Empirical Study of Pedestrian Detection using Deep Learning

by

Ahmet Kapkic

Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master

of

Computing and Information Systems

YOUNGSTOWN STATE UNIVERSITY

May, 2021

Empirical Study of Pedestrian Detection using Deep Learning

Ahmet Kapkic

I hereby release this thesis to the public. I understand that this thesis will be made available from the OhioLINK ETD Center and the Maag Library Circulation Desk for public access. I also authorize the University or other individuals to make copies of this thesis as needed for scholarly research.

Signature:

Ahmet Kapkic, Student Date

Approvals:

Dr. Yong Zhang, Thesis Advisor Date

Dr. John Sullins, Committee Member Date

Dr. Feng George Yu, Committee Member Date

Dr. Salvatore A. Sanders, Dean of Graduate Studies Date

DEDICATION

To my parents, for their everlasting support;

To my professors who guided me not only with words, but also with their actions.

And to the memory of Prof. Coskun Bayrak.

ABSTRACT

Detecting pedestrians in public settings is a major research topic in both Computer Vision and Artificial Intelligence communities. It has found applications in a wide range of areas such as vehicle driving with autonomous control systems, video surveillance, and navigating robots, etc. Over the past decade, a great progress has been made in the development of efficient algorithms and the availability of large-scale data set, especially the advancement of Deep Learning method. In this thesis, the performance of a few state-of-the-art methods were evaluated by conducting empirical experiments with different settings and dataset configurations on pedestrian detection. The experiments were carried out using several Deep Learning models in the framework of both baseline and special configurations, including the Faster R-CNN, Mask R-CNN, and Cascade R-CNN methods. The experimental results show that the Mask R-CNN with a ResNet50 barebone yields the best performance in terms of its larger AP improvement and fewer resource requirement. This work provides a solid foundation upon which more sophisticated comparative studies can be conducted that utilize new algorithms/models and larger data set.

ACKNOWLEDGEMENTS

I would like to thank Dr. Yong Zhang for his support with the opportunity to work on this research project. By providing crucial guidance during every part of my graduate experience and giving insights into approaching and solving a challenging problem. His guidance has been an enormous resource for the overall experimental design, testing and thesis writeup.

I would also like to thank the thesis committee members, Dr. John Sullins and Dr. Feng George Yu, for their encouragement and support throughout my academic coursework at Youngstown State University and spare their valuable time serving on the committee.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Detecting pedestrians in public settings is an important goal in object detection. It has been utilized in different applications such as autonomous car driving, video surveillance and navigating robots. Much research has been done in this area, using a wide variety of methods and different detection settings. The use of deep learning to carry out detection is beneficial because object detection and instance segmentation can be conducted in an integrated system. Due to the improvements in autonomous vehicles and the prevalence of video surveillance systems, The task of detecting pedestrians accurately has become increasingly important in automobile industries [1].

This paper aims to analyze different combinations of model configurations and datasets to provide an experiment-based perspective of how the current methods work with real word videos. This thesis is organized as follows: In Chapter-2, a brief literature review is provided that describes the important related works in the field, as well as the availability of several large-scale public datasets to be used. The features of three models are presented: Faster R-CNN, Mask R-CNN and Cascade R-CNN. In Chapter-3, detailed information of video samples and pre-processing procedures are described. Chapter-4 contains the experimental design such as the training and validation setup and the grouping of algorithms, backbones and architectures in each test run. In Chapter-5, experimental results are presented and the impacts of several factors on the models' performances are analyzed. The conclusions drawn from this study are given in Chapter-6.

1.1. TECHNICAL ISSUES

There are several aspects of a detection algorithm: training, testing and actual detecting. A system must balance these aspects to provide an optimal detection system.

Memory requirements: As the datasets and techniques improve, the memory required to carry on training tasks is also increased. This puts pressure on machine of a single GPU.

Precision: Precision is a critical metric in detection. Average Precision (AP) is used when assessing the performances of algorithms and models. In a validation test, there are four values that help us understand how an algorithm works as shown in Fig. 1).

		Actual	
		Positive	***Negative***
Predicted	***Positive***	True Positive (TP)	False Positive (FP)
	Negative	True Negative (TN)	False Negative (FN)

Figure 1: Confusion matrix table for a class.

Precision is formulated as: $\frac{TP}{TP+FP}$

A system with low precision does not ensure whether a detection has been carried out correctly and is not reliable to be used in real applications.

Training time: Preparing a model is a resource-intense task. Ideally, a good model has the lowest training time while not making any sacrifices from accuracy. Batch sizes and iteration counts help increase the accuracy of a model. Constantly increasing these values does not necessarily increase accuracy as there is a point of diminishing returns, and it is essential to find the optimal point between iterations and accuracy.

Realtime-ness: Since pedestrians are considered a part of traffic, the systems' detection must be done in real-time to prevent accidents. This is an important issue since the detection must be carried out quickly and accurately.

1.2.CONTRIBUTIONS

This study focuses on the empirical evaluation of the Faster R-CNN[2], Mask R-CNN[3] and Cascade R-CNN[4] methods with respect to the performances of two popular precision ratings: bounding box and segmentation in the context of detecting pedestrians. The primary contributions are: (i) Determining the impacts of the baselines and their properties on the training time, accuracy, and precision rate with a given dataset with different training/validation splits; (ii) Comparing the performances of three different models by the standard metrics under the same or similar experimental setups and running environments; (iii) Picking an optimal set of model, baseline and dataset features on training systems to detect pedestrians.

CHAPTER 2

METHODOLOGY

2.1. RELATED WORK

There is a lot of literature and research done on object detection and image detection datasets. As the methods, size of the data and technological resources increase, libraries and software packages are being developed, each of them working in a different way. Detectron2[5] , an object detection library developed by FAIR (Facebook AI Research), is an excellent example of this, which has been started as Mask R-CNN-benchmark and continued to Detectron before being developed into its current state in almost four years. Libraries like this help other research since they are regularly maintained and usually include different models.

Detecting pedestrians in public is also a research area that is being tackled for different scenarios, from robot navigation[6] to security and surveillance[7]. On a research done by Zhang et al.[8] focuses on the difficulty of training with natural scene images to detect objects in UAV images and introduce a dataset named MOHR. The authors also perform several experiments to evaluate the performance of different object detection models on their dataset. Authors suggest that R-FCN method with a ResNet-101 backbone works best with their dataset, performing 31.32 averaged AP, yet point out the need for further research, especially learning discriminative features for irregular shaped objects.

Precision and speed are vital points in pedestrian detection since an inaccurate or late decision could put people at risk. A study connected by Li and colleagues[1] focuses on this by proposing a human and machine cooperative driving system. By designing a deep Q-network (DQN) and combining it with the cooperative driving scheme in real-time, their

approach estimates collision and warns the user to take control. The work claims to achieve a more efficient warning rate than other single-pooled DQN approaches. There are other works [9], [10] focusing on detecting pedestrians under not ideal conditions like on low resolution images or nighttime images.

2.2. DATASETS

Public datasets play an essential role in research since they are mostly a product of collaborative work and usually have way more detailed annotations and categories. There are also datasets with different focuses and properties, which can be used to add variance.

Microsoft COCO[11], presented by Lin et al., provides a dataset for using object recognition to understand scenes. The researchers have gathered 330 thousand images of everyday scenes containing objects in their natural context. Another object database, Open Images Dataset (at the time of writing V6)[12], published by researchers at GoogleAI, provides more than 9 million images with 3 million annotations and 15 million bounding boxes. On its latest version, V6 + Extensions, also includes almost half a million crowdsourced images. The authors state that the research provides 15 times more bounding boxes than the following largest datasets. Developed by Zhang et al., Widerperson[13] is another dataset focusing on pedestrian detection in the wild. The authors point out the difference between diversity and density between requirements and select their images from various scenarios to provide a diverse dataset. Their dataset includes almost 14 thousand images with about 400 thousand annotations.

Terminologies:

- CNN: Convolutional Neural Network
- FPN: Feature Pyramid Network
- IoU: Intersect over Union ($\frac{A \cap B}{A \cup B}$)
- AP: Average Precision.
    - Box AP: Bounding-box (Detection) AP
    - Mask AP: Masking (Segmentation) AP
    - AP50 AP at IoU=0.50
    - APs AP for small objects: area < 322 px
    - APm AP for medium objects: 322 < area < 962 px
    - APl AP for large objects: area > 962 px
- Lr Sched (Training Schedule): Pre-training, 1x has ~12 rounds of iterations while 3x has ~37 rounds.
- $T_{\text{Iteration}}$ (s/it): Averaged time it takes to go through a single iteration.
- $T_{\text{Interference}}$ (s/img): Averaged time it takes to interference a single image.
- Base Learning Rate (SBL): Initial learning rate, how much the model adapts to errors in each update
- Max Iteration (SMI): Maximum number of iterations
- Batch Size Per Image (RBSPI): Region of Interest (ROI) batch size per image
- Testing Threshold (RHSTT): RoI testing confidence threshold
- Image Per Batch (SIPB): Number of images per batch
- RoI Classes (RNC): Number of Region of Interest classes

2.3. METHODS/ALGORITHMS

There are many different object detection algorithms developed over the years. Some of the models are iterations of the others, improving the base model each time. These algorithms vary between resource intensity, general approach and processing methods. R-CNN and YOLO[14] models are an example of iterative models. There are two tasks related to pedestrian detection.

