ABSTRACT

DESAI, HARDI. Light Weight Single-Shot Refinement Neural Network for Object Detection. (Under the direction of Dr Tianfu(Matt) Wu.)

The performance Deep Neural Networks (DNNs) which improves by the increase in hidden layers inevitably results in over parameterized models, large memory consumption and huge amount of data for training. Wide-scale deployment of object detection networks has been limited due to millions of parameters in the object detection models. In this work, we present a filter correlation based pruning methodology for building a compressed object detection model while maintaining a competitive accuracy. It is a one-shot pruning algorithm with the effective steps of train, prune and retrain. The correlation between filter pairs is increased by regularizing the trained model by a novel loss function before removing one filter from the selected filter pair. We achieve a compressed and lightweight object detection model by pruning the connections as well as weights in the model which could be easily deployed for real time performance without the requirement for any special hardware or software.
Light Weight Single-Shot Refinement Neural Network for Object Detection

by

Hardi Desai

A thesis submitted to the Graduate Faculty of North Carolina State University in partial fulfillment of the requirements for the Degree of Master of Science

Electrical Engineering

Raleigh, North Carolina

2019

APPROVED BY:

Dr Edgar Lobaton

Dr Kevin Han

Dr Manasa Kolla
External Member

Dr Tianfu(Matt) Wu
Chair of Advisory Committee
DEDICATION

To my mentors, grand parents, parents and brother.
BIOGRAPHY

Author is a Master’s of Science student in the Department of Electrical and Computer Engineering at North Carolina State University. She pursued her Masters Thesis under the guidance of Dr. Tianfu Wu. She has collaborated with Conduent Labs (Formerly Xerox Research Center) for her thesis mentored by Dr Manasa Kolla. She was working as an Computer Vision Research Intern at Conduent Labs during this research work. Prior to joining NCSU, Hardi graduated with a Bachelors in Technology in Electronics and Telecommunication Engineering from K.J. Somaiya College of Engineering (Autonomous College under University of Mumbai), India.
ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis advisor Dr. Tianfu Wu and my mentor Dr Manasa Kolla. I am very grateful to them for cooperation, continuous support and patience throughout the thesis. I would like to mention that without Dr Manasa’s valuable guidance and advice, this work would not have been possible. She has been instrumental in keeping me motivated throughout the course of this thesis. I would also acknowledge my colleague at Conduent, Ms Aishwarya Gupta who has been an integral part of this research. Her support and guidance remain invaluable.

I am also indebted to my family and friends at NC State for their encouragement, motivation and endless support.

Last, but not the least, I would like to thank the faculty and staff at NC State and all my co-workers at Conduent who have made my journey memorable.
TABLE OF CONTENTS

LIST OF TABLES ........................................................................................................ vii

LIST OF FIGURES ......................................................................................................... viii

Chapter 1 Introduction ................................................................................................. 1
  1.1 Introduction ........................................... 1
  1.2 Problem Statement ............................ 2
  1.3 Vehicle Passenger Detection System .... 2
  1.4 Object Detection: Challenging Task ... 3
  1.5 Building Compact Object Detection Architecture ... 4
  1.6 Contribution of Thesis ....................... 4
  1.7 Organization of Thesis ...................... 5

Chapter 2 Literature Survey ......................................................................................... 7
  2.1 Object Detection Algorithms .................. 7
    2.1.1 RCCN ........................................... 8
    2.1.2 Fast RCNN .................................... 8
    2.1.3 Faster RCNN ................................ 8
    2.1.4 Deep Dense Face Detector ............... 9
    2.1.5 Single Shot Detector(SSD) ............. 9
    2.1.6 Retina Net .................................. 10
    2.1.7 Couple Net .................................. 10
    2.1.8 RefineDet .................................. 11
  2.2 Literature related to our Model .......... 11
    2.2.1 Leveraging Filter Correlations for Deep Compression Models ... 13

Chapter 3 DataSet Description ................................................................................. 14
  3.1 Dataset Collection and Pre-processing .............................................................. 14
  3.2 Dataset Description and Challenges ................................................................. 15
  3.3 Annotation .............................................. 17

