	5	20	2	0:00:31
10	5	20	2	0:00:03	10	6	20	0	0:00:11
10	6	20	0	0:00:11	10	6	20	1	0:00:52
10	6	20	1	0:00:16	10	6	20	2	0:01:43
10	6	20	2	0:00:11	10	7	20	0	0:00:03
10	7	20	0	0:00:03	10	7	20	1	0:00:10
10	7	20	1	0:00:03	10	7	20	2	0:00:17
10	7	20	2	0:00:01	10	8	20	0	0:00:42
10	8	20	0	0:00:42	10	8	20	1	0:01:53

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
10	8	20	1	0:00:21	10	8	20	2	0:02:30
10	8	20	2	0:00:03	10	1	30	0	0:06:14
10	1	30	0	0:06:14	10	1	30	1	0:08:32
10	1	30	1	0:00:13	10	2	30	0	0:00:18
10	2	30	0	0:00:18	10	2	30	1	0:01:21
10	2	30	1	0:00:26	10	3	30	0	0:00:03
10	3	30	0	0:00:03	10	3	30	1	0:00:07
10	3	30	1	0:00:01	10	4	30	0	0:00:10
10	4	30	0	0:00:10	10	4	30	1	0:00:28
10	4	30	1	0:00:04	10	5	30	0	0:00:11
10	5	30	0	0:00:11	10	5	30	1	0:00:31
10	5	30	1	0:00:06	10	6	30	0	0:00:34
10	6	30	0	0:00:33	10	6	30	1	0:01:43
10	6	30	1	0:00:24	10	7	30	0	0:00:07
10	7	30	0	0:00:07	10	7	30	1	0:00:17
10	7	30	1	0:00:02	10	8	30	0	0:01:14
10	8	30	0	0:01:15	10	8	30	1	0:02:30
10	8	30	1	0:00:16	12	1	15	0	0:00:09
12	1	15	0	0:00:09	12	1	15	1	0:07:30
12	1	15	1	0:05:40	12	1	15	2	0:08:56
12	1	15	2	0:00:05	12	1	15	3	0:10:15
12	1	15	3	0:00:03	12	2	15	0	0:00:02
12	2	15	0	0:00:02	12	2	15	1	0:00:22

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
12	2	15	1	0:00:10	12	2	15	2	0:00:53
12	2	15	2	0:00:07	12	2	15	3	0:01:37
12	2	15	3	0:00:08	12	3	15	0	0:00:00
12	3	15	0	0:00:00	12	3	15	1	0:00:03
12	3	15	1	0:00:02	12	3	15	2	0:00:06
12	3	15	2	0:00:00	12	3	15	3	0:00:09
12	3	15	3	0:00:00	12	4	15	0	0:00:02
12	4	15	0	0:00:02	12	4	15	1	0:00:12
12	4	15	1	0:00:05	12	4	15	2	0:00:25
12	4	15	2	0:00:02	12	4	15	3	0:00:33
12	4	15	3	0:00:01	12	5	15	0	0:00:03
12	5	15	0	0:00:03	12	5	15	1	0:00:13
12	5	15	1	0:00:03	12	5	15	2	0:00:22
12	5	15	2	0:00:01	12	5	15	3	0:00:38
12	5	15	3	0:00:02	12	6	15	0	0:00:06
12	6	15	0	0:00:06	12	6	15	1	0:00:40
12	6	15	1	0:00:16	12	6	15	2	0:01:22
12	6	15	2	0:00:09	12	6	15	3	0:02:04
12	6	15	3	0:00:06	12	7	15	0	0:00:01
12	7	15	0	0:00:01	12	7	15	1	0:00:08
12	7	15	1	0:00:03	12	7	15	2	0:00:15
12	7	15	2	0:00:01	12	7	15	3	0:00:20
12	7	15	3	0:00:00	12	8	15	0	0:00:28

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
12	8	15	0	0:00:28	12	8	15	1	0:01:29
12	8	15	1	0:00:21	12	8	15	2	0:02:37
12	8	15	2	0:00:11	12	8	15	3	0:03:00
12	8	15	3	0:00:01	12	1	20	0	0:02:50
12	1	20	0	0:02:49	12	2	20	0	0:00:05
12	1	20	2	0:00:06	12	2	20	1	0:00:44
12	2	20	0	0:00:05	12	2	20	2	0:01:37
12	2	20	1	0:00:19	12	3	20	0	0:00:01
12	2	20	2	0:00:13	12	3	20	1	0:00:05
12	3	20	0	0:00:01	12	3	20	2	0:00:09
12	3	20	1	0:00:02	12	4	20	0	0:00:05
12	3	20	2	0:00:01	12	4	20	1	0:00:21
12	4	20	0	0:00:05	12	4	20	2	0:00:33
12	4	20	1	0:00:05	12	5	20	0	0:00:05
12	4	20	2	0:00:01	12	5	20	1	0:00:19
12	5	20	0	0:00:05	12	5	20	2	0:00:37
12	5	20	1	0:00:05	12	6	20	0	0:00:14
12	5	20	2	0:00:03	12	6	20	1	0:01:03
12	6	20	0	0:00:14	12	6	20	2	0:02:04
12	6	20	1	0:00:19	12	7	20	0	0:00:03
12	6	20	2	0:00:13	12	7	20	1	0:00:13
12	7	20	0	0:00:03	12	7	20	2	0:00:20
12	7	20	1	0:00:03	12	8	20	0	0:00:50

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
12	7	20	2	0:00:01	12	8	20	1	0:02:16
12	8	20	0	0:00:50	12	8	20	2	0:02:59
12	8	20	1	0:00:26	12	1	30	0	0:07:28
12	8	20	2	0:00:04	12	1	30	1	0:10:13
12	1	30	0	0:07:29	12	2	30	0	0:00:22
12	1	30	1	0:00:16	12	2	30	1	0:01:37
12	2	30	0	0:00:22	12	3	30	0	0:00:03
12	2	30	1	0:00:31	12	3	30	1	0:00:09
12	3	30	0	0:00:03	12	4	30	0	0:00:12
12	3	30	1	0:00:01	12	4	30	1	0:00:33
12	4	30	0	0:00:12	12	5	30	0	0:00:13
12	4	30	1	0:00:05	12	5	30	1	0:00:37
12	5	30	0	0:00:13	12	6	30	0	0:00:40
12	5	30	1	0:00:07	12	6	30	1	0:02:04
12	6	30	0	0:00:40	12	7	30	0	0:00:08
12	6	30	1	0:00:28	12	7	30	1	0:00:20
12	7	30	0	0:00:08	12	8	30	0	0:01:30
12	7	30	1	0:00:03	12	8	30	1	0:03:00
12	8	30	0	0:01:30	14	1	10	0	0:00:07
12	8	30	1	0:00:19	14	1	10	1	0:03:18
14	1	10	0	0:00:07	14	1	10	2	0:08:42
14	1	10	1	0:02:16	14	1	10	3	0:09:59
14	1	10	2	0:01:23	14	1	10	4	0:10:57

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
14	1	10	3	0:00:03	14	1	10	5	0:16:45
14	1	10	4	0:00:02	14	2	10	0	0:00:01
14	2	10	0	0:00:01	14	2	10	1	0:00:06
14	2	10	1	0:00:02	14	2	10	2	0:00:26
14	2	10	2	0:00:07	14	2	10	3	0:00:51
14	2	10	3	0:00:04	14	2	10	4	0:01:18
14	2	10	4	0:00:03	14	2	10	5	0:01:53
14	2	10	5	0:00:04	14	3	10	0	0:00:00
14	3	10	0	0:00:00	14	3	10	1	0:00:01
14	3	10	1	0:00:01	14	3	10	2	0:00:04
14	3	10	2	0:00:01	14	3	10	3	0:00:06
14	3	10	3	0:00:00	14	3	10	4	0:00:08
14	3	10	4	0:00:00	14	3	10	5	0:00:10
14	3	10	5	0:00:00	14	4	10	0	0:00:00
14	4	10	0	0:00:00	14	4	10	1	0:00:06
14	4	10	1	0:00:03	14	4	10	2	0:00:15
14	4	10	2	0:00:02	14	4	10	3	0:00:25
14	4	10	3	0:00:01	14	4	10	4	0:00:34
14	4	10	4	0:00:01	14	4	10	5	0:00:39
14	4	10	5	0:00:00	14	5	10	0	0:00:03
14	5	10	0	0:00:03	14	5	10	1	0:00:06
14	5	10	1	0:00:00	14	5	10	2	0:00:15
14	5	10	2	0:00:02	14	5	10	3	0:00:22

