# REAL-TIME VISION METHODS FOR DEPLOYABLE PEDESTRIAN DETECTION, RE-IDENTIFICATION AND PATH PREDICTION SYSTEMS

by

Matías Mendieta

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Electrical Engineering

Charlotte

2020

Approved by:

Dr. Hamed Tabkhi

Dr. Chen Chen

Dr. Andrew Willis

# ABSTRACT

MATÍAS MENDIETA. Real-time Vision Methods for Deployable Pedestrian Detection, Re-identification and Path Prediction Systems. (Under the direction of DR. HAMED TABKHI)

This article presents two methods, REVAMPT and CARP$^{e}$ Posterum. REVAMPT, or Real-time Edge Video Analytics for Multi-person Privacy-aware Tracking, is an integrated system for privacy-built-in pedestrian re-identification. REVAMPT presents novel algorithmic and system constructs to push deep learning capabilities for pedestrian re-identification at the edge (i.e. the video camera). On the algorithm side, REVAMPT proposes a unified computer vision pipeline for detection and re-identification on a low-power computing device without the need for storing the streaming data. At the same time, it avoids facial recognition, re-identifying pedestrians based on their encoded key features at runtime. For the results and evaluation, this article also proposes a new metric, Accuracy·Efficiency (Æ), for holistic evaluation of deployable systems based on accuracy, performance, and power efficiency. REVAMPT outperforms current state-of-the-art by as much as ten-fold Æ improvement.

A symbiotic task for pedestrian re-identification is path prediction, and therefore we also propose CARP$^{e}$ Posterum, a Convolutional Approach for Real-time Pedestrian Path Prediction. Having insight into the movement of pedestrians is not only important for pedestrian re-identification, but also for ensuring safe operation in a variety of applications including autonomous vehicles and social robotics. Current works in this area utilize complex generative or recurrent methods to capture many possible futures. However, despite the inherent real-time nature of predicting future paths, little work has been done to explore accurate and computationally efficient approaches for this task. To this end, CARP$^{e}$ utilizes a variation of Graph Isomorphism Networks in combination with an agile convolutional neural network design to form a

fast and accurate path prediction approach. Notable results in both inference speed and prediction accuracy are achieved, improving FPS by at least 8x in comparison to current state-of-the-art methods while delivering competitive accuracy on well-known path prediction datasets.

# DEDICATION

To my Heavenly Father and to my parents, John and Tina Marie.

# ACKNOWLEDGEMENTS

Thank you to the National Science Foundation (NSF) for supporting this research under Awards No. 1737586 and 1831795.

Thank you to my advisor, Dr. Hamed Tabkhi, for allowing me the opportunity to conduct this research under his supervision.

Thank you to my committee members, Dr. Chen Chen and Dr. Andrew Willis, for their instruction and guidance throughout my studies.

Thank you to my peers in the TeCSAR laboratory for the fruitful discussions and enjoyable collaboration.

Finally, thank you to my parents and siblings, John, Tina Marie, Esteban and Sophia, for their continual support and encouragement.

TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

Æ Accuracy·Efficiency

CNN Convolutional Neural Network

FPS Frames Per Second

GAN Generative Adversarial Network

GPU Graphics Processing Unit

IDF1 Identification F1

IDP Identification Precision

IDR Identification Recall

IoU Intersection over Union

LSTM Long Short-Term Memory

ONNX Open Neural Network Exchange

PII Personally Identifiable Information

ReID Re-identification

RNN Recurrent Neural Network

RoI Region of Interest

# CHAPTER 1: INTRODUCTION

Recent advances in machine learning, particularly deep learning, have driven the development of more advanced computer vision technologies. This includes everything from simple license plate scanners that search for stolen vehicles, to terrain segmentation for satellite imagery or brain tumor detection in medical CT scans. Until recently, the common approach for accomplishing advanced vision tasks was to perform the operation remotely on large cloud servers and send back the results because of the significant algorithmic computation required.

However, many emerging and societally impactful technologies like self-driving cars, social robotics, and intelligent surveillance systems do not have the luxury to simply rely on the traditional cloud computing paradigm. These technologies require intensive artificial intelligence capabilities like object detection, pose estimation, and path prediction, but in a deployable package with minimal power consumption. On top of these tight device constraints, real-time performance is essential for fast decision-making and safe operation of such systems in their deployed environments. Offloading to the cloud is often not an option for such latency and scalability concerns, and therefore the demand for real-time, edge-capable computer vision models is growing quickly.

Particularly with intelligent surveillance technologies, it is imperative to also consider the technological impact from a social privacy 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, mul-

tiple cities in the US have imposed bans on all deployment of facial recognition and tracking technologies [5]. European Union regulators are also considering new restrictions on AI-driven surveillance [6]. To address both technical and ethical concerns, novel approaches are required in deployable systems design across the entire computing stack from algorithm development to computation mapping, communication, and system-level synchronization.

## 1.1 Problem Statement

In intelligent surveillance, a fundamental task is that of pedestrian re-identification and tracking. This entails the ability to detect pedestrians in a scene, and distinguish them across frames. Such technology serves as a basis for various scene and action understanding applications. However, designing a functional system that meets both ethical and technical constraints for deployment is challenging. A majority of the work in this field heavily relies on the use of face detection [7, 8, 9], or requires offline operation on a bank of stored video [10, 11, 12]. However, face detection and offline operation undermine the privacy of pedestrians. To avoid the need for gathering personally identifiable information (PII) and massive video storage, we need an end-to-end solution that processes frames in real-time without static identity databases.

A valuable addition to a pedestrian re-identification system is the ability to perform path prediction for the pedestrians in the scene. This assists in re-identification after long occlusions and understanding person-scene interactions. However, this task has many challenging properties: 1) When choosing their future steps, pedestrians typically have an intrinsic goal from which they plan accordingly. Capturing this intent from outside observation requires a fundamental understanding of human movement. 2) Person-to-person social interactions often influence the future path of a pedestrian, for example, when avoiding collisions or traveling in groups. Therefore, modeling this social effect is imperative for robust path prediction. 3) Predicting the future is inher-

ently time-sensitive, as the information is only useful for decision making if obtained quickly. Therefore, meeting real-time processing constraints is essential for the safety and usefulness of a path prediction algorithm.

Pioneering works have attempted to incorporate social effects in pedestrian trajectory prediction [13, 14, 15]. These approaches relied mainly on hand-crafted rules, and were often limited in scale and function. More recent works have focused on developing data-driven approaches to tackle the path prediction problem. Social LSTM [16] formed a pooling mechanism with recurrent neural networks (RNNs) to provide social context to the prediction. Since then, many approaches have added the use of Generative Adversarial Networks (GANs) [17] within such frameworks, aiming to model the distribution of possible future trajectories [18, 19, 20]. Most recently, the work of Kosaraju et al. [3] utilized a graph neural network to model the social situation in addition to a RNN-based GAN architecture. However, the existing approaches often have two major shortcomings. First, they rely on very complex models with many parameters, which makes the real-time execution on embedded devices nearly impossible. Second and more importantly, they use multiple runs over video frames that inherently violate the real-time nature of path prediction and limit their applicability to real-world problems.

## 1.2 Contributions

This work introduces two components: 1) REVAMPT, a system performing Real-time Edge Video Analytics for Multi-person Privacy-aware Tracking, and 2) CARP$^{e}$ Posterum, A Convolutional Approach for Real-time Pedestrian Path Prediction.

REVAMPT is a low-power system that is able to detect and re-identify pedestrians within a scene without the use of personally identifiable information. We deploy REVAMPT on cameras equipped with the NVIDIA AGX Xavier [21] 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 the system.

To address the challenges of real-time path prediction, this paper proposes CARP$^e$ Posterum. CARP$^e$ is a data-driven approach which effectively captures both intrinsic and social nonlinearities of human trajectories, within real-time constraints. Figure 1.1 shows the high-level mechanism of CARP$^e$. Our method mainly consists of two networks, a graph neural network and a convolutional neural network, knitted together in an end-to-end fashion with efficiency in mind. CARP$^e$ harnesses the strong and proven discrimitaive power of recently proposed Graph Isomorphism Networks [22] to gather social context, and an intentionally designed CNN architecture for effective path prediction.

The combined contributions of this work are as follows:

- A deployed agile vision system for privacy-aware pedestrian re-identification.
- A novel path prediction method, which captures both nonlinear intrinsic and social effects in real-time.

Overall, REVAMPT achieves a pedestrian re-identification accuracy of 77.39% (only 9.44% below the current state-of-the-art [10]) on the DukeMTMC dataset [23], while achieving more than two times the real-time FPS and consuming 1/6th of the power compared to [10]. A balance was struck between algorithmic Accuracy and system Efficiency, measured by Accuracy·Efficiency (Æ). Our system has high scalability potential in a deployed environment while never sacrificing personal privacy. In pedestrian path prediction, notable results in both inference speed and prediction

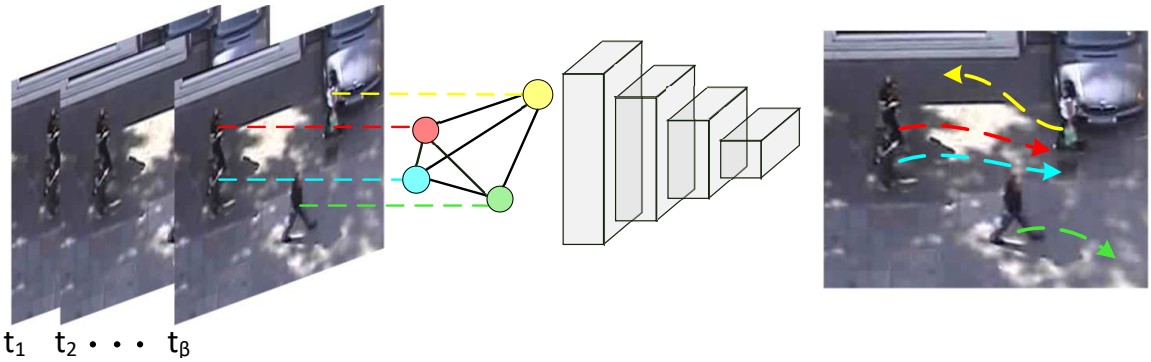



Figure 1.1: A high-level illustration of our proposed method, CARP$^{e}$. Past pedestrian positions from $t_1$ to the current time step $t_\beta$ are fed into the model. This information is propagated through a graph and convolutional neural network in an end-to-end fashion, producing future predicted trajectories for the next $T$ time steps. In the depicted output, we show potential examples of intrinsic nonlinearities (**green**), as well as social effects resulting from collision avoidance (**yellow**) and traveling groups (**cyan** and **red**).

accuracy are achieved, improving FPS by at least 8x in comparison to current state-of-the-art methods while still producing competitive accuracy results on well-known path prediction datasets.