Chapter 4 Proposed Approach .................................................................................... 19
  4.1 Introduction ............................................. 19
  4.2 Terminology ............................................ 20
  4.3 Selection of Correlated Filter Pairs ................................................................. 21
  4.4 Optimizing Selected Correlated Filter Pairs .................................................... 22
  4.5 Selection of Filters for Pruning ................................................................. 23
  4.6 Re-training .............................................. 26
<table>
<thead>
<tr>
<th>Chapter 5</th>
<th>Implementation Details</th>
<th>27</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Timeline</td>
<td>27</td>
</tr>
<tr>
<td>5.2</td>
<td>Methodology</td>
<td>28</td>
</tr>
<tr>
<td>5.3</td>
<td>Hyper Parameters</td>
<td>28</td>
</tr>
<tr>
<td>5.4</td>
<td>Hardware and Software Resources</td>
<td>29</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 6</th>
<th>Results</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Evaluation Metrics</td>
<td>30</td>
</tr>
<tr>
<td>6.2</td>
<td>Results on Caffe Trained Model</td>
<td>31</td>
</tr>
<tr>
<td>6.3</td>
<td>Results on Pytorch Trained Pruned Model</td>
<td>32</td>
</tr>
</tbody>
</table>

| Chapter 7  | Conclusion and Future Work                  | 37 |

| BIBLIOGRAPHY |                                              | 38 |
LIST OF TABLES

Table 3.1  Data Distribution for Test Set ................................. 16
Table 3.2  Data Distribution for Training Set .......................... 16
Table 3.3  Annotation Statistics ........................................... 17

Table 6.1  Results as classifier on Test Dataset using the Classifier Groundtruth . 31
Table 6.2  Results as classifier on Test Dataset using the Object Detector Groundtruth ......................................................... 32
Table 6.3  Results as Object Detector on Test Dataset - Trained in Caffe .......................... 33
Table 6.4  Results of Correlation based One Shot Filter Pruning and Fine Tuning 35
Table 6.5  Results of Correlation based One Shot Filter Pruning when Retrained from Scratch ......................................................... 35
Table 6.6  Results of Correlation based One shot Filter Pruning when Re Trained from Scratch for Pascal VOC ................................. 36
**LIST OF FIGURES**

| Figure 1.1 | Object Detection: Challenging Task [Sta] | 4 |
| Figure 2.1 | SSD Architecture [Liu16] | 10 |
| Figure 2.2 | RefineDet Architecture [Zha18] | 12 |
| Figure 3.1 | Challenges in Dataset | 16 |
| Figure 3.2 | Test Images | 18 |
| Figure 4.1 | Base Network | 21 |
| Figure 4.2 | Pruned Network | 21 |
| Figure 4.3 | Step 1: Selecting Filters to be pruned | 24 |
| Figure 4.4 | Step 2: Selecting Filters to be pruned | 24 |
| Figure 4.5 | Step 3: Selecting Filters to be pruned | 25 |
| Figure 4.6 | Step 4: Selecting Filters to be pruned | 25 |
| Figure 6.1 | Inconsistency in Annotation | 32 |
| Figure 6.2 | Results | 34 |


1.1 Introduction

The field of computer vision started as an undergrad summer project at MIT in late 1960's, to mimic the human visual system, today, it has evolved as a major research area in Artificial Intelligence. And in spite of decades of hard work, researchers are still struggling to achieve the human-level understand-ability and interpret-ability. Deep learning based algorithms have surpassed human accuracy for tasks like Image Classification on ImageNet Data set with an incredibly low error rate. However, algorithms still do not understand the image and are unaware of the properties of object and its interaction with the surrounding
objects. Hence, tasks like Object Detection and Visual Question Answering are still an open research problem. Presently, major challenges for deep learning based computer vision algorithms for different tasks include the performance on the unclean data [DK17], real-time performance and compact models for wide-scale deployment.

1.2 Problem Statement

This thesis research started as exploration of Object Detection Algorithms for the project of Vehicle Passenger Detection System (VPDS) for Conduent. It is an attempt to develop a deep learning based object detection algorithm to detect passengers from monochrome profile face images captured from rear view side images of vehicle in motion. Aim of this master's thesis is to built a compressed object detection algorithm which could produce results in real-time for deployability. According to literature most state-of-the-art deep neural networks are over parameterized [Han16]. Solving this problem for data-hungry neural network is of high significance given the non-availability of data. Our objective is to propose a compact and light weight algorithm which minimizes the empirical parameters while still maintaining competitive accuracy.

1.3 Vehicle Passenger Detection System

High Occupancy Tolls (HOT) or High Occupancy Vehicle (HOV) lanes encourage people to share rides. This helps the transportation agencies to effectively manage congestion and regulate harmful emissions. Current system is based on voluntary compliance by the driver. It has been observed that the majority of these declarations are false to escape tolls and manual system is deficient in detecting the violators. Therefore it is inevitable for an
automated vehicle passenger detection system to effectively enforce the HOV/HOT lanes and thereby managing congestion.