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
14	5	10	3	0:00:01	14	5	10	4	0:00:32
14	5	10	4	0:00:01	14	5	10	5	0:00:44
14	5	10	5	0:00:01	14	6	10	0	0:00:02
14	6	10	0	0:00:02	14	6	10	1	0:00:16
14	6	10	1	0:00:07	14	6	10	2	0:00:46
14	6	10	2	0:00:08	14	6	10	3	0:01:13
14	6	10	3	0:00:04	14	6	10	4	0:02:00
14	6	10	4	0:00:07	14	6	10	5	0:02:23
14	6	10	5	0:00:01	14	7	10	0	0:00:01
14	7	10	0	0:00:01	14	7	10	1	0:00:04
14	7	10	1	0:00:01	14	7	10	2	0:00:10
14	7	10	2	0:00:02	14	7	10	3	0:00:15
14	7	10	3	0:00:01	14	7	10	4	0:00:20
14	7	10	4	0:00:00	14	7	10	5	0:00:24
14	7	10	5	0:00:00	14	8	10	0	0:00:13
14	8	10	0	0:00:13	14	8	10	1	0:00:59
14	8	10	1	0:00:17	14	8	10	2	0:01:45
14	8	10	2	0:00:08	14	8	10	3	0:02:38
14	8	10	3	0:00:07	14	8	10	4	0:03:28
14	8	10	4	0:00:04	14	8	10	5	0:03:31
14	8	10	5	0:00:00	14	1	15	0	0:00:10
14	1	15	0	0:00:10	14	1	15	1	0:08:45
14	1	15	1	0:06:36	14	1	15	2	0:10:28

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
14	1	15	2	0:00:05	14	1	15	3	0:11:57
14	1	15	3	0:00:04	14	2	15	0	0:00:03
14	2	15	0	0:00:03	14	2	15	1	0:00:26
14	2	15	1	0:00:12	14	2	15	2	0:01:03
14	2	15	2	0:00:09	14	2	15	3	0:01:53
14	2	15	3	0:00:09	14	3	15	0	0:00:00
14	3	15	0	0:00:00	14	3	15	1	0:00:04
14	3	15	1	0:00:02	14	3	15	2	0:00:07
14	3	15	2	0:00:01	14	3	15	3	0:00:10
14	3	15	3	0:00:00	14	4	15	0	0:00:02
14	4	15	0	0:00:02	14	4	15	1	0:00:15
14	4	15	1	0:00:06	14	4	15	2	0:00:29
14	4	15	2	0:00:02	14	4	15	3	0:00:39
14	4	15	3	0:00:01	14	5	15	0	0:00:03
14	5	15	0	0:00:03	14	5	15	1	0:00:15
14	5	15	1	0:00:04	14	5	15	2	0:00:26
14	5	15	2	0:00:02	14	5	15	3	0:00:44
14	5	15	3	0:00:02	14	6	15	0	0:00:07
14	6	15	0	0:00:07	14	6	15	1	0:00:47
14	6	15	1	0:00:18	14	6	15	2	0:01:35
14	6	15	2	0:00:10	14	6	15	3	0:02:25
14	6	15	3	0:00:06	14	7	15	0	0:00:02
14	7	15	0	0:00:02	14	7	15	1	0:00:10

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
14	7	15	1	0:00:03	14	7	15	2	0:00:17
14	7	15	2	0:00:01	14	7	15	3	0:00:23
14	7	15	3	0:00:01	14	8	15	0	0:00:33
14	8	15	0	0:00:33	14	8	15	1	0:01:45
14	8	15	1	0:00:24	14	8	15	2	0:03:03
14	8	15	2	0:00:13	14	8	15	3	0:03:31
14	8	15	3	0:00:01	14	1	20	0	0:03:17
14	1	20	0	0:03:17	14	2	20	0	0:00:06
14	1	20	2	0:00:07	14	2	20	1	0:00:51
14	2	20	0	0:00:06	14	2	20	2	0:01:53
14	2	20	1	0:00:22	14	3	20	0	0:00:01
14	2	20	2	0:00:15	14	3	20	1	0:00:06
14	3	20	0	0:00:01	14	3	20	2	0:00:10
14	3	20	1	0:00:02	14	4	20	0	0:00:06
14	3	20	2	0:00:01	14	4	20	1	0:00:25
14	4	20	0	0:00:06	14	4	20	2	0:00:39
14	4	20	1	0:00:06	14	5	20	0	0:00:06
14	4	20	2	0:00:02	14	5	20	1	0:00:22
14	5	20	0	0:00:06	14	5	20	2	0:00:44
14	5	20	1	0:00:05	14	6	20	0	0:00:16
14	5	20	2	0:00:04	14	6	20	1	0:01:14
14	6	20	0	0:00:16	14	6	20	2	0:02:25
14	6	20	1	0:00:23	14	7	20	0	0:00:04

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
14	6	20	2	0:00:15	14	7	20	1	0:00:15
14	7	20	0	0:00:04	14	7	20	2	0:00:23
14	7	20	1	0:00:04	14	8	20	0	0:00:59
14	7	20	2	0:00:01	14	8	20	1	0:02:38
14	8	20	0	0:00:58	14	8	20	2	0:03:30
14	8	20	1	0:00:30	14	1	30	0	0:08:43
14	8	20	2	0:00:04	14	1	30	1	0:11:58
14	1	30	0	0:08:44	14	2	30	0	0:00:26
14	1	30	1	0:00:18	14	2	30	1	0:01:53
14	2	30	0	0:00:26	14	3	30	0	0:00:04
14	2	30	1	0:00:36	14	3	30	1	0:00:10
14	3	30	0	0:00:04	14	4	30	0	0:00:15
14	3	30	1	0:00:02	14	4	30	1	0:00:39
14	4	30	0	0:00:15	14	5	30	0	0:00:15
14	4	30	1	0:00:06	14	5	30	1	0:00:44
14	5	30	0	0:00:15	14	6	30	0	0:00:47
14	5	30	1	0:00:08	14	6	30	1	0:02:24
14	6	30	0	0:00:47	14	7	30	0	0:00:10
14	6	30	1	0:00:33	14	7	30	1	0:00:24
14	7	30	0	0:00:10	14	8	30	0	0:01:45
14	7	30	1	0:00:03	14	8	30	1	0:03:30
14	8	30	0	0:01:44	16	1	10	0	0:00:08
14	8	30	1	0:00:22	16	1	10	1	0:03:45

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
16	1	10	0	0:00:08	16	1	10	2	0:09:59
16	1	10	1	0:02:35	16	1	10	3	0:11:25
16	1	10	2	0:01:35	16	1	10	4	0:12:33
16	1	10	3	0:00:03	16	1	10	5	0:19:08
16	1	10	4	0:00:02	16	2	10	0	0:00:02
16	2	10	0	0:00:02	16	2	10	1	0:00:07
16	2	10	1	0:00:02	16	2	10	2	0:00:29
16	2	10	2	0:00:08	16	2	10	3	0:00:58
16	2	10	3	0:00:05	16	2	10	4	0:01:29
16	2	10	4	0:00:04	16	2	10	5	0:02:09
16	2	10	5	0:00:05	16	3	10	0	0:00:00
16	3	10	0	0:00:00	16	3	10	1	0:00:01
16	3	10	1	0:00:01	16	3	10	2	0:00:04
16	3	10	2	0:00:01	16	3	10	3	0:00:07
16	3	10	3	0:00:00	16	3	10	4	0:00:10
16	3	10	4	0:00:00	16	3	10	5	0:00:11
16	3	10	5	0:00:00	16	4	10	0	0:00:00
16	4	10	0	0:00:00	16	4	10	1	0:00:07
16	4	10	1	0:00:04	16	4	10	2	0:00:17
16	4	10	2	0:00:02	16	4	10	3	0:00:28
16	4	10	3	0:00:02	16	4	10	4	0:00:39
16	4	10	4	0:00:01	16	4	10	5	0:00:45
16	4	10	5	0:00:00	16	5	10	0	0:00:03

