EFFICIENT AND SCALABLE HIGHWAY ASSET DETECTION

by

Ekta Prakash Bhojwani

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

2021

Approved by:

_____
Dr. Hamed Tabkhi

_____
Dr. Omidreza Shoghli

_____
Dr. Chen Chen

_____
Dr. James Conrad

ABSTRACT

EKTA PRAKASH BHOJWANI. Efficient and Scalable Highway Asset Detection.
(Under the direction of
DR. HAMED TABKHI)

. The roads and highways are a valuable asset for the state department of transportation. It takes massive investment and a huge amount of time to maintain the road assets condition. Therefore, it becomes important to automate the maintenance process with minimum manual inspection. There has been significant research in the domain of classical computer vision techniques and machine learning methods concerning highway and road assets maintenance. However, the time consumed to assess and maintain and the amount of manual inspection involved is considerably large. Although, adding automation to speed up the inspection process has been investigated by many studies, they overshadow the importance of scalable and light frameworks detecting the assets like storm drain and drop inlet in real-time. Thus, this thesis focuses on integrating a reliable and scalable AI Deep Learning framework customized for highway assets localization and detection of the assets in a road infrastructure environment mitigating the need for large and bulky model sizes. Furthermore, utilizing the advantage of the less computational cost and the lightweight framework architecture along with reasonably higher accuracy, it is possible to build an end-to-end framework that supports object localization and detection followed by the inference on the mobile edge embedded platforms. In a nutshell, this thesis presents a Deep Learning localization platform, customized to predict the position or the location of the asset items on the Highways such as drop inlets and storm drains. Moreover, it also provides results on the scalability of the localization task to the multi-object detection task with the help the state of the art EfficientDet-D0 model with a test accuracy of 73.4% mAP on the annotated test dataset and achieving validation accuracy of 51.67% mAP on the customized merged data of highway asset item

drop inlet and 5 other COCO classes for object detection application. Additionally, various analyses based on mIOU and classification scores are described in the experimental section below. It also represents that the model framework is edge deployable friendly and can be quantized to an Fp16 lighter version of the model with help of the NVIDIA TensorRT engine showing the benchmarking performance of 50.55 FPS on the NVIDIA Jetson AGX Xavier mobile embedded platform. Moreover, it highlights the challenges and the future scope expansion of the work to the real-time onboard drone visual analytics for Highway assets defect detection.

DEDICATION

This work is wholeheartedly dedicated to my father Mr. Prakash Bhojwani and my mother Mrs. Nisha Bhojwani who have been my source of encouragement and have proven to be committed to stand by me in the challenging situations and have provided me the necessary moral, emotional, spiritual and financial support.

To my brothers, sisters, relatives, mentors, friends and colleagues who have always been available to share their wisdom, advice and enthusiasm to finish this thesis.

ACKNOWLEDGEMENTS

I would like to express my deep and sincere gratitude to my advisor Dr. Hamed Tabkhivayghan, BS, MS, PhD, Assistant Professor, Electrical and Computer Engineering Department, the William States Lee College of Engineering, University of North Carolina at Charlotte, for giving me the opportunity to carry out research and providing invaluable guidance throughout this thesis. His commitment to aim for a task and complete it with thorough details and attention have motivated me to be strong and pursue Thesis. His valuable guidance and his involvement has made this project a successful one. It was a great privilege and an honor to work under him and his research lab, TeCSAR. I would also like to thank his friendship, empathy and a great sense of humor. I am extremely thankful to Dr. Omidreza Shoghli, BS, MS, PhD, Assistant Professor, Civil Engineering Technology and Construction Management for providing me constant support and advice throughout my Thesis journey. His most promising quality of getting things done in an organized and structured way before time has inspired me deeply.

I am indefinitely grateful to the Leidos team, especially Dr. Adrian Burde, BS, MS, PhD, Asset Management Engineer, Leidos, Virginia for his constant acknowledgement and feedack on the research. His industry level oriented attitude has helped me evolve taking this research into a product base environment. I would like to acknowledge Leidos and VDOT for sharing the image dataset. Without the road asset database, training the model would not have been possible I am extremely grateful to my parents for their love, caring and sacrifices for educating and providing me with a future. I am thankful to my relatives and my friends for constantly supporting me. My special thanks to Christopher Neff, PhD candidate, Electrical and Computer Engineering, University of North Carolina at Charlotte, for his patience, ideas and mentoring throughout this work. I would also like to acknowledge Google Meet for providing a platform for online conferencing during my defense presentation and throughout the

last semester, which was unavoidable due to the coronavirus pandemic.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADASYN Adaptive Synthetic Sampling

AI Artificial Intelligence

AP Average Precision

BiFPN Bidirectional Feature Pyramid Network

CNN Convolutional Neural Network

COCO Common Objects in Context

FLOPs Floating point Operations

FP16 Floating Point 16

FP32 Floating Point 32

FPGA Field Programmable Gate Array

FPS Frames Per Second

GLOPS Giga Floating Point Operations

GPU Graphics Processing Unit

GT Ground Truth

IOU Intersection Over Union

mIOU Mean Intersection Over Union

NMS Non Max Suppression

ONNX Open Neural Network Exchange

PASCAL-VOC Visual Open Challenge

SGD Stochastic Gradient Descent

TPU Tensor Processing Unit

TRT TensorRT

VDOT Virginia Department of Transportation

CHAPTER 1: INTRODUCTION

The highways automated assessment, maintenance, and surveillance is one of the highest demands in the Computer Vision community with a perspective to alleviate the manual labor burden, and most importantly reducing the risk of the workers/laborers working [1] [2] in the work zone. Moreover, the assessment is essentially deployed with the help of AI for the broad analysis of the conditions of the highways ensuring the safety of the humans and the environment in a shorter time. The AI automated framework in this thesis involves an end to end pipeline from capturing the frames to pre-processing it, fitting it in framework, and then realizing it for the inference benchmarking on the mobile-edge devices [3] predicting the condition of the asset item. To start with implementing the framework, the dataset [4] [5] is the powerhouse of this pipeline, which requires a significant exploration such as organizing it into the desired structure, curating it to fit to the desired framework, with augmentations and various other processing techniques. After the dataset is organized and structured, to execute the task effectively with a lightweight framework meeting the demands of real-time automated assessment in the near future, a selective Deep Learning Convolutional Neural Network (CNN) [6] [7] framework customized for the desired task is extremely important. Furthermore, it also describes the lightweight model architecture design since the large model sizes [8] [9] and expensive computation costs deter their deployment in many real-world applications such as robotics and self-driving cars where model size and latency are highly constrained. Given these real-world resource constraints, model efficiency [10] [11] becomes increasingly important for object detection. Speaking about the Deep Learning tasks best suited for the implementation of the research study, the best-suited task was starting with

the preliminary approach which is localization, i.e, detecting the position of the asset item in the scene to localizing and classifying the difference between the other asset items known as detection in the Machine Learning and AI community. Figure 1.1 shows an example of multi-scaled object detection involving a drop inlet to show the custom dataset item. Moreover, the Figure 1.2 below shows the example of the highway drop inlets assets of the dataset.

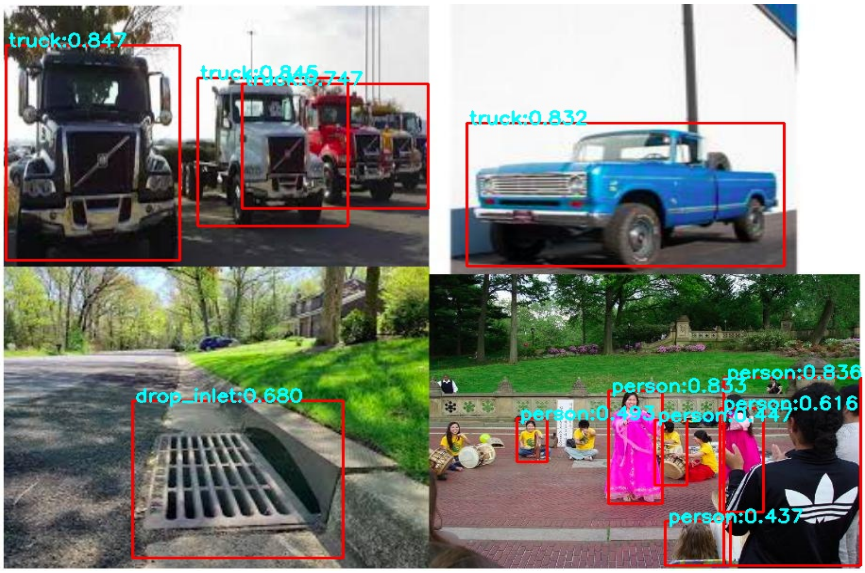

Figure 1.1: Object detection of different objects in a scene

Object Localization and Object Detection [12] [13] [14] [10] are few of the very trending topics and demand huge attention for specialized and customized tasks. With the view to implement such specialized object detection tasks there are many famous open-source datasets available such as COCO 2017 [15], COCO 2014, Pascal-VOC 2017 [16], etc used for object detection, image segmentation [17] [18] and so on. These datasets [19] consists of data in the form of images and annotations or label files to suggest the location of the object present in the image with the label to which category the object belongs. Going forward with the customized tasks, it requires a custom dataset which means integrating the desired data and to work on that data. Broadly speaking, the custom dataset would consist of the images as the desired

objects for the scalable multi-object object detector. Thus, to address the concerns of the importance of the data collection from the Leidos, the industry collaborator with this project, and inspecting its quality and materializing it into an AI framework to complete the assets detection task this thesis study is based on the data processing and the implementation method with the proof of concept that the efficient and scaled framework can be optimized for the future real-time [3] deployment on edge embedded devices.

Figure 1.2: Examples of highway drop inlet asset item

1.1 Motivation

According to the state department of transportation, the highway maintenance [1] [20] and assessment [21] is a must so as to ensure the quality of the highways, the power and the cost efficiency, avoiding the unforeseen natural calamities like floods, cyclones, and overall driving experience for the drivers on the highways. The main motive behind executing the research is to assess the highway assets [2] such as drop inlets, storm drains, pipes, culverts, etc. To assess and maintain the assets with AI and Deep Learning in Computer Vision strategies, the pipeline shown in the figure below, gives the overall picture of the localization and the classification of the highway

asset item as the output. To achieve this, an efficient lightweight model architecture EfficientDet-D0 [21] [21] is chosen. The real time execution pipeline includes the inference framework as discussed below. The framework is benchmarked to validate its performance on the mobile edge embedded platforms with an aim to deploy it on the on-board real-time drone visual analytics in the near future. The current detector proves to be edge deployable with reasonable FPS benchmarking as a preliminary step.