1.4 Object Detection: Challenging Task

Object detection algorithms in literature involved extracting hand crafted features such as HOG, SIFT, SURF and applying powerful machine learning algorithms like SVM for classification. In spite of being highly researched topic, these methods were outperformed by deep convolutional neural networks(CNN) for object detection. The advent of deep neural networks resulted in extinction of these conventional algorithms.

Unlike classification, object detection can not be solved by using standard convolutional neural networks followed by fully connected layer due to the uncertainty in number of occurrences of each object of interest in a given image. A straightforward solution to this problem is to select different regions of interest and then use a CNN to classify if the object is present in the region of interest. Object Detection algorithms should not only detect the presence of the object, but also localize the object. In order to localize the algorithm needs to select from a number of possible regions. The possibility of number of regions is enormous because objects of interest could have different aspect ratio, size and spatial location in the image. The challenges involved in developing an object detection algorithm can be clearly observed from Fig 1.1 where an image has multiple objects with each object having one or more instances in an image.
1.5 Building Compact Object Detection Architecture

Wide-scale deployment of object detection networks has been limited due to millions of parameters in the object detection models. Given the challenges of object detection task, object detection networks have to focus not only on predicting the presence of an object in the image but also the exact location of the object. This leads to fairly complex architecture for object detection networks. Maintaining the accuracy while reducing the parameters of the network remains an open research problem.

1.6 Contribution of Thesis

The contribution of thesis is as follows:

- This work proposes a methodology to built a compressed and light weight object detection model for producing results in real-time by pruning the connections as well as the weights in the model.
- This work encourages pairwise filter correlation based pruning of object detection
algorithm, which namely refers to Single-Shot Refinement Neural Network. It introduces a novel loss function as regularizer for increasing filter correlation before pruning.

• This work empirically demonstrates the presence of redundant filters in current state-of-the-art object detection algorithms and it is possible to achieve equal or higher accuracy with an object detection model with reduced parameters.

• This work is effective exploration of one-shot pruning algorithm for Object Detection

1.7 Organization of Thesis

The thesis has seven chapters. Following paragraphs provide concise information about each chapter.

• Chapter 1 defines the aim of this thesis research with an introduction to the Object Detection Algorithms and Vehicle Passenger Detection System.

• Chapter 2 comprises of a extensive overview of different Object Detection Algorithms. It provides a general overview which led to formulation of the approach proposed in this thesis.

• Chapter 3 describes the data set. It starts with the explanation about the data collection and pre-processing and further discusses the challenges in data.

• Chapter 4 discusses the proposed approach for compressing the Single-Shot Refinement Neural Network.

• Chapter 5 elaborates on the timeline followed, methodology adopted and hyper-parameters selected as a part of this research.
• Chapter 6 consists of an overview of performance metrics and the results.

• Chapter 7 presents conclusions and summary of the research. This chapter also discusses the areas that may require further study and investigation. It is followed by references.
2.1 Object Detection Algorithms

Object Detection is one of the most fundamental problems in computer vision. Object Detection Algorithms can be broadly classified into two different categories - Single Stage Detector and Two Stage Detector. Single stage detectors like Single Shot Detector [Liu16] make fixed number of predictions on grid of different possible region-of-interest(RoI). Two stage detectors have a region proposal network to find the objects and a second network which fine tunes this proposals and outputs the final prediction. Popular examples of two stage detectors include like RCNN, Fast RCNN and Faster RCNN.
In the following section, various approaches used for object detection are discussed.

### 2.1.1 RCCN

Regions with CNN features (RCNN) uses a selective search algorithm to come with 2000 regions. Each of the selected 2000 regions are then passed to convolutional neural net (Alex Net) resulting in 4096 dimensional feature vector. With CNN as feature extractor, the algorithm further uses Support Vector Machine (SVM) in order to classify whether the object is present or not. Along with the prediction of the object in the candidate region, this algorithm also predicts the offset to bounding box. This increases accuracy in case of occlusion. At time of its arrival it produced state-of-art results in terms of accuracy but it could not be implemented in real time due to huge amount of processing time. [Gir14]

### 2.1.2 Fast RCNN

The author of RCNN tried to solve the time efficiency of RCNN. Instead of using the convolutional neural network for each of the 2000 region proposals, Fast RCNN framework shares the convolutional neural network for all the 2000 selectively searched regions. It also replaces SVM classifier with a softmax layer on top of the CNN to output a classification. Maintaining the accuracy of RCNN, Fast RCNN has less execution time. [Gir15]

### 2.1.3 Faster RCNN

The major bottleneck in terms of execution speed of Fast RCNN is generation the 2000 region proposal which is done using the selective search algorithm. So, in Faster RCNN the selective search is replaced by a Region Proposal Network. Region Proposal Network uses a sliding window over the CNN feature map. For every window, it produces $k$ different
bounding box along with its score which indicates how good the bounding box is expected to be. [Ren15]