# CHAPTER 2: RELATED WORK & BACKGROUND

## 2.1 Pedestrian Detection and Re-Identification

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 [24], single shot detectors like SSD [25] and YOLO [26], as well as pose-estimation models like DeeperCut [27] and OpenPose [28]. 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 detected pedestrians throughout the video frames. Therefore, on the re-identification side, recent methods leverage CNNs to extract unique features among persons [29, 30, 31, 32, 33, 34, 35, 36]. The work in [37] 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. [38] uses bidirectional Long Short-Term Memory (LSTM) cells in a Recurrent Neural Network (RNN) to learn the spatial and temporal behavior of people throughout the video. Triplet loss [39, 40, 41] is another promising technique to train the network with the goal of clustering classes in a way that IDs of the same category have minimum distance among each other, while examples from different categories are separated by a large margin.

ReID approaches [11] and [12] are monocular video pedestrian tracking systems that use offline lifted multicut and subgraph decomposition methods. [42] proposed a method for single camera multi-target tracking in terms of the Binary Integer Program, and can incur online, real-time results. However, the system utilizes simple

HSV histograms for appearance features, and is not robust to a variety of environments. [43] presents a RNN-based tracking algorithm to encode long-term cues and generate similarity scores for ReID. [44] and [45] apply single object trackers to the multi-person tracking problem, with [45] introducing a dynamic CNN with shared and target specific layers for appearance feature extraction. The current state-of-the-art in complete person re-identification systems is DeepCC [10]. This approach uses OpenPose for detections, a deep learning triplet loss ReID network for visual data association, and trajectory tracklets.

Overall, the current state of pedestrian re-identification algorithms struggle with lack of privacy preservation and/or limited focus. First, the idea of privacy preservation and online functionality are lost with many current approaches. 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 [7, 8, 9, 46, 47], 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. Second, even for current online approaches, very little investigation is done with respect to feasibility of deployment. Many of these approaches only focus on part of the complete vision pipeline needed, and provide low inference speeds on large, high power GPUs. In contrast to existing work, this article proposes a shift to deployable, non-personal and data private pedestrian re-identification. REVAMPT accomplishes these tasks online on low-power edge devices.

## 2.2 Pedestrian Path Prediction

Early works in pedestrian path prediction focused on the development of Gaussian processes and energy models to understand human behavior and movement [13, 14, 15, 48, 49]. However, these methods often require many predefined rules for pedestrian interactions, and are limited to predicting a short time into the future.

Recurrent architectures are a common option for recent works in path prediction, given their theoretical ability to capture an infinite history of inputs [50, 51]. [16, 52] utilize LSTM-based RNNs at the forefront of their approaches to understand and predict human trajectories. Liang et al. [4] propose an LSTM-based joint trajectory and activity prediction system that incorporates scene segmentation maps, pedestrian visual features, and person keypoints to better inform both tasks. However, in practice, the infinite history capabilities of RNNs are largely absent and the forced sequential operation within the RNN limits its parallelization potential in modern hardware [53]. This challenges the effectiveness of RNNs versus purely convolutional networks in sequence modeling tasks, and makes such RNN approaches not ideal for deployable real-time inference.

Many recent works have focused on incorporating generative models for the path prediction problem. In [18], Gupta et al. built on the work of [16] by integrating a social pooling mechanism into an LSTM-based GAN network. [20] takes this work further, combining scene-level visual features with attention modules for physical and social relations. [19] aims specifically to capture the latent decision, or intrinsic elements, of pedestrian movements in a generative fashion with statistical sub-networks. Most recently, Kosaraju et al. form an RNN-based generative approach that includes the use of Bicycle-GANs [54] and Graph Attention Networks (GAT) [55]. These generative approaches typically rely on the ability to repeatedly inference the model and generate many samples per pedestrian, which neglects the inherent real-time

constrains of practical path prediction.

In contrast to these works, our method relies neither on recurrent nor generative architectures. We use a convolutional approach for hardware-friendliness and deterministic real-time inference. For social context, we formulate a Graph Neural Network (GNN) based on recent theoretical work in GNNs [22] to maximize its disciminative power. In this way, we tackle both intrinsic and social effects of pedestrian path prediction while achieving real-time inference capabilities on low-power devices.

## 2.3 Graph Neural Networks

Graph Neural Networks (GNNs) have seen great progress in recent years. Naturally, GNNs aim to take advantage of the powerful learning ability of neural networks for non-euclidean data. Data representations for molecular models or social media interactions are naturally inclined to graph representations, and therefore require a unique neural network definition. Typically, each node in the graph holds a feature, which is operated on across the graph structure. Such operations are utilized for a variety of objectives in the graph such as forming new node features, performing node classification, or completing graph-level classification [56].

GNNs are implemented with two major approaches, spectral and spatial [56]. Spectral methods, such as [57, 58], utilize mathematical formulations rooted in graph signal processing to perform a graph Fourier Transform and subsequent convolution. However, these methods assume the graph to be undirected and static. This limitation hinders the viability of spectral methods in many domains where dynamic graph structures are preferred or required.

Spatial methods. e.g. [22, 59, 55], do not require such assumptions, as they operate on each node relative to its neighbors, allowing for dynamic graph structures. Typically, for a given node, the features of its neighbors are aggregated, and then combined with the current node feature. The aggregate and combine operations dif-

fer in the various spatial GNN formulations. This process can be repeated to form more abstract node representations, as well as increase the reach of a given node in a sparsely-connected graph. GraphSAGE [59] proposed an inductive learning framework with max-pooling aggregation across node features. Graph Attention Networks (GAT) [55] perform the aggregation and combine steps together using a weighted sum approach with attention. However, these GNN formulations have been largely based on empirical evidence, without a supporting theoretical foundation for optimal behavior. In their recent work [22], Xu et al. investigate the theoretical properties of GNNs to determine optimal discriminative power based on the Weisfeiler-Lehman (WL) graph isomorphism test [60]. They present a new GNN formulation, termed Graph Isomorphism Networks (GIN), that are provably among the most expressive GNN variants. In this work, we use the GIN operators as a basis for our graph network and reformulate for use in path prediction.

## 2.4 Preliminary: Graph Isomorphism Networks

For graph convolutional operations, the typical aim is to analyze a graph structure and the features of its nodes, producing meaningful representations in an embedding space across different graphs. Ideally, for a GNN to be maximally discriminative, two separate nodes should only map to the same location in the embedding space if all aspects of node and neighborhood are identical. These aspects include both the node features and neighborhood structure. Therefore, we expect to gather a unique feature embedding in all other cases. Testing for discriminative power in a GNN can be analogously drawn to the task of graph isomorphism tests, or distinguishing whether two graphs are topologically identical.

A well known test for determining such properties is the WL isomorphism test, as it has been found to effectively classify a broad class of graphs [60]. Therefore, Xu et al. [22] use the WL isomorphism test as a theoretical guide in determining

the distriminative power of GNNs. In their work, the authors find that a GNN is as powerful as the WL isomorphism test if its aggregation (and graph level readout operation, as used in node classification) is injective. Therefore, the authors define a joint aggregation/combination operator as shown in Equation 2.1.

$$h_i' = \phi\left((1+\epsilon)\cdot h_i + \sum_{j\in N(i)} h_j\right) \tag{2.1}$$

Here, node features $h_j$ are from nodes in the neighborhood $N(i)$. $h_i$ is the feature for node $i$, $\epsilon$ is a trainable parameter, $\phi$ indicates an Multilayer Perception (MLP), and $h_i'$ is the updated node feature. In this work, we employ the findings of [22] and this joint aggregation/combination operator to formulate a graph for the pedestrian path prediction problem, as will be detailed in Section 4.1.2.

# CHAPTER 3: REVAMPT

## 3.1 Privacy Requirements and Threat Modeling

This section describes the privacy threat models which REVAMPT 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 the system 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 and its generated meta-data 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 REVAMPT:

(1) REVAMPT 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) REVAMPT's re-identification algorithm works on an encoded feature representation of an individual (without using facial recognition algorithms). These features represent the visual and structural attributes of an individual, generating an abstract and temporary encoding that has essentially no meaning outside the constraints of our system. By utilizing these feature representations, REVAMPT 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 [7, 8, 9, 46, 47].

We design REVAMPT with respect to these defined privacy protection policies. Section 3.2 and Section 3.3 present algorithmic constructs and deployable system design of REVAMPT.

## 3.2 REVAMPT: Algorithmic Constructs

This section presents algorithmic constructs to enable real-time pedestrian re-identification while satisfying our privacy model detailed in Section 3.1.

Figure 3.1 outlines the full algorithmic pipeline. The pipeline consists of three primary phases: (1) Detection, (2) Feature Extraction, and (3) ID Ranking and Assignment. For the detection part, we chose OpenPose [28] 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 locations. In the feature extraction portion, discriminative features are generated for detection representation. These features are utilized, along with spatial positioning, to match detections in the Local Database. The rest of this section discusses the technical details of our proposed pipeline integration.

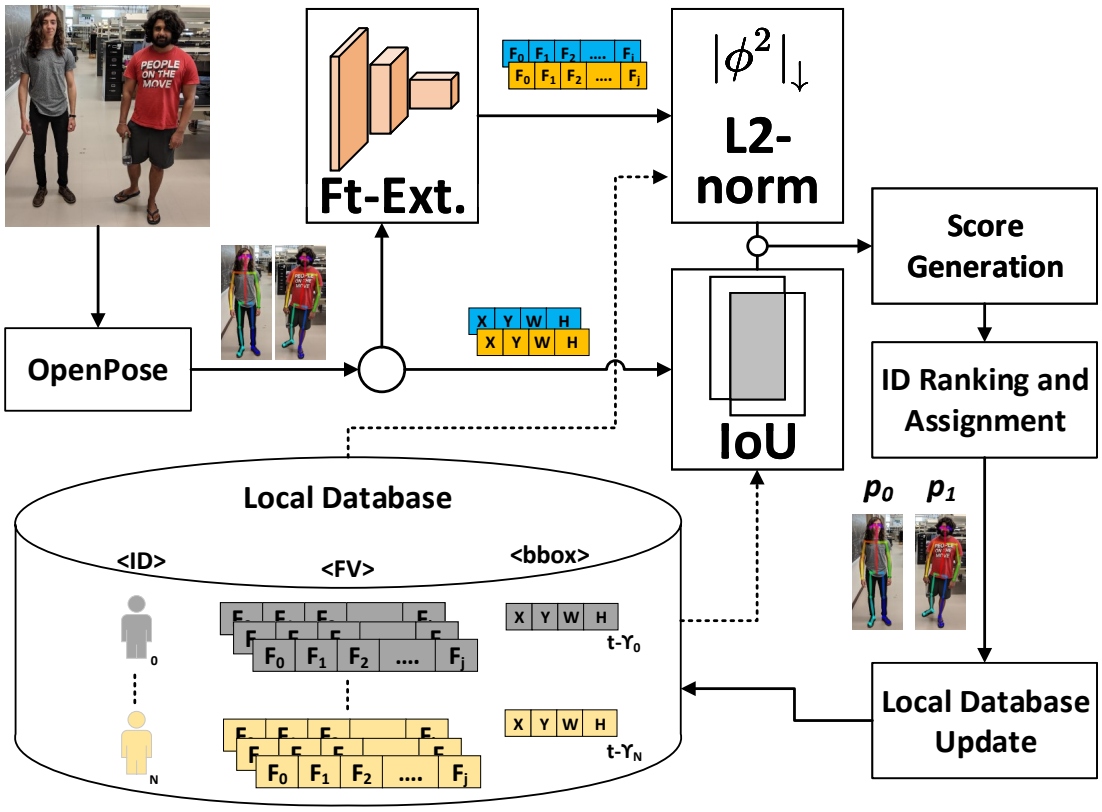



Figure 3.1: Algorithm Pipeline on the Edge

### 3.2.1 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 3.1. For this task, we needed a deep convolution network capable of 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 [61] and MobileNet-V2 [62] 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 REVAMPT, we employ the feature extraction network and training method proposed in [63]. The model is based on MobileNet-V2, with the fully connected layer changed to a 2D average pooling with a kernel size of (8, 4) in order to make the output of the network a 1x1280 vector as the embedded appearance features. As in [10, 63], we also use the triplet loss function [41] 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:

