# PEDESTRIAN DETECTION USING DEEP RESIDUAL NETWORK WITH TRANSFER LEARNING AND FASTER RCNN

A Thesis

Presented to

The Faculty of the Department of Computer Science

California State University, Los Angeles

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

By

Naveen Kumar Subramani

August 2018

The thesis of Naveen Kumar Subramani is approved.

Mohammad Pourhomayoun, Ph. D., Committee Chair

Zilong Ye, Ph. D

Raj Pamula, Ph. D., Department Chair

California State University, Los Angeles

August 2018

ABSTRACT

Pedestrian Detection Using Deep Residual Network with Transfer Learning and Faster RCNN

By

Naveen Kumar Subramani

With the rise of Artificial Intelligence over the past decade, many useful applications have been developed and in use. Computer vision is one of the core areas of Machine Learning. Computer vision could solve a range of tasks in many areas like medical computer vision, manufacturing machine vision, in autonomous vehicles and so on. Pedestrian detection is a challenging problem in computer vision. It is also an important task in video monitoring system. The objective of this thesis is to build a true pedestrian detector. The robust model is based on deep residual neural network using Transfer Learning. Trained model can be used to detect pedestrians on traffic videos in real time.

# ACKNOWLEDGMENTS

I would like to thank Professor Dr. Mohammad Pourhomayoun for his knowledge, advice, patience, and support for this research project. Thanks for giving me the opportunity to be part of the Data Science Research Lab here. Also, I would like to thank the Computer Vision and Deep Learning community for sharing knowledge and resources.

TABLE OF CONTENTS

LIST OF TABLES

Table

LIST OF FIGURES

Figure

CHAPTER 1

Introduction

## 1.1 Background

Road accidents happen all over the place, especially around cities. Pedestrians and cyclists are more prone to accidents than anyone else. Technology could help city planners and Government bodies to come up with more reliable, safe, affordable, environment friendly solutions.

Object detection in a real time video is a promising technology. It is one of the core areas of computer vision and machine learning. Object detection can help solve this problem in many ways like pedestrian detection, face detection, cyclist detection, and human pose estimation and so on.

Pedestrian detection is widely popular these days. Many self-driving cars has this kind of technology pre-installed. At this point of time, it is necessary that surveillance street cameras have this technology. It can be made to detect, count and track. With more available data of demographics information, this technology could even be able to identify people. [1]

There are many types of object detection architectures. It comes down to feature extraction methods. Most of them falls into two categories: Hand engineering based (Classic Machine Learning models) and deep neural networks based (Deep Learning models).

HOG (Histograms of Gradient) + Linear SVM (Support Vector Machines) is a classic Machine Learning approach which falls under hand-engineering method [2]. The OpenCV library has a pre-trained model for person detection. The model is based on

HOG + Linear SVM. It can be readily used to perform pedestrian detection. But there is a problem of false detection with two bounding boxes (overlapping of the same person) and occlusion between pedestrians. Also, when there is motion blur, the HOG Descriptor did not perform well.

Since the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), deep neural networks became widely popular [7]. With the rise of computing power and public datasets, more and more deep learning application have been developed over the years. Convolutional neural network (CNN) is one of the class of deep neural network which is used for learning image data [2]. There are many recent developments in Region based Convolutional Neural Network (RCNN) networks. This is an end to end feature extraction method. In comparison with Region based CNN based network, Faster RCNN performs better than Fast RCNN and RCNN for Object Detection purpose [4]. This thesis also discusses the results produced by this model trained on this network.

Transfer learning is the adaption of pre-trained CNN models to similar tasks in the same domain, by fine-tuning parameters or extracting features of the pre-trained CNN models [8]. The transfer learning approach enables to develop powerful models, by building on the results of top experts in the deep learning field. Pre-trained deep learning image models are available, which are a great building block for an object detector. This thesis focusses on creating a model to detect pedestrians by exploiting complex high dimensional feature of a pre-trained model with a few labeled examples that are available for training.

## 1.2 Research Goal

The goal of this thesis is to create a robust deep learning model that accurately detects pedestrian in an image and video stream data. One of the objectives of this research is to build a deep learning model understands the difference between pedestrian and non-pedestrian person in an image. This thesis discusses the infrastructure, development environment, dataset preparation, developed algorithms and methods, training and deploying procedures, and other challenges.