Object Detection (Bounding Box): Object detection aims to create a bounding box around a class to detect it. Since the final result is a bounding box, it does not tell anything about the shape of an object or differentiate between different instances of the same class. Fig. 2 gives an example of this, while a bounding box shows a box with “person”, this does not give any information if the box includes a whole person or just a part of them.

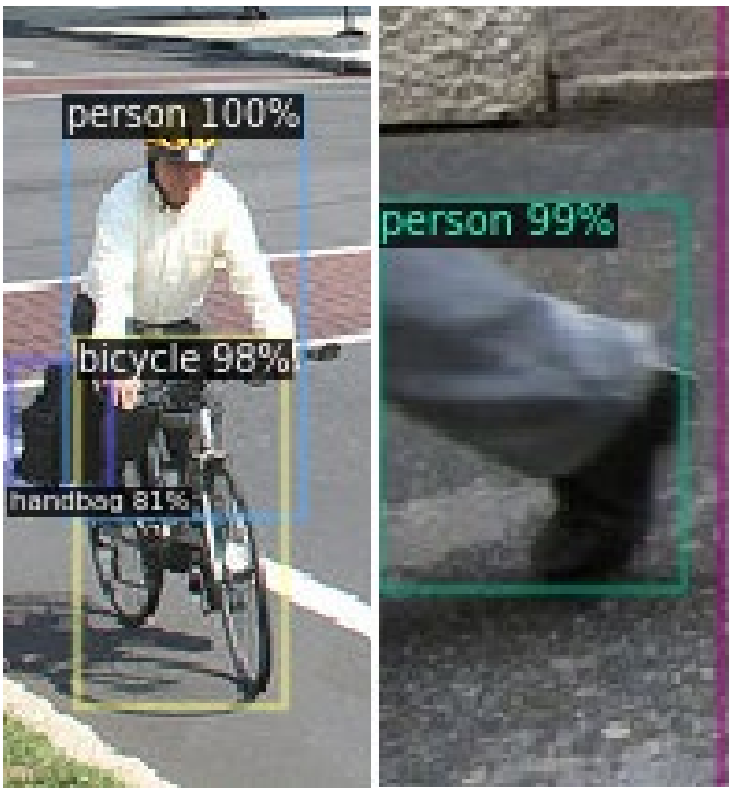

Figure 2: Object detection output on a sample image.

Figure 3: Object detection output on a sample image.

Instance Segmentation: Segmentation classifies objects by creating pixel-level masks. This represents an object way more accurately than object detection. Different from semantic segmentation, instance segmentation tries to understand and find each instance of the same class. Fig. 4 demonstrates the pixel-level mask by providing a comparison to bounding box output.

Figure 4: Difference between object detection and instance segmentation

Figure 5: Instance segmentation output on a sample image.

2.3.1. FAST R-CNN

Fast R-CNN[12], a study done by Ross Girschick, improves on a previous R-CNN (Region-based Convolutional Neural Network) by giving input image directly to CNN, generating a convolutional feature map and using a region of interest pool layer. The author suggests that the research provides train speedup times by up to 18.3 times over R-CNN.

2.3.2. FASTER R-CNN

A different study made by Ren and his colleagues proposed an improved version of Fast R-CNN, called Faster R-CNN, which benefits from a newly introduced RPN (Region Proposal Network) to cut the costs of region proposals. The authors state that these improvements provide a detection frame rate of 5fps and an increased median Average Precision value.

2.3.3. MASK R-CNN

In the study of Mask R-CNN, He et al. introduced a branch for recognizing bounding boxes to the aforementioned Faster R-CNN, improving it further. Also, it enables image segmentation by including mask and pixel-to-pixel alignment, which was missing in Faster

R-CNN. Mask R-CNN handles segmentation branches in parallel to the detection branch. Authors claim that with these changes, Mask R-CNN outperforms all single-model entries on every task at the COCO challenge at the time.

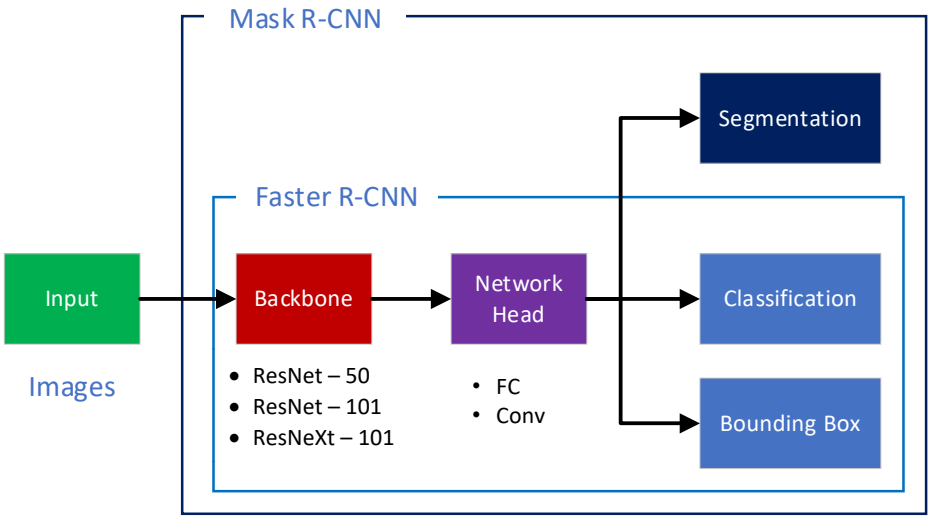

Figure 6: General architecture of Faster R-CNN and Mask R-CNN methods.

2.3.4. CASCADE R-CNN

Developed by Cai et al., Cascade R-CNN[15] is a cascade approach based on Faster R-CNN, using multiple detection branches to achieve a better detection performance. Since the Cascade R-CNN approach includes multiple detection branches, authors have extended their previous Cascade R-CNN approach to do instance segmentation by adding a mask head to the cascade, opposed to Mask R-CNN's parallel branch architecture[4]. Authors state that their approach increases AP with ResNet-50 by 3.6 on object detection (from 37.7) and 1.5 on instance segmentation (from 33.9) using COCO 2017 dataset.

2.3.5. YOLO

Another object detection system developed by Redmon et al. proposes a one-stage real-time object detection (bounding box) system[14], focusing on speed and accuracy. YOLO outperforms RetinaNet-50 and 101 methods in both mAP-50 ratings and in processing

time. Since YOLO is a single-stage method, it takes a lot less time to process. The authors also propose that their system outperforms Fast and Faster R-CNN methods in AP on COCO test-dev 2015 dataset[16].