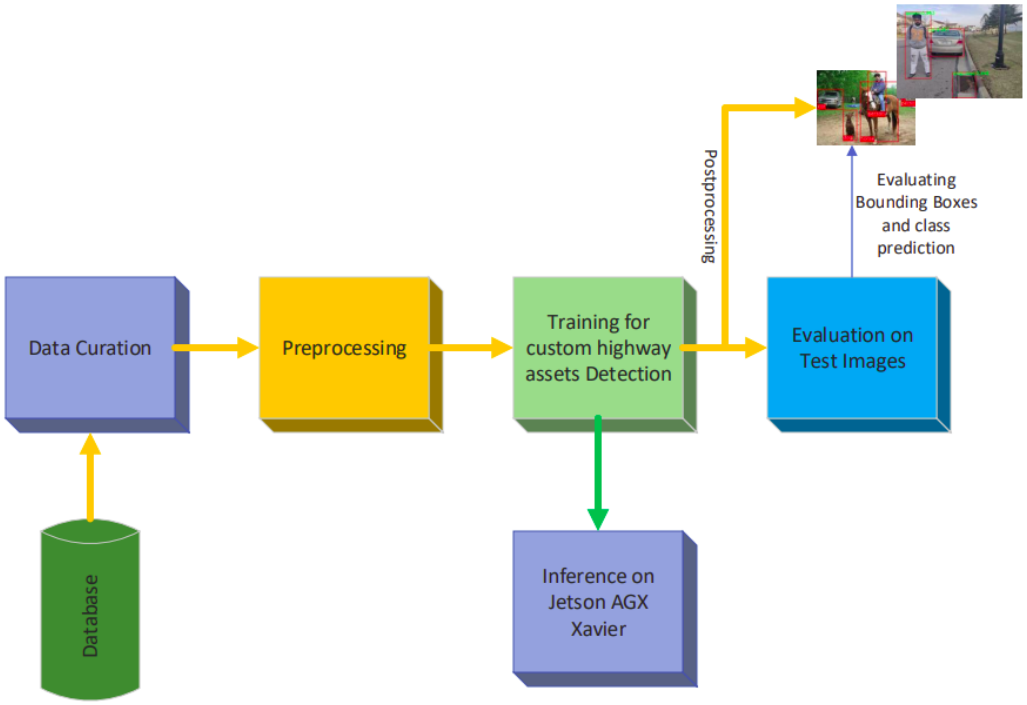

Figure 1.3: End-to-end pipeline for Highway Assets Detection

1.2 Contributions

EfficientDet is a family of lightweight scalable networks for high-resolution and efficient object detection with great accuracy and efficiency. It unifies the state of the art backbone EfficientNet network, which avails the flexibility of tuning the width, depth of the layers and the input resolution and the Bidirectional Feature Pyramid Network [22] [23] [24] followed by the Class and Box prediction network. The EfficientDet[22] framework for customised task depends on the custom dataset essentially. A popular

method transfer learning is implemented, where the new customised dataset is fine-tuned based on the pre-trained weights of the pre-trained model, which is trained on a large open source COCO2017 dataset. For the sensible and the best training model, the hyperparameters tuning is involved. The evaluation and the inference pipeline proves to achieve high performance FPS. For the clarity of the main steps taken in order to make this project a successful one, as shown in the Figure 1.3. In summary, this study has the following major contributions:

Towards front-end algorithms:

- Curating, analyzing and pre-processing the dataset to make it fit for the highway assets detection task. Moreover, filtering out the COCO dataset into 5 classes and merging the highway drop inlets class along with the filtered COCO dataset so as to prove the multi-object detector scalibility of the model.
- Building a customized dataloader where the images and the annotation of the dataset are fed to the model and arranging the dataset format accordingly.
- Building an augmentation pipeline to mitigate the effect of meagre dataset on the training framework. The augmentation pipeline includes offline augmentation such as Random scale, Random zoom in and zoom out, and so on.
- Evaluating the qualitative results based on the results generated by the model on the random test dataset identifying the objects in the image based on the categories and the confidence scores so as to make it easily available for the inference pipeline.
- Creating optimized C++ vision pipeline with TensorRT models for efficient agile execution on embedded edge devices.
- Evaluation of vision pipeline on NVIDIA Jetson Xavier platform as an early proof of concept for real-time on-board processing.

1.3 Thesis Outline

The Thesis is based on the collaboration with the Leidos and VDOT industry outlined as follows: Chapter 2 gives a detailed report on background related work of Transfer Learning pipeline, class imbalance methods and inference optimization with the current market trend in the field of inference on the embedded platforms. Chapter 3 introduces the network architecture of EfficientDet - both localization of 1 asset item and object detection. It explains the backbone CNN as well as the Architecture details and how is it customised for the task. Chapter 4 is the Inference Optimization description on NVIDIA Jetson AGX Xavier. Chapter 5 is the experimental results section where an exhaustive evaluation of the framework on the Leidos dataset of highway asset items and the relevant COCO classes is conducted. Experimental results also include inference benchmarking performance on TensorRT engine for this project's developed custom model.

CHAPTER 2: RELATED WORK

This section is an amalgamated set of related work relevant to the thesis. First, it represents the background of the transfer learning useful to train the networks with smaller datasets. It also includes the challenges with a small or a meager dataset. The background highlights popular class balancing techniques to mitigate the class imbalance problem. Lastly, a survey for new and emerging optimizing framework for the inference has been discussed.

2.1 Road pavement Damage detection

There has been research in the field concerning the pavement damage detection. In particular, some works focus on detecting only the existence of the damage regardless of its type [25] [21]. Other works focus on classifying the road damages into a few types. For example, [26] devised an approach for detecting two directional cracks (i.e., horizontal and vertical), while [27] developed another approach for detecting three categories of damages, namely horizontal, vertical, and crocodile. Due to the fact that differentiating among damage types is critical for proper road maintenance planning, [28] have implemented an approach for a thorough classification of road damage types. There have been research with road damage classification and detection tasks which [28] [25] use bulky networks. However, the accuracy and efficiency is the main concern for identifying the damage. The [25] [26] uses YOLO [29] which is a network generating requiring many bounding box regression anchors making the network bulky and not suitable for mobile edge deployment. While the work in [30] highlights on the automated technology for damage detection but uses a large dataset to analyse. While, we prove in our study that the automated detection can work based

on the meagre dataset availability of highway assets.

2.2 Road Asset Classification

There have been a significant contribution to this study which delivered solutions on the Road Asset Classification using transfer learning approach with multi-level classification. For the classification, the dataset used had 14 road asset items including the pavements [31], ditches, pipes, culverts, drains, etc. The road asset images dataset collected for this work were originally taken for manual inspection purposes, and mostly all of the images were from defected road assets which made the classification task more challenging comparing to when all images collected were from non-defective [21] road assets because defects could be of various forms and degrees. The challenges mentioned above made the model design a more complex task. Considering these challenges, a clear representation learning of similar and dissimilar features [32] among images under each class and between different classes was studied to further the classification [32] task. Hence, a very productive approach i.e., transfer learning was adopted to solve these challenging tasks. Moreover, in this way the model was potential to overcome the sparsity and data limitation challenges. ImageNet [19] as a base knowledge source provided valuable information regarding general features [33] [34] [35] of the objects that existed in the scenes of the target dataset i.e., the road assets dataset. This helped the model to recognize intra-class similarity and inter-class differences [36]. Thus, learning about such parameters, and addressing the concerns of the data limitations, this work included model representation to learning the differences between the contextual features [5] [4] of the classes. To reduce the negative effect of classes with high inter-class confusion, using a separate classifier [37] [32] as the second stage of network is proposed in this study. First the main classifier is used to classify all road assets. The results of main classifier is analyzed by generating a confusion matrix. Based on confusion analysis, challenging classes are picked to be classified by a binary classifier [38]. The dataset for main classifier

is modified and the challenging classes are combined under the input dataset of main classifier. In the case of this study, since unpaved ditch has a low accuracy and high confusion rate with paved ditch [31], these two classes are combined as ditch in the main classifier and a binary classifier is used to classify them. It can improve the accuracy for unpaved ditch and reduce the negative effect of unpaved ditch on paved ditch and as a result an increase in paved ditch class accuracy will be gained. This happens due to the fact that binary classifier will be specialized in distinguishing between paved ditch and unpaved ditch by optimizing the networkâs weights only on ditch images. Hence,this previous work was a great motivation to lead forward the task in the case of road assets classification.

2.3 Transfer learning methods

Transfer Learning is a broad topic under the Deep Learning [5] [4] umbrella. In the same way a human learns from previous experiences to generalize new similar situations, the deep learning framework attempts to learn the previous knowledge to generalize new tasks. This phenomenon of transferring the acquired information to a new model or task is called transfer learning[33] [34] [35] [39] . It has been practiced in many different ways such as weight transfer for super- vised learning [40] and policy transfer for reinforcement learning [41] . One way that transfer learning can be used for CNNs is to primarily train a CNN [6] [8] on a huge dataset [5] [4] with similar basic visual features to the target dataset for that includes data for the new or customized task. Utilizing pre-trained [35] [8] [42] networks not only reduces the training time significantly but also enables the CNN to learn from small datasets with high sparsity. In the context of object detector, transfer learning is helpful with the freezing the network layers or using the pre-trained backbone [9] [8] [43] and feature aggregation networks[24] [44]. While the class label prediction and bounding box regression network can be made to start learning from weight initialization weights.

2.4 Class Imbalance

Any dataset with unequal distribution between its majority and minority classes can be considered to have class imbalance, and in real-world applications, the severity of class imbalance can vary from minor to severe (high or extreme). A dataset can be considered imbalanced [45] [46] if the classes, e.g.,healthy and defect cases, are not equally represented. The majority class makes up most of the dataset, whereas the minority class, with limited dataset representation, is often considered the class of interest. [47] Most standard algorithms assume or expect balanced class distributions or equal misclassification costs. Therefore, when presented with complex imbalanced data sets, these algorithms fail to properly represent the distributive characteristics of the data and resultantly provide unfavorable accuracies across the classes of the data. There are famous data sampling techniques like Oversampling[48][47], which is increasing the samples of the rare classes, it can be data augmentation, Random oversampling simply replicates randomly the minority class examples. Random oversampling is known to increase the likelihood of occurring overfitting[49] [50] [51]. On the other hand, the major drawback of Random undersampling is that this method can discard useful data. There is one more technique, for instance Synthetic Minority Oversampling Technique (SMOTE) [47] [45] [52] [53] [48] This method is considered a state-of-art technique and works well in various applications. This method generates synthetic data based on the feature space similarities between existing minority instances. To create a synthetic instance, it finds the K-nearest neighbors [54] of each minority instance, randomly selects one of them, and then calculate linear interpolations to produce a new minority instance in the neighborhood. One more technique ADASYN [47] [45] generates samples of the minority class according to their density distributions. More synthetic data is generated for minority class samples that are harder to learn, compared to those minority samples that are easier to learn. It calculates the K-nearest neighbors[54] of each minority instance, then gets the class ratio

of the minority and majority instances to generate new samples. By repeating this process, it adaptively shifts the decision boundary to focus on those samples that are difficult to learn.

2.5 Optimizing Inference Engine

There have been huge demands for the Computer Vision applications in the domain of Healthcare, surveillance, safety, self-driving cars, defense, and many more. The Deep Learning inference framework helps in realizing these tasks into a product. Therefore, to gauge the real-time performance, there have been immense research in the Deep Learning Acceleration [55] [56] [57]. Hardware companies like NVIDIA [58], Intel [57] are focusing on the development of Deep Learning Accelerators optimized for inference engines. The TPU [59] [60] are the other hardware accelerators for inferencing. Despite having a much smaller and lower power chip, the TPU has 25 times as many MACs and 3.5 times as much on-chip memory as the K80 GPU. The TPU is about 15X - 30X faster at inference than the K80 GPU [61] and the Haswell CPU. The TPU has a higher inference speed because it uses a mix of 8-bit weights and 16-bit activations (or vice versa), the Matrix Unit computes at half-speed, and it computes at a quarter-speed when both are 16 bit. This offers huge acceleration at low power. Similarly, FPGAs [57] have started to implement the deep learning framework exploration meeting the high-intensity inference acceleration based on OpenCL and hardware kernel languages such as Vivado based on massive task parallelism.