2.1.4 Deep Dense Face Detector

With an aim to built a single deep convolutional neural network based classifier for face detection from multiple views, researchers from Yahoo proposed the Deep Dense Face Detector. Built on fine tuned AlexNet architecture on AFLW architecture, this algorithm uses sliding window approach to obtain the final predictions. The network consists of 8 layers with first 5 convolutional layers and 3 fully connected layers. For the localization of the detected regions, non max suppression is used. Images are scaled up and down to detect faces at different sizes. [Far15]

2.1.5 Single Shot Detector (SSD)

Single Shot Detector (SSD) is a single stage detector. The key idea is to spread out the default bounding boxes of different scales and aspect ratios to multiple layers of different resolutions within a single convolutional neural network. This is clearly observed in SSD architecture diagram depicted in Fig. 2.1. This would handle objects of different scales naturally. Similar to Faster RCNN, the detector generates the confidence score for the presence of the object in the each of the default boxes along with the adjustments to the default box for better accuracy. For class imbalance issue, SSD uses Online Hard Negative Examples Mining (OHEM) [Shr16]. SSD, when introduced, proved to be desired choice for object detection due to its excellent speed-vs-accuracy trade-off. [Liu16]
2.1.6 Retina Net

As most single stage detector face the issue of class imbalance issue, Retina Net focused over addressing this issue using Focal Loss and thereby improving the performance of object detection algorithms. It is a single stage detector consisting of a backbone convolutional neural network for feature extraction along with two sub modules. The first sub module is for the object classification on the backbone's output and second one is for convolutional bounding box regression. They prove that Focal Loss is more effective than OHEM. [Lin18]

2.1.7 Couple Net

While most object detection algorithms as discussed above focus on local features, Couple Net structure of convolutional neural network proposes to use global as well as local features for object detection, thereby introducing contextual information. In this algorithm, the object proposals obtained by the Region Proposal Network (RPN) are fed into the the coupling module which consists of two branches. One branch adopts the position sensitive Region of Interest (PSRoI) pooling to capture the local part information of the object, while the other employs the RoI pooling to encode the global and context information. Next,
different coupling strategies and normalization ways are adopted to make full use of the complementary advantages between the global and local branches.[Zhu17]

2.1.8 RefineDet

Built on SSD architecture, Single-Shot Refinement Neural Network (RefineDet) is current state-of-art single stage object detection network. It consists of two interconnected modules - Anchor Refinement Module (ARM) and Object Detection Module (ODM). Anchor Refinement Module is responsible for filtering out negative anchors for reduction in the search space for the classifier and coarsely adjusting the locations and sizes of anchors to provide better initialization for the subsequent regressor. Object Detection Module detects the object by improving the regression of ARM module and predicting the multi-class label. The network has Transfer Connection Block for transferring the features from ARM Module to ODM Module. The whole network is trained end-to-end using the multitask loss function. The is explained by the Fig. 2.2 which explains the architecture diagram for RefineDet.[Zha18]

2.2 Literature related to our Model

RefineDet[Zha18] provided a significant improvement over Single Shot Detector (SSD)[Liu16] by separating the Anchor Refinement Module and Object Detection Module for prediction. This work is built on RefineDet architecture for object detection.

It has been researched that we have significant number redundant connections in complex deep neural networks. The extra connections in deep neural networks builds on computational complexity and is also responsible for over-fitting the trained model. More recently, [FC18] suggested that deep neural networks comprises of sub networks which
when trained from scratch in isolation can achieve the same accuracy as original network.

Notable neural network pruning technique was proposed by Song Han and Et al. by suggesting Dense-Sparse-Dense [Han16] approach for Deep Neural Networks. [Xie17] proved the presence of redundant filters in SSD by visualization of filters using the deconvolutional method [ZF14]. [AK17] also demonstrated a method for channel reduction for real-time inference with a reasonable accuracy drop. Methods for pruning neural networks sometimes require special hardware or sparse libraries. This research is more focused towards formulating ideas which could prune neural networks without any special requirements. [Sin18] suggests an approach to prune neural networks by removing highly correlated filters. Another noteworthy framework for filter pruning is discussed in [Sin19] where sparsity is induced in the model in consecutive channels before filter selection and filter pruning.