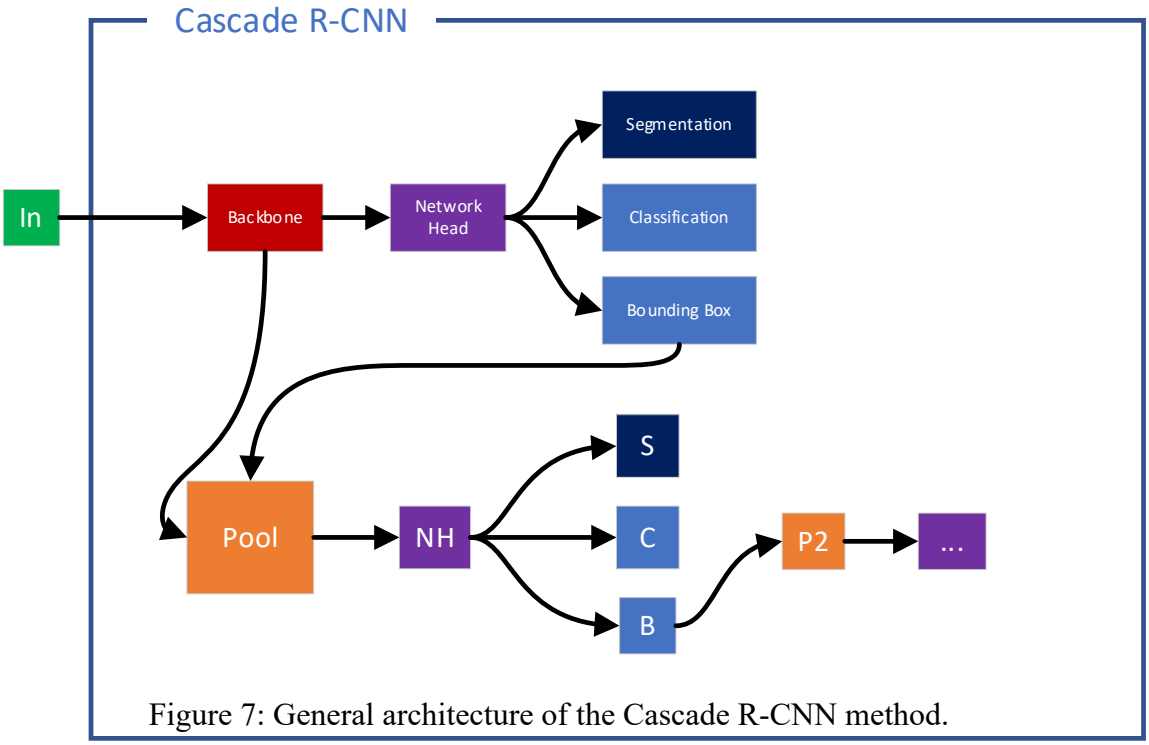

Figure 7: General architecture of the Cascade R-CNN method.

2.3.6. PANOPTIC SEGMENTATION

Proposed by Krillov et al., panoptic segmentation aims to provide a coherent segmentation by unifying semantic and instance segmentation towards real-world vision systems[17]. To achieve this, the authors define a set of rules to join these two segmentation methods together, including proposing a new metric called Panoptic Quality and setting a minimum IoU threshold. Using three datasets that have both dense semantic and instance semantic annotations[18]–[20], authors compare the performance of their model by doing several experiments; comparing AP results to their PQ metric and human-machine prediction performances.

2.4. METHODOLOGY

The proposed methodology consists of six steps, as explained in Fig 8.; model selection, where the models mentioned in Related Work are compared and selected; baseline comparison, where the baselines are compared in regard to their memory requirement, speed and accuracy. After listing and comparing datasets and models, a dataset is created and filtered. Datasets and models are trained with datasets to develop a comparative table, described in Fig. 9. The model-baseline couple with the highest accuracy and execution time is considered as the optimal couple and evaluated in the last step.

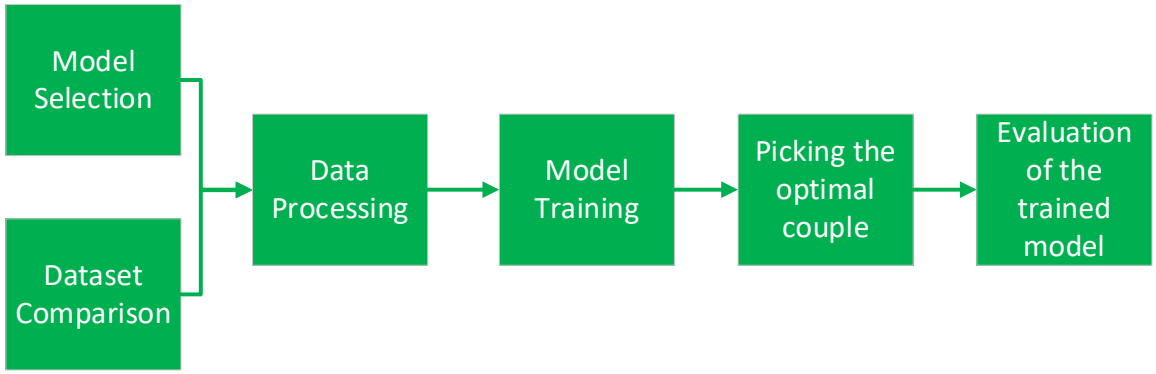

Figure 8: Proposed methodology

		METHODS				
		A	B	C	D	E
BASELINES	X	Accuracy: Runtime:				
	Y					
	Z					
	Q					
	P					

Figure 9: Sample Model-Baseline couple table

CHAPTER 3

DATA PROCESSING

This section describes dataset preparation steps, from acquiring to be readily used in the experiment.

3.1. DATASET

During the course of the work, the COCO image dataset has been utilized as the dataset provides enough images to train and take less space than other datasets with more images. The COCO dataset provides 66,808 images with “person” annotation in total, separated as 64,115 train and 2,693 validation images. A few utility programs were developed in Python to acquire, split and prepare the datasets to help with the research. Fig. 10 to 12 provides images from the COCO dataset at different stages of the work.

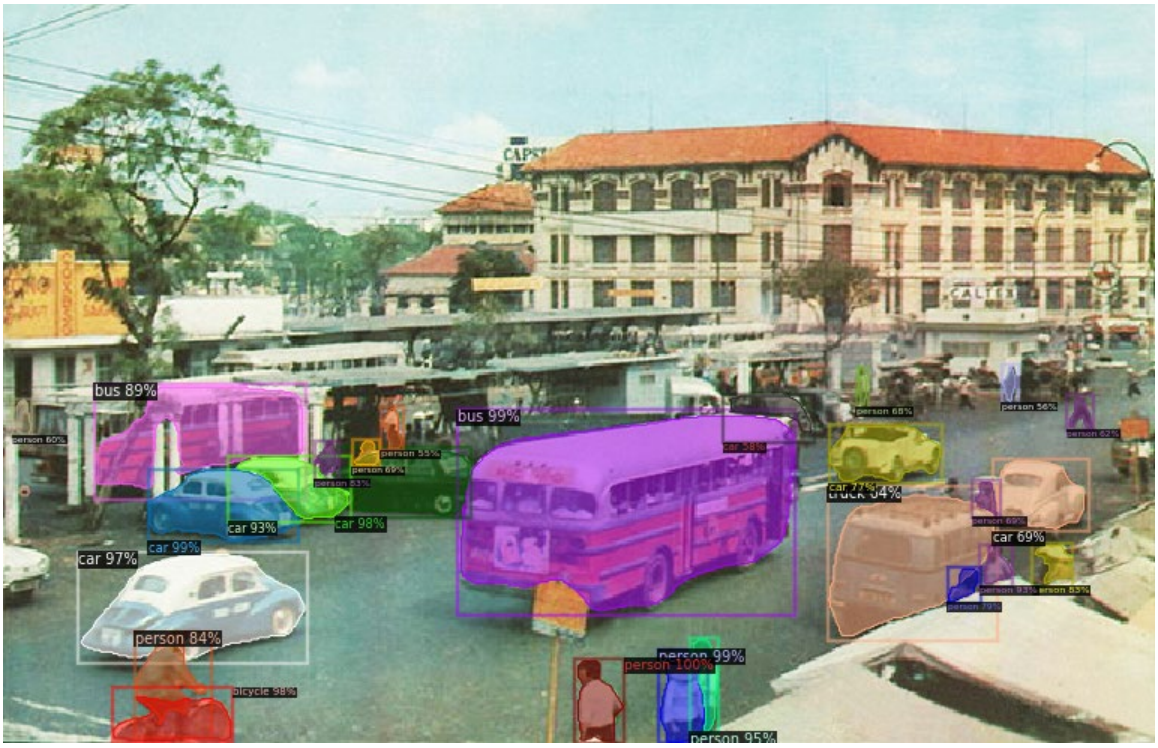

Figure 10: An unfiltered image from the COCO dataset

Figure 11: Sample “person” images from the edited COCO dataset.

Figure 12: A processed image from “validation” set using Faster R-CNN.

3.2.SAMPLE PREPARATION

A few utility programs were developed in Python to acquire, split and prepare the datasets to help with the research. A dataset is generally constructed as two folders of images train/validation and two annotation files consisting of annotations of respective folders. A downloader utility used to get only images with “person” from 123,000 images, a filter utility to extract unrelated (non-person) annotations in images, and a separator

utility to change sizes of training-validation annotation files. These utilities allow us to use and split the dataset however we need.

Using the created dataset utilities, several subsets of train images are selected to achieve different balances with the validation dataset, including an 80/20 split between training and validation images to provide an insight on the effect of dataset split on training performance.

CHAPTER 4

EXPERIMENT DESIGN

This section describes the experimental details during the training/validation phases with various parameter setups. All experiments were carried out on different configurations using the Detectron2 packages installed under Linux environments, a local machine with Nvidia 1660Ti, CUDA 11.2 with 6 GB GPU memory, and on a Google Colab machine with a Tesla T4, CUDA 11.2 with 15 GB's of GPU Memory. Datasets are stored locally for the local machine and on Google Drive, for Google Colab machines to reduce network and data transmission costs.