$$Loss = \sum_{i=1}^{n} \left[ \alpha + \|f_i^a - f_i^p\|^2 - \|f_i^a - f_i^n\|^2 \right]_+, \tag{3.1}$$

where $\alpha$ 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 a hypersphere of radius $\alpha$. 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 follow [63] and assign error-friendly operations, such as convolution and General Matrix Multiply (GeMM) operations, to half precision (16-bit) floating point accuracy and apply mixed precision training [64] to minimize the potential error incurred by half precision operations.

### 3.2.2 Integration of Vision Pipeline

In order for the re-identification task to be accomplished on the edge, all stages must be integrated seamlessly together. Referring back to Figure 3.1, 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 3.1). The output of this network provides the encoded 1x1280 feature vector for each detection. REVAMPT performs the L2-norm operation between the features extracted from the current frame to those of previously seen entries in the Local Database. For each detection, the bounding box information is also propagated for IoU comparison with the stored bounding box (from t-$\Upsilon$, where $\Upsilon$ is the time since that entry was last seen) of $N$ Local Database entries. The intuition for maintaining these bounding boxes is to provide spatial locality, such that detections are matched with entries that not only match in the embedded space (feature vector), but also in location. 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 generate a score for each detection to database entry using a weighted sum of the IoU and L2-norm results, such that a lower score indicates a closer match.

Final matching decisions are made using the generated scores and a re-ranking approach to ensure optimal ID assignment. 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 Local Database is updated with assigned labels, keypoints, feature vectors, and bounding box from the processed frame. For each entry in the Local Database, a gallery of feature vectors can be stored to provide

a cluster representation for an entry. This is done to allow for more robust comparison when performing the L2-norm functions. Although not depicted in Figure 3.1, REVAMPT maintains the keypoints and confidences for assessing the quality of a detection, as will be described in Section 3.3.1.

## 3.3 REVAMPT: 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 system. How data flows between algorithms, when and how to utilize said algorithms, 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 unified system. With REVAMPT's focus on privacy, it was important that the system is designed around never storing any personally identifiable information. Algorithmic selections and the design of the system's processing flow hinged around that constraint.

### 3.3.1 System Hyperparameters and Processing Flow

Figure 3.2 shows the logical processing flow of one frame of data on the edge, beginning 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 percentage of keypoints equal to $\theta_{key}$, and each of those keypoints a confidence value of at least $\theta_{conf}$. Algorithm 1 demonstrates this operation and Table 3.1 presents the system configurable hyperparameters.

For every valid detection, all possible matches for ReID are gathered from the Local Database, as discussed in Section 3.2.2. The match scores (weighted sum scores of

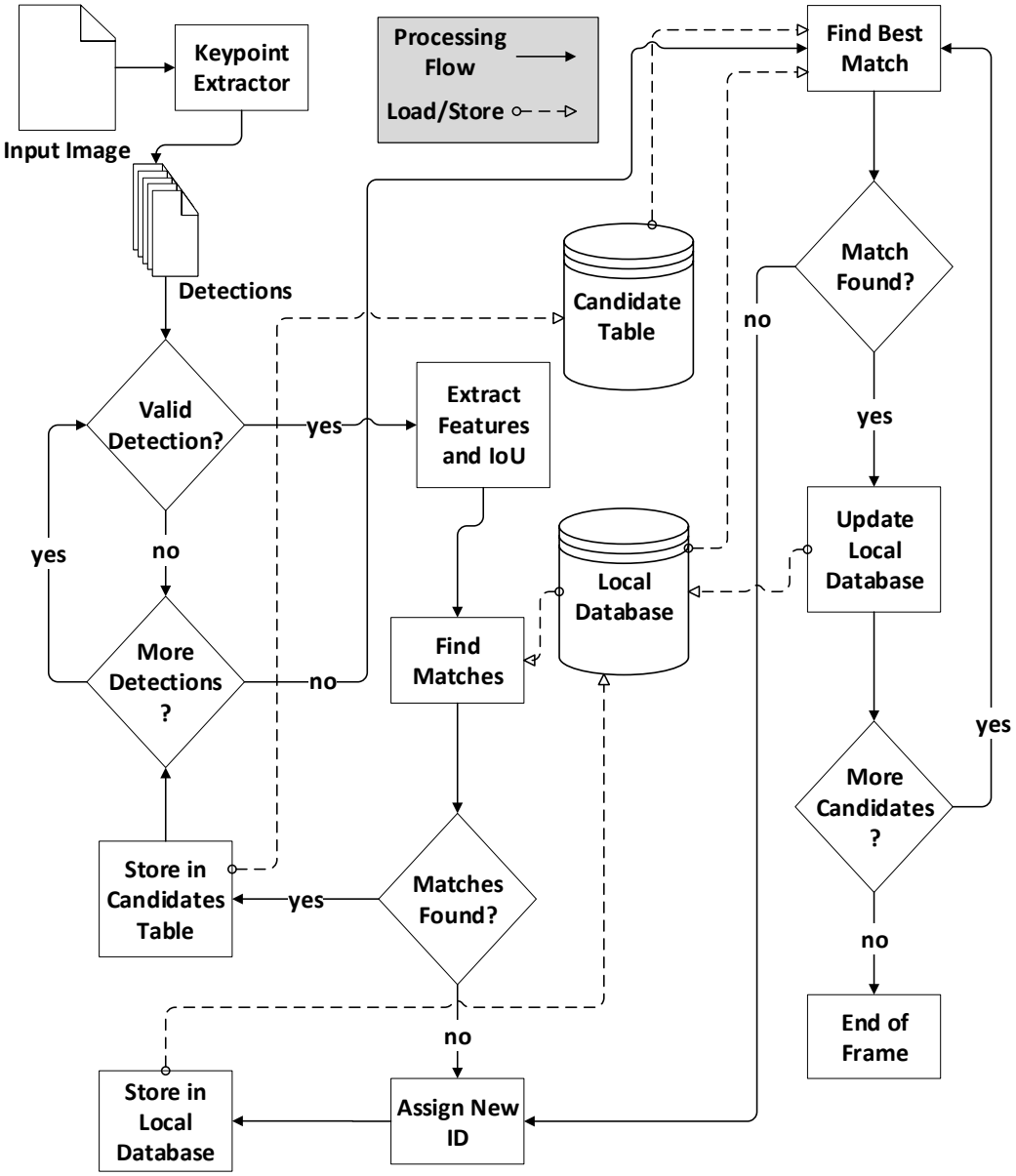



Figure 3.2: Processing Flow of the Edge

Table 3.1: System Parameters

| Parameter | Description |
|---|---|
| $\theta_{key}$ | Minimum percentage of keypoints for valid detection |
| $\theta_{conf}$ | Confidence threshold for valid keypoint |
| $\theta_{update}$ | Valid keypoint threshold for feature vector update |
| $\theta_{score}$ | Match score threshold |
| $\bar{D}$ | Detection from keypoint extractor |
| $\bar{V}$ | Valid Detections |
| $\widehat{DB}$ | Local database |
| $v$ | Valid detection in $\bar{V}$ |
| $e$ | Entry Local Database |
| $\tilde{C}$ | Candidate Table |
| $\zeta$ | Global variable to keep track of new ID |

IoU and L2-norm) for ($v$, $e$) pairs are gathered and stored in the Candidate Table $\tilde{M}$. 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 (bounding box coordinates). However, it only updates the feature vector if the new feature vector is better representative of the object (meaning obtained with more keypoints than previously had) or has at least $\theta_{update}$ valid keypoints.

### 3.3.2 Database

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 have passed since that pedestrian was detected. When an object has not been seen by the system after some time (as indicated by life), the object's ID is removed 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.

**Algorithm 1** Validating Detections

**Input:** $\bar{D}, \theta_{conf}, \theta_{key}$
**Output:** $\bar{V}$
$\bar{V} \leftarrow \emptyset$
**for** $d$ in $\bar{D}$ **do**
    $numKeyPoints$ = findValidKeyPoints($d$, $\theta_{conf}$)
    **if** $numKeyPoints \geq \theta_{key}$ **then**
        $\bar{V} \leftarrow \bar{V} \cup \{d\}$
    **end if**
**end for**

**Algorithm 2** Finding Best Matches

**Input:** $\bar{V}, \widehat{DB}$
**Output:** $\widehat{newID}, \widehat{reID}$
$\widehat{newID} \leftarrow \emptyset, \widehat{reID} \leftarrow \emptyset$
$\tilde{C}$ = getCandidateTable($\bar{V}$, $\widehat{DB}$)
**while** $min(\tilde{C}) \leq \theta_{score}$ **do**
    $\widehat{reID} \leftarrow \widehat{reID} \cup \{v, e\}$
    removeValidDetectionAndEntry($v, e, \tilde{C}$)
**end while**
**while** thereIsCandidate($\tilde{C}$) **do**
    $\zeta = \zeta + 1$
    $\widehat{newID} \leftarrow \widehat{newID} \cup \{\zeta\}$
    removeValidDetectionAndEntry($v$,$\emptyset$, $\tilde{C}$)
**end while**