As this research is inspired by [Sin18], this method is discussed in detail in the following
2.2.1 Leveraging Filter Correlations for Deep Compression Models

In this paper, the authors present a novel method for pruning of deep convolutional neural networks. They propose to discard filters based on pair wise filter correlation. The trained model is optimized to increase the pair wise filter correlation for selected filters before discarding one of the filter from the pair. The idea here is to transfer the information from one filter to other by highly correlating the filter pair. This leads to very little information loss while pruning. Post discarding one filter from each pair per layer, the model is fine-tuned in order to regain the original accuracy. It is an iterative pruning approach which is continued till a significant accuracy drop is not observed.
CHAPTER 3

DATASET DESCRIPTION

3.1 Dataset Collection and Pre-processing

Dataset consists of rear view side images of vehicle captured from the deployment site with an vehicle speed ranging from stop to 100 mph. Previous work included extracting the Region of Interest - ROI (i.e. Front and the Rear window patch) from the captured image. The Front window patch consists of the view from the front windshield of the car with driver seat and front passenger seat. And the rear window patch of the car consists of the photo from the rear side window from which the profile view of the passengers sitting at the rear seat is visible. Extraction of ROI’s is an important pre-processing step as the
captured images consists of a lot of redundant information (such as roads and other cars). This background information can potentially increase the computational complexity of convolutional neural network.

### 3.2 Dataset Description and Challenges

This research is specifically focused on the rear window data set (ie the extracted ROI’s) for detecting persons from its profile view. Most of the images are dark and naked eyes cannot detect a person seating on rear seats without pre-processing of the image. ROI’s of all the images consist of profile view of the face. In such a case, it is not possible to observe all the facial landmarks. This is clearly observed in Fig 3.1. The dataset also consists of images which are too dark or too bright. Sometimes, the window for rear seat is closed. Another major challenge for the rear view profile faces is occlusion. Images in Fig 3.2 depict the same.

Given the intrinsic nature of this problem, it is not surprising to observe that for most cases we do not have any person seating on the rear seat. Hence, number of cases of single occupancy are far higher than the having an occupancy of 3 passengers. The dataset possess a class imbalance problem and the selected algorithm should be able to provide high accuracy for all the classes in order to deploy the system in real world. The data distribution in test dataset is tabulated in Table 3.1 illustrate the class imbalance.

The training data set consisted of 4880 images and 1224 images are used validation. For testing, we have one set of images consists of 7061 images and another set of images consisting of 2761 images. Table 3.2 further provides detailed statistics of the dataset mentioning the number of images for each class present in training and testing data set. Ground truth for the object detection task was present in form of xmls files in Pascal VOC data format.
Table 3.1 Data Distribution for Test Set

<table>
<thead>
<tr>
<th>Number of Occupants</th>
<th>Number of images</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5514</td>
</tr>
<tr>
<td>1</td>
<td>993</td>
</tr>
<tr>
<td>2</td>
<td>495</td>
</tr>
<tr>
<td>3</td>
<td>59</td>
</tr>
<tr>
<td>Total</td>
<td>7061</td>
</tr>
</tbody>
</table>

Figure 3.1 Challenges in Dataset

Table 3.2 Data Distribution for Training Set

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Training</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Class 1</td>
<td>3543</td>
<td>899</td>
</tr>
<tr>
<td>Class 2</td>
<td>1227</td>
<td>298</td>
</tr>
<tr>
<td>Class 3</td>
<td>105</td>
<td>26</td>
</tr>
<tr>
<td>Class 4</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>4800</td>
<td>1224</td>
</tr>
</tbody>
</table>
3.3 Annotation

In order to evaluate the accuracy on the test dataset, 7061 images were manually annotated. This exercise proved to be useful. The proposed approach was finalised based on the insights gathered from annotation task. Statistics for the annotation task are summarized in Table 3.3

<table>
<thead>
<tr>
<th>Total number of images annotated</th>
<th>7061</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of images having at least 1 person</td>
<td>1298</td>
</tr>
<tr>
<td>Number of images having more than 1 person</td>
<td>994</td>
</tr>
<tr>
<td>Number of images having full occupancy (i.e. 3 person)</td>
<td>59</td>
</tr>
<tr>
<td>Average aspect ratio</td>
<td>1</td>
</tr>
</tbody>
</table>

Following is the summary of different observations:

- The model is supposed to handle a class imbalance issue as the probability of rear seat occupancy is very less.

- The rear seat is occupied by children (with or without car seats). In such cases we can not observe the entire face but only part of forehead or hair is captured. This is clearly observed in Fig 3.2 (a).

- The model has to learn a lot of contextual information. Annotator used information like the seat belt to annotate the person as the data is very blurry. Features like nose, ears and spectacles oval shaped blob confirm the presence of person which can be perceived in 3.2 (d).
• Occlusion is also an issue for rear seat occupants. Ears which are important landmark for detection of profile face, may or may not be visible (possibly due to headphones or loose hair of a female passenger). This can be noticed in Fig 3.2 (c) and 3.2 (b).