CHAPTER 3: Deep Learning Training Framework

This chapter consists of the methods involved in implementing the pipeline and the architecture used to integrate it. It starts with the Deep Learning algorithm required followed by the Data curation to run the algorithm, the architectural design, compound scaling methodology, training setup of the architecture and the following chapter highlights on deployment of the architecture method on the hardware module Jetson AGX Xavier.

3.1 Deep Learning Algorithm

The department of health, transportation, highways, etc require a lot of maintenance and assessment [1] due to the rigorous activities and task intensiveness. Due to their exhaustiveness of the resources and the labor inspection, there is a need for automated tasks to prevent the risk of the laborers and the manual inspection involved. Thus, in this study, before starting to automate the task it is important to realize the availability of the resources and the data to be analyzed for the completion of the task. The research is based on developing an algorithm for the VDOT highway assets items inspection. The first and foremost task is to sense the availability of the database since it is an automated Deep Learning task that requires massive data [5] [4] to execute the application. The data collection and management is expensive which comes at the cost of resources required and the safety of the workers because of the data collection on the highways. Hence, the data collected was just bound to a single highway asset drop inlet with the resourcefulness being the concern. Therefore, the preliminary step was to just predict the location [62] of the single asset item with respect to the image in the dataset. This process is known as [63] the object

localization. Furthermore, to study the scalability of the model, learning its capacity to train the multiple objects of different categories, an algorithm using the state of the art EfficientDet [22] architecture has experimented thoroughly. The fine-tuning [64] [33] of the architecture and the necessary steps taken are dictated in the chapter below.

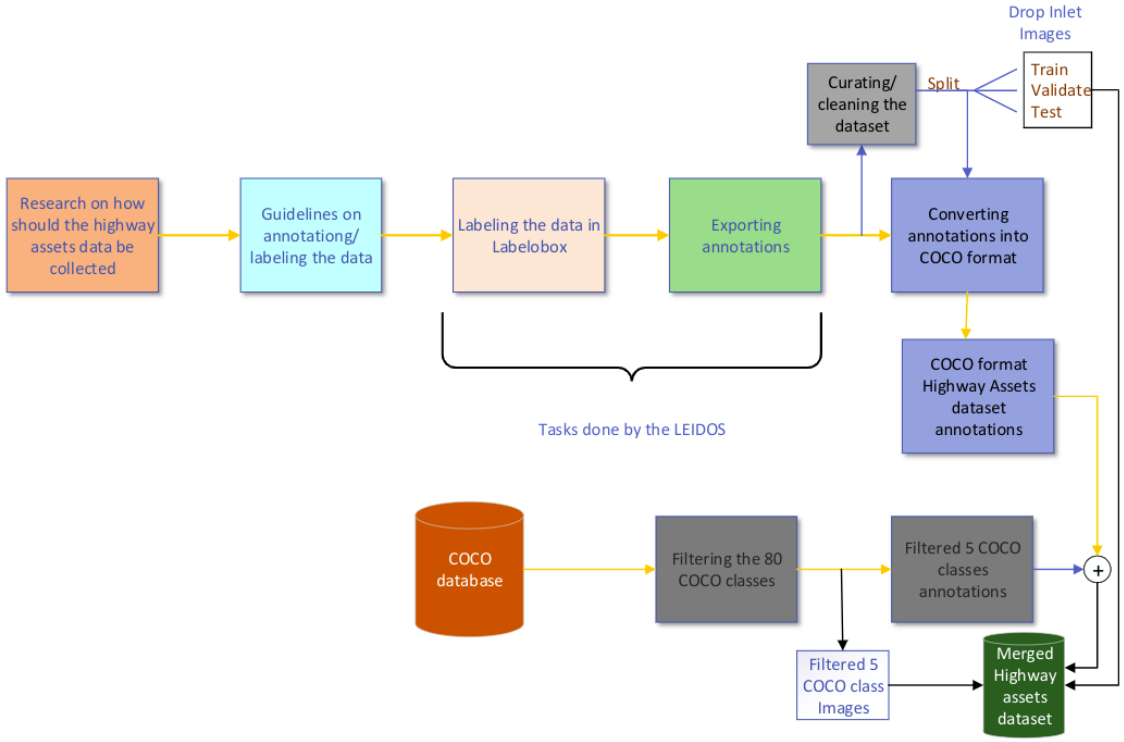

Figure 3.1: Dataset curation and organization

3.2 Data Curation

Dataset organization and management is a skillful art to run an AI pipeline. The equation of the data to training the model is quite proportional. The better the data the better the accuracy and the learning capability of the model. Thus, for a successful implementation of the scalable multi-object detection architecture, a reasonable number of classes images samples were taken for training and the corresponding evaluation. By dataset organization, it deeply means that the images and their annotations

are organized and cleaned to fit it to a particular annotations format. As discussed in the previous sections, the framework is mainly built for the custom dataset collected by the industry partner Leidos, i.e., the drop inlet highway asset item, the dataset was labeled in the Labelbox [65] labeling tool followed by the generation of an export annotations ready file. This annotations file is then converted to COCO[66] [67] [68] format annotations with an aim to maintain the consistency between further added classes to the dataset. This helped in building a model for the single class object detection since the availability of the single class. Furthermore, to create a scalable multi-class framework, the COCO classes were merged with few classes from the popular COCO dataset. To make a flexible training setup, it was just important to filter particular categories relevant to highway objects. Thus, 5 other classes from the COCO dataset [66] [15] namely the bus, car, person, train, truck were merged with the existing highway asset dataset bringing the total to a dataset of 6 classes. The training and the evaluation results are based on these 6 classes mentioned in the Experimental Results section. After merging the dataset as shown in Figure 3.1, the data before feeding to the training pipeline is preprocessed thoroughly to fit in the framework which is discussed in the chapter later. Apart from all the dataset organization which was a 6 classes COCO format annotated dataset the number of the samples of the images in each class was studied thoroughly to gauge the class balance [46] [45] across the dataset. The number of training and validation samples given below show the number of the images originally in the dataset. Notably, the training samples, for the drop inlet class depict that there can be a reasonable class imbalance which means the ratio of the samples concerning the samples in the other classes could be less than or equal to 1. Like the drop inlet images are significantly less than the sample images in each of the other classes. This class imbalance [47] [52] [53] could hurt the performance of the model over training and evaluation such that the class with very less number of samples could be hard to classify. Therefore, to mitigate this

Table 3.1: EfficientDet scaling configuration

Model	Input size (R_{input})	Backbone network	BiFPN $\#channels$ W_{bifn}	BiFPN $\#layers$ D_{bifpn}	Box/class $\#layers$ D_{class}
D0 ($\phi = 0$)	512	$B0$	64	3	3
D1 ($\phi = 1$)	640	$B1$	88	4	3
D2 ($\phi = 2$)	768	$B2$	112	5	3
D3 ($\phi = 3$)	896	$B3$	160	6	4
D4 ($\phi = 4$)	1024	$B4$	224	7	4
D5 ($\phi = 5$)	1280	$B5$	288	7	4
D6 ($\phi = 6$)	1280	$B6$	384	8	5
D7 ($\phi = 7$)	1536	$B7$	384	8	5

issue, class balance comes in handy. To solve these issues, the plausible solutions are adding focal loss [69] or augmentation techniques referenced in the chapter Related Work Data Augmentation section. We necessarily try many online augmentations such as Random Rotate, Random Flip, Random Scale, Random Translate, and many more. But to augment 1 class amongst all the classes, offline augmentation, which means populating the data with more number of images as a part of the dataset such that the ratio of the class with the meager number of images approaches to 1 with respect to other classes in the database. Different results with various techniques are mentioned below. In this way, the data is structured into a particular format and stored on the disk. The data management visualization is important to understand the gravity of the concept. Hence, it is visualized in the Figure 3.1.

3.3 Network Architecture

The widely acknowledged compound scaling and the lightweight factor of the Google founded EfficientNet and EfficientDet architecture are accepted typically with an aim to build the framework customized applications like detection, classification, and segmentation as an end product ready to be deployed in real-time. However, the beauty of this architecture family is the fact that it is scalable detection architecture with both higher accuracy and better efficiency across a wide spectrum of resource constraints (e.g., from 3B to 300B FLOPs). The EfficientDet configuration over different models of the family is shown in Table 3.1 below. The paper [22] mentions

Table 3.2: EfficientDet Parameters

Model	Input size	#Params	FLOPs
EfficientDet-D0	512	3.9M	2.5B
EfficientDet-D1	640	6.6M	6.1B
EfficientDet-D2	768	8.1M	11B
EfficientDet-D3	896	12M	25B
EfficientDet-D4	1024	21M	55B
EfficientDet-D5	1280	34M	135B
EfficientDet-D6	1280	52M	226B
EfficientDet-D7	1536	52M	325B

how the other object detectors like single stage object detectors like SSD [12] and the anchor free generation [14] are efficient but they compromise accuracy which is a very important factor when it comes to the precise and correct bounding box prediction and its corresponding class label. EfficientDet architecture is light, efficient, and proven to be the state of the art object detector since it beats the YOLO-V4 [29] [13] and RetinaNet [69] in the terms of accuracy and is lighter when compared to the parameters of the other popular object detectors. Acknowledging the useful considerations regarding the EfficientDet from the description above, it is possible to build an end to end pipeline with fulfilling the requirements of detecting the object from the distance with high-resolution input without compromising the accuracy and efficiency. The Table 3.2 shows the different configurations of EfficientDet family. From the Table 3.2 and the Table 3.1, the EfficientDet-D0 looks the best fit as this was the lightest model with a decent input image resolution with 3M parameters weights configuration. Moreover, the motive behind selecting the lightest model was to deploy it on the drone mountable low power edge embedded device shortly. The current chapter focuses on how the EfficientDet-D0 is customized to fit in the pipeline for this research and also discusses briefly the key components of the architecture. The key components are 1) Backbone, 2) BiFPN Network,3) Class/Box prediction network

Backbone Network Previous works for the object detection scale the base network by using bigger backbone networks like ResNet [9] and VGGNet [8], using large

input resolution, or using multi-scale training to achieve high accuracy. However, these methods rely on scaling only a single dimension, which has inadequate effectiveness. Thus, to tackle the scaling method, and maintaining the efficiency, the EfficientNet family [44], show outstanding performance on image classification by jointly scaling the width, depth, and input image resolution. Thus, the EfficientDet architecture is the extension of this state of the art backbone network. Starting with the baseline EfficientNet-B0, the EfficientNet family scales up to B7 with the help scaling coefficients below:

$$\begin{aligned} depth : d &= 1.2^{\phi} \\ width : w &= 1.1^{\phi} \\ resolution : r &= 1.15^{\phi} \end{aligned} \tag{3.1}$$

Every EfficientDet-Dx detector architecture has a corresponding scaling of EfficientNet-Bx backbone where x ranges from 0 to 7. The main purpose of the extractor in this work is to serve the purpose of deep feature extraction followed by the Feature Pyramid network. The EfficientDet Architecture consists of two main components - Backbone + BiFPN network. As you can see in the figure, 3.2 the BiFPN Layer only interacts with the feature maps extracted by the backbone network at level 3-7 of the backbone network.

Feature Pyramid Network The Feature Pyramid network [70] [23] [24] is a medium for fusing the features extracted from the backbone. The object detector architectures mainly rely on the feature fusion techniques. There have been advancements from feature fusion to scaled feature fusion. The SSD [?] detector has its independent technique of fusing the first 4 layers of the feature network. The fusion technique changes with the difference in the scalability of the model. Like the NAS-FPN [70] [24] networks maintains a cross-scale feature fusion technique but it takes significant hours of training these bulky networks with a significant drop in efficiency.