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
16	5	10	0	0:00:03	16	5	10	1	0:00:07
16	5	10	1	0:00:01	16	5	10	2	0:00:17
16	5	10	2	0:00:02	16	5	10	3	0:00:26
16	5	10	3	0:00:01	16	5	10	4	0:00:37
16	5	10	4	0:00:01	16	5	10	5	0:00:50
16	5	10	5	0:00:01	16	6	10	0	0:00:02
16	6	10	0	0:00:02	16	6	10	1	0:00:18
16	6	10	1	0:00:08	16	6	10	2	0:00:53
16	6	10	2	0:00:10	16	6	10	3	0:01:23
16	6	10	3	0:00:04	16	6	10	4	0:02:17
16	6	10	4	0:00:08	16	6	10	5	0:02:44
16	6	10	5	0:00:02	16	7	10	0	0:00:01
16	7	10	0	0:00:01	16	7	10	1	0:00:04
16	7	10	1	0:00:01	16	7	10	2	0:00:11
16	7	10	2	0:00:02	16	7	10	3	0:00:17
16	7	10	3	0:00:01	16	7	10	4	0:00:22
16	7	10	4	0:00:00	16	7	10	5	0:00:27
16	7	10	5	0:00:00	16	8	10	0	0:00:15
16	8	10	0	0:00:15	16	8	10	1	0:01:07
16	8	10	1	0:00:20	16	8	10	2	0:02:00
16	8	10	2	0:00:09	16	8	10	3	0:03:01
16	8	10	3	0:00:08	16	8	10	4	0:03:59
16	8	10	4	0:00:05	16	8	10	5	0:04:00

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
16	8	10	5	0:00:00	16	1	15	0	0:00:11
16	1	15	0	0:00:11	16	1	15	1	0:10:00
16	1	15	1	0:07:32	16	1	15	2	0:12:00
16	1	15	2	0:00:06	16	1	15	3	0:13:40
16	1	15	3	0:00:04	16	2	15	0	0:00:03
16	2	15	0	0:00:03	16	2	15	1	0:00:30
16	2	15	1	0:00:14	16	2	15	2	0:01:11
16	2	15	2	0:00:10	16	2	15	3	0:02:09
16	2	15	3	0:00:11	16	3	15	0	0:00:00
16	3	15	0	0:00:00	16	3	15	1	0:00:04
16	3	15	1	0:00:02	16	3	15	2	0:00:08
16	3	15	2	0:00:01	16	3	15	3	0:00:11
16	3	15	3	0:00:00	16	4	15	0	0:00:02
16	4	15	0	0:00:02	16	4	15	1	0:00:17
16	4	15	1	0:00:07	16	4	15	2	0:00:33
16	4	15	2	0:00:03	16	4	15	3	0:00:44
16	4	15	3	0:00:01	16	5	15	0	0:00:04
16	5	15	0	0:00:04	16	5	15	1	0:00:17
16	5	15	1	0:00:05	16	5	15	2	0:00:30
16	5	15	2	0:00:02	16	5	15	3	0:00:50
16	5	15	3	0:00:03	16	6	15	0	0:00:08
16	6	15	0	0:00:08	16	6	15	1	0:00:53
16	6	15	1	0:00:21	16	6	15	2	0:01:48

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
16	6	15	2	0:00:12	16	6	15	3	0:02:46
16	6	15	3	0:00:07	16	7	15	0	0:00:02
16	7	15	0	0:00:02	16	7	15	1	0:00:11
16	7	15	1	0:00:04	16	7	15	2	0:00:20
16	7	15	2	0:00:01	16	7	15	3	0:00:27
16	7	15	3	0:00:01	16	8	15	0	0:00:37
16	8	15	0	0:00:37	16	8	15	1	0:01:59
16	8	15	1	0:00:28	16	8	15	2	0:03:30
16	8	15	2	0:00:15	16	8	15	3	0:04:00
16	8	15	3	0:00:01	16	1	20	0	0:03:45
16	1	20	0	0:03:45	16	2	20	0	0:00:07
16	1	20	2	0:00:08	16	2	20	1	0:00:58
16	2	20	0	0:00:07	16	2	20	2	0:02:09
16	2	20	1	0:00:26	16	3	20	0	0:00:01
16	2	20	2	0:00:18	16	3	20	1	0:00:06
16	3	20	0	0:00:01	16	3	20	2	0:00:11
16	3	20	1	0:00:02	16	4	20	0	0:00:07
16	3	20	2	0:00:01	16	4	20	1	0:00:28
16	4	20	0	0:00:07	16	4	20	2	0:00:45
16	4	20	1	0:00:07	16	5	20	0	0:00:07
16	4	20	2	0:00:02	16	5	20	1	0:00:26
16	5	20	0	0:00:07	16	5	20	2	0:00:50
16	5	20	1	0:00:06	16	6	20	0	0:00:18

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
16	5	20	2	0:00:04	16	6	20	1	0:01:24
16	6	20	0	0:00:18	16	6	20	2	0:02:46
16	6	20	1	0:00:26	16	7	20	0	0:00:04
16	6	20	2	0:00:17	16	7	20	1	0:00:17
16	7	20	0	0:00:04	16	7	20	2	0:00:27
16	7	20	1	0:00:04	16	8	20	0	0:01:07
16	7	20	2	0:00:01	16	8	20	1	0:03:01
16	8	20	0	0:01:07	16	8	20	2	0:04:00
16	8	20	1	0:00:34	16	1	30	0	0:10:00
16	8	20	2	0:00:05	16	1	30	1	0:13:36
16	1	30	0	0:09:58	16	2	30	0	0:00:30
16	1	30	1	0:00:21	16	2	30	1	0:02:10
16	2	30	0	0:00:30	16	3	30	0	0:00:04
16	2	30	1	0:00:41	16	3	30	1	0:00:11
16	3	30	0	0:00:04	16	4	30	0	0:00:17
16	3	30	1	0:00:02	16	4	30	1	0:00:45
16	4	30	0	0:00:17	16	5	30	0	0:00:17
16	4	30	1	0:00:07	16	5	30	1	0:00:50
16	5	30	0	0:00:17	16	6	30	0	0:00:54
16	5	30	1	0:00:09	16	6	30	1	0:02:45
16	6	30	0	0:00:54	16	7	30	0	0:00:11
16	6	30	1	0:00:38	16	7	30	1	0:00:27
16	7	30	0	0:00:11	16	8	30	0	0:01:59

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
16	7	30	1	0:00:03	16	8	30	1	0:04:01
16	8	30	0	0:02:00	18	1	10	0	0:00:09
16	8	30	1	0:00:25	18	1	10	1	0:04:14
18	1	10	0	0:00:09	18	1	10	2	0:11:12
18	1	10	1	0:02:54	18	1	10	3	0:12:50
18	1	10	2	0:01:46	18	1	10	4	0:14:08
18	1	10	3	0:00:04	18	1	10	5	0:21:38
18	1	10	4	0:00:02	18	2	10	0	0:00:02
18	2	10	0	0:00:02	18	2	10	1	0:00:08
18	2	10	1	0:00:02	18	2	10	2	0:00:33
18	2	10	2	0:00:09	18	2	10	3	0:01:05
18	2	10	3	0:00:06	18	2	10	4	0:01:40
18	2	10	4	0:00:04	18	2	10	5	0:02:25
18	2	10	5	0:00:06	18	3	10	0	0:00:00
18	3	10	0	0:00:00	18	3	10	1	0:00:01
18	3	10	1	0:00:01	18	3	10	2	0:00:05
18	3	10	2	0:00:01	18	3	10	3	0:00:08
18	3	10	3	0:00:00	18	3	10	4	0:00:11
18	3	10	4	0:00:00	18	3	10	5	0:00:13
18	3	10	5	0:00:00	18	4	10	0	0:00:01
18	4	10	0	0:00:01	18	4	10	1	0:00:08
18	4	10	1	0:00:04	18	4	10	2	0:00:19
18	4	10	2	0:00:02	18	4	10	3	0:00:32