• Manual annotation of images required pre-processing of the images. All the images present in Fig. 3.2 are pre-processed and faces are redacted for privacy purposes.
PROPOSED APPROACH

4.1 Introduction

Deep Neural Networks (DNN) have outperformed the classical vision methods in most of the fields and applications. Whether its images, videos or text, deep neural networks are being used either to extract deep features or to provide the end-to-end solution. The performance of DNNs should improve with increase in depth, but it inevitably results in increase in number of learnable parameters and model size. Thus, the growing size of the neural networks, large memory consumption and huge amounts of training dataset required are some of the challenges associated with DNNs.
Recently, a lot of work is going in the direction of compressing the neural network to reduce model size, memory consumption and data requirement. A DNN can be compressed by removing insignificant weights without changing the network structure or can be compressed by reducing the least significant channels/filters. In the later case, pruning of redundant/least-important channels or filters results in decreased FLOPs as well as reduced memory requirement. Also, the deep neural network will be most effective when it has highly de-correlated filters and each filter is significant irrespective of the presence/absence of any other filter. Thus, inspired by the works of [Han16], [AK17], [FC18], we focus at compressing the deep neural networks by removing the highly correlated filters.

As per [Sin18], we try to maximize the correlation between the filters and prune one of the two highly correlated filters. To overcome the drop in accuracy, we retrain the pruned model from scratch to regain the performance drop.

4.2 Terminology

The original cost function for model $M$ is denoted by $C(\Theta)$ with $\Theta$ being the model parameters. Model $M$ has $K$ convolutional layers in total. Convolutional layers are denoted by $L_i$ where $i$ ranges from $[1, 2, 3, .., K]$. The number of filters for each layer $L_i$ is equal to the number of output channels denoted by $c_o$. An individual filter for a given layer $L_i$ is denoted by $f_j$ where $j$ ranges from $[1, 2, 3, .., n]$ where $n$ is number of output channels. Filters for every layer are denoted by $F_{L_i}$ where $F_{L_i} = \{f_1, f_2, .., f_{c_o}\}$. Also, the dimension of individual filter would be $(h_k, w_k, c_{in})$ where $h_k$, $w_k$ and $c_{in}$ are height, width and number of input channels, respectively.

Fig 4.1 and Fig 4.2 illustrates a the model architecture before and after pruning between two consecutive layers. Here $f_2$ is the filter which is prunned from $L_i$ and its corresponding
filter in layer $L_{i+1}$ is also pruned.

![Figure 4.1 Base Network](image1)

![Figure 4.2 Pruned Network](image2)

### 4.3 Selection of Correlated Filter Pairs

Inspired from concept of [Sin18], for every layer $L_i$ a fixed number of filter pairs are selected from which one filter could be pruned. This selection is based upon the correlation coefficient of filter pairs for a given layer $L_i$. So each filter $f_i$ for the layer $L_i$ is flattened and
correlation coefficient between the given filter \( f_i \) and all the remaining filters of the layer \( L_i \) is calculated. All the filter pairs are then ranked based upon the correlation value. And the top \( n \) filter pairs are selected for further optimisation.

Correlation coefficient is a statistical measure suggesting the degree of association between two random variables. Mathematically, it is defined as

\[
\rho_{X,Y} = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}
\]

Here, \( X \) and \( Y \) are random variables having expectation \( \mu_X \) and \( \mu_Y \) and standard deviation \( \sigma_X \) and \( \sigma_Y \) respectively.

It should be noted that there is a possibility of filter \( f_i \) to be repeated in different filter pairs for a given layer \( L_i \). For example, for layer \( L_i \), \( n \) filter pairs are selected. Let the first filter pair be (1,2) and second filter pair (2,5) where 1,2 and 5 refers to filter position. Correlation for the pair (1,5) is not necessarily high. This should not be a concern while selection of top \( n \) correlated filter pairs.

### 4.4 Optimizing Selected Correlated Filter Pairs

The idea here is to have minimum information loss due to model compression. In order to transfer the information from one filter to other filter in a given filter pair, the correlation coefficient for the selected filter pair has to maximised to at least 0.9. This is achieved by fine-tuning the trained model \( M \) with a new regularizer \( R \) for the correlation coefficient \( \rho_{X,Y} \).