A deep residual network is built using Transfer Learning. The model detector is originally built to detect pedestrians. It is possible to build any type of Object Detection model based on the idea proposed in this thesis. The datasets and libraries are publicly available for academic use. GPU computing source is mandatory for this thesis to efficiently work with different deep neural architectures.

# CHAPTER 2

# Deep Learning

## 2.1 Convolutional Neural Network

For an Object Detection problem, in traditional machine learning method like HOG + SVM, the features are extracted using hand- engineering methods [1]. It follows a divide and conquer approach. In Convolutional Neural Network or generally Deep Learning models, these features are learned by deep neural networks. It follows an end to end approach in training the process. Convolution Neural Networks works well for Image and Video data [12]. It has seen unprecedented results. Convolutional Neural Network achieves good results in Image classification and object detection problems. Object detection is Image classification + Localization. [3]

For CNN, input is an image and output are bounding box(coordinates) and labels (classification names). An image is a 2d array of pixels. These numbers are the input to the convolutional layers. To classify and label, each image must go through multiple layers of filters, pooling layers, fully connected layers and SoftMax function for training, validating and testing. ResNet is one of the popular CNN architecture with good results [9]. In this thesis, an adapted ResNet architecture is used.

### 2.1.1 Faster RCNN

Faster RCNN (Faster Region based Convolution Neural Network) is a CNN which is recently published model and it works well for the Object Detection problem in real time [9]. Many Object detecting models are built based on Faster R-CNN.

Faster RCNN is originally derived from its parent Region based Convolutional Network (RCNN). RCNN works on the principle of proposing regions by selective

search and compute CNN on those regions [4]. The drawback of RCNN is that, network must be trained 3 times individually for feature extraction (CNN), regression (regressor) and classification (SVM), which is expensive and slow. The immediate successor to RCNN is Fast RCNN. Fast RCNN eliminates this problem training all three networks (feature extraction, regression and classification) combined. But, the region proposal method in Fast RCNN is still slow and Faster RCNN addresses this problem. [5]

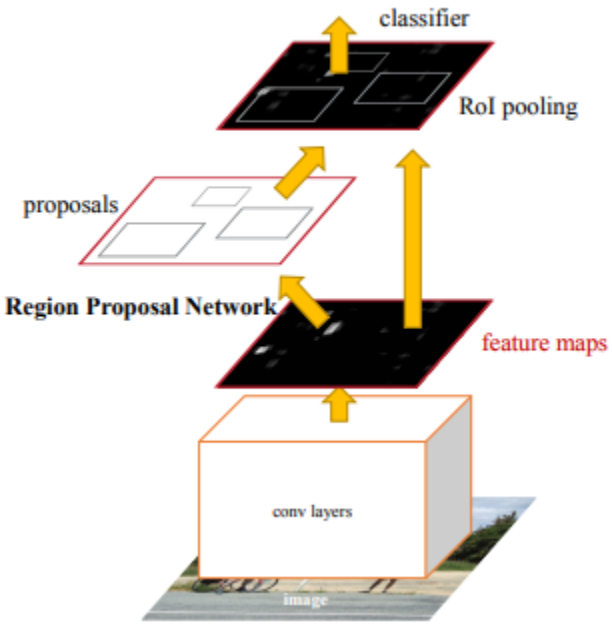



*Figure 1*. Faster RCNN (From [6])

In Faster RCNN, selective search is replaced by CNN itself. In *Figure 1*, it is shown that, CNN is applied for Region Proposal. Again, the same CNN layers are used for Classifier. This makes the whole process faster. Even though this method is faster than all previous models and higher accuracy is achieved. [6]

The pre-trained Faster RCNN model is trained on the Microsoft Common Objects in Context (MS COCO) dataset. The MS COCO dataset consists of 90 classes. It includes

person as one of the class but not pedestrian. It has bicycle as one of the class but not cyclist. [13]

### 2.1.2 Deep Residual Network

Deep networks often face vanishing gradient problem. Depth of layers is important in deep learning. Residual networks solve this problem by mapping a residual function as H(x) = F(x)+x, whereas in a plain block, the function is mapped as H(x) = F(x). It reuses the weights and learns from the previous layer [9] which solves the degrading problem.