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
18	4	10	3	0:00:02	18	4	10	4	0:00:44
18	4	10	4	0:00:01	18	4	10	5	0:00:50
18	4	10	5	0:00:00	18	5	10	0	0:00:04
18	5	10	0	0:00:04	18	5	10	1	0:00:08
18	5	10	1	0:00:01	18	5	10	2	0:00:19
18	5	10	2	0:00:03	18	5	10	3	0:00:29
18	5	10	3	0:00:01	18	5	10	4	0:00:42
18	5	10	4	0:00:01	18	5	10	5	0:00:56
18	5	10	5	0:00:01	18	6	10	0	0:00:02
18	6	10	0	0:00:02	18	6	10	1	0:00:20
18	6	10	1	0:00:09	18	6	10	2	0:01:00
18	6	10	2	0:00:11	18	6	10	3	0:01:34
18	6	10	3	0:00:05	18	6	10	4	0:02:34
18	6	10	4	0:00:09	18	6	10	5	0:03:06
18	6	10	5	0:00:02	18	7	10	0	0:00:01
18	7	10	0	0:00:01	18	7	10	1	0:00:05
18	7	10	1	0:00:02	18	7	10	2	0:00:13
18	7	10	2	0:00:02	18	7	10	3	0:00:19
18	7	10	3	0:00:01	18	7	10	4	0:00:25
18	7	10	4	0:00:00	18	7	10	5	0:00:30
18	7	10	5	0:00:00	18	8	10	0	0:00:17
18	8	10	0	0:00:17	18	8	10	1	0:01:16
18	8	10	1	0:00:22	18	8	10	2	0:02:14

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
18	8	10	2	0:00:11	18	8	10	3	0:03:23
18	8	10	3	0:00:09	18	8	10	4	0:04:28
18	8	10	4	0:00:05	18	8	10	5	0:04:30
18	8	10	5	0:00:00	18	1	15	0	0:00:13
18	1	15	0	0:00:13	18	1	15	1	0:11:14
18	1	15	1	0:08:29	18	1	15	2	0:13:29
18	1	15	2	0:00:07	18	1	15	3	0:15:23
18	1	15	3	0:00:05	18	2	15	0	0:00:03
18	2	15	0	0:00:03	18	2	15	1	0:00:33
18	2	15	1	0:00:16	18	2	15	2	0:01:20
18	2	15	2	0:00:11	18	2	15	3	0:02:25
18	2	15	3	0:00:12	18	3	15	0	0:00:00
18	3	15	0	0:00:00	18	3	15	1	0:00:05
18	3	15	1	0:00:03	18	3	15	2	0:00:09
18	3	15	2	0:00:01	18	3	15	3	0:00:13
18	3	15	3	0:00:00	18	4	15	0	0:00:03
18	4	15	0	0:00:03	18	4	15	1	0:00:19
18	4	15	1	0:00:07	18	4	15	2	0:00:37
18	4	15	2	0:00:03	18	4	15	3	0:00:50
18	4	15	3	0:00:01	18	5	15	0	0:00:04
18	5	15	0	0:00:04	18	5	15	1	0:00:19
18	5	15	1	0:00:05	18	5	15	2	0:00:34
18	5	15	2	0:00:02	18	5	15	3	0:00:56

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
18	5	15	3	0:00:03	18	6	15	0	0:00:09
18	6	15	0	0:00:09	18	6	15	1	0:01:01
18	6	15	1	0:00:23	18	6	15	2	0:02:02
18	6	15	2	0:00:13	18	6	15	3	0:03:06
18	6	15	3	0:00:08	18	7	15	0	0:00:02
18	7	15	0	0:00:02	18	7	15	1	0:00:13
18	7	15	1	0:00:04	18	7	15	2	0:00:22
18	7	15	2	0:00:01	18	7	15	3	0:00:30
18	7	15	3	0:00:01	18	8	15	0	0:00:42
18	8	15	0	0:00:42	18	8	15	1	0:02:15
18	8	15	1	0:00:31	18	8	15	2	0:03:55
18	8	15	2	0:00:17	18	8	15	3	0:04:30
18	8	15	3	0:00:01	18	1	20	0	0:04:13
18	1	20	0	0:04:14	18	2	20	0	0:00:08
18	1	20	1	0:02:23	18	2	20	1	0:01:05
18	1	20	2	0:00:09	18	2	20	2	0:02:25
18	2	20	0	0:00:08	18	3	20	0	0:00:01
18	2	20	1	0:00:29	18	3	20	1	0:00:08
18	2	20	2	0:00:20	18	3	20	2	0:00:13
18	3	20	0	0:00:01	18	4	20	0	0:00:08
18	3	20	1	0:00:03	18	4	20	1	0:00:32
18	3	20	2	0:00:01	18	4	20	2	0:00:50
18	4	20	0	0:00:08	18	5	20	0	0:00:08

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
18	4	20	1	0:00:08	18	5	20	1	0:00:29
18	4	20	2	0:00:02	18	5	20	2	0:00:56
18	5	20	0	0:00:08	18	6	20	0	0:00:21
18	5	20	1	0:00:07	18	6	20	1	0:01:35
18	5	20	2	0:00:05	18	6	20	2	0:03:06
18	6	20	0	0:00:21	18	7	20	0	0:00:05
18	6	20	1	0:00:29	18	7	20	1	0:00:19
18	6	20	2	0:00:19	18	7	20	2	0:00:30
18	7	20	0	0:00:05	18	8	20	0	0:01:15
18	7	20	1	0:00:05	18	8	20	1	0:03:23
18	7	20	2	0:00:01	18	8	20	2	0:04:30
18	8	20	0	0:01:16	18	1	20	1	0:12:52
18	8	20	1	0:00:38	18	1	20	2	0:15:26
18	8	20	2	0:00:06	18	1	30	0	0:11:12
18	1	30	0	0:11:13	18	1	30	1	0:15:26
18	1	30	1	0:00:24	18	2	30	0	0:00:33
18	2	30	0	0:00:33	18	2	30	1	0:02:26
18	2	30	1	0:00:47	18	3	30	0	0:00:05
18	3	30	0	0:00:05	18	3	30	1	0:00:13
18	3	30	1	0:00:02	18	4	30	0	0:00:19
18	4	30	0	0:00:19	18	4	30	1	0:00:51
18	4	30	1	0:00:07	18	5	30	0	0:00:19
18	5	30	0	0:00:19	18	5	30	1	0:00:56

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
18	5	30	1	0:00:10	18	6	30	0	0:01:01
18	6	30	0	0:01:01	18	6	30	1	0:03:07
18	6	30	1	0:00:43	18	7	30	0	0:00:13
18	7	30	0	0:00:13	18	7	30	1	0:00:30
18	7	30	1	0:00:04	18	8	30	0	0:02:14
18	8	30	0	0:02:14	18	8	30	1	0:04:31
18	8	30	1	0:00:28	20	1	10	0	0:00:10
20	1	10	0	0:00:10	20	1	10	1	0:04:41
20	1	10	1	0:03:14	20	1	10	2	0:12:30
20	1	10	2	0:01:58	20	1	10	3	0:14:12
20	1	10	3	0:00:04	20	1	10	4	0:15:43
20	1	10	4	0:00:03	20	1	10	5	0:23:55
20	2	10	0	0:00:02	20	2	10	0	0:00:02
20	2	10	1	0:00:02	20	2	10	1	0:00:08
20	2	10	2	0:00:10	20	2	10	2	0:00:37
20	2	10	3	0:00:06	20	2	10	3	0:01:12
20	2	10	4	0:00:05	20	2	10	4	0:01:52
20	2	10	5	0:00:06	20	2	10	5	0:02:42
20	3	10	0	0:00:00	20	3	10	0	0:00:00
20	3	10	1	0:00:01	20	3	10	1	0:00:01
20	3	10	2	0:00:01	20	3	10	2	0:00:05
20	3	10	3	0:00:00	20	3	10	3	0:00:08
20	3	10	4	0:00:00	20	3	10	4	0:00:12