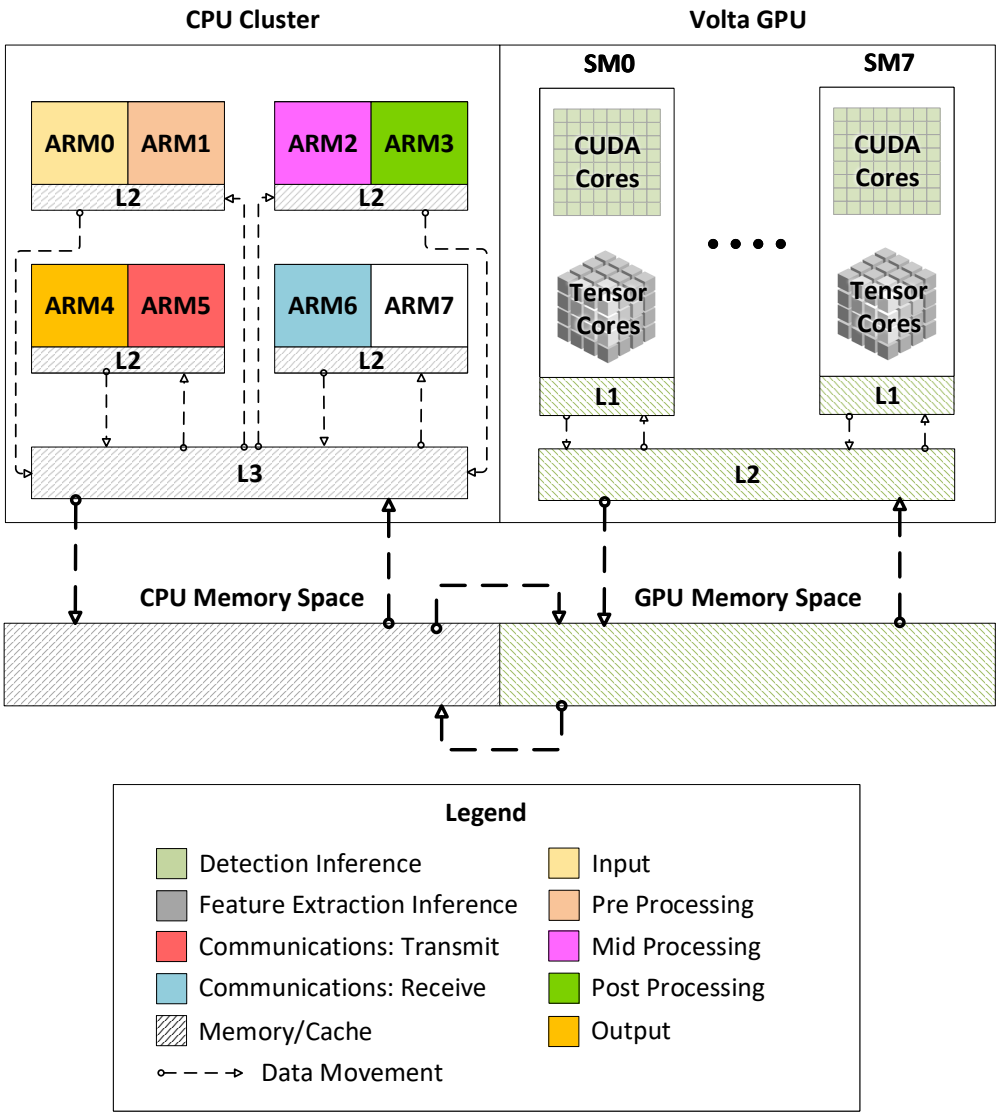



Figure 3.3: Mapping of Processes to Edge Resources

### 3.3.3 Computation and Optimization

To achieve real-time performance on the edge, we chose Nvidia AGX Xavier SoCs [21]. The Xavier is equipped with many advanced components that are leveraged for REVAMPT, 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 3.3 shows the how the different processes in REVAMPT are mapped to the

Xavier resources. REVAMPT is split into five stages with different tasks as follows - 1) Input: Image retrieval from the camera. 2) Pre-processing: Image resizing and other transformations. 3) Mid-processing: Detection network inference. 4) Post-processing: keypoint formation and ReID feature extraction. 5) Output: ID assignment and database updates.

Each stage of the detection framework is mapped to a separate ARM Core. Two cores are kept available for the addition of transmit and receive threads in an expansion to a distributed system setting as in [65]. 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 is converted from the ONNX (Open Neural Network Exchange) format 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 MobileNet-V2. All code on the edge was developed in C++ for computational efficiency, enhanced execution, and mapping control.

## 3.4 Experimental Results and Evaluation

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

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

### 3.4.1 Algorithm Evaluation

In order to validate the accuracy of the full algorithm pipeline, the edge algorithms were ported to Python 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 complete pedestrian re-identification systems, DeepCC, on the same trainval_mini validation set. For all experiments, we measure Identification Precision (IDP), Identification Recall (IDR), and Identification F1 (IDF1) with truth-to-result matching, as proposed in [66]. This metric maps one ground truth ID to one generated ID from the system, and measures how long the tracker correctly matches pedestrians. The one-to-one mapping of ground truth ID to generated ID is chosen using a bipartite matching method. Intuitively, IDP measures how often a generated ID matches to its mapped ground truth ID, and IDR measures how often a ground truth ID matches to its mapped generated ID. IDF1 is simply the harmonic mean of IDP and IDR. Detection misses are computed in accordance to the truth-to-result matching method, with the IoU threshold at 0.3. In accordance with Table 3.1, the values for system hyperparameters are as follows: $\theta_{key} = 0.4$, $\theta_{conf} = 0.35$, $\theta_{match} = 2.25$, $\theta_{update} = 0.8$. Note that $\theta_{key}$, $\theta_{conf}$, and $\theta_{update}$ are percentages expressed from zero to one.

#### 3.4.1.1 Accuracy

Figure 3.4 shows the IDF1 for each camera individually, as well as the average across all cameras (Avg) for both approaches. Overall, REVAMPT only drops 9.44% average IDF1 compared to DeepCC, with DeepCC at 86.83% and REVAMPT at 77.39%. We expect that REVAMPT will have a slightly lower accuracy because, as an online system, it performs ReID within a short temporal window in accordance

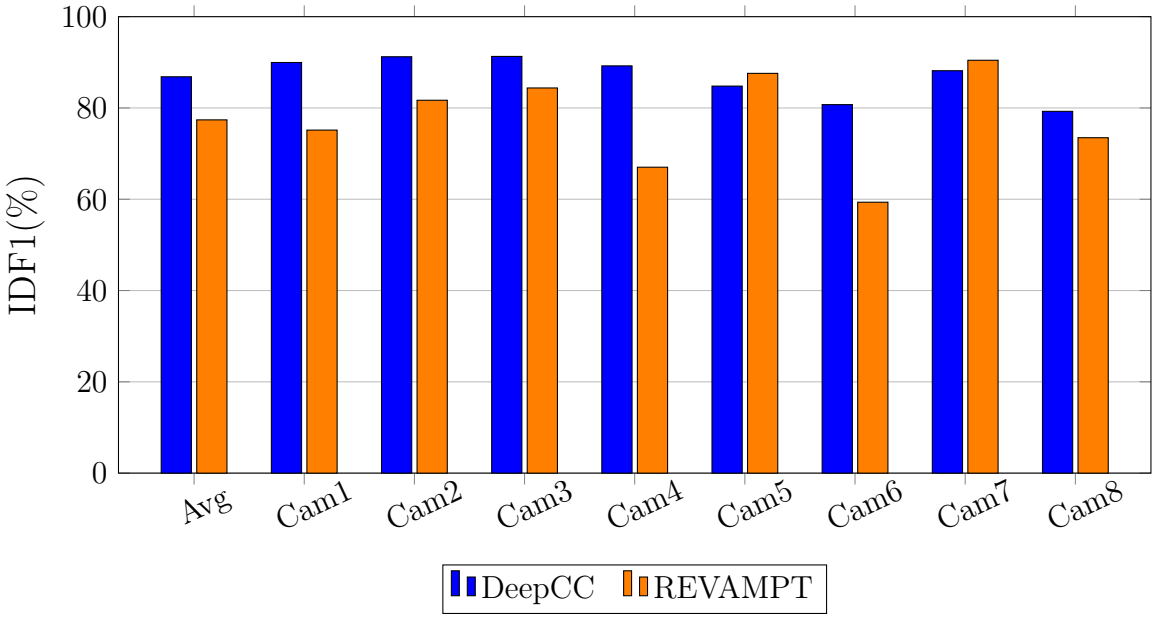



Figure 3.4: IDF1 Results for All Cameras

with the spontaneous nature of online operation. On the other hand, the offline DeepCC algorithm is able to look forward and backward in time across the entire video set, allowing for further optimizations in ID matching and assignment without the constraints of real-time operation. However, on further analysis of the results, we find that REVAMPT performs very comparatively to DeepCC with the exception of cameras 4 and 6. Quantitatively, REVAMPT is only 5.33% lower in average IDF1 compared to DeepCC in all other cameras.

To obtain further insight into the performance of REVAMPT, Figure 3.5 shows IDP versus IDR for REVAMPT and DeepCC. DeepCC maintains groupings around 85% for both IDP and IDR. REVAMPT maintains high IDP, around 90%; however, the IDR is less consistent across cameras. This means that the system generally minimizes false positives, but is less effective at minimizing false negatives. One major cause for this result is simply failing to detect the pedestrian in the frame. If the person is not detected, the ground truth instance will have no match, and therefore incur a false negative. The second reason for lower IDR performance is ID switching. When a pedestrian is occluded for a longer period of time, REVAMPT is unable to maintain

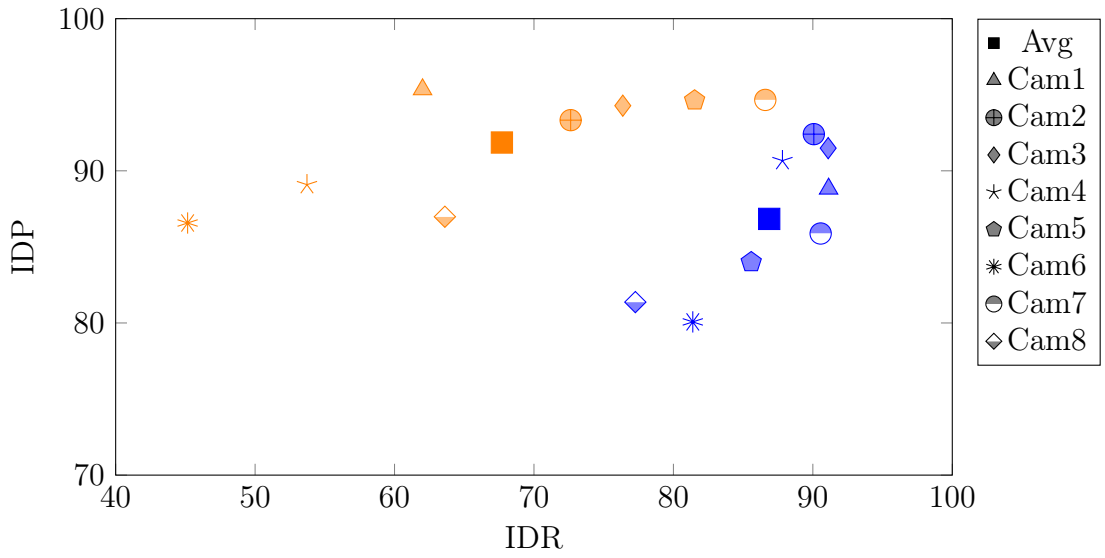



Figure 3.5: Precision (IDP) and Recall (IDR) for DeepCC (blue) and REVAMPT (orange)

spatial correlation with the pedestrian and its last known location. Therefore, once the pedestrian reappears, REVAMPT may give the pedestrian a new ID and incur false negatives for the ground truth instance relative to its original generated ID. To maintain spatial correlation under long occlusion, REVAMPT will require a method for predicting the future position of the pedestrian based on previous observations, and therefore be able make sense of the pedestrian's movement and location during occlusion. We will discuss a proposed path prediction method for eventual integration with REVAMPT in Chapter 4.