4.1. TRAINING

The training phase has two steps:

STEP-1: Prepare the dataset:

Locally stored datasets are prepared with annotation files in COCO JSON format. JSON format helps to parse the data using the utility programs to prepare the dataset to be imported by detectron2's library.

COCO Standard Data Format	**Object Detection Extension**

{"info" : info, "images" : [image],
"annotations" : [annotation], "licenses" : [license],}
 	
info{ "year": int,
"version" : str, "description" : str,
"contributor" : str, "url" : str,
"date_created" : datetime,}
 	
image{"id" : int, "width" : int,
"height" : int, "file_name" : str,
"license" : int, "flickr_url" : str,
"coco_url" : str, "date_captured" : datetime,}
 	
license{"id" : int, "name" : str, "url" : str, }
	annotation{
	"id" : int,
	"image_id" : int,
	"category_id" : int,
	"segmentation" : RLE or [polygon],
	"area" : float,
	"bbox" : [x,y,width,height],
	"iscrowd" : 0 or 1,
	}
	
	categories[{
	"id" : int,
	"name" : str,
	"supercategory" : str,
	}]

Figure 13: COCO Data Format

STEP-2: Train a model:

In this step, a baseline is picked and fine-tuned on the prepared dataset. Baselines: During the work, baselines from detectron2's model zoo are used. A few of the baselines with different backbone combinations and baseline features are selected to provide a comparison. These are,

1) Algorithms:
    - Faster R-CNN, provides only object (bounding box) detection
    - Mask R-CNN, supports object segmentation and object detection
    - Cascade R-CNN, is a cascade of Mask R-CNN model.
2) Backbones:
    - FPN: Uses a Resnet + Feature Pyramid Network (FPN) backbone with standard convolution layer and Fully-connected (FC) heads for bounding box and mask predictions.
    - Conv4 (C4): A ResNet conv4 backbone with a conv5 head. This backbone has also used by the Faster R-CNN paper.
    - Dilated C5 (DC5): A ResNet conv5 backbone, including dilations in conv5, and a standard convolution and Fully Connected layers as head for bounding box and mask predictions.
3) ResNet Network Architectures:
    - R-50: Original ResNet-50 model, by Microsoft Research Asia. 50 layers deep.
    - R-101: Original ResNet-101 model. 101 layers deep.
    - X-101: ResNeXt-101-32x8d model. 101 layers deep.

The main difference between ResNet and ResNeXt is that ResNeXt has more detailed layers in each stage of convolution; and includes 32 grouped convolutions each iteration. This results in a more detailed processing of an image, but requires more memory.

4) Baseline training schedule: 1x/3x. This defines how much a baseline has been pretrained. A 3x schedule represents 3 COCO epochs of training.

A table with the complete baselines is provided below.

TABLE 1: COMPLETE BASELINE TABLE

Model ID	Name	Lr sched	Train time (s/iter)	Inference time (s/im)	Train mem (GB)	Box AP	Mask AP
			Faster R-CNN				
137257644	R50-C4	1x	0.551	0.102	4.8	35.7	-
137847829	R50-DC5	1x	0.38	0.068	5	37.3	-
137257794	R50-FPN	1x	0.21	0.038	3	37.9	-
137849393	R50-C4	3x	0.543	0.104	4.8	38.4	-
137849425	R50-DC5	3x	0.378	0.07	5	39	-
137849458	R50-FPN	3x	0.209	0.038	3	40.2	-
138204752	R101-C4	3x	0.619	0.139	5.9	41.1	-
138204841	R101-DC5	3x	0.452	0.086	6.1	40.6	-
137851257	R101-FPN	3x	0.286	0.051	4.1	42	-
139173657	X101-FPN	3x	0.638	0.098	6.7	43	-
			Mask R-CNN				
137259246	R50-C4	1x	0.584	0.11	5.2	36.8	32.2
137260150	R50-DC5	1x	0.471	0.076	6.5	38.3	34.2
137260431	R50-FPN	1x	0.261	0.043	3.4	38.6	35.2
137849525	R50-C4	3x	0.575	0.111	5.2	39.8	34.4
137849551	R50-DC5	3x	0.47	0.076	6.5	40	35.9
137849600	R50-FPN	3x	0.261	0.043	3.4	41	37.2
138363239	R101-C4	3x	0.652	0.145	6.3	42.6	36.7
138363294	R101-DC5	3x	0.545	0.092	7.6	41.9	37.3
138205316	R101-FPN	3x	0.34	0.056	4.6	42.9	38.6
139653917	X101-FPN	3x	0.69	0.103	7.2	44.3	39.5
			Cascade R-CNN (C-R-CNN)				
138602847	C-R-CNN	1x	0.317	0.052	4	42.1	36.4
144998488	C-R-CNN	3x	0.328	0.053	4	44.3	38.5

A subset of baselines is picked to provide the ideal comparison, a set of criteria is set by these research goals:

- C1: Algorithms: Comparison of Faster, Cascade and Mask R-CNN algorithms, visualizing the difference between algorithms.
- C2: ResNet training models: How do R-50, R-101, and X-101 affect the performance?
- C3: Backbone comparisons: FPN/C4/DC5, which one does perform better?
- C4: LR schedule difference: How does 1x and 3x differ?

Baselines are codenamed for logging and comparison purposes. The resulting baseline subset is shown in the table below:

TABLE 2: SELECTED BASELINE SET

Codename	Algorithm	Model ID	Name	Lr sched	Train mem (GB)	Box AP	Mask AP
F3	Faster R-CNN	137257794	R50-FPN	1x	3	37.9	-
F6	Faster R-CNN	137849458	R50-FPN	3x	3	40.2	-
M1	Mask R-CNN	137259246	R50-C4	1x	5.2	36.8	32.2
M2	Mask R-CNN	137260150	R50-DC5	1x	6.5	38.3	34.2
M3	Mask R-CNN	137260431	R50-FPN	1x	3.4	38.6	35.2
M6	Mask R-CNN	137849600	R50-FPN	3x	3.4	41	37.2
M9	Mask R-CNN	138205316	R101-FPN	3x	4.6	42.9	38.6
M10	Mask R-CNN	139653917	X101-FPN	3x	7.2	44.3	39.5
CR1	Cascade R-CNN	138602847	C R-CNN	1x	4	42.1	36.4
CR2	Cascade R-CNN	144998488	C R-CNN	3x	4	44.3	38.5

M3 is selected as the base model, which will be compared with others to answer the research goals. Following subsets are created to answer the research goals:

- C1: F3-M3-CR1
- C2: M6-M9-M10
- C3: M1-M2-M3

- C4: F3-F6, M3-M6, CR1-CR2

We have several properties to adjust in addition to the baseline model:

- SOLVER.IMS_PER_BATCH (SIPB): Number of images per batch, 2 for each GPU in training. Increases memory as batch size increases.
- SOLVER.BASE_LR (SBL): Base Learning Rate, 0.001 selected as default to keep it away from overshooting while keeping the convergence time short.
- SOLVER.MAX_ITER (SMI): Maximum number of training iterations, 900 chosen as default.
- ROI.BATCH_SIZE_PER_IMAGE (RBSPI): Region of Interest (ROI) batch size per image, 512 given as default.
- ROI.NUM_CLASSES (RNC): Region of Interest classes, which is 1 for this project because there's only one class (Person) to train.

4.2. TESTING

STEP-3: Inference & Evaluation on trained model:

After training a model, the path to the model is given to calculate inference on the validation dataset. In addition to the previous properties, an RoI testing threshold is added at this point.

ROI_HEADS.SCORE_THRESH_TEST (RHSTT): RoI testing threshold, 0.6 as default. A lower value increases AP but slows down the inference process. Different values may be chosen and experimented on to find an ideal point.

These are also the model variables that will be adjusted during the work to observe their effect on accuracy and training time on pedestrian detection. To observe this, baseline M3 will be run with values of:

TABLE 3: EXPERIMENTAL BASELINE PROPERTIES

	SIPB	SBL	SMI	RBSPI	RNC	RHSTT
Run#1	2*	0.001	900	512	1**	0.6
Run#2	2	**0.002**	900	512	1	0.6
Run#3	2	**0.0005**	900	512	1	0.6
Run#4	2	0.001	**1800**	512	1	0.6
Run#5	2	0.001	**450**	512	1	0.6
Run#6	2	0.001	900	**1024**	1	0.6
Run#7	2	0.001	900	**256**	1	0.6
Run#8	2	0.001	900	512	1	**0.8**
Run#9	2	0.001	900	512	1	**0.4**

*SIPB will be kept at 2, according to the GPU and memory limitations.