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
20	3	10	5	0:00:00	20	3	10	5	0:00:14
20	4	10	0	0:00:01	20	4	10	0	0:00:01
20	4	10	1	0:00:05	20	4	10	1	0:00:09
20	4	10	2	0:00:03	20	4	10	2	0:00:21
20	4	10	3	0:00:02	20	4	10	3	0:00:35
20	4	10	4	0:00:01	20	4	10	4	0:00:50
20	4	10	5	0:00:00	20	4	10	5	0:00:56
20	5	10	0	0:00:04	20	5	10	0	0:00:04
20	5	10	1	0:00:01	20	5	10	1	0:00:08
20	5	10	2	0:00:03	20	5	10	2	0:00:21
20	5	10	3	0:00:01	20	5	10	3	0:00:32
20	5	10	4	0:00:01	20	5	10	4	0:00:46
20	5	10	5	0:00:01	20	5	10	5	0:01:02
20	6	10	0	0:00:03	20	6	10	0	0:00:03
20	6	10	1	0:00:10	20	6	10	1	0:00:22
20	6	10	2	0:00:12	20	6	10	2	0:01:06
20	6	10	3	0:00:05	20	6	10	3	0:01:44
20	6	10	4	0:00:11	20	6	10	4	0:02:51
20	6	10	5	0:00:02	20	6	10	5	0:03:25
20	7	10	0	0:00:01	20	7	10	0	0:00:01
20	7	10	1	0:00:02	20	7	10	1	0:00:05
20	7	10	2	0:00:02	20	7	10	2	0:00:14
20	7	10	3	0:00:01	20	7	10	3	0:00:21

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
20	7	10	4	0:00:01	20	7	10	4	0:00:28
20	7	10	5	0:00:00	20	7	10	5	0:00:34
20	8	10	0	0:00:19	20	8	10	0	0:00:19
20	8	10	1	0:00:24	20	8	10	1	0:01:24
20	8	10	2	0:00:12	20	8	10	2	0:02:30
20	8	10	3	0:00:10	20	8	10	3	0:03:46
20	8	10	4	0:00:06	20	8	10	4	0:04:58
20	8	10	5	0:00:00	20	8	10	5	0:04:59
20	1	15	0	0:00:14	20	1	15	0	0:00:14
20	1	15	1	0:09:27	20	1	15	1	0:12:37
20	1	15	2	0:00:08	20	1	15	2	0:15:00
20	1	15	3	0:00:05	20	1	15	3	0:17:02
20	2	15	0	0:00:04	20	2	15	0	0:00:04
20	2	15	1	0:00:18	20	2	15	1	0:00:37
20	2	15	2	0:00:12	20	2	15	2	0:01:29
20	2	15	3	0:00:13	20	2	15	3	0:02:42
20	3	15	0	0:00:00	20	3	15	0	0:00:00
20	3	15	1	0:00:03	20	3	15	1	0:00:05
20	3	15	2	0:00:01	20	3	15	2	0:00:10
20	3	15	3	0:00:00	20	3	15	3	0:00:14
20	4	15	0	0:00:03	20	4	15	0	0:00:03
20	4	15	1	0:00:08	20	4	15	1	0:00:21
20	4	15	2	0:00:04	20	4	15	2	0:00:42

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
20	4	15	3	0:00:01	20	4	15	3	0:00:56
20	5	15	0	0:00:05	20	5	15	0	0:00:05
20	5	15	1	0:00:06	20	5	15	1	0:00:21
20	5	15	2	0:00:02	20	5	15	2	0:00:38
20	5	15	3	0:00:03	20	5	15	3	0:01:03
20	6	15	0	0:00:10	20	6	15	0	0:00:10
20	6	15	1	0:00:26	20	6	15	1	0:01:07
20	6	15	2	0:00:15	20	6	15	2	0:02:16
20	6	15	3	0:00:09	20	6	15	3	0:03:28
20	7	15	0	0:00:02	20	7	15	0	0:00:02
20	7	15	1	0:00:05	20	7	15	1	0:00:14
20	7	15	2	0:00:02	20	7	15	2	0:00:25
20	7	15	3	0:00:01	20	7	15	3	0:00:34
20	8	15	0	0:00:47	20	8	15	0	0:00:47
20	8	15	1	0:00:34	20	8	15	1	0:02:30
20	8	15	2	0:00:19	20	8	15	2	0:04:21
20	8	15	3	0:00:02	20	8	15	3	0:05:00
20	1	20	0	0:04:43	20	1	20	0	0:04:40
20	1	20	1	0:02:39	20	2	20	0	0:00:09
20	1	20	2	0:00:10	20	2	20	1	0:01:13
20	2	20	0	0:00:09	20	2	20	2	0:02:42
20	2	20	1	0:00:32	20	3	20	0	0:00:01
20	2	20	2	0:00:22	20	3	20	1	0:00:08

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
20	3	20	0	0:00:01	20	3	20	2	0:00:14
20	3	20	1	0:00:03	20	4	20	0	0:00:09
20	3	20	2	0:00:01	20	4	20	1	0:00:35
20	4	20	0	0:00:09	20	4	20	2	0:00:56
20	4	20	1	0:00:09	20	5	20	0	0:00:08
20	4	20	2	0:00:02	20	5	20	1	0:00:32
20	5	20	0	0:00:08	20	5	20	2	0:01:03
20	5	20	1	0:00:08	20	6	20	0	0:00:23
20	5	20	2	0:00:05	20	6	20	1	0:01:45
20	6	20	0	0:00:23	20	6	20	2	0:03:27
20	6	20	1	0:00:32	20	7	20	0	0:00:05
20	6	20	2	0:00:22	20	7	20	1	0:00:21
20	7	20	0	0:00:05	20	7	20	2	0:00:34
20	7	20	1	0:00:05	20	8	20	0	0:01:24
20	7	20	2	0:00:01	20	8	20	1	0:03:46
20	8	20	0	0:01:24	20	8	20	2	0:05:00
20	8	20	1	0:00:43	20	1	20	1	0:14:16
20	8	20	2	0:00:06	20	1	20	2	0:17:07
20	1	30	0	0:12:28	20	1	30	0	0:12:29
20	1	30	1	0:00:26	20	1	30	1	0:17:07
20	2	30	0	0:00:37	20	2	30	0	0:00:37
20	2	30	1	0:00:52	20	2	30	1	0:02:42
20	3	30	0	0:00:05	20	3	30	0	0:00:05

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
20	3	30	1	0:00:02	20	3	30	1	0:00:14
20	4	30	0	0:00:21	20	4	30	0	0:00:21
20	4	30	1	0:00:08	20	4	30	1	0:00:56
20	5	30	0	0:00:21	20	5	30	0	0:00:21
20	5	30	1	0:00:11	20	5	30	1	0:01:03
20	6	30	0	0:01:07	20	6	30	0	0:01:08
20	6	30	1	0:00:47	20	6	30	1	0:03:27
20	7	30	0	0:00:14	20	7	30	0	0:00:14
20	7	30	1	0:00:04	20	7	30	1	0:00:34
20	8	30	0	0:02:30	20	8	30	0	0:02:30
20	8	30	1	0:00:31	20	8	30	1	0:05:00
25	1	15	0	0:00:18	25	1	15	0	0:00:18
25	1	15	0	0:00:18	25	1	15	0	0:00:18
25	1	15	1	0:11:54	25	1	15	1	0:15:57
25	1	15	2	0:00:10	25	1	15	2	0:19:05
25	1	15	3	0:00:07	25	1	15	3	0:21:46
25	2	15	0	0:00:05	25	2	15	0	0:00:05
25	2	15	1	0:00:22	25	2	15	1	0:00:47
25	2	15	2	0:00:16	25	2	15	2	0:01:54
25	2	15	3	0:00:17	25	2	15	3	0:03:26
25	3	15	0	0:00:00	25	3	15	0	0:00:00
25	3	15	1	0:00:04	25	3	15	1	0:00:07
25	3	15	2	0:00:01	25	3	15	2	0:00:13