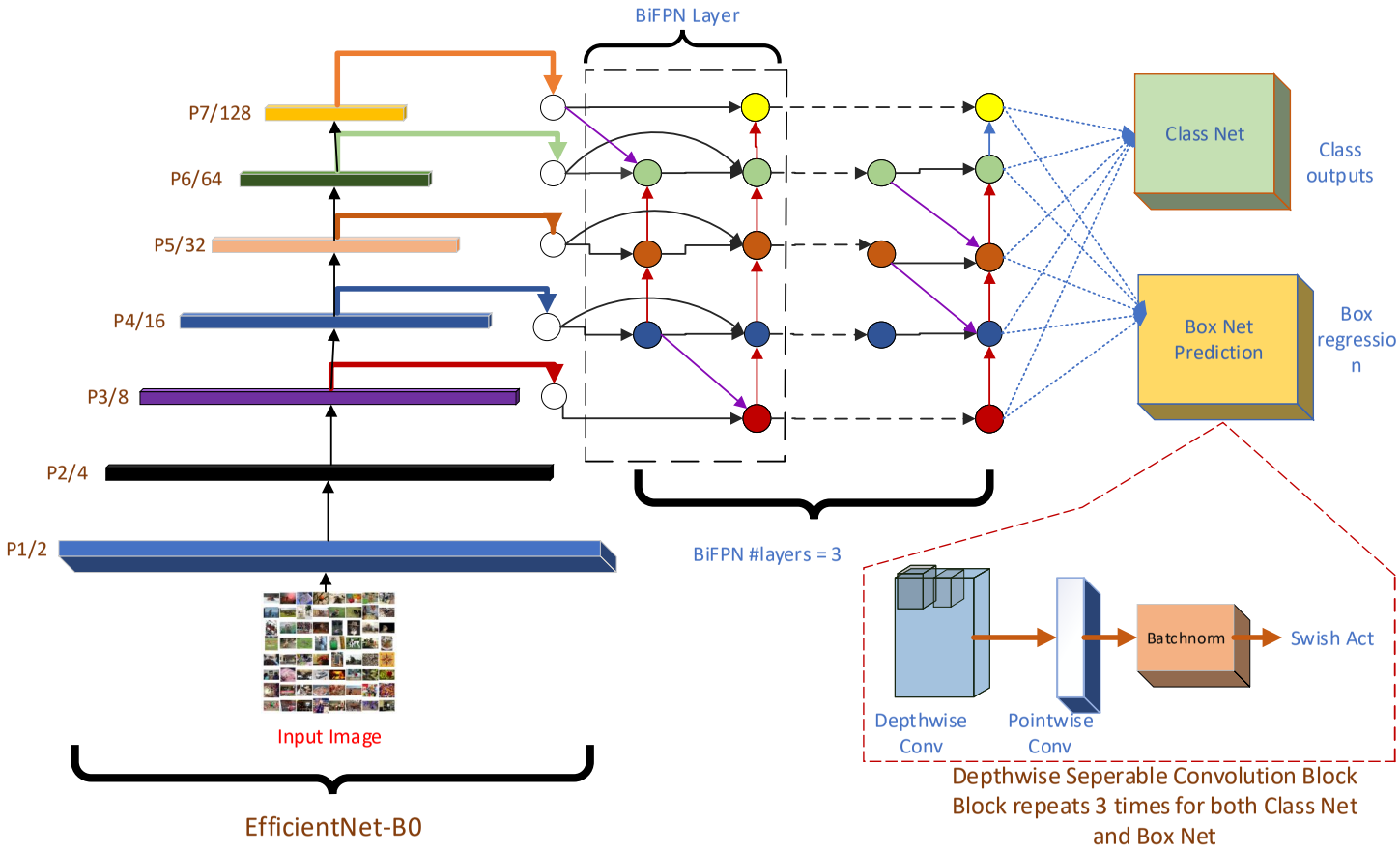

Figure 3.2: A detailed illustration of the custom EfficientDet-D0 architecture for multi-object detection. Consisting of a 7 stages scalable backbone EfficientNet-B0, and feature maps taken from 5 stages P3-P7 (denoted by different colors), and a BiFPN network with showing the multi-scale fusion and the BiFPN layer block is repeated 3 times and circles representing nodes.

In the same way, the EfficientDet detector relies on the multi-scale BiFPN network, which accounts for the top-down and bottom-up path feature network. It is clear in the Figure 3.2 that each bidirectional (top-down bottom-up) path is treated as one feature network layer, and repeat the same layer multiple times to enable more high-level feature fusion.

There are bottom-up and top-down connections between the feature maps at different levels. Thus, there could be a need to *Upsample* or *Downsample* the features. In Figure 3.3, the connection indicated by the red arrows pointing up between the nodes is the Upsample and the purple arrows represents the Downsample operation. Each Node inside a BiFPN layer can accept either 2 or 3 inputs and it combines them to produce a single output. Since the BiFPN Network consists of multiple BiFPN Layers the number of the BiFPN layers depends on the size of the EfficientDet. This scaling of the BiFPN based on the sizes is determined with the help of Compound Scaling mentioned below. In the block below 3.3, the output node like mentioned above is a way of summation of the feature input nodes. But before, the input nodes are added, it is made sure that they are of the same channel and size. In order to make the input channels equal to the output channels, a *Convolution2D* block for 1x1 convolution between the channels followed by a *Batchnorm2d* and *Activation* is implemented. Furthermore, in Figure 3.3, the nodes at the output layer are the fusion of all the input nodes. So, a fusion of all nodes is possible with [71] Depthwise Separable Convolution.

3.4 Compound scaling

The Deep Learning community has been contributing to scaling the large bulky networks and architecture into a lighter, and efficient framework being able to lower the computational cost. This also enables real-time visual analytics on the power resource-constrained devices. Thus the EfficientNet family has overpowered other state of the art networks because of its flexibility to scale the depth, width, and the

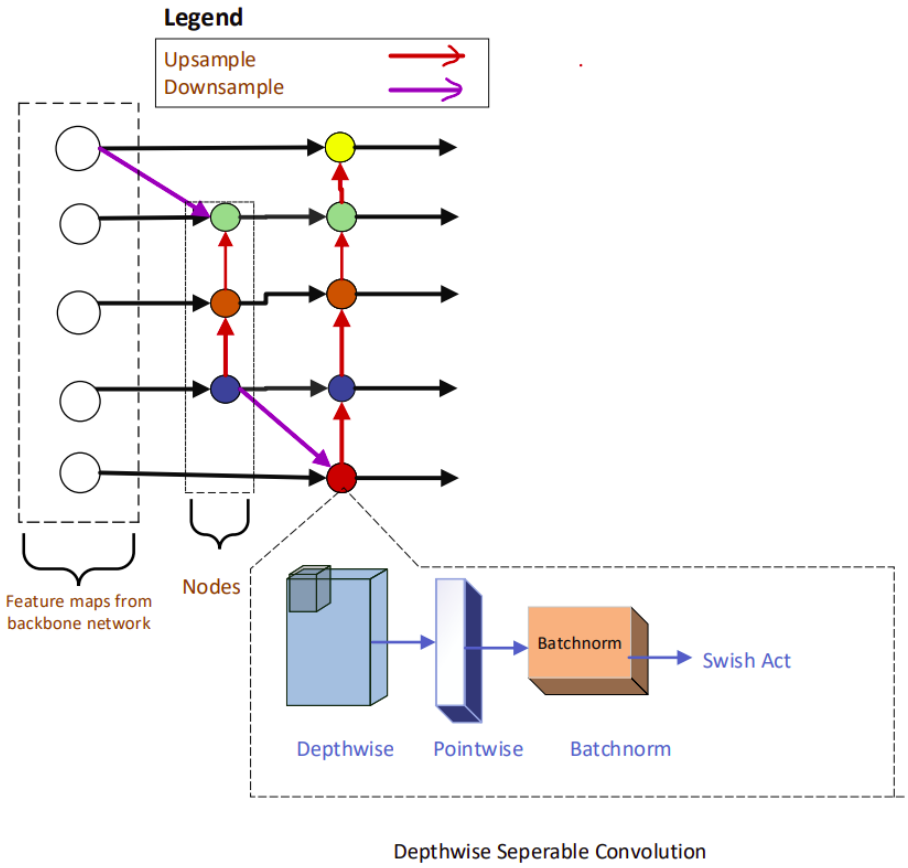

Figure 3.3: A detailed illustration of the BiFPN block of the architecture which represents Upsample and Downsample operations on the feature maps nodes. The circle components between the connections is the node

.

resolution network parameters with the help of a Grid search algorithm. However, the Efficientdet network jointly scales the depth, width, input resolution, the BiFPN network, and the class/box prediction network using a compound coefficient ϕ based on the Compound scaling method mentioned below. The formulation behind the compound scaling according to scalability of all the components for Depth, Width, input resolution, BiFPN layers, and class/box prediction network based on the respective model is as below.

Input Image Resolution The EfficientNet layers downsample the original input image resolution by 128 times. Thus, the input resolution of EfficientDet must be

divisible by 128, and is linearly scaled down as shown in Equation 3.2

$$R_{input} = 512 + \phi \cdot 128 \tag{3.2}$$

BiFPN Depth Scaling The BiFPN layers is linearly increased and approximated to start with the small integer. Hence, it starts with 3 for the D0 model as shown in the Equation 3.3.

$$D_{bifpn} = 3 + \phi \tag{3.3}$$

BiFPN Width Scaling The width or the number of channels of the BiFPN is decided on the heuristic grid search approach, picking the best value as 1.35 as the width scaling factor multiplied by 64 channels as shown in the Equation 3.4

$$W_{bifpn} = 64 \cdot (1.35)^{\phi} \tag{3.4}$$

Depth for Class/Box Network The scale of the repeated layers in the block for the class and the box prediction network are more likely scaled linearly using the Equation 3.5

$$D_{class} = 3 + \frac{\phi}{3} \tag{3.5}$$

3.5 Training setup

To integrate the aforementioned architecture customized according to the intended task, the Hyperparameters tuning is explored along with the following steps involved, First, The backbone EfficientNet-B0suitable for this research,is a pre-trained on the very popular and humongous 'Imagenet' dataset of 1000 classes. This provides well-trained weights to ensure smooth learning of the features of the dataset with a lower learning rate hyperparameter. Second, the BiFPN against the classical FPN in the context of our drop inlet localization and detection would stand helpful in the follow-

ing ways:

1. In the drop inlet detection application, features like grid orientation, brightness, and the surrounding concrete area should be taken into consideration. Hence, using multi-scale feature extraction can benefit from the accurate detection task.

2. Feature Pyramid Network (FPN)[23] uses a top-down approach to sum up multi-scale features that can be used for fusing multiscale extracted features. In the FPN, different scales do not necessarily contribute equally to the output features that can lead to some missing features in the drop-inlet detection process. thus the BiFPN would help in giving the features equal importance making the framework robust.