For given layer \( L_i \), correlation coefficients of all the filter pairs selected are summed and divided by number of filter pairs selected for the layer. This is the normalised filter
coefficient for the layer $L_i$. The regularizer $R$ is defined as negative log of normalised filter correlation:

$$R = -\log\left(\frac{\sum_{n=1}^{N} \rho_{X,Y}}{N}\right)$$

(4.1)

With the new regularizer $R$, Model $M$ is then trained with the following parameters.

$$\Theta = \Theta \left( C(\Theta) + \lambda \times R \right)$$

(4.2)

Here, $\lambda$ is a hyper parameter, $C(\theta)$ is original cost function with $\Theta$ being the original parameters. For all our experiments $\lambda$ has been set to 1.

### 4.5 Selection of Filters for Pruning

After the correlation coefficient’s between the selected filter pairs have been maximized by the optimization step, one filter from each filter pair can be removed for a light weight architecture. Since correlation is not associative, we might have filters repeated in different filter pairs. This has been clearly explained in section 4.3. Our approach uses the following method to select the filters which are to pruned.

All the filters are considered in a graphical structure with each filter connected to all the filters, it is highly correlated with. Each individual filter $f_i$ is a node in the graph. The weight for each of the filter is equal to number of connections it has. In other words, weight of a given filter is the total number of filters with which the given filter has a high correlation coefficient. The filter having maximum weight is selected to be pruned and the weights of all the remaining filters connected to given filter are updated. This process is carried out iteratively till a point when the weight of all the remaining filters in the graph is 0. Hence
we are able to select the required filters for pruning.

This process is clearly explained in following Figure 4.3, 4.4, 4.5 and 4.6. Here, the weights or the number of connection for each of the filters is clearly mentioned as $W_i$ where $i$ refers to the filter index. The iterative process of selecting the filters to be pruned is illustrated step wise.

**Figure 4.3** Step 1: Selecting Filters to be pruned

**Figure 4.4** Step 2: Selecting Filters to be pruned
Figure 4.5 Step 3: Selecting Filters to be pruned

Figure 4.6 Step 4: Selecting Filters to be pruned
4.6 Re-training

Lastly, we retrain the pruned architecture using the initial weights. The approach to fine-tune the trained model $M$ was discarded after observing a consistent drop in accuracy over multiple experiments. [Liu18] suggested that pruned architecture is more significance than weights obtained by training the over parameterized model. Training the pruned network from scratch resulted in better trained model over fine-tuning empirically.
CHAPTER 5

IMPLEMENTATION DETAILS

5.1 Timeline

In order to develop a good intuition, an intensive literature survey was conducted with state-of-the-art object detection approaches. Different algorithms and approaches were critically analyzed given the challenges in the dataset and deployability of the selected algorithm as the key factors. SSD [Liu16] was the state-of-the-art algorithm at start of this research. It handled the class imbalance using OHEM [Shr16] and provided a balance between time and accuracy.

As the authors of SSD had made caffe [Jia14] based code-base publicly available, the first
step involved testing the accuracy on the available code-base. Results were generated on SSD with VGG-16 as well as ResNet as backend architecture. RefineDet[Zha18] architecture, which is clearly an improvement over the SSD, was later published and the network was shifted from SSD to RefineDet.

End-to-end research framework of pytorch [Pas17] provided a better platform for experimenting with custom loss function and filter correlation based pruning of the architecture. This lead to change in code-base from caffe to pytorch.

5.2 Methodology

All the experiments were conducted on Single Shot Refinement Neutral Network with VGG as base architecture. An attempt was made to built the RefineDet architecture with ResNet backend. All the convolutional layers in the base architecture except the first input layer are pruned. The first convolutional channel extracts features from the image. Prunning the first layer leads to loss of input features. Hence, we do not select the first convolutional layer while prunning.

Predictions are made on the layer Conv_4_3, Conv_5_3, Conv_fc_7 and Conv_6_2 for VGG16 base architecture in RefineDet. As we prune all the convolutional layers in base architecture, all the connections from the convolutional layer to the Transfer Connection Block and Anchor Refinement Module are also pruned at the selected filter index.

5.3 Hyper Parameters

Initialization of the the base VGG-16 model is done using the ImageNet weights. Extra added layers are randomly initialized using the Xavier method. Network is trained using Stochastic
Gradient Descent with momentum set to 0.9 and weight decay as 0.0005. Learning rate is set to 0.001 for first 40 epochs, 0.0001 for next 10 epochs and 0.00001 for next 10 epochs. We start with a very low learning rate and keep it increasing gradually for the first epoch to 0.001. We use a batch size of 16 and number of workers (used in pytorch for loading data in parallel) as 32. Input images are resized to 320X320. For training, the jaccard index is set to 0.5.