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
25	3	15	3	0:00:01	25	3	15	3	0:00:19
25	4	15	0	0:00:04	25	4	15	0	0:00:04
25	4	15	1	0:00:10	25	4	15	1	0:00:27
25	4	15	2	0:00:05	25	4	15	2	0:00:53
25	4	15	3	0:00:01	25	4	15	3	0:01:11
25	5	15	0	0:00:06	25	5	15	0	0:00:06
25	5	15	1	0:00:07	25	5	15	1	0:00:27
25	5	15	2	0:00:03	25	5	15	2	0:00:48
25	5	15	3	0:00:04	25	5	15	3	0:01:21
25	6	15	0	0:00:13	25	6	15	0	0:00:13
25	6	15	1	0:00:33	25	6	15	1	0:01:27
25	6	15	2	0:00:19	25	6	15	2	0:02:55
25	6	15	3	0:00:12	25	6	15	3	0:04:26
25	7	15	0	0:00:03	25	7	15	0	0:00:03
25	7	15	1	0:00:06	25	7	15	1	0:00:18
25	7	15	2	0:00:02	25	7	15	2	0:00:32
25	7	15	3	0:00:01	25	7	15	3	0:00:43
25	8	15	0	0:01:01	25	8	15	0	0:01:01
25	8	15	1	0:00:44	25	8	15	1	0:03:12
25	8	15	2	0:00:25	25	8	15	2	0:05:38
25	8	15	3	0:00:02	25	8	15	3	0:06:34
25	1	20	0	0:06:00	25	1	20	0	0:06:00
25	1	20	1	0:03:19	25	2	20	0	0:00:11

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
25	1	20	2	0:00:12	25	2	20	1	0:01:33
25	2	20	0	0:00:11	25	2	20	2	0:03:27
25	2	20	1	0:00:41	25	3	20	0	0:00:02
25	2	20	2	0:00:28	25	3	20	1	0:00:11
25	3	20	0	0:00:02	25	3	20	2	0:00:18
25	3	20	1	0:00:04	25	4	20	0	0:00:11
25	3	20	2	0:00:01	25	4	20	1	0:00:45
25	4	20	0	0:00:11	25	4	20	2	0:01:11
25	4	20	1	0:00:12	25	5	20	0	0:00:11
25	4	20	2	0:00:03	25	5	20	1	0:00:41
25	5	20	0	0:00:11	25	5	20	2	0:01:21
25	5	20	1	0:00:10	25	6	20	0	0:00:30
25	5	20	2	0:00:07	25	6	20	1	0:02:16
25	6	20	0	0:00:30	25	6	20	2	0:04:26
25	6	20	1	0:00:42	25	7	20	0	0:00:07
25	6	20	2	0:00:28	25	7	20	1	0:00:27
25	7	20	0	0:00:07	25	7	20	2	0:00:43
25	7	20	1	0:00:07	25	8	20	0	0:01:49
25	7	20	2	0:00:02	25	8	20	1	0:04:50
25	8	20	0	0:01:49	25	8	20	2	0:06:28
25	8	20	1	0:00:55	25	1	20	1	0:17:51
25	8	20	2	0:00:08	25	1	20	2	0:21:26
25	1	30	0	0:15:56	25	1	30	0	0:15:57

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
25	1	30	1	0:00:33	25	1	30	1	0:21:54
25	2	30	0	0:00:48	25	2	30	0	0:00:47
25	2	30	1	0:01:06	25	2	30	1	0:03:26
25	3	30	0	0:00:07	25	3	30	0	0:00:07
25	3	30	1	0:00:03	25	3	30	1	0:00:19
25	4	30	0	0:00:27	25	4	30	0	0:00:27
25	4	30	1	0:00:11	25	4	30	1	0:01:12
25	5	30	0	0:00:27	25	5	30	0	0:00:27
25	5	30	1	0:00:15	25	5	30	1	0:01:20
25	6	30	0	0:01:27	25	6	30	0	0:01:27
25	6	30	1	0:01:02	25	6	30	1	0:04:27
25	7	30	0	0:00:19	25	7	30	0	0:00:18
25	7	30	1	0:00:06	25	7	30	1	0:00:43
25	8	30	0	0:03:11	25	8	30	0	0:03:15
25	8	30	1	0:00:41	25	8	30	1	0:06:29
30	1	15	0	0:00:23	30	1	15	0	0:00:22
30	1	15	1	0:14:43	30	1	15	1	0:19:14
30	1	15	2	0:00:12	30	1	15	2	0:23:00
30	1	15	3	0:00:08	30	1	15	3	0:26:08
30	2	15	0	0:00:06	30	2	15	0	0:00:06
30	2	15	1	0:00:27	30	2	15	1	0:00:57
30	2	15	2	0:00:19	30	2	15	2	0:02:17
30	2	15	3	0:00:21	30	2	15	3	0:04:07

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
30	3	15	0	0:00:01	30	3	15	0	0:00:01
30	3	15	1	0:00:04	30	3	15	1	0:00:08
30	3	15	2	0:00:01	30	3	15	2	0:00:15
30	3	15	3	0:00:01	30	3	15	3	0:00:22
30	4	15	0	0:00:05	30	4	15	0	0:00:05
30	4	15	1	0:00:13	30	4	15	1	0:00:32
30	4	15	2	0:00:05	30	4	15	2	0:01:04
30	4	15	3	0:00:02	30	4	15	3	0:01:25
30	5	15	0	0:00:07	30	5	15	0	0:00:07
30	5	15	1	0:00:09	30	5	15	1	0:00:32
30	5	15	2	0:00:04	30	5	15	2	0:00:58
30	5	15	3	0:00:05	30	5	15	3	0:01:36
30	6	15	0	0:00:16	30	6	15	0	0:00:16
30	6	15	1	0:00:40	30	6	15	1	0:01:43
30	6	15	2	0:00:23	30	6	15	2	0:03:28
30	6	15	3	0:00:14	30	6	15	3	0:05:19
30	7	15	0	0:00:04	30	7	15	0	0:00:04
30	7	15	1	0:00:07	30	7	15	1	0:00:21
30	7	15	2	0:00:02	30	7	15	2	0:00:38
30	7	15	3	0:00:01	30	7	15	3	0:00:52
30	8	15	0	0:01:12	30	8	15	0	0:01:12
30	8	15	1	0:00:53	30	8	15	1	0:03:50
30	8	15	2	0:00:30	30	8	15	2	0:06:42

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
30	8	15	3	0:00:02	30	8	15	3	0:07:40
30	1	20	0	0:07:13	30	1	20	0	0:07:14
30	1	20	2	0:00:15	30	2	20	0	0:00:13
30	2	20	0	0:00:13	30	2	20	1	0:01:52
30	2	20	1	0:00:50	30	2	20	2	0:04:08
30	2	20	2	0:00:34	30	3	20	0	0:00:02
30	3	20	0	0:00:02	30	3	20	1	0:00:13
30	3	20	1	0:00:04	30	3	20	2	0:00:22
30	3	20	2	0:00:01	30	4	20	0	0:00:13
30	4	20	0	0:00:13	30	4	20	1	0:00:54
30	4	20	1	0:00:14	30	4	20	2	0:01:26
30	4	20	2	0:00:04	30	5	20	0	0:00:13
30	5	20	0	0:00:13	30	5	20	1	0:00:49
30	5	20	1	0:00:12	30	5	20	2	0:01:36
30	5	20	2	0:00:08	30	6	20	0	0:00:35
30	6	20	0	0:00:35	30	6	20	1	0:02:42
30	6	20	1	0:00:50	30	6	20	2	0:05:18
30	6	20	2	0:00:33	30	7	20	0	0:00:08
30	7	20	0	0:00:08	30	7	20	1	0:00:32
30	7	20	1	0:00:08	30	7	20	2	0:00:52
30	7	20	2	0:00:02	30	8	20	0	0:02:09
30	8	20	0	0:02:09	30	8	20	1	0:05:48
30	8	20	1	0:01:07	30	8	20	2	0:07:42