We note that cameras 4 and 6 are noticeably lower in IDR than most others. This is consistent with the results in Figure 3.4, and indicate that much of the accuracy drop is caused by low IDR in these cameras. Table 3.2 reports the number of person-to-person occlusions that occur for each camera within the trainval_mini set, as well as the average ground truth bounding box area. Cameras 4 and 6 have the lowest average bounding box areas, indicating that they require detection at longer ranges with pedestrians being smaller in the images. Because of this, a higher detection resolution is needed to avoid missed detections. We chose to run the detection network

at a relatively low resolution at an attempt to balance reasonable runtime speed and detection accuracy, but we study different resolutions further in Section 3.4.3. Additionally, the scene dynamics of camera 6 make occlusions particularly common, as shown by the large number of person-to-person occlusions in Table 3.2. Maintaining spatial correlation with a path prediction method will likely improve the IDR in camera 6 and others by improving ID retention from person-to-person occlusions, as well as potential person-object occlusions that occur in the cameras.

### 3.4.2 System Evaluation

For all system evaluation measurements, the complete end-to-end REVAMPT pipeline was inferenced. We also compare against DeepCC [10]. Because DeepCC is not integrated into a deployable system, we used OpenPose at the maximum accuracy configuration to simulate it for power and FPS measurements. This simplified representation provides a favorable scenario for DeepCC, as its ResNet-50 feature extraction and gallery matching functions would incur additional latencies and power requirements. Real-time candidate matching is built into REVAMPT, so it is included in all reported measurements.

#### 3.4.2.1 Power Consumption and Computation Efficiency

For measuring the power consumption on the Xavier, Tegrastats was used. For power consumption on the Titan V and V100 GPUs, we utilized the NVIDIA System Management Interface. AMD $\mu$Prof was used to measure CPU (Ryzen Threadripper 1920X) idle power for the Titan V and V100 systems. For both methods, 1080p 60 FPS video was read frame-by-frame from memory. A brief warm up of 20 frames

Table 3.2: Camera Scene Analysis

| **Measurement** | **Cam 1** | **Cam 2** | **Cam 3** | **Cam 4** | **Cam 5** | **Cam 6** | **Cam 7** | **Cam 8** |
|---|---|---|---|---|---|---|---|---|
| Occlusions (IoU $\geq$ 0.5) | 7845 | 1230 | 4029 | 1341 | 3104 | 39434 | 2677 | 4040 |
| Avg Area (pixels) | 21092 | 18450 | 26344 | 15150 | 23118 | 14037 | 22693 | 16667 |

was allowed before power was sampled over 100 frames. Measurements for FPS were taken directly from the OpenPose GUI.

Table 3.3: FPS and Power Consumption of Real-Time Inference

| **System** | REVAMPT | DeepCC | DeepCC | DeepCC |
|---|---|---|---|---|
| **Device** | Xavier | Titan V | 2xTitan V | V100 |
| **FPS** ↑ | 5.1 | 2.5 | 4.7 | 2.7 |
| **Power** ↓ | 33.31W | 200W | 365W | 224W |
| **Detailed Xavier Power Consumption** | | | | |
| CPU | GPU | DDR | SOC | Total |
| 2.67W | 19.49W | 2.60W | 8.55W | 33.31W |

Table 3.3 presents the power consumption and FPS for REVAMPT and DeepCC. Here we can see that for real-time applications, REVAMPT outperforms DeepCC on each GPU setup we tested. Even using two Titan V's, DeepCC is only able to reach 4.7 FPS. Meanwhile, REVAMPT can reach 5.1 FPS on an embedded GPU platform. In addition, REVAMPT consumes only 17% of the power of DeepCC on a single Titan V, or 9% for the dual Titan setup. Figure 3.6 presents computation efficiency, which is FPS processing per watt. DeepCC has an Efficiency between 0.012 and 0.013 FPS/Watt in all configurations. In comparison, REVAMPT has an Efficiency of 0.153 FPS/Watt. When looking at Efficiency, REVAMPT performs an order of magnitude better than DeepCC for real-time applications. This is because REVAMPT was built from the ground up to perform in real-time, both algorithmically and systemically.

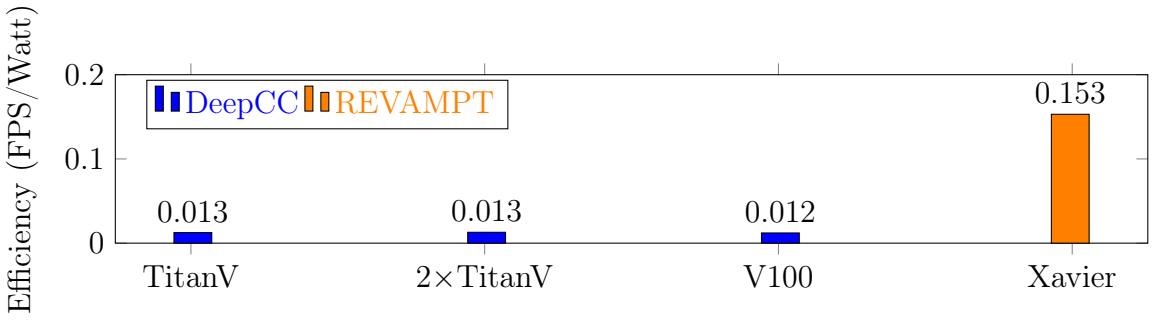



Figure 3.6: Efficiency of each test case.

#### 3.4.2.2 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.

In the case of REVAMPT, Accuracy would take the form of F1, while Efficiency is measured in FPS/Watt. Figure 3.7 shows the Æ Coverage for REVAMPT and DeepCC (Titan V). Here you can see that while DeepCC outperforms REVAMPT in terms of IDR and IDF1 Accuracy, REVAMPT has a significantly higher Æ Mark (11.85 vs 1.13) and more than 1.75x the total Æ Coverage (81.53% vs 46.57%). This is because our optimizations allow us to operate at twice the framerate, 17% of the power, and only 9.4% lower in F1 accuracy.

Table 3.4: Design Configuration Analysis

| **Measurement** | $\mathbf{P_2}$ | $\mathbf{P_3}$ | $\mathbf{C_D}$ | $\mathbf{R_{480}}$ | $\mathbf{R_{192}}$ | $\mathbf{R_{128}}$ |
|---|---|---|---|---|---|---|
| Power(W) ↓ | **7.91** | 12.08 | 33.30 | 34.42 | 29.12 | 25.71 |
| FPS ↑ | 1 | 3 | 5 | 3 | 10 | **15** |
| Accuracy ↑ | 76.95% | 77.00% | 77.39% | **82.33%** | 50.64% | 10.18% |

### 3.4.3 Design Flexibility and Adaptation

REVAMPT can further be configured to prioritize accuracy, FPS, or power consumption, as illustrated in Table 3.4. The default configuration of REVAMPT is

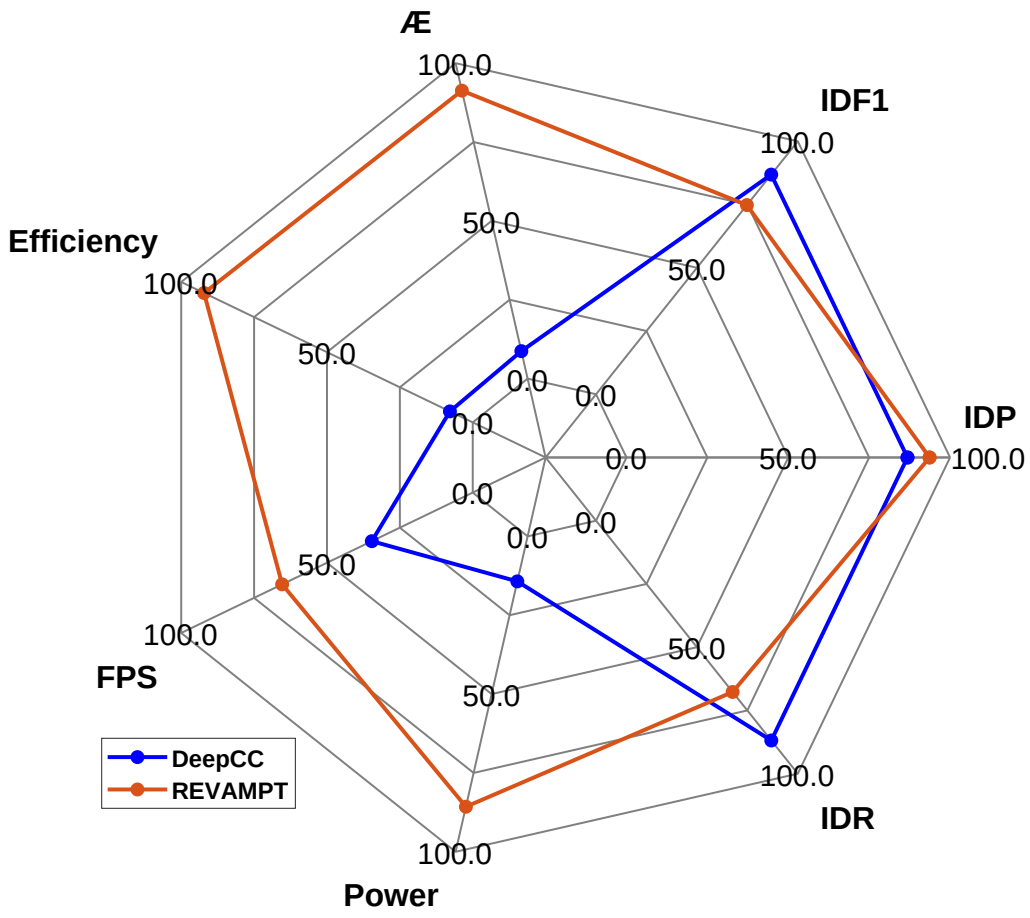



Figure 3.7: Æ Coverage

shown as $C_D$, with an input resolution of 576x320 for the keypoint extractor, and power consumption as seen in Table 3.3. We analyze five additional design configurations by modifying the input resolution, as well as the power restriction levels on the Xavier device. Configuration $R_{480}$, $R_{192}$, and $R_{128}$ are the proposed REVAMPT with modified input resolutions at 848x480, 336x192, and 224x128. Configuration $P_2$ and $P_3$ are the proposed REVAMPT configured with Power Mode 2 and 3 ($C_D$ uses Power Mode 0) provided by the Xavier device [67].