**RNC is held at 1 due to the class size.

All of these runs are compared in terms of detection performance to find an optimal baseline run configuration.

4.3. DATASET SPLIT

In addition to the baseline property runs, two more runs of M3 are conducted with the same baseline properties but with different training-validation splits. This is done to observe the effect of splits in performance and precision, if there is any. The splits are done using custom utility tools, and three total splits are created.

- Split1: A split of 20560 training and 2693 validation images (88%/12%). This is the default split used in all of the runs.
- Split2: All images with “person” class in the COCO dataset. 64115 training and 2693 validation images (96%/4%).
- Split3: A split with 80% and %20 between training and validation images.

TABLE 4: BASELINE SPLITS

	Split	SBL	SMI	RBSPI	RHSTT
Run#1	20560/2693	0.001	900	512	0.6
Run#10	64115/2693	0.001	900	512	0.6
Run#11	10772/2693	0.001	900	512	0.6

CHAPTER 5

RESULTS AND DISCUSSIONS

Throughout the experiments, selected ten baselines have trained dozens of times separately to compare their differences. Table 5 below provides the results of chosen runs carried out as described in previous chapters. As the baselines are undertrained, every initial run except an experimental run saw an increase in AP values. On top of comparing baselines, nine combinations of baseline properties are also experimented with to provide an insight into their effect on performance. During this chapter, the impact of baselines with regard to their detection algorithm, res-net models, backbones and LR schedules are explained separately. Baseline properties are compared with base learning rates, training iterations, RoI batch sizes and testing thresholds. Notable differences in results are also demonstrated with Tensorboard outputs, Fig. 14, 15, 19, and 21.

TABLE 5: EXPERIMENTAL RESULTS

Code-name	**Name**	**Lr sched**	**Box AP**	**Mask AP**	**Train Box AP**	**Train Mask AP**	$\Delta_{Box\ AP}$	$\Delta_{Mask\ AP}$	$T_{Iteration}$ **(s/it)**	$T_{Interference}$ **(s/img)**	**Total Time (h:mm:ss)**
M1	R50-C4	1x	36.8	32.2	49.998	40.849	13.198	8.649	1.5486	0.362128	0:39:09
M2	R50-DC5	1x	38.3	34.2	49.845	42.124	11.545	7.924	1.0256	0.220827	0:25:06
M3	R50-FPN	1x	38.6	35.2	52.227	45.077	13.627	9.877	0.5469	0.207233	0:17:19
M6	R50-FPN	3x	41	37.2	54.119	45.569	13.119	8.369	0.5443	0.125886	0:13:42
M9	R101-FPN	3x	42.9	38.6	54.537	47.233	11.637	8.633	0.7421	0.162362	0:18:16
M10	X101-FPN	3x	44.3	39.5	56.582	48.472	12.282	8.972	3.6404	0.732266	1:26:49
CR1	R50-FPN	1x	42.1	36.4	56.399	45.692	14.299	9.292	0.681	0.156877	0:17:07
CR2	R50-FPN	3x	44.3	38.5	57.808	47.96	13.508	9.46	0.67	0.173187	0:17:40
F3	R50-FPN	1x	37.9	-	50.882	-	12.982	-	1.2647	0.31936	0:33:01
F6	R50-FPN	3x	40.2	-	53.157	-	12.957	-	1.2541	0.320109	0:32:53

After comparing ten baselines with nine different configurations, the results are presented.

5.1. IMPACT OF BASELINES

Baselines have several different properties, which requires a detailed evaluation of them. In this section, previously defined criteria are evaluated and compared in separate parts.

5.1.1 IMPACT OF DETECTION ALGORITHM (C1)

The detection algorithm is one of the essential factors in detection problems. On top of their performance improvements, algorithms also allow different approaches and functionalities to improve the scale of pedestrian detection further. As Mask R-CNN supports instance segmentation on top of the bounding box object detection present in Faster R-CNN and Fast R-CNN, Mask R-CNN is mode adaptable. The experiments also show that Mask R-CNN models train better in both Box AP and Mask AP. As shown in Fig. 14 and 15, there is not much difference in accuracy, false-negative, and false-positive values; since the image classification is done with R-50. Mainly, increases in AP values and iteration times are focused on.

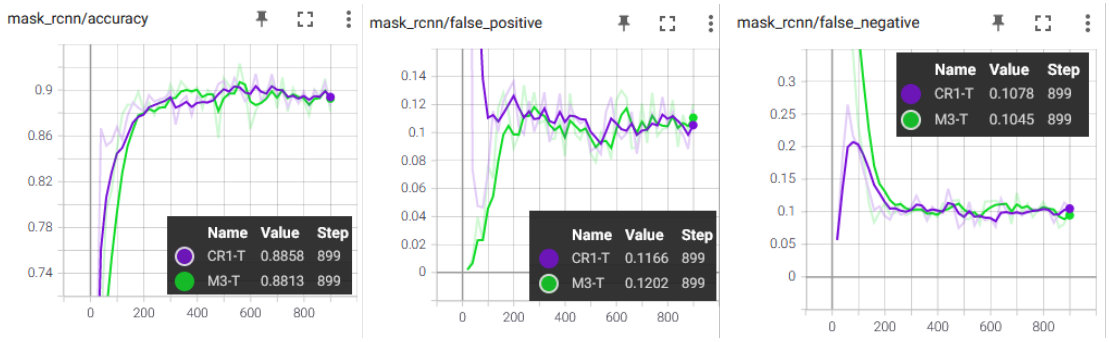

Figure 14: Accuracy, False-Positive/Negative comparisons of Mask R-CNN (red), Cascade R-CNN (purple)

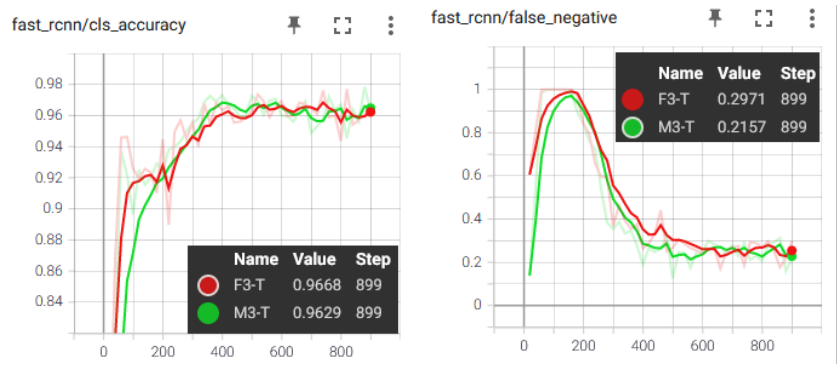

Figure 15: Accuracy and False-Negative comparison of Mask R-CNN (red) and Faster R-CNN (green)

As shown in Table 5., the difference in AP increase in Mask R-CNN (M3) and Cascade R-CNN (CR1) has very similar increases in AP values between three baselines, the difference between the two being only 0.37% more. However, each iteration of CR1 is 25.92% slower than M3. Faster R-CNN has the lowest values of AP increase and iteration time. Based on these results, it can be said that Mask R-CNN is a better option in terms of speed/train performance than Faster R-CNN and Cascade R-CNN in general. Except, in bounding box AP, Cascade R-CNN provides the highest AP.

TABLE 6: IMPACT OF DETECTION ALGORITHM (C1)

Code-name	Name	Box AP	Mask AP	Train Box AP	Train Mask AP	$\Delta_{Box\ AP}$	$\Delta_{Mask\ AP}$	$T_{Iteration}$ (s/it)	$T_{Interference}$ (s/img)	Total Time (h:mm:ss)
M3	R50-FPN	38.6	35.2	52.227	45.077	13.627	9.877	0.5469	0.207233	0:17:19
CR1	R50-FPN	42.1	36.4	56.399	45.692	14.299	9.292	0.681	0.156877	0:17:07
F3	R50-FPN	37.9	-	50.882	-	12.982	-	1.2647	0.31936	0:33:01

5.1.2. IMPACT OF RES-NET MODELS (C2)

By utilizing different layer depths and setting, ResNet models provide similar AP increases before and after training, and it is observed that iteration time and memory usage makes the most notable difference. Table 7 shows that X101 takes 700% more time than the R50 baseline for each iteration and has a lower increase in AP. However, a more complex model provides a better AP; thus, it is more suitable for stable systems that do not

require re-training models. In terms of iteration speed, AP increase and memory usage, R50 provides a better increase-performance tradeoff with a lower memory requirement.