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
30	8	20	2	0:00:10	30	1	20	1	0:21:27
30	1	20	1	0:03:59	30	1	20	2	0:25:46
30	1	30	0	0:19:07	30	1	30	0	0:19:06
30	1	30	1	0:00:40	30	1	30	1	0:26:14
30	2	30	0	0:00:57	30	2	30	0	0:00:57
30	2	30	1	0:01:20	30	2	30	1	0:04:07
30	3	30	0	0:00:08	30	3	30	0	0:00:08
30	3	30	1	0:00:04	30	3	30	1	0:00:22
30	4	30	0	0:00:32	30	4	30	0	0:00:32
30	4	30	1	0:00:13	30	4	30	1	0:01:26
30	5	30	0	0:00:32	30	5	30	0	0:00:32
30	5	30	1	0:00:17	30	5	30	1	0:01:36
30	6	30	0	0:01:44	30	6	30	0	0:01:43
30	6	30	1	0:01:13	30	6	30	1	0:05:19
30	7	30	0	0:00:22	30	7	30	0	0:00:21
30	7	30	1	0:00:07	30	7	30	1	0:00:52
30	8	30	0	0:03:49	30	8	30	0	0:03:49
30	8	30	1	0:00:49	30	8	30	1	0:07:41
40	1	15	0	0:00:29	40	1	15	0	0:00:29
40	1	15	1	0:19:12	40	1	15	1	0:25:29
40	1	15	2	0:00:16	40	1	15	2	0:30:39
40	1	15	3	0:00:11	40	1	15	3	0:35:06
40	2	15	0	0:00:07	40	2	15	0	0:00:07

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
40	2	15	1	0:00:37	40	2	15	1	0:01:16
40	2	15	2	0:00:26	40	2	15	2	0:03:05
40	2	15	3	0:00:28	40	2	15	3	0:05:32
40	3	15	0	0:00:01	40	3	15	0	0:00:01
40	3	15	1	0:00:06	40	3	15	1	0:00:11
40	3	15	2	0:00:02	40	3	15	2	0:00:21
40	3	15	3	0:00:01	40	3	15	3	0:00:30
40	4	15	0	0:00:06	40	4	15	0	0:00:06
40	4	15	1	0:00:17	40	4	15	1	0:00:43
40	4	15	2	0:00:07	40	4	15	2	0:01:26
40	4	15	3	0:00:02	40	4	15	3	0:01:55
40	5	15	0	0:00:10	40	5	15	0	0:00:10
40	5	15	1	0:00:13	40	5	15	1	0:00:44
40	5	15	2	0:00:05	40	5	15	2	0:01:18
40	5	15	3	0:00:07	40	5	15	3	0:02:10
40	6	15	0	0:00:21	40	6	15	0	0:00:21
40	6	15	1	0:00:53	40	6	15	1	0:02:20
40	6	15	2	0:00:31	40	6	15	2	0:04:45
40	6	15	3	0:00:19	40	6	15	3	0:07:06
40	7	15	0	0:00:05	40	7	15	0	0:00:05
40	7	15	1	0:00:11	40	7	15	1	0:00:29
40	7	15	2	0:00:03	40	7	15	2	0:00:51
40	7	15	3	0:00:02	40	7	15	3	0:01:10

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
40	8	15	0	0:01:37	40	8	15	0	0:01:36
40	8	15	1	0:01:12	40	8	15	1	0:05:11
40	8	15	2	0:00:40	40	8	15	2	0:08:56
40	8	15	3	0:00:03	40	8	15	3	0:10:20
40	1	20	0	0:09:41	40	1	20	0	0:09:42
40	1	20	2	0:00:20	40	2	20	0	0:00:18
40	2	20	0	0:00:18	40	2	20	1	0:02:30
40	2	20	1	0:01:07	40	2	20	2	0:05:33
40	2	20	2	0:00:45	40	3	20	0	0:00:03
40	3	20	0	0:00:03	40	3	20	1	0:00:18
40	3	20	1	0:00:06	40	3	20	2	0:00:30
40	3	20	2	0:00:02	40	4	20	0	0:00:18
40	4	20	0	0:00:18	40	4	20	1	0:01:13
40	4	20	1	0:00:19	40	4	20	2	0:01:56
40	4	20	2	0:00:05	40	5	20	0	0:00:18
40	5	20	0	0:00:17	40	5	20	1	0:01:07
40	5	20	1	0:00:16	40	5	20	2	0:02:10
40	5	20	2	0:00:11	40	6	20	0	0:00:48
40	6	20	0	0:00:48	40	6	20	1	0:03:38
40	6	20	1	0:01:08	40	6	20	2	0:07:11
40	6	20	2	0:00:45	40	7	20	0	0:00:11
40	7	20	0	0:00:11	40	7	20	1	0:00:44
40	7	20	1	0:00:12	40	7	20	2	0:01:10

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
40	7	20	2	0:00:03	40	8	20	0	0:02:55
40	8	20	0	0:02:53	40	8	20	1	0:07:46
40	8	20	1	0:01:31	40	8	20	2	0:10:22
40	8	20	2	0:00:13	40	1	20	1	0:28:48
40	1	20	1	0:05:20	40	1	20	2	0:34:27
40	1	30	0	0:25:42	40	1	30	0	0:25:34
40	1	30	1	0:00:54	40	1	30	1	0:35:11
40	2	30	0	0:01:16	40	2	30	0	0:01:16
40	2	30	1	0:01:46	40	2	30	1	0:05:32
40	3	30	0	0:00:11	40	3	30	0	0:00:11
40	3	30	1	0:00:05	40	3	30	1	0:00:30
40	4	30	0	0:00:43	40	4	30	0	0:00:43
40	4	30	1	0:00:18	40	4	30	1	0:01:55
40	5	30	0	0:00:45	40	5	30	0	0:00:44
40	5	30	1	0:00:24	40	5	30	1	0:02:11
40	6	30	0	0:02:21	40	6	30	0	0:02:19
40	6	30	1	0:01:38	40	6	30	1	0:07:09
40	7	30	0	0:00:29	40	7	30	0	0:00:29
40	7	30	1	0:00:09	40	7	30	1	0:01:10
40	8	30	0	0:05:10	40	8	30	0	0:05:10
40	8	30	1	0:01:05	40	8	30	1	0:10:23
50	1	15	0	0:00:37	50	1	15	0	0:00:37
50	1	15	1	0:24:16	50	1	15	1	0:31:59

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
50	1	15	2	0:00:20	50	1	15	2	0:38:42
50	1	15	3	0:00:14	50	1	15	3	0:44:11
50	2	15	0	0:00:09	50	2	15	0	0:00:09
50	2	15	1	0:00:45	50	2	15	1	0:01:35
50	2	15	2	0:00:32	50	2	15	2	0:03:50
50	2	15	3	0:00:35	50	2	15	3	0:06:55
50	3	15	0	0:00:01	50	3	15	0	0:00:01
50	3	15	1	0:00:07	50	3	15	1	0:00:13
50	3	15	2	0:00:02	50	3	15	2	0:00:26
50	3	15	3	0:00:01	50	3	15	3	0:00:37
50	4	15	0	0:00:08	50	4	15	0	0:00:08
50	4	15	1	0:00:21	50	4	15	1	0:00:54
50	4	15	2	0:00:09	50	4	15	2	0:01:47
50	4	15	3	0:00:03	50	4	15	3	0:02:23
50	5	15	0	0:00:12	50	5	15	0	0:00:12
50	5	15	1	0:00:15	50	5	15	1	0:00:54
50	5	15	2	0:00:06	50	5	15	2	0:01:36
50	5	15	3	0:00:08	50	5	15	3	0:02:41
50	6	15	0	0:00:26	50	6	15	0	0:00:26
50	6	15	1	0:01:07	50	6	15	1	0:02:52
50	6	15	2	0:00:38	50	6	15	2	0:05:51
50	6	15	3	0:00:24	50	6	15	3	0:08:50
50	7	15	0	0:00:06	50	7	15	0	0:00:06