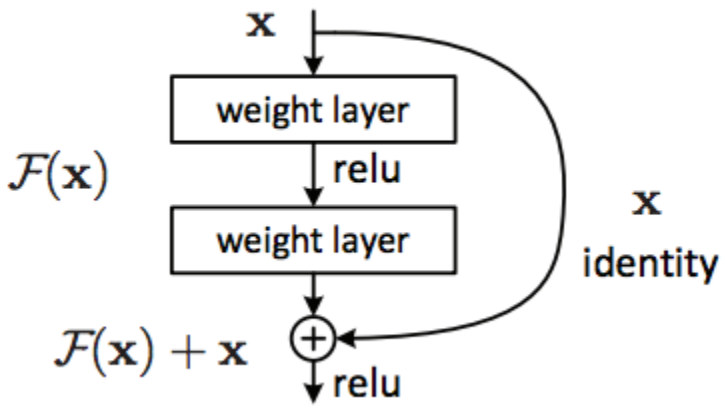



*Figure 2*. Building Block of Residual Networks (From [9])

It makes training time less, which also helps in parallel computations of huge multi-dimensional arrays of numbers (tensors).

## 2.2 Transfer Learning

### 2.2.1 Challenges

In this thesis, there are multiple constraint to train a pedestrian model.

1. Resource Constraint

   It takes multiple GPUs to train a deep network. To compute millions of parameters is simply impossible with a regular CPUs. It is discussed in later chapter in detail about the challenges faced in training

2. Time Constraint

   Even with multiple GPUs, to train a deep neural network, it will still take weeks of just training. It leaves us limited time to experiment with our model.

3. Data Constraint

   The biggest bottleneck for this thesis is data. There are only few labelled pedestrian datasets available for academic use. There are almost no true cyclist datasets for academic use from computer vision community. The problem with these datasets is they are too specific. A general, diverse dataset is hard to find, unless. With few labelled data, and a deep network, model is going to overfit. The model must generalize.

### 2.2.2 Why Transfer Learning?

Data makes Deep Learning possible. More success from Machine Learning comes from supervised learning which means more labelled data. Transfer Learning (TL) is the process of transferring weights/parameters, essentially knowledge from a domain to another domain. It transfers weights of a model from a domain space to a similar domain

space to execute a different job [8]. The advantage comes in not only in not training a very large dataset but also making it possible to train in a short time and more experiments. The other main advantage is transferring knowledge from one domain to another while retaining features.

There are many Transfer Learning methods, one such is using a pretrained Convolutional Neural Network model [8] which is different from other methods. A similar approach is carried out to train the deep neural networks. In the pre-trained model of CNN, there are many low-level features that can be retained. “person” is one of the 90 class in the pre-trained mode [9]. The low-level features of a person can be transferred to a “pedestrian” class. Now, it makes it easy to tune high-level features. Parameters and feature representations can be transferred. Feature Extractor and Fine Tuning can be done.

This thesis is mainly focused on applying Transfer Learning in feature extractor. Using a Faster RCNN with ResNet-101 pre-trained model feature is extracted, and a classification layer is added at the end to detect “pedestrian” class. The pre-trained model is trained on 90 different classes on MS COCO dataset. “person” is one among the class [13]. Since my model is trying to detect pedestrian, much of the knowledge from a model trained on a similar domain is helpful. It is helpful in saving resource and time. Although, experiments were done in fine tuning the last few layers by modifying last few layers (high level features) in the pre-trained model, there was not any significant change in results.

CHAPTER 3

Method and Procedures

## 3.1 Development Environment

Any Deep learning models consume huge compute power. In this thesis, both CPUs and GPUs are used where ever necessary.

For Hardware dependencies, Google Cloud Platform is used for distributed learning. The platform has access to multiple GPUs and other infrastructures like storage disks and Machine learning APIs. It's available for 24x7.

For Software dependencies [13], a list of libraries, frameworks and development kit with versions that are required is shown in Table 1. *Software Dependencies.*

Table 1.

*Software Dependencies*

| **Libraries** | **Version** | **Purpose** |
|---|---|---|
| TensorFlow | 1.4 and above | For computation |
| Python | 3 and above | Programming Language for TensorFlow |
| Google Cloud SDK | latest | Interface between Cloud and workstation |
| Object Detection API | latest | To configure and deploy trained models |
| Jupyter Notebook | latest | For visualization and numerical simulation |

### 3.2 Dataset Preparation

#### 3.2.1 MS COCO Dataset

The Faster RCNN model which is trained on Microsoft Common Objects in Context (COCO) dataset has about 90 classification names. The MS COCO dataset bagged the first place in 2015 object detection competition. [7] [11]. The frozen weights from a pre-trained model on MS COCO datasets is used. These frozen weights and learning parameters are transferred into a different set of pedestrian datasets and thus producing a new model that detects a pedestrian class.