TABLE 7: IMPACT OF RES-NET MODELS (C2)

Code-name	Name	Box AP	Mask AP	Train Box AP	Train Mask AP	$\Delta_{Box\ AP}$	$\Delta_{Mask\ AP}$	$T_{Iteration}$ (s/it)	$T_{Interference}$ (s/img)	Total Time (h:mm:ss)
M6	R50-FPN	41	37.2	54.119	45.569	13.119	8.369	0.5443	0.125886	0:13:42
M9	R101-FPN	42.9	38.6	54.537	47.233	11.637	8.633	0.7421	0.162362	0:18:16
M10	X101-FPN	44.3	39.5	56.582	48.472	12.282	8.972	3.6404	0.732266	1:26:49

Figure 16: Detection made by M10

5.1.3. IMPACT OF BACKBONES (C3)

When comparing backbone training and AP performances, the main difference the time it takes to train a baseline. FPN halves the total time compared to C4, with almost three times faster iteration times. While DC5 is 1.5 times faster than C4, it does not affect Box and Mask AP as much as C4.

TABLE 8: IMPACT OF BACKBONES (C3)

Code-name	Name	Box AP	Mask AP	Train Box AP	Train Mask AP	$\Delta_{Box\ AP}$	$\Delta_{Mask\ AP}$	$T_{Iteration}$ (s/it)	$T_{Interference}$ (s/img)	Total Time (h:mm:ss)
M1	R50-C4	36.8	32.2	49.998	40.849	13.198	8.649	1.5486	0.362128	0:39:09
M2	R50-DC5	38.3	34.2	49.845	42.124	11.545	7.924	1.0256	0.220827	0:25:06
M3	R50-FPN	38.6	35.2	52.227	45.077	13.627	9.877	0.5469	0.207233	0:17:19

5.1.4. IMPACT OF LR SCHEDULE (C4)

LR Schedule determines the number of epochs a model has been pre-trained. As algorithms differ in training performance and benefit differently from the number of trains, they are compared separately. Baselines have chosen in regard to their barebone configs. It is observed that there's next to no difference between $\boldsymbol{T_{Iteration}}$ and $\boldsymbol{T_{Interference}}$ when it comes to the LR schedule. when it comes to the LR schedule. However, less trained models are trained faster, whereas higher LR sched baselines have higher AP scores.

TABLE 9: LR IMPACT ON MASK AND CASCADE R-CNN (C4)

Code-name	Name	Lr sched	Box AP	Mask AP	Train Box AP	Train Mask AP	$\Delta_{Box\ AP}$	$\Delta_{Mask\ AP}$	$T_{Iteration}$ (s/it)	$T_{Interference}$ (s/img)	Total Time (h:mm:ss)
M3	R50-FPN	1x	38.6	35.2	52.227	45.077	13.627	9.877	0.5469	0.207233	0:17:19
M6	R50-FPN	3x	41	37.2	54.119	45.569	13.119	8.369	0.5443	0.125886	0:13:42
CR1	R50-FPN	1x	42.1	36.4	56.399	45.692	14.299	9.292	0.681	0.156877	0:17:07
CR2	R50-FPN	3x	44.3	38.5	57.808	47.96	13.508	9.46	0.67	0.173187	0:17:40

On the other hand, LR does not provide any significant changes at Box AP, iteration and interference time on Faster R-CNN.

TABLE 10: LR IMPACT ON FASTER R-CNN (C4)

Code-name	Name	Lr sched	Box AP	Train Box AP	$\Delta_{Box\ AP}$	$T_{Iteration}$ (s/it)	$T_{Interference}$ (s/img)	Total Time (h:mm:ss)
F3	R50-FPN	1x	37.9	50.882	12.982	1.2647	0.31936	0:33:01
F6	R50-FPN	3x	40.2	53.157	12.957	1.2541	0.320109	0:32:53

5.2. IMPACT OF BASELINE PROPERTIES

There are several properties that change how a model is trained and performed. For testing, baseline M3 has been selected. Run#1 is the control model, where every value is at its defaults.

- Run#2 and #3 demonstrates the effect of SBI, which is the base learning rate.
- Run#4 and #5 demonstrates the effect of SMI, count of maximum training iterations. Iteration numbers are doubled and halved to show the impact, to provide a point to create an ideal model.
- Run#6 and #7 shows the difference achieved by changing RBSPI, RoI batch size per image.
- And Run#8 and #9 shows how changing testing threshold (RHSTT) value affect interference performance.

Since the research was done on single GPU machines, differences in the number of images per batch (SIPB) were not observed. Likewise, as the dataset was filtered to include a single class, multiple class AP values (RNC) are also not provided.

5.2.1. Base Learning Rate (SBL):

Learning rate is the rate of how a model reacts to errors by changing weights in each update. The learning rate is doubled on Run#2 and halved on Run#3. The results show a lower learning rate resulted in improved Box AP and Mask AP values. However, on experimental runs where SBL is 0, a dramatic decrease in AP has been observed since the model does not improve itself. This shows that the SBL value should be experimented with each dataset-model as there's no definite ideal value.

TABLE 11: IMPACT OF SBL

M3 Runs	**SBL**	Train box AP	Train mask AP	$\Delta_{Box\ AP}$	$\Delta_{Mask\ AP}$	$T_{Iteration}$ (s/it)	$T_{Interference}$ (s/img)	**Total Time (h:mm:ss)**
Run#1	**0.001**	52.227	45.08	13.63	9.877	0.5469	0.20723	0:17:19
Run#2	**0.002**	51.239	44.06	12.64	8.858	0.5535	0.14662	0:14:45
Run#3	**0.0005**	53.141	45.26	14.54	10.06	0.5322	0.12728	0:13:35

5.2.2. Training Iterations:

A lower count of iterations has a minimal effect on iteration per second value; however, having a high number of iterations provide higher Box AP scores at the cost of time. On an experimental run with 9000 iterations, 53 Box AP has been reached in an hour and 25 minutes which further proves it.

TABLE 12: IMPACT OF ITERATION COUNT

M3 Runs	**Iteration Count (SMI)**	Train box AP	Train mask AP	$\Delta_{Box\ AP}$	$\Delta_{Mask\ AP}$	$T_{Iteration}$ (s/it)	$T_{Interference}$ (s/img)	**Total Time (h:mm:ss)**
Run#1	**900**	52.227	45.08	13.63	9.877	0.5469	0.20723	0:17:19
Run#4	**1800**	52.429	44.38	13.83	9.178	0.5423	0.1287	0:21:56
Run#5	**450**	51.191	43.71	12.59	8.51	0.5554	0.1323	0:09:59
Run#Exp1	**9000**	52.919	44.85	14.32	9.65	0.5404	0.128306	1:26:42

5.2.3. RoI Batch Size:

Increasing batch sizes directly affects Box AP values, as it improves the control model's Bounding box AP by 0.8.

TABLE 13: IMPACT OF ROI BATCH SIZE

M3 Runs	**RoI Batch (RBSPI)**	Train box AP	Train mask AP	$\Delta_{Box\ AP}$	$\Delta_{Mask\ AP}$	$T_{Iteration}$ (s/it)	$T_{Interference}$ (s/img)	**Total Time (h:mm:ss)**
Run#1	**512**	52.227	45.08	13.63	9.877	0.5469	0.20723	0:17:19
Run#6	**1024**	53.037	44.39	14.44	9.188	0.5775	0.12238	0:14:03
Run#7	**256**	51.718	44.53	13.12	9.328	0.5059	0.13482	0:13:31

5.2.4. Testing Threshold:

Changes in testing threshold show themselves in results, as it provides a threshold for predictions, a high testing rate results in higher confidence in detected objects. Still, it can also miss predictions with lower confidence, resulting in a lower AP. Figures 11 and 12 demonstrate the effect of using different thresholds on validation outputs.