$R_{480}$ does provide a higher accuracy than other configurations, with a reduction in FPS and slight increase in power consumption. This configuration particularly improves the IDR of cameras 4, 6, and 8, which require the longest detection range

as found in Table 3.2. This configuration confirms the intuitions discussed in Section 3.4.1.1, bringing the accuracy of REVAMPT just 4.5% below the offline DeepCC algorithm.

The keypoint extraction resolution for $R_{128}$ properly extract keypoints for only persons close to the camera, resulting in low accuracy for DukeMTMC. $R_{192}$ offers an option for additional throughput at an accuracy loss. For its balance across all areas, configuration $C_D$ was chosen for the analyses of this report. With deployment in an edge environment, $C_D$ would likely require PoE Type 3. The $P_3$ and $P_2$ configurations show how REVAMPT could be adapted to the power levels of PoE Type 2 and Type 1 for deployment, with minimal loss in accuracy.

## 3.5 Discussion

Overall, REVAMPT is a deployable pedestrian detection and re-identification system capable of low-power, privacy-aware operation. However, there are areas for improvement, particularly on the accuracy front. As discussed in Section 3.4.1.1, REVAMPT would benefit most from reduction in false negatives during inference. Currently, REVAMPT can be configured to run at higher resolutions to mitigate false negatives incurred by missed detections. However, reducing false negatives incurred by ID switching still remains a challenge. Therefore, in the next chapter (Chapter 4), we will propose CARP$^e$ Posterum: A Convolutional Approach for Real-time Pedestrian Path Prediction. This method serves as a potential solution to the loss of spatial context after occlusions for REVAMPT, and stretches beyond this into applications like ensuring safe navigation of social robots and self-driving vehicles.

# CHAPTER 4: CARP$^{e}$ Posterum

## 4.1 CARP$^{e}$ Posterum: Method

The task of pedestrian path prediction is to predict the position of a pedestrian for $T$ time steps in the future given the past $\beta$ observed positions of the pedestrian. The goal is to accomplish this task as accurately as possible, while maintaining real-time inference capabilities. Two major factors that contribute to the future trajectory of a pedestrian are the intrinsic location goal of that pedestrian and the social context of the environment. We therefore aim to develop a model to capture these factors using the observed trajectories of all $P$ pedestrians in the scene at a given time step.

For the remainder of this chapter, we will distinguish these various elements as follows: Past pedestrian trajectories take the form of absolute coordinates $A$ and relative coordinates $R$, defined as $A_i = \{(x_i^t, y_i^t) \,|t = 1, \cdots, \beta\}$ and $R_i = \{(x_i^t - x_i^1, y_i^t - y_i^1) \,|t = 1, \cdots, \beta\}$, $\forall i \in \{1, 2, \cdots, P\}$. The future trajectories $Y$ of the pedestrians are predicted, and output as $Y_i = \{(x_i^t, y_i^t) \,|t = \beta + 1, \cdots, T\}$, $\forall i \in \{1, 2, \cdots, P\}$. These predictions are compared with the ground truth future trajectories $\hat{Y}$ for evaluation.

### 4.1.1 CARP$^{e}$ Posterum: Model Overview

Overall, CARP$^{e}$'s model consists of two main segments: 1) the Graph Module and 2) the Prediction Module. Figure 4.1 visualizes the full data mechanisms and module internals of CARP$^{e}$. The role of the graph module is to produce features for each observed pedestrian that incorporate a broader social context across the scene. These features, along with the original observed trajectories of the pedestrians

are both utilized by the prediction module to produce the future trajectories for each pedestrian. In this task, all trajectories for all pedestrians in the scene are inferred simultaneously, taking $P$ pedestrian features as input and outputting $P$ future trajectories in a single pass. We will explain each module and their functional details in the following sections.

### 4.1.2 CARP$^{e}$ Posterum: Graph Module

#### 4.1.2.1 Graph Formulation

A graph $G = (V, E)$ is constructed, where $V$ and $E$ are the sets of nodes and edges respectively. All $P$ pedestrians in the scene are represented as nodes in $V = \{V_0, V_1, \cdots, V_P\}$. Each pedestrian in the graph has a corresponding node feature $h_i$ held within the graph structure. The GNN performs joint aggregation and combination operations on $G$ to produce an output set of node features $h' = \{h'_0, h'_1, \cdots, h'_P\}$.

To form the input node feature, $A_i$ and $R_i$ are concatenated for a given pedestrian, and inferenced through a single fully-connected layer $\rho$. The absolute coordinates define the global position of the pedestrian, while the relative features act as a normalized form of input to better understand the pedestrian's past movement pattern.

#### 4.1.2.2 Graph Operation

Upon obtaining the node features, the graph is constructed as previously described. In order to maintain global context, the graph is fully connected. This allows the network to learn the relevant information needed, rather than predefining with handcrafted rules how relational connections should be made. To collect information across the graph, we define an aggregation and combination operation. The joint operation is represented in Equation 4.1, based on the GIN operation described in Section 2.4. In [22], Xu et al. only employ one MLP in their base operation. However, we reason that

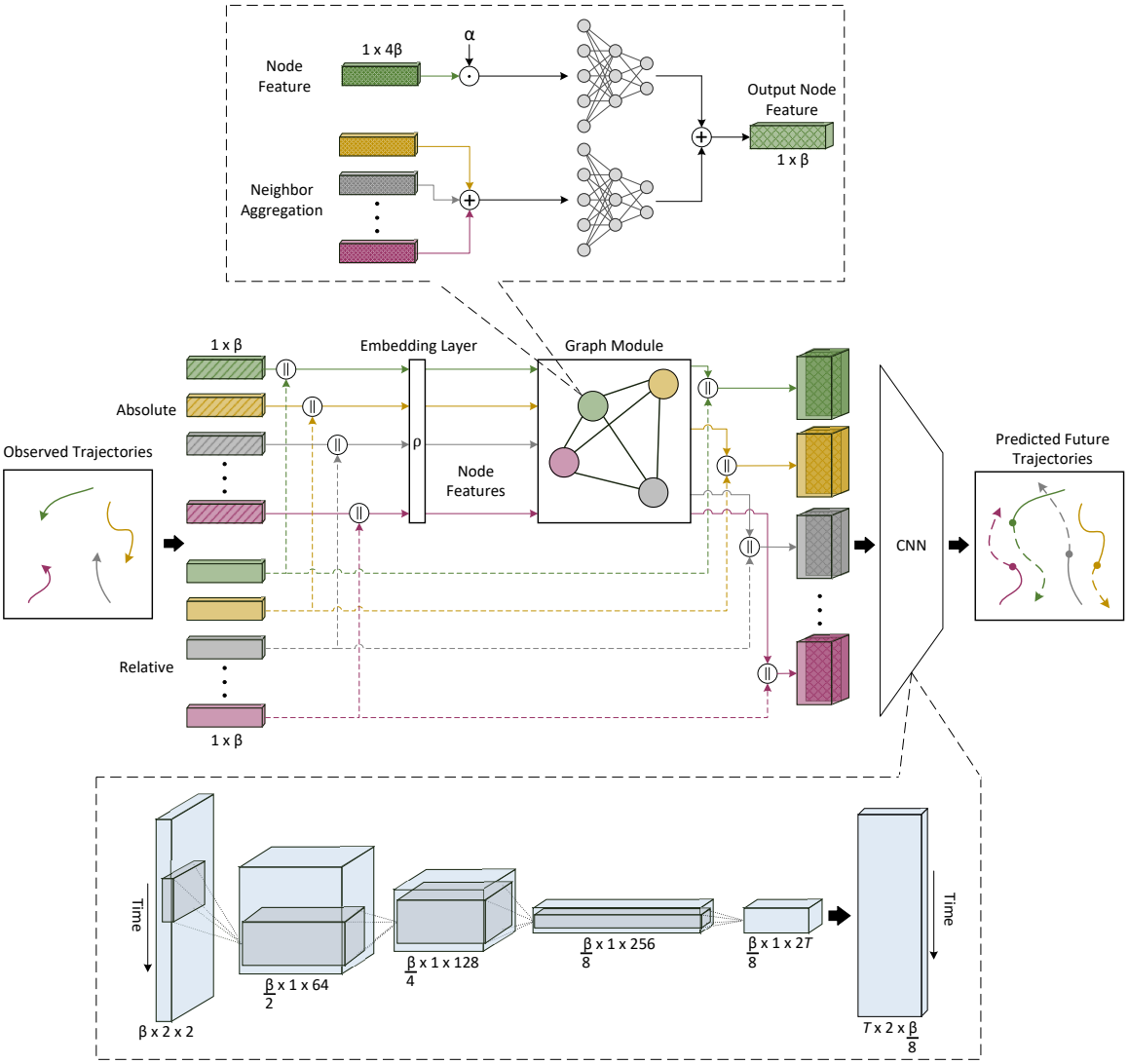



Figure 4.1: An overview of the full model architecture and data mechanisms for CARP$^{e}$. Listed dimensions are in the form *row* × *column* × *channel*. Observed positions for $\beta$ time steps are formed into trajectories for each of the $P$ pedestrians present in the scene. Embedded absolute and relative trajectories are inputted into the graph module to gather social context features. The Prediction Module CNN then utilizes these features along with the relative coordinates of each pedestrian to produce informed future trajectory predictions. In streaming applications, these trajectories are predicted every frame (equivalent to a time step) by appending the current frame information with a stored coordinate history to form the observation, and then inferencing the model.

abstracting the representation of the target node and the social context separately before combining will enable a deeper understanding and integration of neighboring nodes in context. Therefore, CARP$^{e}$'s graph operator performs an MLP operation on the summed neighborhood features and the node features with two separate MLPs $\phi_0$ and $\phi_1$. The MLP architectures are of two layers each for $\phi_0$ and $\phi_1$ in order to satisfy the universal approximation theorem [68, 69] and the recommendations for GIN operations as defined in [22].

$$h'_i = \phi_0\Big(\alpha \cdot h_i\Big) + \phi_1\Bigg(\sum_{j \in N(i)} h_j\Bigg) \tag{4.1}$$

This process is illustrated in the top portion of Figure 4.1, where $\alpha = 1 + \epsilon$ from Equation 2.1. Only a single graph operation is completed across the graph. This is done for two reasons. First, because the graph is fully connected, all pedestrians are accounted for in a single operation. Second, in aiming for real-time feasibility, limiting the number of operations allows our method to efficiently operate at scale. The output node features $h'_i \ \forall i \in \{1, 2, \cdots, P\}$ are subsequently employed in the Prediction Module.