#### 3.2.2 KITTI Dataset

Dataset is the most crucial piece in machine learning. There are many pedestrian datasets available publicly for academic use. In this thesis, KITTI dataset [16] are used for training and validating. KITTI datasets comes with RGB images with 512 x 512-pixel dimensions [10]. These datasets have well annotated, labelled data for Object detection. The overall performance of the dataset using a Faster RCNN is pretty good [10]

The dataset has a list of labelled data for the image. Each labelled data has bounding box coordinates and classification names.

Some of the testing dataset is provided by LA city streets video cameras. Video camera data from mission road and alameda station road is used. Also, pictures from Google images is also used to test the model under any environment.

### 3.2.3 Preprocessing Dataset

Unlike the traditional machine learning algorithms for image classification, CNN architecture-based models need less pre-processing of data. Some of the images which are blurry, cut off in the edges. These images were cleaned in the dataset.

Using ImageDataGenerator class from Keras library, new data is generated by transforming and enhancing the original dataset for reflections and color intensities. []. However, the original labelled data is still saved for these images which saves time and effort.

The final statistics of data used for this thesis is in Table 2. *Dataset Image Statistics*

Table 2.

*Dataset Image Statistics*

| **Dataset** | **Size** |
|---|---|
| KITTI Original Dataset | 7480 |
| After Data Cleaning | 7050 |
| After Data Augmentation and labelling | 7180 |
| Training | 6710 |
| Validation | 500 |

### 3.2.4 TF Records

TensorFlow uses a TF Record data format for its computation. The most important reason for converting existing dataset into a TF Record format is faster

computation. It converts any type of data into a serialized binary format. In this thesis, there is both image and floating-point integer data type are discussed. Labels are list of floating-point integer type. Converting to TF Record makes it easy handle both the data types. *Figure 3*. Describes the flowchart of generating TF Records

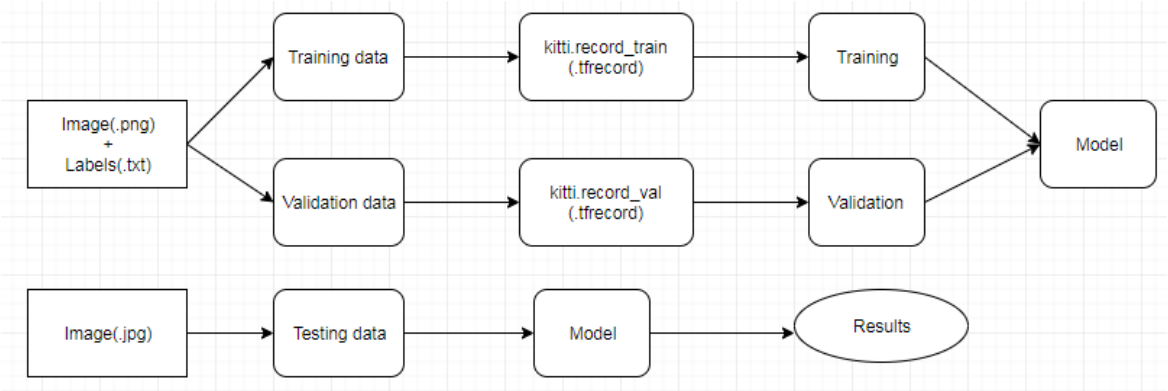



*Figure 3*. Flowchart of the dataset model

### 3.3 Training and Validation

#### 3.3.1 Network Architecture

In this thesis, an adaptive ResNet-101 architecture is used. ResNet is one of the earlier CNN architecture built in Deep Learning era that was using hardware accelerators. It produced promising results in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) with Top-5 error rate of 3.6% [9]. The network is also known for performing well in detection, localization, and segmentation problems. The previous chapter also discusses hoe degrading problem and efficiency of using ResNet architecture. It has about 152 layers.

#### 3.3.2 Model Configuration

The config file has the all hyperparameters and other pipeline configuration. It has the model parameters, training parameters and eval parameters for training and

evaluation. Table 3. Shows the configuration settings. The Model section has the number of classes to be configured and set to 1. The learning rate is initially set to a 0.001 and as the training process approaches towards the minimal loss learning rate shortens. Total number of steps is configured to 80000.