TABLE 14: IMPACT OF TESTING THRESHOLD

M3 Runs	**Testing Threshold**	Train box AP	Train mask AP	$\Delta_{Box\ AP}$	$\Delta_{Mask\ AP}$	$T_{Iteration}$ (s/it)	$T_{Interference}$ (s/img)	**Total Time (h:mm:ss)**
Run#1	**0.6**	52.227	45.08	13.63	9.877	0.5469	0.20723	0:17:19
Run#8	**0.8**	52.009	44.44	13.41	9.243	1.3914	0.35865	0:36:39
Run#9	**0.4**	51.535	44.4	12.94	9.199	1.3922	0.34576	0:36:05

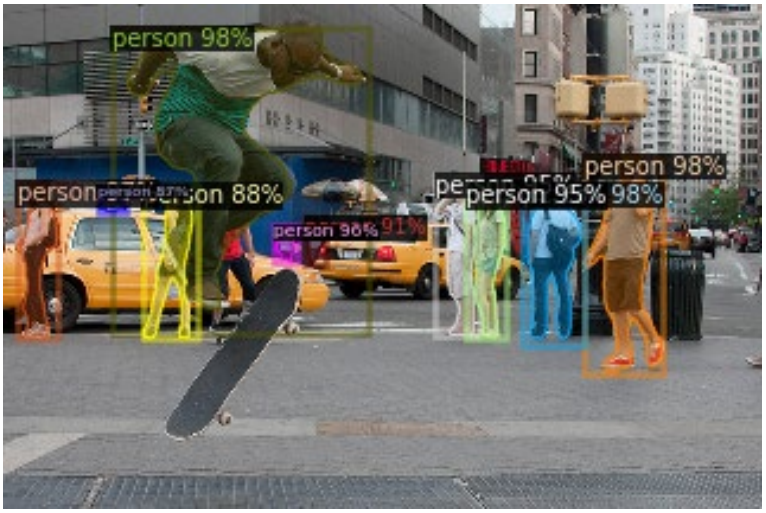

Figure 17: A detection from #Run8 (Threshold 0.8)

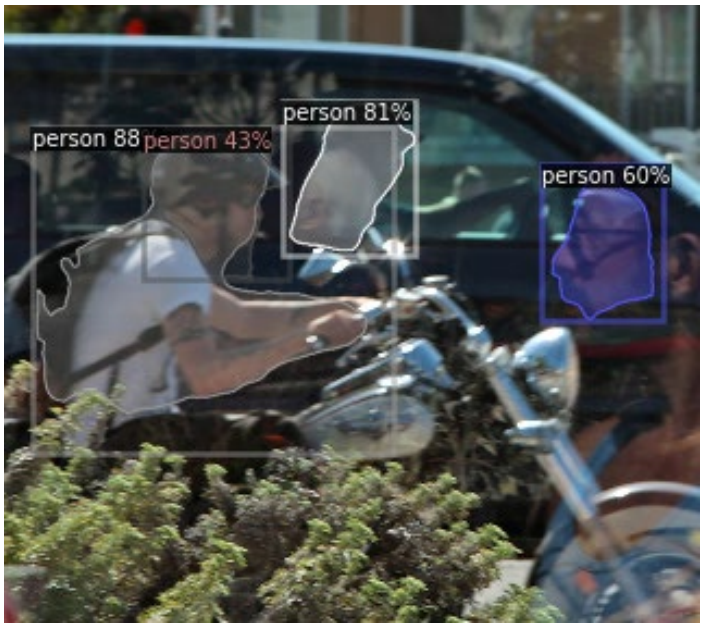

Figure 18: A detection from #Run4 (Threshold 0.4)

Having experimented with different values, it is observed that:

- Most notable factor in Box and Mask AP's observed by decreasing the learning rate.
- Halving the iteration count reduced the training time by half, but it has lowered the AP values as a result.
- Increasing RoI batch sizes provided a better result in Box AP and the fastest interference time per image.

A well-performing model should have a low learning rate, high iterations, and bigger RoI batch sizes. But as the iterations increase, the time the model takes to develop increases diminishingly.

TABLE 15: SUMMARIZATION OF DIFFERENT BASELINE VARIABLES

M3 Runs	Train box AP	Train mask AP	$\Delta_{Box\ AP}$	$\Delta_{Mask\ AP}$	$T_{Iteration}$ (s/it)	$T_{Interference}$ (s/img)	**Total Time (h:mm:ss)**
Run#1	52.227	45.08	13.63	9.877	0.5469	0.20723	0:17:19
Run#2	51.239	44.06	12.64	8.858	0.5535	0.14662	0:14:45
Run#3	**53.141**	**45.26**	**14.54**	**10.06**	**0.5322**	**0.12728**	0:13:35
Run#4	52.429	44.38	13.83	9.178	0.5423	0.1287	0:21:56
Run#5	*51.191*	*43.71*	*12.59*	*8.51*	*0.5554*	*0.1323*	0:09:59
Run#6	53.037	44.39	14.44	9.188	0.5775	0.12238	0:14:03
Run#7	51.718	44.53	13.12	9.328	0.5059	0.13482	0:13:31
Run#8	52.009	44.44	13.41	9.243	1.3914	0.35865	0:36:39
Run#9	51.535	44.4	12.94	9.199	1.3922	0.34576	0:36:05

5.3. IMPACT OF TRAIN-VALIDATION SPLIT

Another two runs of M3 (M3 Run#10 and #11) has been done to show how different splits of training data would result in training.

M3 Run#1 itself has 20,560 train, 2,693 validation images. The rate of training images to validation images is 7.5/1. M3 Run#10 includes all images in the COCO dataset with the "person" class, filtered by utility tools. With 64,115 train and 2,693 validation images, the rate of train-validation is 23/1. M3 Run#11 has run with an 80/20 split, including 10,772 training and 2,693 validation images. As it can be seen from Table 16, the results do not give any insight on the impact of a split, as most of the accuracy and model is reached on first iterations. After the optimal accuracy is achieved, the model re-evaluates itself around the same rates, which can be seen in Fig. 19.

TABLE 16: IMPACT OF SPLIT

M3 Runs	Split	Train box AP	Train mask AP	$\Delta_{Box\ AP}$	$\Delta_{Mask\ AP}$	$T_{Iteration}$ (s/it)	$T_{Interference}$ (s/img)	**Total Time (h:mm:ss)**
Run#1	**20560/2693**	52.227	45.077	13.627	9.877	0.5469	0.207233	0:17:19
Run#10	**64115/2693**	50.749	44.181	12.149	8.399	0.5542	0.134859	0:14:15
Run#11	**10772/2693**	51.752	44.035	13.152	8.835	0.5597	0.132962	0:14:14

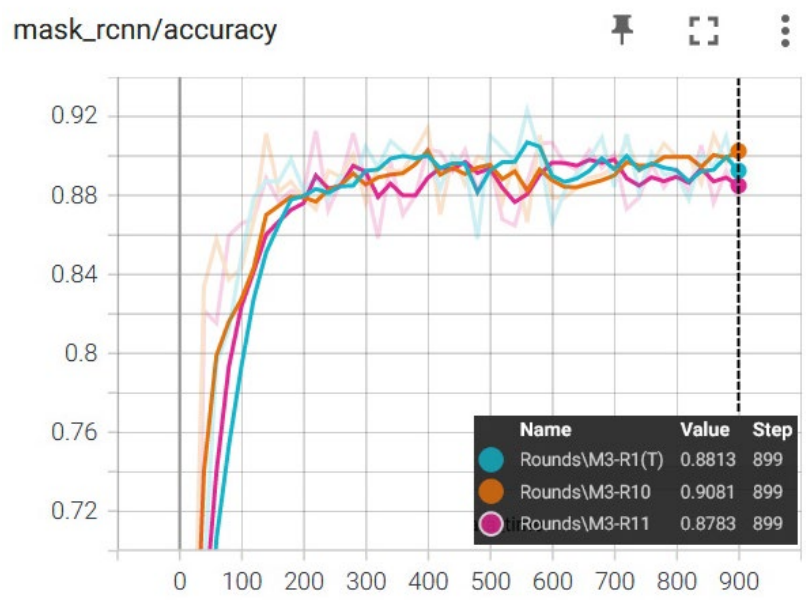

Figure 19: Split comparison between Run#1, Run#10 and Run#11

After running and analyzing these experiments, a baseline with ideal properties have been created and run to prove experiment results. Test baseline is selected from M6, as it has the best properties (Mask R-CNN, R50-FPN) and is more trained than M3. Table 17 below provides a comparison between initial runs and the run with changed baseline properties. Table 18 includes detailed AP values for comparison.

TABLE 17: EXPERIMENTAL BASELINE COMPARISON

Runs	Box AP	Mask AP	$\Delta_{Box\ AP}$	$\Delta_{Mask\ AP}$	SBL	SMI	RBSPI	RHSTT	$T_{Iteration}$ (s/it)	$T_{Interference}$ (s/img)	Total Time (h:mm:ss)
M6 (initial)	54.119	45.569	**13.119**	**8.369**	0.001	900	512	0.6	0.5443	0.125886	0:13:42
M6-F	55.302	47.249	**14.299**	**10.049**	0.0005	2700	1024	0.6	0.5911	0.126910	0:32:17

TABLE 18: EXPERIMENTAL BASELINE AP COMPARISON

Baseline	Object Detection						Instance Segmentation					
	AP	***AP50***	***AP75***	***APs***	***APm***	***APl***	***AP***	***AP50***	***AP75***	***APs***	***APm***	***APl***
M6 (init)	54.119	83.45	58.94	37.7	60.783	70.24	45.569	79.88	47.854	27.426	50.437	63.623
M6-F	55.302	83.98	60.25	37.91	62.645	72.106	47.249	81.01	50.654	28.578	51.767	65.598

The AP and increment results are similar to M10's, which has an X101-FPN backbone. But the improvement shows itself on runtime, whereas M10 uses 7 GB of memory and takes 1:27:17 to train, our experimental M6-F uses 5.6 GB and takes 0:32:17 to achieve similar results. The difference between the baseline trained M6 and modified trained M6 is shown in Fig. 20 and 21.