### 4.1.3 CARP$^{e}$ Posterum: Prediction Module

Typically, Recurrent Neural Networks (RNNs) are employed as the basis for state-of-the-art path prediction methods. Given the theoretical ability of RNNs to capture information along infinitely many time steps, such architectures have been frequently chosen for sequence-based problems, particularly the LSTM variant. However, recent works in the sequence modeling domain [53, 70] have found convolutional architectures to be advantageous over RNNs in many ways. Convolutional approaches to sequence modeling often form conceptually simpler networks, have more stable gradients and allow for greater parallelization, while producing comparable or improved accuracies

on sequence data. Additionally, CNNs are well established for effectively capturing correlations in the spatial domain [71]. The task of path prediction presents itself as both a spatially and temporally sensitive task, receiving and predicting $(x_i^t, y_i^t)$ coordinate values across time. Therefore, we find that utilizing a convolutional architecture rather than an RNN may be more effective for the path prediction problem, while offering desirable hardware-friendly characteristics for real-time inference.

To this end, we form our Prediction Module with a simple CNN design to maximize spatial and temporal understanding, taking full advantage of the convolutional architecture approach. In our model, we first provide as input $S_i \in \mathbb{R}^{\beta \times 2 \times 2}$, formed from the relative feature for a pedestrian $R_i \in \mathbb{R}^{2\beta}$ and their corresponding output node feature $h_i' \in \mathbb{R}^{2\beta}$ concatenated together. In $S_i$, the $(x_i^t, y_i^t)$ coordinate pairs are mapped with temporal order in the rows. We therefore map both the spatial coordinate information and temporal context into the 2D domain, where the CNN can advantageously correlate. This input structure provides the ability to naturally analyze the observed trajectory at various time granularites, adjusting filter size and stride, in a hardware-friendly fashion. In RNNs, such analysis would be impractical and computationally inefficient, requiring multiple LSTMs per pedestrian. To provide additional social context to the input of the Prediction Module, the output node feature $h_i'$ is placed as the second channel in $S_i \in \mathbb{R}^{\beta \times 2 \times 2}$.

The layers of the network are designed to capture changes in velocity and position with a bottom-up approach. First, a 2x2 filter is convolved across the input, as illustrated in Figure 4.1. By convolving across just two time steps for each kernel, we emphasis model awareness of the high frequency movement and velocity changes over the observed period. As the feature progresses through the network, 2x1 filters are employed to find lower frequency trends, gathering the context of the trajectory across more time steps in the subsequent compressed representations. After the third network layer, CARP$^{\text{e}}$ produces a tensor in $\mathbb{R}^{\frac{\beta}{8} \times 1 \times 2T}$ that obtains a holistic under-

standing of the observed trajectory. A subsequent 1x1 convolution transforms this feature into the predicted trajectory.

## 4.2 Experiments

### 4.2.1 Evaluation Methodology

We evaluate our model on two widely-used datasets in the path prediction domain, ETH [1] and UCY [2]. The ETH dataset is split into two portions (ETH, HOTEL), and UCY is split into three portions (UNIV, ZARA1, ZARA2). All portions are from distinct scenes other than ZARA1 and ZARA2, which are the same scene at different times. These datasets consist of a variety of pedestrian navigation situations, including many nonlinear behaviours and social interactions. We utilize the same data and evaluation procedures as in [18], and commonly used in path prediction works [3, 4, 16, 20]. Therefore, a leave-one-out approach is applied for training and testing among the five scenarios. The data is collected as real-world coordinates in meters, with observations taken for 8 time steps (3.2 seconds) and predictions made for the next 12 time steps (4.8 seconds). Two metrics are utilized for quantitative evaluation on the ETH/UCY datasets:

- Average Displacement Error (ADE) - The average L2 distance between the ground truth $(x, y)$ positions $\hat{Y}$ and predicted $Y$ for all $T$ predicted time steps over all $P$ pedestrians.

$$\text{ADE} = \frac{\sum_{i=1}^{P} \sum_{t=1}^{T} \left\| \hat{Y}_i^t - Y_i^t \right\|_2}{P * T} \tag{4.2}$$

- Final Displacement Error (FDE) - The average L2 distance between the ground truth $(x, y)$ positions $\hat{Y}$ and predicted $Y$ for only the final time step $T$ over all

$P$ pedestrians.

$$\text{FDE} = \frac{\sum_{i=1}^{P} \left\| \hat{Y}_i^T - Y_i^T \right\|_2}{P} \tag{4.3}$$

All inference timing analyses are run with a *frame* batch size of one to accurately measure latency and throughput for a realistic streaming input scenario. Note that all pedestrians in a scene at time $t$ are processed simultaneously, and therefore each singular *frame* input still inherently requires a *pedestrian* batch of $P$ trajectories.

### 4.2.2 Implementation Details

For the dimensions mentioned in Figure 4.1, $\beta = 8$ and $T = 12$, in accordance to the evaluation methodology. The MLPs $\phi_0$ and $\phi_1$ contain two hidden layers with input dimensions of $4\beta$ and $2\beta$. The embedding layer $\rho$ has an input dimension of $2\beta$ and upscales by 2. We implemented the model in PyTorch and trained it on an Nvidia Titan V GPU. An open-source PyTorch extension library for graph convolution [72] was used as the basis for implementing the Graph Module. The model was trained end-to-end with a *frame* batch size of 64 for 80 epochs. We use the Adam [73] optimizer with a learning rate of 0.01 and a gradient clip of 5. A mean squared error loss was used for training.

### 4.2.3 Quantitative Results

#### 4.2.3.1 Comparison Approaches

We compare our model to common baseline methods and current state-of-the-art approaches in path prediction. Baseline methods include *Linear*, a simple linear regressor, and Social LSTM [16] (*S-LSTM*), a classic method utilizing LSTMs and social pooling. Social GAN [18] adds generative models to the Social LSTM approach. *SGAN-P* and *SGAN* indicate the variants with and without the social pooling module as reported in [18]. *Sophie* [20] employs an LSTM-based GAN module with social

and physical attention. Social BiGAT [3] (*S-BiGAT*) incorporates LSTMs, Bicycle-GANs [54] and physical attention, with GAT [55] networks to model social elements. *Next* [4] is a state-of-the-art approach that employs visual pedestrian features and scene segmentation maps for an LSTM-based prediction module with focal attention to make informed trajectory predictions.

#### 4.2.3.2 Accuracy Analysis

ADE and FDE results are reported in Table 4.1. It is common for generative approaches in this domain to predict 20 possible trajectories for each pedestrian, and use the closest prediction to ground truth in evaluation. However, since we are considering evaluation within a real-time context, an analysis of the single trajectory prediction results is much more applicable. Therefore, we compare with the K=1 results for all approaches, where K is the number of predictions per pedestrian.

As seen in Table 4.1, CARP$^{e}$ performs very well against all other methods. In ADE, CARP$^{e}$ achieves within 0.02 meters of the best state-of-the-art approach *Next* on average. In FDE, our approach outperforms all other methods. The Prediction Module design to emphasize understanding of velocity change across both the coordinate and social context features allows our method to adjust the final prediction positions accordingly.

Similar to *S-BiGAT*, CARP$^{e}$ employs a GNN for gathering insight into the so-

Table 4.1: ADE and FDE results for all five scenarios in the ETH [1] and UCY [2] datasets. Results followed by * are the K=1 accuracies as reported in the analyzes of [3].

| Dataset | Linear | S-LSTM | SGAN-P | SGAN | Sophie | Next | S-BiGAT | CARP$^{e}$ |
|---|---|---|---|---|---|---|---|---|
| **ETH** | 1.33 / 2.94 | 1.09 / 2.35 | – | 1.13 / 2.21 | – | 0.88 / 1.98 | – | **0.80** / **1.48** |
| **HOTEL** | 0.39 / **0.72** | 0.79 / 1.76 | – | 1.01 / 2.18 | – | **0.36** / 0.74 | – | 0.52 / 1.00 |
| **UNIV** | 0.82 / 1.59 | 0.67 / 1.40 | – | **0.60** / 1.28 | – | 0.62 / 1.32 | – | 0.61 / **1.23** |
| **ZARA1** | 0.62 / 1.21 | 0.47 / 1.00 | – | **0.42** / 0.91 | – | **0.42** / 0.90 | – | **0.42** / **0.84** |
| **ZARA2** | 0.77 / 1.48 | 0.56 / 1.17 | – | 0.52 / 1.11 | – | **0.34** / 0.75 | – | **0.34** / **0.69** |
| **AVG** | 0.79 / 1.59 | 0.72 / 1.54 | 0.85 / 1.76* | 0.74 / 1.54 | 0.71 / 1.46* | **0.52** / 1.14 | 0.61 / 1.33* | 0.54 / **1.05** |

cial context. *S-BiGAT* uses GAT operations in its graph, which operate with simple single-layer operations on node features. However, as mathematically shown in [22], such single-layer operations are insufficient for robust graph learning. Instead, CARP$^{e}$ employs a GIN-based formulation with MLP operations to maximize the discriminative power of its graph module. This design choice in GNN gives CARP$^{e}$ a competitive edge over *S-BiGAT*, as revealed in the average ADE and FDE results of Table 4.1.

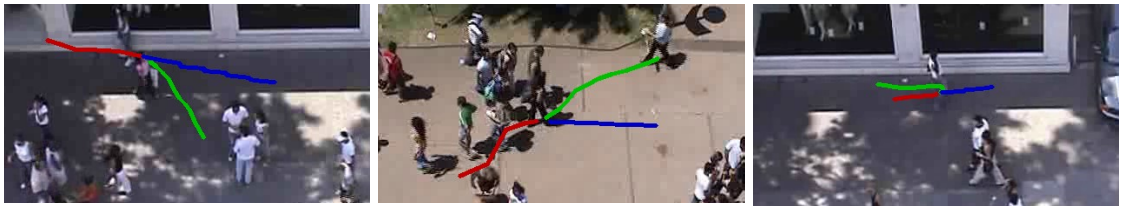
### 4.2.4 Qualitative Results

Figure 4.2 illustrates the function of CARP$^{e}$ with qualitative examples. The first row of Figure 4.2 shows cases of solely intrinsic nonlinearities. In these examples, the pedestrian takes indirect paths with changes in direction and/or speed. In the failure case (b.3), the observed path seems to foreshadow a change in speed alone, but the ground truth indicates that the pedestrian will soon change direction drastically. This abrupt adjustment is not anticipated by CARP$^{e}$. However, with some additional information a few time steps later, CARP$^{e}$ is able to understand the nonlinear behavior and anticipate the future positions of the pedestrian as shown in (a.2). Examples (a.1) and (a.3) illustrate how CARP$^{e}$ captures an understanding of speed and direction variations to predict the pedestrian's navigation intent.

The first two columns of Figure 4.2 provide samples encompassing intrinsic and social nonlinearities. Examples (a.5) and (a.6) show situations where the depicted pedestrian alters their trajectory in response to simple social states, including nonlinear change of direction in response to their neighbor's movement as they travel together. Sample (a.4) shows a complex crowd scenario, where CARP$^{e}$ is able to determine a likely path through the crowd and achieve a correct prediction.