Table 3.

*Model Hyper Parameters*

| **Parameter** | **Value** |
|---|---|
| num_clases | 1 |
| batch_size | 1 |
| initial_learning_rate | 0.001 |
| learning_rate at Step:50000 | 0.0001 |
| learning_rate at Step:70000 | 0.00001 |
| num_steps | 80000 |

### 3.3.3 Challenges

The main challenge was high computational complexity of training process. Initially, to train the model, CPU with intel i7 processor of 4 GB RAM configuration is used. It was really time consuming to train and compute huge parameters. With original ResNet-101 architecture, this would have been even more difficult to train millions of parameters and weights. The configured model has 80000 steps in total to successfully train the model. The time taken for a single step is about 10 minutes, which is highly

inefficient. *Figure 4*. Shows the time taken for a single step in CPU, which is about 10 minutes.


*Figure 4*. Training model in CPUs

### 3.3.4 GPU Training

After experiencing an inefficient performance from CPUs, it was a obvious decision to train it on Google Cloud Platform (GCP). A GCP project is created with necessary APIs, Storage and Development environment. All the necessary files were uploaded to cloud and the model was set to train in multiple GPUs in the Google Cloud Platform. *Figure 5*. Shows all the necessary files. GPU was configured with ML engines to use 1 master and 4 worker servers for distributed training on cloud.


*Figure 5*. List of Files in the Storage disk

The total time that took for training and evaluating was about 8 hours. Table 4. Shows steps taken and loss at that stage of training. Loss was converging towards in the end with less than 1%.

Table 4.

*Iteration Count and Loss*

| **Iteration Count (Steps)** | **Loss** |
|---|---|
| 1 | 1.9404 |
| 261 | 1.771 |
| 1236 | 1.1483 |
| 44021 | 0.235 |
| 60871 | 0.0698 |
| 79989 | 0.0407 |

### 3.3.5 Exporting Model

The final step of training this model and the first step before experimenting with this model is to save and export inference graph. *Figure 6.* Shows how the training process ends successfully and model is saved to disk.


*Figure 6.* Training Model

The platform allows to save the model in multiple checkpoints through the training process. I chose latest checkpoint in the training model at step 80003 and exported it. *Figure 7.* Shows the model check point for export. The checkpoint usually consists of these three files.


*Figure 7.* Model Checkpoint

Google Cloud SDK allows to export the graph from cloud to local machine, using export_inference_graph.py file from Object Detection API, the

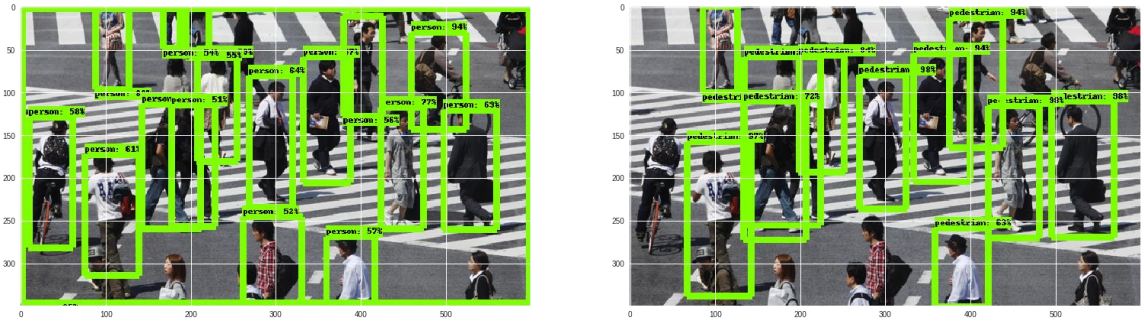
Image tensor data in the model check point is exported as frozen inference graph. At this point, the model graph is saved, exported, frozen and ready to experiment.