The minimum level for the input to the BiFPN layer is the 3^{rd} level of the Backbone network and the maximum level is the 7th level from the backbone network as shown in the Figure 3.2. For the training, the necessary *upsample* and *downsample* shown by the red and violet arrows in the 3.3 looks after the proper resampling of the feature maps. The EfficientDet-D0 according to the equation 3.3 has D_{bifpn} i.e., the BiFPN block is repeated 3 times is clearly mentioned in the 3.2. These features can now be fed into the class and box network. In this network, the class network is modified to include the number of classes in the labeled dataset. Since there is only 1 class in the terms of the object localization the pre-trained [33] [34] coco head is set to 1. Similarly, for the 6 classes, the head class net is modified to 6 with the help of Depthwise Separable Convolution [71] as shown in the Figure 3.2. The box net is responsible for the regression [72] of the bounding box coordinates. The output of this part is the anchors' generation, inspired by RetinaNet architecture style [69] takes place, where the number of scales and the anchors scales is set according to the resolution of the bounding box predictions desired. There are 9 anchors [14] [73] generated of 3 scales with 3 aspect ratios [(1.0, 1.0), (1.4, 0.7), (0.7, 1.4)] which is

selected as per the dataset. Moreover, the Focal Loss is employed to mitigate the class imbalance problem. A study on 2 different γ values of (1.5, 2) to learn its effect on training and evaluation. The Focal Loss [69] focuses on the hard and rate training samples which is the drop inlet class in our case.

CHAPTER 4: Effcient Onboard Execution on Edge Devices

While training a model for the customized dataset, and looking forward to evaluating on the test dataset, one needs to visualize the training data, clean it up, and train again unless the best results to try reducing the bias-variance tradeoff. After the model is trained to perfection it is ready for production. The production process involves 3 stages. First, converting the model trained in the choice of framework which is Pytorch [74] in the favor of the Thesis into an intermediate understandable machine learning graph, which is then converted to the optimized machine learning hardware language followed by some post-processing essentially based on the task and application. Thus, the machine learning hardware language is the pivotal aspect of this pipeline. For this, the very recent and advanced NVIDIA has served the inference [3] optimizations. The NVIDIA TensorRT [58] is an SDK for high-performance deep learning inference. It includes a deep learning inference optimizer and runtime that delivers low latency and high-throughput for deep learning inference applications. The core of NVIDIA TensorRT is a C++ library that facilitates high performance inference on NVIDIA GPUs. TensorRT takes a trained network, which consists of a network definition and a set of trained parameters, and produces a highly optimized runtime engine which performs inference for that network. The Figure 4.1 represents the actual conversion in simplified block diagram representation. The block indicates the conversion process and it also depicts that finally the TensorRT engine is sent to the Inference pipeline for actual evaluation and benchmarking.

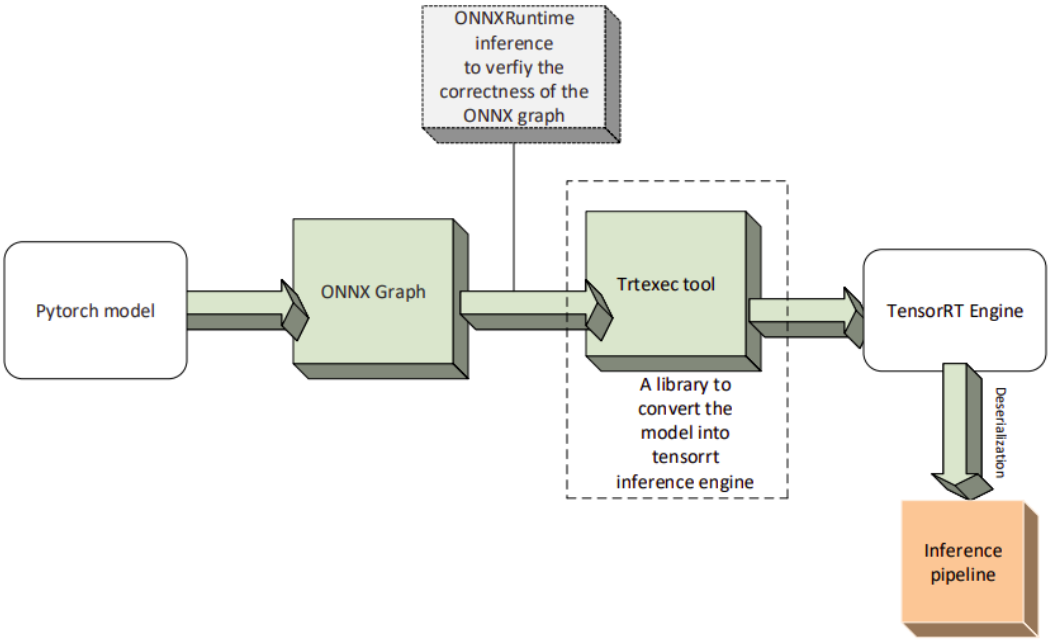

Figure 4.1: Model to TensorRT engine conversion process

4.1 Pytorch model to TensorRT conversion pipeline

ONNX The training framework is discussed in chapter 3 above. Starting with the deployment production process, the first step is converting the Pytorch model into the ONNX [75] graph format. ONNX stands for Open Neural Network Exchange. It is a compatible graph for the trained equivalent model in any framework of choice It is an open format built to represent machine learning models. In this study, the Pytorch ONNX [76] API is used to convert the perfectly trained model into an ONNX graph. The model conversion to ONNX is executed on the NVIDIA TitanV hardware.

ONNXRuntime The huge benefit of having a common format of the graph at run-time is that the software or hardware that loads the model at run time only needs to be compatible with ONNX. Thus, the ONNXruntime libraries were installed on the same hardware NVIDIA TitanV for the compatibility with the ONNX graph. To validate the ONNX model, there have been equal weightage to the onnxruntime [77] output tensors. These output tensors are compared to the Pytorch saved network

definition model. The output tensors generated by the runtime engine are pretty similar to the PyTorch model outputs which then confirms the deployment of the model on the edge device

Hardware module There has been tremendous research in the field of Machine Learning benchmarking performance describing the necessity of the Deep Learning hardware accelerators and low power devices to tackle the resource constraints and reducing the latency for the model kernel execution in real-time. Thus, the hardware chip manufacturing and design companies like NVIDIA, Intel, Xilinx, etc let the user avail the hardware inferencing on the specialized deep learning accelerators like GPUs and FPGAs. Thus, taking the maximum advantage of the available hardware resources suitable for the execution for our framework, considering the low power requirement and the inferencing capability, hardware module NVIDIA Jetson AGX Xavier is used for the inference and benchmarking performance as shown in Figure 4.3 below. The specs of the Jetson AGX Xavier is in the Table 4.1.

trtexec tool In this thesis, the framework is deployed converting the ONNX model in the TensorRT engine using the TensorRT Python API and trtexec tool. The trtexec tool [78] is a favorable library designed by the TensorRT committee [58] which enables to use the C++ backend for the conversion of the ONNX graph into a serialized TensorRT engine. It is a tool that has all the necessary components of the TensorRT library. It also helps in profiling, time tracing the layers, and quantizing the model to build the TensorRT engine.

TensorRT Engine After, the ONNX model is checked minutely to avoid errors if any, the model is converted to an engine optimized specifically to carry out the inference application. TensorRT achieves maximum inference throughput by generating an optimal runtime engine. The TensorRT enables to quantize the FP32 trained model into INT8 (8-bit integer) or FP16 (16-bit floating point) arithmetic quantized model. This decrease in precision can significantly speedup inference at a small cost

of accuracy. Other kinds of optimizations include minimizing GPU memory footprint [58] by reusing memory, fusing layers and tensors, selecting the appropriate data layers based on hardware, and so forth. Various customized plugins could be used to further optimize the process [58].

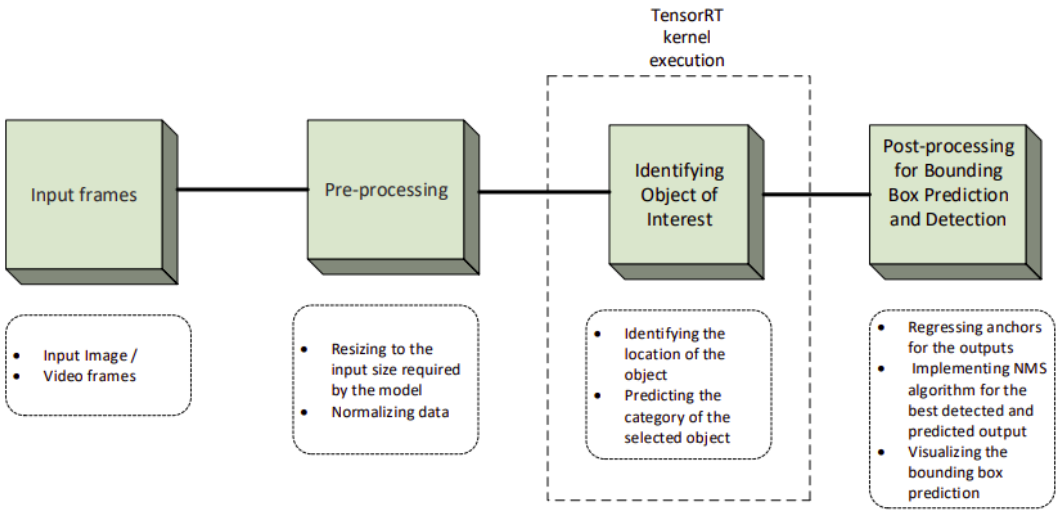

Figure 4.2: Visual Analytics Pipeline

4.2 Benchmarking pipeline

As we learned in the section above regarding the conversion of the model trained on the custom dataset to the TensorRT engine, it is crucial to know the workflow of how the TensorRT kernel for the object detection task is executed based on the engine file and what are the steps taken to run the kernel for the detection task and inference benchmarking on each of the processes. The experimental results in chapter 4 shows these numbers. In the Figure 4.2 below, each block speaks about the task involved. Hence, just to elaborate briefly, the description is as follows:

Pre-processing The engine might accept a particular format of the model to reach out to the best output. Also, the engine would only accept the input frame size for which the ONNX graph was converted. Hence, it is important to resize the input and then normalize it.

TensorRT Kernel execution The current capacity of this research is to execute the task-specific TensorRT kernel with accepting the preprocessed input data frames and to generate some intermediary output before finally visualizing the desired final output bounding box prediction. Hence, the inferencing benchmarking is the time taken by the kernel TensorRT engine to execute the frames per second. In this region particularly the engine uses the compressed model engine weights to identify the object of interest that is to localize the presence of the object and predicting its category to which it belongs. Hence, the TensorRT optimization in this research is only for the kernel execution part. While the pre-processing and post-processing is carried out using some native Pytorch and Python Numpy methods.

Post-processing The results generated as the outputs from the TensorRT engine framework can now be post-processed to generate the output in the form of a bounding box and the classification probability of the images or frames passed as an input to the framework. In this research, the Post-processing step was implemented with the help of native Pytorch Anchors and NMS algorithms.

Figure 4.3: Jetson AGX module

Table 4.1: Jetson AGX Device Configuration

GPU	512-core Volta GPU with Tensor Cores
CPU	8-core ARM v8.2 64-bit CPU, 8MB L2 4MB L3
Memory	32GB 256-Bit LPDDR4x | 137GB/s
DL Accelerator	(2x) NVDLA Engines

CHAPTER 5: EXPERIMENTAL RESULTS

In this section, the method to train and evaluate the model utilizing the EfficientDet-D0 architecture with a range of experiments observed in the research are discussed. The COCO API is integrated for further evaluation on the custom dataset i.e., inclusive of highway asset item drop inlet and the subset of COCO classes. Then, the respective quantitative and qualitative experiment results are presented. The Experimental results also show the transition from the single-stage i.e., Object localization to multi-stage scaled object detector. To achieve the inference on the edge devices, FPS performance numbers are reported for the TensorRT engine equivalent of the trained Pytorch model.