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
50	7	15	1	0:00:12	50	7	15	1	0:00:36
50	7	15	2	0:00:04	50	7	15	2	0:01:03
50	7	15	3	0:00:02	50	7	15	3	0:01:26
50	8	15	0	0:02:00	50	8	15	0	0:02:00
50	8	15	1	0:01:28	50	8	15	1	0:06:24
50	8	15	2	0:00:49	50	8	15	2	0:11:10
50	8	15	3	0:00:04	50	8	15	3	0:12:49
50	1	20	0	0:12:02	50	1	20	0	0:12:02
50	1	20	2	0:00:25	50	2	20	0	0:00:22
50	2	20	0	0:00:22	50	2	20	1	0:03:06
50	2	20	1	0:01:23	50	2	20	2	0:06:54
50	2	20	2	0:00:57	50	3	20	0	0:00:04
50	3	20	0	0:00:04	50	3	20	1	0:00:22
50	3	20	1	0:00:07	50	3	20	2	0:00:37
50	3	20	2	0:00:02	50	4	20	0	0:00:22
50	4	20	0	0:00:22	50	4	20	1	0:01:31
50	4	20	1	0:00:24	50	4	20	2	0:02:23
50	4	20	2	0:00:06	50	5	20	0	0:00:22
50	5	20	0	0:00:21	50	5	20	1	0:01:23
50	5	20	1	0:00:20	50	5	20	2	0:02:41
50	5	20	2	0:00:14	50	6	20	0	0:00:59
50	6	20	0	0:00:59	50	6	20	1	0:04:31
50	6	20	1	0:01:24	50	6	20	2	0:08:53

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
50	6	20	2	0:00:55	50	7	20	0	0:00:13
50	7	20	0	0:00:13	50	7	20	1	0:00:54
50	7	20	1	0:00:14	50	7	20	2	0:01:27
50	7	20	2	0:00:04	50	8	20	0	0:03:34
50	8	20	0	0:03:34	50	8	20	1	0:09:38
50	8	20	1	0:01:52	50	8	20	2	0:12:47
50	8	20	2	0:00:16	50	1	20	1	0:35:59
50	1	20	1	0:06:40	50	1	20	2	0:43:07
50	1	30	0	0:31:52	50	1	30	0	0:31:50
50	1	30	1	0:01:07	50	1	30	1	0:43:41
50	2	30	0	0:01:35	50	2	30	0	0:01:35
50	2	30	1	0:02:12	50	2	30	1	0:06:55
50	3	30	0	0:00:13	50	3	30	0	0:00:13
50	3	30	1	0:00:06	50	3	30	1	0:00:37
50	4	30	0	0:00:54	50	4	30	0	0:00:54
50	4	30	1	0:00:22	50	4	30	1	0:02:24
50	5	30	0	0:00:54	50	5	30	0	0:00:54
50	5	30	1	0:00:29	50	5	30	1	0:02:41
50	6	30	0	0:02:53	50	6	30	0	0:02:52
50	6	30	1	0:02:02	50	6	30	1	0:08:51
50	7	30	0	0:00:36	50	7	30	0	0:00:36
50	7	30	1	0:00:11	50	7	30	1	0:01:27
50	8	30	0	0:06:24	50	8	30	0	0:06:24

Table 5: SDS time for each camera at each time segment in **R²MMT**. Time is in format h:mm:ss.

Standard Configuration					**Configuration with Memory**				
K	**Camera**	τ	t	**Time**	**K**	**Camera**	τ	t	**Time**
50	8	30	1	0:01:21	50	8	30	1	0:12:49

.0.0.2 Person Crop Distribution by Camera

This shows the distribution of person crops generated by R^{2}MMT on DukeMTMC-video [1] for each camera for each time segment for each value of τ. This is used in Equations 3 and 4 of the main paper. Note that *Percent* here is noted for that particular camera.

Table 6: Distribution of person crops for each camera at each time segment.

Camera 1	t	**Number**	**Percent**	**Camera 2**	t	**Number**	**Percent**
Total		29121		Total		11360	
τ=10	0	2681	0.092	τ=10	0	1227	0.108
	1	12511	0.430		1	1269	0.112
	2	9556	0.328		2	2709	0.238
	3	1714	0.059		3	2151	0.189
	4	1355	0.047		4	1897	0.167
	5	1304	0.045		5	2107	0.185
τ=15	0	3235	0.111	τ=15	0	1618	0.142
	1	21513	0.739		1	3587	0.316
	2	2388	0.082		2	3019	0.266
	3	1985	0.068		3	3136	0.276
τ=20	0	15192	0.522	τ=20	0	2496	0.220
	1	11271	0.387		1	4860	0.428

Camera 1	t	**Number**	**Percent**	**Camera 2**	t	**Number**	**Percent**
	2	2659	0.091		2	4004	0.352
τ=30	0	24749	0.850	τ=30	0	5205	0.458
	1	4373	0.150		1	6155	0.542
Camera 3	t	**Number**	**Percent**	**Camera 4**	t	**Number**	**Percent**
Total		3250		Total		6402	
τ=10	0	212	0.065	τ=10	0	631	0.099
	1	813	0.250		1	1873	0.293
	2	915	0.282		2	1406	0.220
	3	536	0.165		3	1179	0.184
	4	501	0.154		4	914	0.143
	5	273	0.084		5	399	0.062
τ=15	0	516	0.159	τ=15	0	1467	0.229
	1	1424	0.438		1	2443	0.382
	2	762	0.234		2	1606	0.251
	3	548	0.169		3	886	0.138
τ=20	0	1025	0.315	τ=20	0	2504	0.391
	1	1451	0.446		1	2585	0.404
	2	774	0.238		2	1313	0.205
τ=30	0	1940	0.597	τ=30	0	3910	0.611
	1	1310	0.403		1	2492	0.389
Camera 5	t	**Number**	**Percent**	**Camera 6**	t	**Number**	**Percent**
Total		6802		Total		12951	
τ=10	0	1771	0.260	τ=10	0	1424	0.110
	1	705	0.104		1	2676	0.207
	2	1453	0.214		2	2961	0.229
	3	926	0.136		3	1918	0.148

Camera 1	t	**Number**	**Percent**	**Camera 2**	t	**Number**	**Percent**
	4	967	0.142		4	2773	0.214
	5	980	0.144		5	1199	0.093
τ=15	0	1875	0.276	τ=15	0	2709	0.209
	1	2054	0.302		1	4352	0.336
	2	1324	0.195		2	3302	0.255
	3	1549	0.228		3	2588	0.200
τ=20	0	2476	0.364	τ=20	0	4100	0.317
	1	2379	0.350		1	4879	0.377
	2	1947	0.286		2	3972	0.307
τ=30	0	3929	0.578	τ=30	0	7061	0.545
	1	2873	0.422		1	5890	0.455
Camera 7	t	**Number**	**Percent**	**Camera 8**	t	**Number**	**Percent**
Total		4967		Total		15685	
τ=10	0	805	0.162	τ=10	0	3720	0.237
	1	1144	0.230		1	4217	0.269
	2	1247	0.251		2	2947	0.188
	3	728	0.147		3	2678	0.171
	4	603	0.121		4	2064	0.132
	5	440	0.089		5	57	0.004
τ=15	0	1326	0.267	τ=15	0	5859	0.374
	1	1870	0.376		1	5025	0.320
	2	1045	0.210		2	3741	0.239
	3	726	0.146		3	1058	0.067
τ=20	0	1949	0.392	τ=20	0	7937	0.506
	1	1975	0.398		1	5625	0.359

Camera 1	t	**Number**	**Percent**	**Camera 2**	t	**Number**	**Percent**
	2	1043	0.210		2	2121	0.135
τ=30	0	3196	0.643	τ=30	0	10884	0.694
	1	1771	0.357		1	4799	0.306