Figure 20: Detections between Baseline M6 and Modified M6

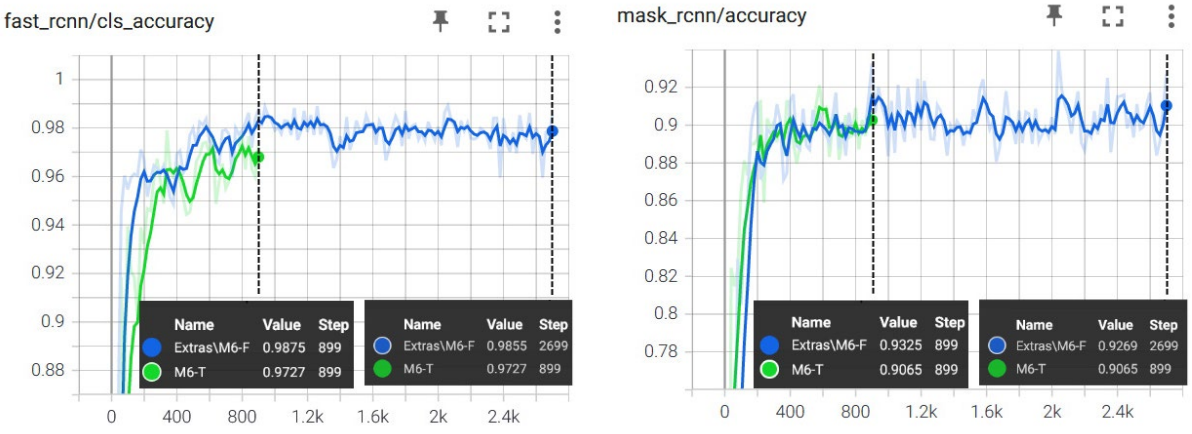

Figure 21: Mask R-CNN accuracy of Modified M6 (Blue) and Baseline M6 (Green)

CHAPTER 6

CONCLUSIONS

This thesis compares the performances of several pedestrian detection methods with different model parameter values and configuration settings. It should be noted that the performance of a detection model may vary largely depending upon the data size and computing power used. The experimental results of this study are obtained using a local server and therefore the conclusions cannot be extended to a different computing domain without making appropriate adjustments. The primary findings are summarized as below:

1. Baseline models with Mask R-CNN yields a higher AP with the minimal running time.
2. Cascade R-CNN uses more memory, takes more time for iterations and interference, but generates the highest AP.
3. Faster R-CNN baselines require less memory, but do not support object segmentation and perform worse on bounding box than the Mask R-CNN models.
4. If a well-rounded model is desired, a baseline with these features can be considered:
    - Mask R-CNN, ResNet R-50, and FPN backbone.
    - High LR, as it was pre-trained with higher initial AP values. This can be an issue when introducing new classes.
    - Iteration Count: 1800-2700 may be a good starting choice.
    - Base Learning Rate: 0.001 to 0.00025 for a single GPU.
    - RoI Batch Size: At least 512 for a practical dataset.
5. ResNet models require increasingly more memory as the layer depth increases. X-101 takes more time and resources to improve AP than other R-101 variants. Using COCO dataset, R-50 finishes training faster and does not require much memory.

6. A baseline model of a FPN backbone yields better results than that of C4 and DC5 alternatives.
7. Testing threshold seems not a significant factor of affecting the AP results.

REFERENCES

[1] J. Li, L. Yao, X. Xu, B. Cheng, and J. Ren, "Deep reinforcement learning for pedestrian collision avoidance and human-machine cooperative driving," *Inf. Sci.*, vol. 532, pp. 110–124, Sep. 2020, doi: 10.1016/j.ins.2020.03.105.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *ArXiv150601497 Cs*, Jan. 2016, Accessed: Apr. 20, 2021. [Online]. Available: http://arxiv.org/abs/1506.01497.

[3] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *ArXiv170306870 Cs*, Jan. 2018, Accessed: Apr. 20, 2021. [Online]. Available: http://arxiv.org/abs/1703.06870.

[4] Z. Cai and N. Vasconcelos, "Cascade R-CNN: High Quality Object Detection and Instance Segmentation," *ArXiv190609756 Cs*, Jun. 2019, Accessed: Apr. 20, 2021. [Online]. Available: http://arxiv.org/abs/1906.09756.

[5] Yuxin Wu, Alexander Kirillov, Francisco Massa and Wan-Yen Lo, & Ross Girshick. (2019). Detectron2. https://github.com/facebookresearch/detectron2

[6] A. Mateus, D. Ribeiro, P. Miraldo, and J. C. Nascimento, "Efficient and robust Pedestrian Detection using Deep Learning for Human-Aware Navigation," *Robot. Auton. Syst.*, vol. 113, pp. 23–37, Mar. 2019, doi: 10.1016/j.robot.2018.12.007.

[7] H. Huang, Y. Xu, Y. Huang, Q. Yang, and Z. Zhou, "Pedestrian tracking by learning deep features," *J. Vis. Commun. Image Represent.*, vol. 57, pp. 172–175, Nov. 2018, doi: 10.1016/j.jvcir.2018.11.001.

[8] H. Zhang, M. Sun, Q. Li, L. Liu, M. Liu, and Y. Ji, "An empirical study of multi-scale object detection in high resolution UAV images," *Neurocomputing*, vol. 421, pp. 173–182, Jan. 2021, doi: 10.1016/j.neucom.2020.08.074.

[9] Y. Jin, Y. Zhang, Y. Cen, Y. Li, V. Mladenovic, and V. Voronin, "Pedestrian detection with super-resolution reconstruction for low-quality image," *Pattern Recognit.*, vol. 115, p. 107846, Jul. 2021, doi: 10.1016/j.patcog.2021.107846.

[10] X. Dai *et al.*, "Multi-task faster R-CNN for nighttime pedestrian detection and distance estimation," *Infrared Phys. Technol.*, vol. 115, p. 103694, Jun. 2021, doi: 10.1016/j.infrared.2021.103694.

[11] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," *ArXiv14050312 Cs*, Feb. 2015, Accessed: Apr. 20, 2021. [Online]. Available: http://arxiv.org/abs/1405.0312.

[12] A. Kuznetsova *et al.*, "The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale," *Int. J. Comput. Vis.*, vol. 128, no. 7, pp. 1956–1981, Jul. 2020, doi: 10.1007/s11263-020-01316-z.

[13] S. Zhang, Y. Xie, J. Wan, H. Xia, S. Z. Li, and G. Guo, "WiderPerson: A Diverse Dataset for Dense Pedestrian Detection in the Wild," *ArXiv190912118 Cs*, Sep. 2019, Accessed: Apr. 20, 2021. [Online]. Available: http://arxiv.org/abs/1909.12118.

[14] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *ArXiv180402767 Cs*, Apr. 2018, Accessed: Apr. 20, 2021. [Online]. Available: http://arxiv.org/abs/1804.02767.

[15] Z. Cai and N. Vasconcelos, “Cascade R-CNN: Delving into High Quality Object Detection,” *ArXiv171200726 Cs*, Dec. 2017, Accessed: Apr. 20, 2021. [Online]. Available: http://arxiv.org/abs/1712.00726.

[16] J. Redmon and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” *ArXiv161208242 Cs*, Dec. 2016, Accessed: Apr. 25, 2021. [Online]. Available: http://arxiv.org/abs/1612.08242.

[17] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, “Panoptic Segmentation,” *ArXiv180100868 Cs*, Apr. 2019, Accessed: Apr. 20, 2021. [Online]. Available: http://arxiv.org/abs/1801.00868.

[18] M. Cordts *et al.*, “The Cityscapes Dataset for Semantic Urban Scene Understanding,” *ArXiv160401685 Cs*, Apr. 2016, Accessed: Apr. 23, 2021. [Online]. Available: http://arxiv.org/abs/1604.01685.

[19] B. Zhou *et al.*, “Semantic Understanding of Scenes through the ADE20K Dataset,” *ArXiv160805442 Cs*, Oct. 2018, Accessed: Apr. 23, 2021. [Online]. Available: http://arxiv.org/abs/1608.05442.

[20] G. Neuhold, T. Ollmann, S. R. Bulò, and P. Kontschieder, “The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 5000–5009, doi: 10.1109/ICCV.2017.534.