5.1 Object Localization

Dataset. As the name suggests the objective of the training setup was to first analyze the training model based on Leidos claimed annotated highway asset item dataset i.e., the single object drop inlets class, each image having 1 or 2 drop inlets in each image and thus the corresponding bounding box annotations for the respective item in the image were fed to the model for the training and evaluation. The purpose of the training was to localize the drop inlet as a preliminary approach. For this setup, the database comprised of nearly 612 images for the training and 172 images for the validation while 87 test images for evaluating the test accuracy.

Training. The training framework was built on a EfficientDet-D0 model pre-trained on the COCO2017 dataset which has 80 classes in total. The HeadNet part of the model i.e, the class net and the box net prediction network was modified to let the COCO head of the pre-trained model on 80 classes change to the 1 class in the

training setup. The total number of images for the training is 612 images while for the validation are 180 images and 88 test unseen images. The model was trained using the Adam weight optimizer [79].In the training setup since the drop inlet localization was altogether a newer object for the pre-trained model, we ran the model with a smaller learning rate $1e-5$ for 10 warmup epochs which was linearly increased to 0.05 followed by training on additional 80 epochs with a cosine scheduler [80] which reduces and adjusts the learning rate by a factor of 10 after every 30 epochs. Furthermore, the evaluation and the test accuracy are discussed below

Evaluation The single object localization was evaluated on the metrics based on class scores and mIOU when the dataset had intended to have groundtruth annotations. The results in the Table 5.2 indicates quantitative results for both *val* and *test* images. The images were also evaluated on the zoom-in augmentation which means the original test image size was resized to a small resolution and the aspect ratio of the background zoom image was maintained like that of the original image. The zoom augmented image could be referred to as the image in the right column in the Figure 5.2.

Testing. Similarly, for the test unseen images, the localization was observed. The test evaluation was observed for the test images having groundtruth annotations and thus average of IOU could be calculated over the test images. The IOU is calculated with the help of the equation 5.1, where B_{gt} is the groundtruth bounding box coordinates while B_p is the predicted bounding box coordinates on evaluation. The results in the Table 5.2 shows the averages over the IOU of test images in the dataset. Taking the industrial collaboration project point into the consideration, the results were also analyzed based on the standard deviation (STD) and the variance factor on the test images. The testing for baseline images is the normal input image

Table 5.1: Evaluation on Test set for drop inlet class

Class label	AP	AP^{50}	AP^{75}	AP^{S}	AP^{M}	AP^{L}	AR
drop inlet	74.6	96.5	88.4	-	-	74.6	78.7

Table 5.2: Bounding box prediction evaluation metrics on drop inlet asset dataset

Experiment	Image size	Baseline	Zoom$_{aug}$
mIOU *test*	800 ×600	88.6%	89.6%
mIOU *val*	800 ×600	91.4 %	92.3%
Standard deviation *test*	800 ×600	7.32%	4.47 %
Variance *test*	800 ×600	0.535%	0.200 %

with no augmentations. The testing is also done on the Zoom augmented images where the test images are augmented so that the asset item in the image could be visualized at a distance. Table 5.2 shows that the IOU increases with the increase in the pixel area of the image which validates the scaling capability of the model.

$$IOU_{B_{gt},B_p} = \frac{AreaofOverlap(B_{gt} \cap B_p)}{AreaofUnion(B_{gt} \cup B_p)} \tag{5.1}$$

5.2 Multi-class detection

The results for this section are based on fine-tuning the 6 classes i.e., the custom dataset on the COCO pre-trained model. The 6 classes account for drop inlet, person, car, bus, train and truck.

Dataset. Object detection is a famous Computer vision method, an algorithm that demands rigorous work to execute. The data being the pillar of the algorithm, open-source labeled benchmarks specialized for object detection tasks are widely available such as PASCAL-VO2017C Challenge[16] and COCO2017 dataset [15]. The COCO2017 is a database of 80 classes based on common objects with common context in the surrounding with a variety of training and validation images, while the PASCAL-VOC [16] 2007 consists of 20 classes. Thus, taking the advantage of the abundant open-source labeled dataset, it was possible to create a merged custom dataset of 6 classes including the highway asset drop inlet merged with the 5 different COCO classes namely bus, car, person, truck, and train. The reason behind selecting these classes was the relevancy to the highway scene and make more sense in the

Table 5.3: Number of labeled Train and validation images across the different classes

Sr.No	Class label	#*Train* samples	#*val* samples
1	*drop inlet*	612	172
2	*bus*	3953	190
3	*car*	12252	536
4	*person*	64116	2694
5	*train*	3589	158
6	*truck*	6127	251

real-world. The number of the samples per class is demonstrated in the Table 5.3 2017

Training. The training model architecture is the lightest model of the EfficientDet family. The EfficientNet[44] being the backbone of the EfficientDet family, is already trained on the ImageNet dataset[**?**] and a selective best suited pre-trained model already on the COCO dataset is trained for the multi-object detection. Moreover, pre-trained [33] [34] weights on MS COCO[15] are converted to Pytorch have been utilized as initial weights. Furthermore, all layers are trained, and the weights of the layers are not frozen. During training before each image is fed to the network, random resizepad is used to resize and pad the input images to the desired resolutions based on the current EfficientNet-B0 model i.e., 512*512 input resolution while the random horizontal or random vertical flip, random scale, and random aspect ratio and Random Zoom are some of the advanced online augmentation techniques applied on the dataset to serve the training in the best possible learning of the model. Color jitter was also used to randomly change the brightness, contrast, saturation, and hue of the RGB channels using principle component analysis [6]. The images are then normalized using per channel mean and standard deviation. The model was trained using Stochastic Gradient Descent [81] with a weight decay of $4e-5$ and momentum of 0.9 with optimizer epsilon of $1e-3$. The weights were initialized using the variance scaling method underwent five warm-up epochs with a learning rate of $1e-4$ that increased linearly until it reached 0.012. The network was then trained for an

additional 100 epochs and followed the cosine annealing [80] step decay learning rate scheduler that reduces the learning rate by a factor of 10 every 30 epochs. For the loss function, the smooth l1 loss as the box regression loss and focal loss 5.2 were employed[69] [82] to mitigate the class imbalance effect for the classification loss. The quantitative and qualitative results below show the improvised effect using focal loss with $\alpha = 0.5$ and $\gamma = 1.5$ and $\alpha = 0.5$ and $\gamma = 2$. However, the evaluation based on training the model with Cross Entropy Loss other than the Focal Loss is as given below.

$$FL_{p_t} = -\alpha(1 - p_t)^{\gamma} \log(p_t) \tag{5.2}$$

Evaluation For the results, average precision and recall scores is reported: AP (mean of AP scores at IOU = 0.50, 0.55,$\cdots$, 0.90, 0.95), AP^{50} (AP at IOU = 0.50), AP^{75}, AP^{M} for medium objects, AP^{L} for large objects, and AR (mean of recall scores). Also, the results varying with increasing the gamma values are shown. There is one more Table showing the per-class accuracy of the samples in the images. Since, there was no availability of COCO *test-dev*, the results are based on *val* set. Since the custom dataset was split into the test set, the test-dev for only 1 class drop inlet

Table 5.4: Per class accuracy on custom COCO val dataset.

Class label	mAP	AP^{50}	AP^{75}	AP^{S}	AP^{M}	AP^{L}	AR
evaluation on custom dataset *val* images for model trained on 6 classes							
drop inlet	74.3	97.6	83.6	-	-	74.3	80
bus	59.3	74.4	66.7	13.1	37.6	79.3	50.1
car	28.0	48.3	28.3	12.5	52.3	63.6	14.5
person	46.1	72.8	48.3	18.8	57.0	73.9	18.4
train	61.8	83.0	70.5	22.2	26.8	66.7	60.9
truck	28.8	46.1	31.6	9.0	30.3	47.7	30.4
evaluation on COCO *val* images for model trained on COCO 80 classes							
bus	59.7	74.0	67.4	7.8	38.5	79.1	49.1
car	26.9	47.3	27.5	11.8	50.1	63.6	13.4
person	43.7	70.8	45.3	17.5	53.5	71.3	17.9
train	62.5	83.6	71.4	23.2	32.7	66.9	59.7
truck	29.5	48.0	30.7	8.2	26.5	52.2	30.5

could be calculated. The Table 5.5 refers to the models trained within this study with $\gamma = 1.5$ and $\gamma = 2$ with $\alpha = 0.25$ as in the equation 5.2. Since the training carried on with $\gamma = 2$ showed better results on evaluation it was considered as the perfectly trained model for further evaluation. To study, how much accuracy AP does each class category contributes, the results were explored like in the Table 5.4. For each category in the Table 5.4, the input image for the evaluation is 512 and the experiment is conducted into 2 parts. The first 6 rows are for the evaluation on the custom *val* dataset i.,e, the merged dataset of subset of COCO and the drop inlet custom class. While, the second part of the Table 5.4 is explored to validate the consistency of the custom model compared to the COCO pre-trained EfficientDet-D0 model. On observing the Table 5.4 , for instance, though the number of Person class in the *train* set was the highest as in the Table 5.3 the accuracy for that particular class is lesser than few other classes. Hence, on examining the results for the pre-trained model evaluation, it could be confirmed that the results for the custom training are consistent with the COCO dataset trained model. It gives the reasoning that since the custom model is also a fine-tuned model from the COCO pre-trained model, the trend of the accuracy is the same. Since, the *test*set was possibly available for the drop inlet class, we evaluate the *test* data for drop inlets for the model trained on 6 classes. The results are as shown in the Table 5.1. However, the results also include evaluation on training based on using no Focal Loss where only the cross entropy loss was used for the classification loss. The Table 5.5 shows the 3rd row as using cross entropy over focal loss for a deep ablation study. The FL denotes training and evaluation based on Focal Loss while CE represents Cross Entropy in that case.

Table 5.5: Evaluation metric results on custom val dataset.

Method	Input size	mAP	AP^{50}	AP^{75}	AP^{S}	AP^{M}	AP^{L}	AR
FL ($\gamma = 1.5$)	512	48.89	69.98	54.1	13.1	40.62	66.9	42.2
FL ($\gamma = 2$)	512	49.75	70.5	54.9	15.0	41.0	67.4	42.4
CE	512	47.75	68.34	53.64	12.76	40.03	65.98	41.76

5.3 Qualitative Results

In this section, qualitative results for the Object localization on the Leidos Drop Inlet Dataset is shown and also the multi-object detection with all the different classes of the merged database is shown in the figures below.

5.3.1 Qualitative Results of IOU metric on drop inlet

This is a preliminary qualitative result where the IOU is defined as a metric to prove that the model localizes the object correctly in the image. The IOU is the overlap between the groundtruth and the predicted bounding box by the model architecture.In the Figure 5.1 and Figure 5.2. The images are *test* images with groundtruth annotations represented by the green bounding box, while the object localized is shown in the red bounding box. The 2 figures are represented with a variety of background i.e., highway concrete, and also with a grass background to portray the reality of the images concerning the highway environment. Starting with, the images on the left are the baseline images as part of the dataset. The column on the right shows the zoom out images with the object and the corresponding localization. The images are augmented as zoom out to validate that the model can also localize the object from distance and when it is not closer to the camera perspective. It is noticeable that the model still performs well and does not lose the IOU metric even if the image is zoom out.