### 3.4 Results

This Chapter contains the Results from pre-trained model using Faster RCNN with ResNet-101 trained on MS COCO dataset and an adapted Faster RCNN model that is trained on KITTI dataset in this thesis is compared. After loading the model in Object detection API, the detector generates detection results for each image with a bounding box. In the *Figure 8*. (a) Shows the results of Faster RCNN Pre-trained Model. The model detects person class, bicycle class and a car class, among the 90 classification names. The main problem with this model is that it cannot differentiate/classify person from pedestrian. The model works well to classify between a person and a bicycle, but it does not solve the problem to differentiate/classify between person and pedestrian

(a) (b)

*Figure 8*. Pedestrian, Cyclist, and Car (a) Faster RCNN Model (b) Transfer Learning Model with ResNet (From [14])



(a) (b)

*Figure 9.* Pedestrians, Crowd, and Cyclist (a) Faster RCNN Model (b) Transfer Learning Model with ResNet (From [15])

However, In the *Figure 8*. (b)Transfer Learning Model trained in this Thesis, the model detects for only pedestrian class and ignores all other classes. Also, the problem of the existing model that cannot differentiate/classify non-pedestrian person from pedestrian is solved in this thesis. Similar results can be seen in *Figure 9*. with Pedestrians, Crowd, and Cyclist. Since truncated images were removed in our dataset for training process, the model doesn't detect truncated image boundaries.

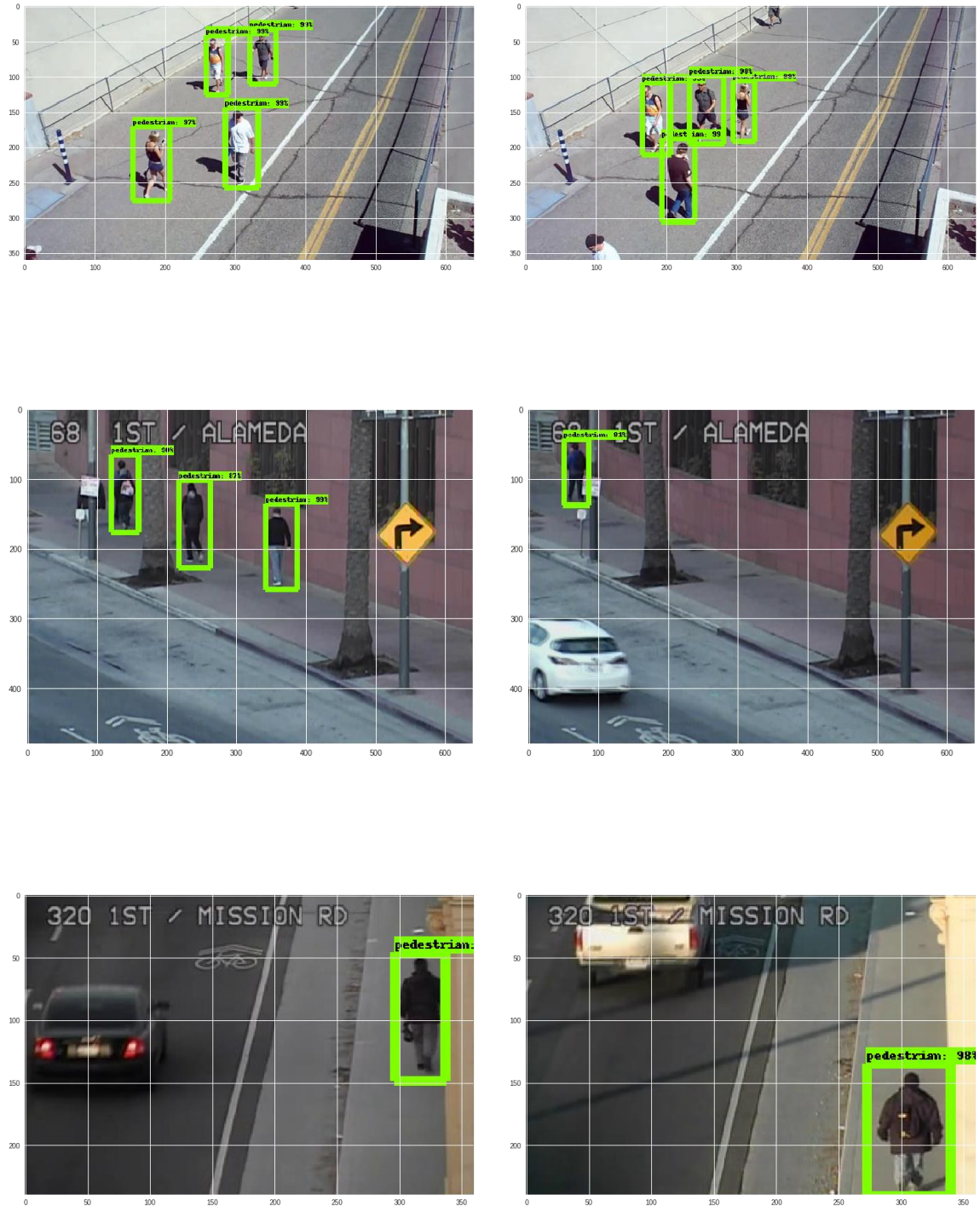



*Figure 10*. Results from different Street Cameras

The Transfer Learning model also produced promising result in detecting pedestrians in LA City Street Cameras. Results are shown In *Figure 10*.