In (b.1), we show a failure case in which CARP$^{e}$ predicts the pedestrian navigating behind a stationary group. However, the pedestrian actually chooses to navigate in

(a) Successful examples



(b) Failure examples

Figure 4.2: **Red** indicates the observed trajectory, **blue** is the predicted trajectory, and **green** is the ground truth. Images are referenced in to the text as (a.1) to (a.6) and (b.1) to (b.3) from top-left to bottom right.

front of the group instead. The complex crowd scenario, illustrated in (b.2), contains many pedestrians moving individually and in groups. The observed trajectory of the individual traveller is varied in directional intent, and the traveller is nearing collision with a group of pedestrians moving left to right. It is difficult to predict a deterministic route under this intense uncertainty, and therefore CARP$^{e}$ takes the safe bet and assumes a path consistent with the social norm (moving left to right with the group). However, the person decides to travel in front of the group and progress upward across the pathway. In samples (b.1) and (b.2), we note that having visual features of the pedestrians would provide insight into their body position and intended direction. Such modeling would potentially assist in prediction for such scenarios, and real-time capable integration of this information may be a worthwhile direction for future work.

### 4.2.5 Real-time Analysis

Path prediction is inherently a time-sensitive task. A crucial characteristic of a path prediction algorithm is its ability to perform real-time inference, particularly on low-power embedded devices. Every fraction of a second is crucial for improving the safety of deployable technologies like self-driving cars and social robots. Therefore, we analyze CARP$^e$ in comparison to current state-of-the-art approaches for such characteristics. In Table 4.2, we first compare the FPS of CARP$^e$ on an Nvidia P100 GPU as a baseline. CARP$^e$ far surpasses the performance of all other methods, by at least 38x across the board.

CARP$^e$ achieves such improvements for two reasons. First, CARP$^e$ is designed with real-time inference in mind, eliminating extraneous operations and focused on optimizing the computation expense to accuracy ratio. Second, CARP$^e$ employs a convolutional rather than recurrent architecture. All other methods base their approach on LSTM cells, which limit their hardware utilization capabilities. Instead, CARP$^e$ is able to take full advantage of the parallel computing capacities of modern hardware, and is thereby well suited for real-world deployment.

*Sophie* and *S-BiGAT* do not have their models or latency numbers publicly available, and therefore we do not report their FPS performance. However, it can be noted that both *Sophie* and *S-BiGAT* add additional layers of complexity to the *SGAN* approach. These include the addition of scene-level feature extraction directly on the image using the computationally heavy VGG-19 [74] network, as well as additional scene and social attention mechanisms. Therefore, we can conclude that not only

Table 4.2: FPS comparison on the Nvidia P100 GPU. Numbers are report as an average per frame across both ETH and UCY datasets.

| **Device** | **S-LSTM** | **SGAN-P** | **SGAN** | **Next** | **CARP$^e$** |
|---|---|---|---|---|---|
| **P100** | 0.38 | 6.67 | 20.00 | 19.46 | **762.14** |

will these approaches experience similar computational difficulties as the other RNN-based architectures, they will also incur additional latencies due to the use of large scene feature extractors and attention networks.

In Table 4.3, we thoroughly compare real-time performance of CARP$^{e}$ against the best state-of-the-art approach *Next*. First, we analyze the number of floating point operations (FLOPs) and parameters for each approach. CARP$^{e}$ is substantially more efficient, reducing the number of FLOPs and parameters by more than 97%. We also consider the FPS performance on a low-power embedded device, the Nvidia Jetson Nano. For both GPU and single core CPU inference, CARP$^{e}$ provides an over 17x and 8x speedup respectively in comparison to *Next*.

Because of its RNN-based design, *Next* is not able to effectively utilize the parallel capabilities of modern hardware, as evident by its almost equal CPU and GPU FPS numbers. In all reported numbers in Tables 4.2 and 4.3, we only consider the trajectory generator portion of the *Next* approach. However, this trajectory generator relies on additional scene segmentation features and visual pedestrian features, both of which require large networks for extraction (Xception [75] and ResNet-101 [76] based architectures). Therefore, we compare using an optimistic scenario for *Next*.

Table 4.3: Detailed comparison with Next [4]. FPS numbers reported on the Nvidia Jetson Nano embedded device for both GPU and single core CPU. The Nano a 128-core Maxwell GPU and ARM A57 CPU with a power consumption of approximately 10 Watts in our tests.

| **Approach** | **MFLOPs** | **Parameters** | **FPS (GPU)** | **FPS (CPU)** |
|---|---|---|---|---|
| **Next** | 53.08 | 3.95M | 5.61 | 5.50 |
| **CARP$^{e}$** | **1.14** | **0.10M** | **95.87** | **48.11** |

# CHAPTER 5: CONCLUSIONS

Overall, REVAMPT is a deployable pedestrian detection and re-identification system capable of low-power, privacy-aware operation. For the results and evaluation, this article introduces a new two-part metric, Accuracy·Efficiency (Æ). REVAMPT outperforms current state-of-the-art more than ten-fold in Æ. We also proposed CARP$^{e}$ Posterum, a convolutional approach for real-time pedestrian path prediction. Distinct from prior work, CARP$^{e}$ is able to produce accurate future trajectory predictions within real-time constraints. CARP$^{e}$ is an agile CNN that operates across the temporal context of observations in the spatial domain, maximizing both feature correlation and parallel hardware utilization. We also employ a discriminative graph neural network based on GIN operators to gather social context features, providing additional insight into the predictive model. Through quantitative and qualitative evaluations, we show that CARP$^{e}$ has the ability to capture nonlinear intrinsic and social effects within its novel architecture design, achieving competitive accuracy results in comparison with the current state-of-the-art methods while enabling 8x to 38x improvements in FPS.

Future directions include integrating CARP$^{e}$ into REVAMPT, and improving for resilience against noise in detections. Also, the keypoint information gathered by REVAMPT could be useful to integrate into CARP$^{e}$ to better understand pedestrian directional intent. For REVAMPT specifically, designing an intelligently-scaled human pose estimator using very recent network concepts like EfficientNet [77] could replace OpenPose and provide similar accuracy at higher inference rates.

# REFERENCES


[1] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 261–268, Sep. 2009.

[2] A. Lerner, Y. Chrysanthou, and D. Lischinski, "Crowds by example," *Comput. Graph. Forum*, vol. 26, pp. 655–664, 2007.

[3] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, H. Rezatofighi, and S. Savarese, "Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, eds.), pp. 137–146, Curran Associates, Inc., 2019.

[4] 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 Conference on Computer Vision and Pattern Recognition*, pp. 5725–5734, 2019.

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

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

[7] 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.

[8] 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.

[9] "Surveillance solutions."

[10] 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.

[11] S. Tang, M. Andriluka, B. Andres, and B. Schiele, "Multiple people tracking by lifted multicut and person re-identification," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3701–3710, 2017.

[12] S. Tang, B. Andres, M. Andriluka, and B. Schiele, "Subgraph decomposition for multi-target tracking," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5033–5041, 2015.

[13] D. Helbing and P. MolnÃ¡r, "Social force model for pedestrian dynamics," *Physical Review E*, vol. 51, p. 4282â4286, May 1995.

[14] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 261–268, Sep. 2009.

[15] K. Yamaguchi, A. C. Berg, L. E. Ortiz, and T. L. Berg, "Who are you with and where are you going?," in *CVPR 2011*, pp. 1345–1352, June 2011.

[16] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[17] 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 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2672–2680, Curran Associates, Inc., 2014.

[18] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2255–2264, June 2018.

[19] Y. Li, "Which way are you going? imitative decision learning for path forecasting in dynamic scenes," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[20] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese, "Sophie: An attentive gan for predicting paths compliant to social and physical constraints," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

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

[22] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *International Conference on Learning Representations*, 2019.

[23] 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.

[24] 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.

[25] 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.

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

[27] 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.

[28] 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.

[29] 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.

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

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

[32] 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.

[33] 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, 2018.

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

[35] 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, no. 11, pp. 5683–5695, 2018.

[36] 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, 2017.

[37] 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, no. 4, pp. 2042–2054, 2017.

[38] 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, no. 3, pp. 1366–1377, 2019.

[39] 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.

[40] 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.

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

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

[43] 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.

[44] J. Zhu, H. Yang, N. Liu, M. Kim, W. Zhang, and M. Yang, "Online multi-object tracking with dual matching attention networks," *CoRR*, vol. abs/1902.00749, 2019.

[45] Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu, "Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism," *CoRR*, vol. abs/1708.02843, 2017.

[46] 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.

[47] 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.

[48] M. K. C. Tay and C. Laugier, *Modelling Smooth Paths Using Gaussian Processes*, pp. 381–390. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[49] S. Pellegrini, A. Ess, and L. Van Gool, "Improving data association by joint modeling of pedestrian trajectories and groupings," in *Computer Vision – ECCV 2010* (K. Daniilidis, P. Maragos, and N. Paragios, eds.), (Berlin, Heidelberg), pp. 452–465, Springer Berlin Heidelberg, 2010.

[50] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[51] A. Graves, "Generating sequences with recurrent neural networks," 2013.

[52] H. Manh and G. Alaghband, "Scene-lstm: A model for human trajectory prediction," *CoRR*, vol. abs/1808.04018, 2018.

[53] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018.

[54] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman, "Toward multimodal image-to-image translation," in *Advances in neural information processing systems*, pp. 465–476, 2017.

[55] P. VeliÄkoviÄ, G. Cucurull, A. Casanova, A. Romero, P. LiÃ², *andY. Bengio*, "$Graphattentionnetworks,''$ $inInternationalConferenceonLearningR$

[56] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," 2019.

[57] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[58] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," 2016.

[59] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in neural information processing systems*, pp. 1024–1034, 2017.

[60] B. Weisfeiler and A. A. Lehman, "A reduction of a graph to a canonical form and an algebra arising during this reduction," *Nauchno-Technicheskaya Informatsia*, vol. 2, no. 9, pp. 12–16, 1968.

[61] 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.

[62] 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.

[63] 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.

[64] 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.

[65] C. Neff, M. Mendieta, S. Mohan, M. Baharani, S. Rogers, and H. Tabkhi, "Revamp$^2$t: Real-time edge video analytics for multi-camera privacy-aware pedestrian tracking," 2019.

[66] 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.

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

[68] K. Hornik, M. Stinchcombe, H. White, *et al.*, "Multilayer feedforward networks are universal approximators.," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[69] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

[70] M. Elbayad, L. Besacier, and J. Verbeek, "Pervasive attention: 2d convolutional neural networks for sequence-to-sequence prediction," 2018.

[71] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[72] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[73] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[74] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[75] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *CoRR*, vol. abs/1610.02357, 2016.

[76] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

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