5.3.2 Qualitative results on random images with no groundtruth annotations

This is the first and foremost basic result evaluated on the random unseen *test* images i.e., with no groundtruth annotations to check if the model is robust enough to localize any drop inlet images. The image below shows the variety of drop inlet images with a variety of concrete and drop inlet grids colors and textures. The confidence score i.e., the probability of the localization and the presence of the asset image is shown as the metric in the Figure 5.3

Figure 5.1: Qualitative results of object localization IOU for drop inlet

5.3.3 Qualitative Results of Multi-object Detection

The multi-object detection results based on the classification score i.e., the probability of the correctly classified object in the model with respect to other classes in the image is given. The results are based on the custom *test* images which are customised and self created to involve the drop inlet along with other classes in the dataset. The qualitative results show the baseline model when the training is conducted on the images as shown in the Table 5.3.The figure on the left in the Figure 5.4 shows the evaluation based on the training conducted under $\gamma = 1.5$ of the Focal Loss as mentioned in the Training section above. The other important observation is when we tackle the dropped accuracy of the model with the help of Focal Loss which deals with applying more weight on the rare and hardly classified examples. The

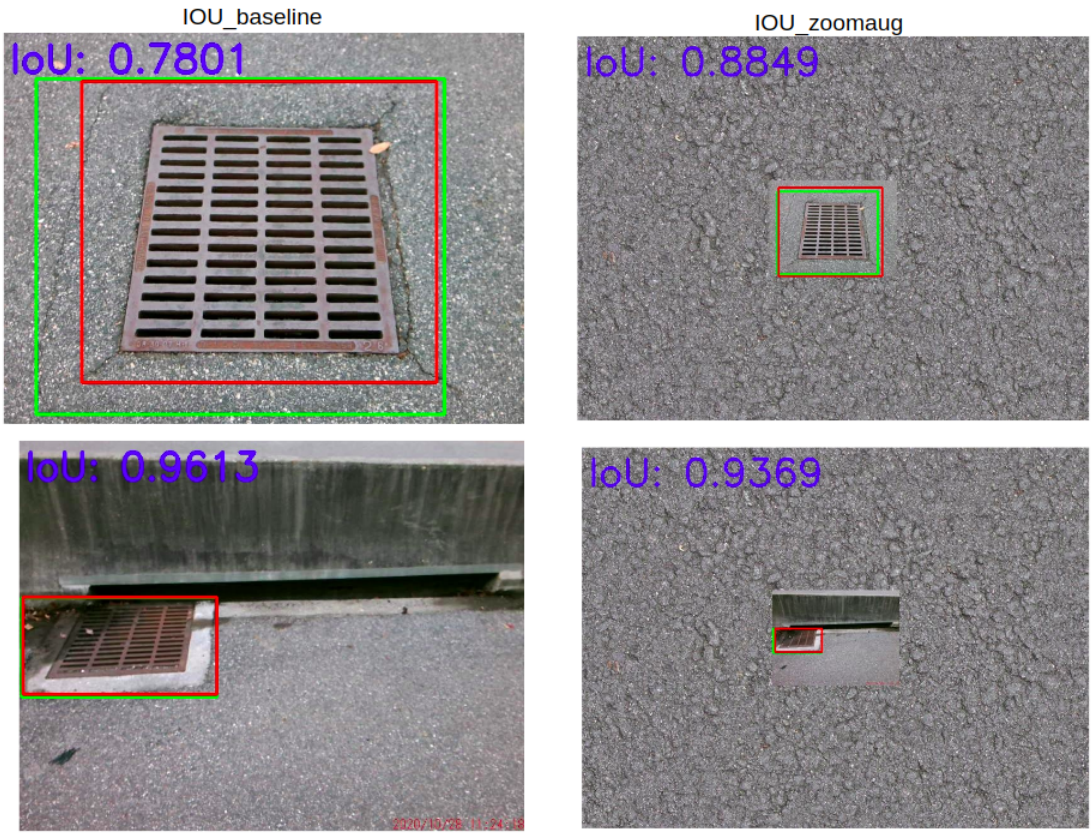

Figure 5.2: Qualitative results of object localization IOU for drop inlet with concrete background on zoom image

image on the right in the 5.4 shows well that how the rare drop inlet class i.e., the class having fewer images are classified as more weightage to the hard and rare class. Thus, we see an improved detection over implementing focal loss with $\gamma = 2$ value in the Focal loss function. The Figure 5.5 also shows result based on the test image which is customized to include all the categories of the dataset including highway drop inlet and few more categories.

5.4 Inference on mobile edge embedded platform

The benchmarking of the model on the edge devices is the most important step to validate the efficiency of the lightweight model under resource constraints. Thus, after the model is readily available with the accurate hyperparameters tuning and validated as a part of evaluation on a massively powerful GPU server, it is possible to

Figure 5.3: Qualitative results showing the basic localization on random unseen images with variety of drop inlet

deploy the model to realize it in real-time on edge platforms. Thus, the results below show the benchmarking [3] performance so that it can be validated for the future onboard drone mountable processing implementation.

5.4.1 Inference Benchmarking

The precise and accurate evaluated Pytorch model is converted to an onnx model as mentioned in the chapter above, and then it is converted to a hardware-specific engine file that helps in inference acceleration. The optimization and Deep Learning Hardware accelerator used in this study are namely, the NVIDIA TensorRT library and NVIDIA Jetson AGX device. The reason why it could be benchmarked on low power devices is the availability of the model quantized into FP16 weights. The model is trained with FP32 weights on the Lambda Quad server with 4 NVIDIA TitanV

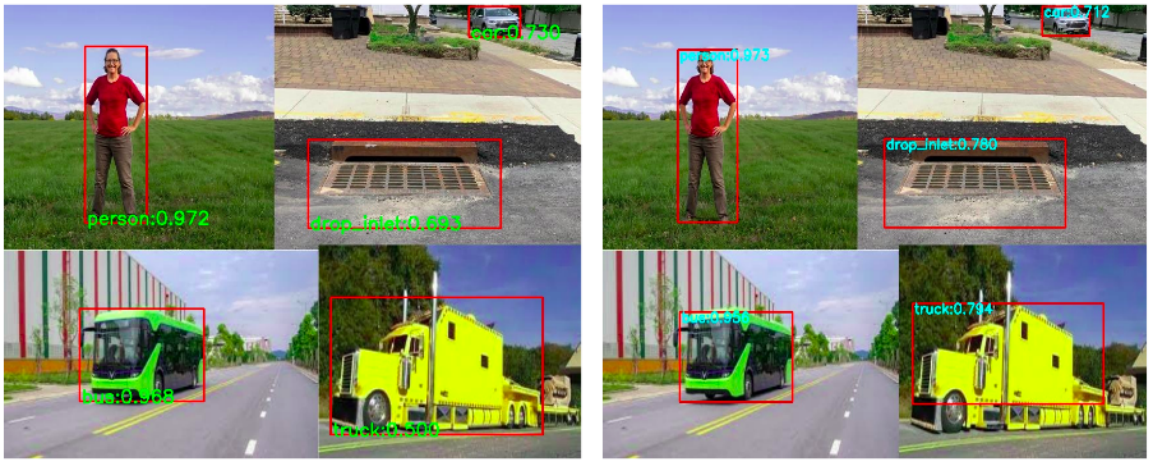

Figure 5.4: Multi-object detection in a scene

.

Table 5.6: Inference Benchmarking on TensorRT kernel

Model	Batch size	Input size	FLOPS	#Params	Runtime	FPS
Native FP32	1	512	2.5B	3M	228.83ms	4.37
FP32 TRT	1	512	2.5B	3M	59.6ms	16.78
FP16 TRT	1	512	2.5B	3M	19.78ms	50.55

devices. The conversion and the benchmarking was implemented in the TensorRT Python API. Table 5.6 below shows the difference in the FPS performance compared to the native Pytorch implementation. Table 5.6 below shows the average inference time required by the model to preprocess the input image and execute the engine to generate the output tensors of the model. However, as shown in the Figure 4.2, the post-processing step is necessary for the visualization of the evaluation, the results below in the Table 5.6 does not account for the runtime latency for the post-processing since that would be explored in the near future.

5.4.2 Challenges in the current pipeline

The current challenges in the pipeline are, as shown in the qualitative results, the probability i.e., the class confidence score of the customized data class drop inlet is less than the confidence of the other classes in the dataset and fails to predict some drop inlets in the multi-object scene. One more challenge is sometimes the model still fails to predict/detect some objects from a distance since the training was conducted

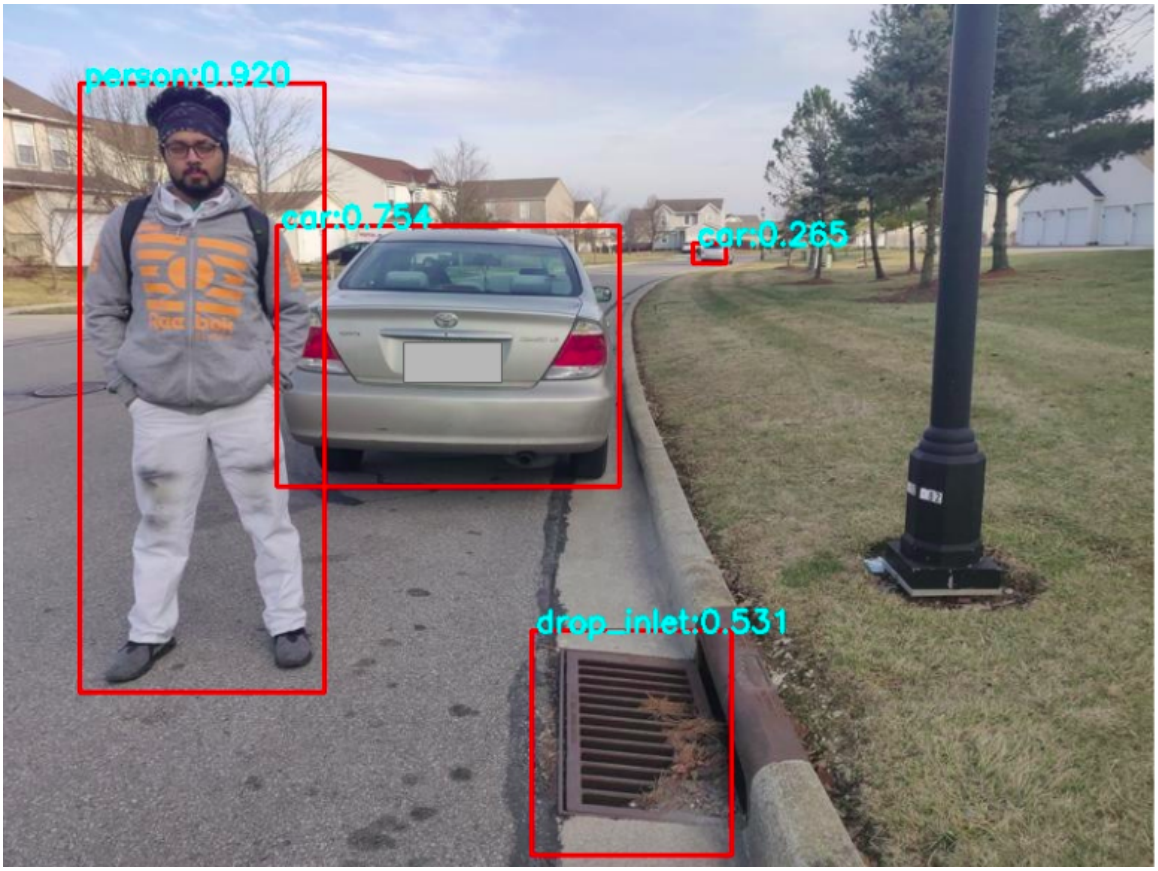

Figure 5.5: A custom test image including drop inlet and relevant objects where the model detects the object with their classification score probability

.

on the COCO dataset images and the drop inlet images which are captured from a close camera perspective. The other challenge lies in the validation of the frames on the NVIDIA Jetson AGX Xavier device. There is a significant accuracy drop observed while running evaluation with TensorRT execution kernel on the Jetson edge device when compared to the evaluation results on the Lambda Quad server.