# CHAPTER 4

## Conclusion

A true pedestrian detector in a short space of time and computing resource. The model learned well using the frozen weights. It is a robust model that can classify a pedestrian from a non-pedestrian person.

The built model performs well for general with some false positives by detecting cyclists as pedestrian. To make it more robust, to completely remove false positives and even detect truncated images. In future we will work on more customized datasets for person, pedestrian, cyclists, bicycle and lanes.

The built model is based on a general pipeline. The same pipeline and configurations can be used to build to detect any arbitrary object. Even more a robust performance can be achieved with customized datasets for pedestrians and cyclists. As part of future work, I would like to build a robust model that will detect cyclists and pedestrians and is able to detect cyclist on the bicycle lane, person walking with cycle on the pedestrian lane and cyclist on the pedestrian lane. With future advancement in computing power (GPUs, TPUs) and customized datasets, it is possible to experiment more accurate results in Object Detection and Computer Vision domain in the coming years.

REFERENCES


1. C. Yuanyuan, G. Shuqin, Z. Biaobiao, D. K.-L (2013). A Pedestrian Detection and Tracking System Based on Video Processing Technology. *2013 Global Congress on Intelligent Systems.* doi: 10.1109/GCIS.2013.17
2. R. Benenson, M Omran, J. Hosang, B Schiele (2014). Ten years of Pedestrian detection, what have we learnt? *European Conference on Computer Vision 2014.* doi:10.1007/978-3-319-16181-5_47
3. P. Dollar, C. Wojek, B. Schiele, P. Perona (2011). Pedestrian Detection: An Evaluation of the State of the Art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. [Online]. 34(4), pp. 743-761. doi:10.1109/TPAMI.2011.155
4. R. Girshick, J. Donahue, T. Darrell, J. Malik (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*. doi:10.1109/cvpr.2014.81
5. R. Girshick, (2015). Fast R-CNN. *2015 IEEE International Conference on Computer Vision (ICCV).* doi:10.1109/ICCV.2015.169.
6. S. Ren, K. He, R. Girshick, J. Sun (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* [Online]. 39(6), pp. 1137-1149. doi:10.1109/TPAMI.2016.2577031
7. K. Alex, S. Ilya, H. E. Geoffrey (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems, 1097-1105.* doi:10.1145/3065386.

8. P. J. Sinno, Y. Qiang (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering.* [Online]. 22(10), pp. 1345-1359 doi:10.1109/TKDE.2009.191
9. K. He, X. Zhang, S. Ren, J. Sun (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition.* doi:10.1109/CVPR.2016.90
10. Zhang, Shanshan & Benenson, Rodrigo & Schiele, Bernt. (2017). CityPersons: A Diverse Dataset for Pedestrian Detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition.* doi: 10.1109/CVPR.2017.474.
11. T. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, P. Dollár (2015). Microsoft COCO: Common Objects in Context. *arXiv Preprint arXiv:1405.0312.* Retrieved from: arxiv.org/abs/1405.0312v3
12. I. Goodfellow, Y. Bengio, A. Courville. *Deep Learning*. Boston: MIT Press, (2016)
13. TensorFlow model resources, https://github.com/tensorflow/models
14. *Transportation*, Fred Ryan, 2017 [Online]. Available: https://www.washingtonpost.com/local/trafficandcommuting/waymos-ceo-on-fatal-autonomous-uber-crash-our-car-would-have-been-able-to-handle-it/2018/03/25/4cc97550-3046-11e8-8abc-22a366b72f2d_story.html?utm_term=.894097bc4644. [Accessed: 02-Jul-2018]
15. *Pedestrian Perfection*. Hayley Romer, 2011 [Online]. Available: https://www.theatlantic.com/national/archive/2011/05/pedestrian-perfection-the-11-most-walk-friendly-us-cities/238337/. [Accessed: 02-Jul-2018]
16. KITTI datasets resources, http://www.cvlibs.net/datasets/kitti/eval_object.php