CHAPTER 6: CONCLUSIONS AND FUTURE WORK

6.1 Conclusion

In this paper, a scalable and efficient way of integrating the EfficientDet-D0 architecture for the highway asset detection is presented. The study shows that how a subset of COCO data can be merged along with the custom annotated dataset under the resource constraints. This merging technique can be useful in the highway assessment when there is an interaction of common classes and the highway asset item. It also proves that the network architecture is robust to unseen test images. It localizes and detects with minimum negligible false detections. The research also proposes a proof of concept that how an efficient model can be quantized to FP16 weights model and further can be deployed in the real-time environment since the model had showed significant benchmarking performance on the low-power NVIDIA Jetson AGX Xavier.

6.2 Future Work

The thesis has the potential to be continued in the future. In this research study, the focus was on creating lightweight highway asset detection. The current architecture can be expanded to integrate multi-class multi-level asset items and evaluate further on. The study was a preliminary proof of concept for the Leidos and VDOT on how the highway assets can be assessed based on AI and Deep Learning algorithms. However, nothing is preventing modifying the model to identify and detect the defects of the highway assets with predicting the degree of defects in the asset item [21]. Moreover, the research does not only limit to detection and its evaluation, but it has a great vision ahead with deploying the perfect model on the drone for real-time edge

video analytics. The current inference benchmarking in the Python API can rather be realized into a low-level C++ language API for the best hardware optimization and increased accuracy. The further benchmarking runs can involve the post-processing of predicted the bounding boxes as the part of the custom plugin in the TensorRT library so that the Inference can be calculated for an end to end pipeline i.e., input the image, preprocessing, deserializing the model, and then post-processing based on the advantage of TensorRT optimization for the real-world applications.

REFERENCES

[1] C.-H. Wei and Schonfeld, "A research framework for cost-effective highway maintenance planning," in *Proceedings /Public Works Management Policy, 2(4)*, pp. 340–349.

[2] K. Ananraya and V. Ammarapala, "The development of highways assets management system," in *2010 7th International Conference on Service Systems and Service Management*, pp. 1–6, 2010.

[3] V. J. Reddi, C. Cheng, D. Kanter, P. Mattson, G. Schmuelling, C.-J. Wu, B. Anderson, M. Breughe, M. Charlebois, W. Chou, R. Chukka, C. Coleman, S. Davis, P. Deng, G. Diamos, J. Duke, D. Fick, J. S. Gardner, I. Hubara, S. Idgunji, T. B. Jablin, J. Jiao, T. S. John, P. Kanwar, D. Lee, J. Liao, A. Lokhmotov, F. Massa, P. Meng, P. Micikevicius, C. Osborne, G. Pekhimenko, A. T. R. Rajan, D. Sequeira, A. Sirasao, F. Sun, H. Tang, M. Thomson, F. Wei, E. Wu, L. Xu, K. Yamada, B. Yu, G. Yuan, A. Zhong, P. Zhang, and Y. Zhou, "Mlperf inference benchmark," 2020.

[4] M. Najafabadi, F. Villanustre, T. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *Journal of Big Data*, vol. 2, 12 2015.

[5] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," *EURASIP Journal on Advances in Signal Processing*, vol. 2016, 05 2016.

[6] 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*, NIPSâ12, (Red Hook, NY, USA), p. 1097â1105, Curran Associates Inc., 2012.

[7] 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.

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

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

[10] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. DollÃ¡r, "Designing network design spaces," 2020.

[11] Y. Wu, L. Liu, C. Pu, W. Cao, S. Sahin, W. Wei, and Q. Zhang, "A comparative measurement study of deep learning as a service framework," *IEEE Transactions on Services Computing*, pp. 1–1, 2019.

[12] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *Lecture Notes in Computer Science*, p. 21â37, 2016.

[13] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020.

[14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.

[15] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. DollÃ¡r, "Microsoft coco: Common objects in context," 2014.

[16] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, pp. 98–136, Jan. 2015.

[17] 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.

[18] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," *CoRR*, vol. abs/1612.01105, 2016.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[20] Z. Zheng, Y. Wei, and Y. Yang, "University-1652: A multi-view multi-source benchmark for drone-based geo-localization," 2020.

[21] R. Fan, M. J. Bocus, Y. Zhu, J. Jiao, L. Wang, F. Ma, S. Cheng, and M. Liu, "Road crack detection using deep convolutional neural network and adaptive thresholding," *CoRR*, vol. abs/1904.08582, 2019.

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

[23] 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.

[24] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," *CoRR*, vol. abs/1803.01534, 2018.

[25] S. Bang, S. Park, H. Kim, Y.-s. Yoon, and H. Kim, "A deep residual network with transfer learning for pixel-level road crack detection," 07 2018.

[26] L. Zhang, F. Yang, Y. Daniel Zhang, and Y. J. Zhu, "Road crack detection using deep convolutional neural network," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3708–3712, 2016.

[27] B. Akarsu, M. KARAKÃSE, K. PARLAK, E. Akin, and A. SARIMADEN, "A fast and adaptive road defect detection approach using computer vision with real time implementation," *International Journal of Applied Mathematics, Electronics and Computers*, pp. 290–290, 12 2016.

[28] H. Maeda, Y. Sekimoto, T. Seto, T. Kashiyama, and H. Omata, "Road damage detection and classification using deep neural networks with smartphone images: Road damage detection and classification," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, 06 2018.

[29] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.

[30] D. Arya, H. Maeda, S. K. Ghosh, D. Toshniwal, A. Mraz, T. Kashiyama, and Y. Sekimoto, "Transfer learning-based road damage detection for multiple countries," 2020.

[31] M. Skibniewski and C. Hendrickson, "Automation and robotics for road construction and maintenance," *Journal of Transportation Engineering-asce - J TRANSP ENG-ASCE*, vol. 116, 05 1990.

[32] M. Heidarysafa, K. Kowsari, D. E. Brown, K. J. Meimandi, and L. E. Barnes, "An improvement of data classification using random multimodel deep learning (RMDL)," *CoRR*, vol. abs/1808.08121, 2018.

[33] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," 2018.

[34] T. . W. Weiss, K.and Khoshgoftaar, "A survey of transfer learning," *Journal of Big Data*, 2016.

[35] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Transfer learning for time series classification," *2018 IEEE International Conference on Big Data (Big Data)*, Dec 2018.

[36] R. Pilarczyk and W. Skarbek, "On intra-class variance for deep learning of classifiers," *Foundations of Computing and Decision Sciences*, vol. 44, no. 3, pp. 285 – 301, 01 Sep. 2019.

[37] K. Trapeznikov, V. Saligrama, and D. A. Castañón, "Cost sensitive sequential classification," *CoRR*, vol. abs/1205.4377, 2012.

[38] T. Ishida, G. Niu, and M. Sugiyama, "Binary classification from positive-confidence data," in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.

[39] I. L. M. Oquab, L. Bottou and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[40] A. Kolesnikov, X. Zhai, and L. Beyer, "Revisiting self-supervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[41] L. Ren, J. Lu, Z. Wang, Q. Tian, and J. Zhou, "Collaborative deep reinforcement learning for multi-object tracking," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[42] Y. Guo, H. Shi, A. Kumar, K. Grauman, T. Rosing, and R. Feris, "Spottune: Transfer learning through adaptive fine-tuning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[43] 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.

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

[45] K. T. B. R. e. a. Leevy, J.L., "A survey on addressing high-class imbalance in big data," *Big Data 5*, vol. 106, pp. 5–42, November 2018.

[46] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, p. 249â259, Oct 2018.

[47] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[48] M. A. Arefeen, S. T. Nimi, and M. S. Rahman, "Neural network based undersampling techniques," 2019.

[49] S. Salman and X. Liu, "Overfitting mechanism and avoidance in deep neural networks," 2019.

[50] X. Ying, "An overview of overfitting and its solutions," *Journal of Physics*, 2019.

[51] R. Webster, J. Rabin, L. Simon, and F. Jurie, "Detecting overfitting of deep generative networks via latent recovery," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[52] H. Taherdoost, "Sampling methods in research methodology; how to choose a sampling technique for research," *International Journal of Academic Research in Management*, vol. 5, pp. 18–27, 01 2016.

[53] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, p. 321â357, Jun 2002.

[54] P. Cunningham and S. J. Delany, "k-nearest neighbour classifiers: 2nd edition (with python examples)," 2020.

[55] A. X. M. Chang, A. Zaidy, V. Gokhale, and E. Culurciello, "Compiling deep learning models for custom hardware accelerators," *CoRR*, vol. abs/1708.00117, 2017.

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

[57] J. Liao, L. Cai, Y. Xu, and M. He, "Design of accelerator for mobilenet convolutional neural network based on fpga," in *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, vol. 1, pp. 1392–1396, 2019.

[58] https://docs.nvidia.com/deeplearning/tensorrt/api/index.html.

[59] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, (New York, NY, USA), pp. 1–12, ACM, 2017.

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

[61] S. Ayubian, S. Alawneh, and J. Thijssen, "Gpu-based monte-carlo simulation for a sea ice load application," 07 2016.

[62] P. Perera and V. M. Patel, "Learning deep features for one-class classification," vol. abs/1801.05365v2, 2019.

[63] J. Brownlee, "A gentle introduction to object recognition with deep learning," 2019.

[64] W. Ouyang, X. Wang, C. Zhang, and X. Yang, "Factors in finetuning deep model for object detection," 2016.

[65] https://labelbox.com/.

[66] https://github.com/cocodataset/cocoapi/tree/master/PythonAPI.

[67] https://github.com/ashnair1/COCO-Assistant.

[68] https://github.com/immersive-limit/coco-manager.

[69] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *CoRR*, vol. abs/1708.02002, 2017.

[70] G. Ghiasi, T. Lin, R. Pang, and Q. V. Le, "NAS-FPN: learning scalable feature pyramid architecture for object detection," *CoRR*, vol. abs/1904.07392, 2019.

[71] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

[72] Y. He, C. Zhu, J. Wang, M. Savvides, and X. Zhang, "Bounding box regression with uncertainty for accurate object detection," 2019.

[73] J. Wang, K. Chen, S. Yang, C. C. Loy, and D. Lin, "Region proposal by guided anchoring," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[74] https://pytorch.org/docs/stable/torch.html.

[75] https://onnx.ai/.

[76] https://pytorch.org/docs/stable/onnx.htmltorch-onnx.

[77] https://github.com/microsoft/onnxruntime.

[78] https://github.com/NVIDIA/TensorRT/tree/master/samples/opensource/trtexec.

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

[80] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," 2017.

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

[82] D. Sarkar, A. Narang, and S. Rai, "Fed-focal loss for imbalanced data classification in federated learning," 2020.