5.4 Hardware and Software Resources

All the experiments were conducted on cluster of 4 Nvidia GPU’s Tesla K80. Initial experiments were conducted on caffe and later pytorch version 0.3 is used.
6.1 Evaluation Metrics

Metrics used for evaluation is Average Precision as defined by Pascal VOC challenge [Eve10]. Since we have one object (i.e., person), Mean Average Precision (MAP) is same as the Average Precision (AP). We use the 11-point interpolated Average Precision for calculating accuracy of Object Detector Algorithm. The Intersection of Union ratio is set to 0.5 while training. Due to disparity in annotation of training data-set and test data-set even the correctly predicted bounding box have Jaccard index lower than 0.5. Hence, for testing the Jaccard similarity coefficient is set to 0.3.
6.2 Results on Caffe Trained Model

At the start of this project, focus was on increasing the accuracy of detecting full occupancy on the rear seat.

The test dataset used originally had 7061 images and the ground-truth consisted of number of passengers visible from the rear side view window. We did not have the bounding box dimensions as the ground-truth at the beginning of the project for the test dataset. All the 7061 images were manually annotated for the groundtruth as required for testing the object detector. Table 6.1 discusses the results on the Test Dataset with the original groundtruth as classifier and Table 6.2 discusses the results on the Test Dataset with the manually annotated groundtruth as object detector.

<table>
<thead>
<tr>
<th>Groundtruth</th>
<th>One Occupant</th>
<th>Two Occupant</th>
<th>Three Occupant</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSD (VGG)</td>
<td>89.27%</td>
<td>87.48%</td>
<td>91.55%</td>
</tr>
<tr>
<td>SSD (ResNet)</td>
<td>90.01%</td>
<td>88.89%</td>
<td>93.44%</td>
</tr>
<tr>
<td>RefineDet(VGG)</td>
<td>80.51%</td>
<td>77.77%</td>
<td>94.70%</td>
</tr>
<tr>
<td>RefineDet(ResNet)</td>
<td>89.94%</td>
<td>86.24%</td>
<td>94.97%</td>
</tr>
</tbody>
</table>

For the test data-set, results as object detector were formulated after the manual annotation. During training the Jaccard index is set 0.5. However, with IOU threshold set as 0.5, results observed were very poor. Hence, we plotted the groundtruth bounding box and predicted bounding box over the images for analytical purpose. The difference in annotation scheme was observed on plotting the bounding boxes over the image. This inspired to
Table 6.2 Results as classifier on Test Dataset using the Object Detector

<table>
<thead>
<tr>
<th>Groundtruth</th>
<th>One Occupant</th>
<th>Two Occupant</th>
<th>Three Occupant</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSD (VGG)</td>
<td>82.35%</td>
<td>83.66%</td>
<td>96.27%</td>
</tr>
<tr>
<td>SSD (ResNet)</td>
<td>92.52%</td>
<td>90.92%</td>
<td>94.83%</td>
</tr>
<tr>
<td>RefineDet(VGG)</td>
<td>79.54%</td>
<td>95.59%</td>
<td>99.16%</td>
</tr>
<tr>
<td>RefineDet(ResNet)</td>
<td>91.26%</td>
<td>88.38%</td>
<td>95.97%</td>
</tr>
</tbody>
</table>

decrease the IOU Threshold while testing. An example image showing the difference can be observed in Fig 6.1. For all the experimental results hence with, Jaccard similarity index is set to 0.3. Table 6.3 provides the MAP as Object Detector for different IOU threshold while training.

![Figure 6.1 Inconsistency in Annotation](image)

6.3 Results on Pytorch Trained Pruned Model

Table 6.4 provides results for one-shot-pruning with train-prune-fine tune approach. It is observed that we could remove upto 334 filters in one shot while maintaining a competitive
Table 6.3 Results as Object Detector on Test Dataset - Trained in Caffe

<table>
<thead>
<tr>
<th>IOU Threshold</th>
<th>0.3</th>
<th>0.35</th>
<th>0.4</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSD (VGG)</td>
<td>42.50%</td>
<td>34.85%</td>
<td>25.32%</td>
<td>9.02%</td>
</tr>
<tr>
<td>SSD (ResNet)</td>
<td>59.49%</td>
<td>54.80%</td>
<td>47.80%</td>
<td>28.75%</td>
</tr>
<tr>
<td>RefineDet(VGG)</td>
<td>81.11%</td>
<td>76.52%</td>
<td>67.96%</td>
<td>43.00%</td>
</tr>
<tr>
<td>RefineDet(ResNet)</td>
<td>81.26%</td>
<td>75.90%</td>
<td>67.85%</td>
<td>43.66%</td>
</tr>
</tbody>
</table>

accuracy. During the optimization training, the model should be selected according to the validation accuracy and not the absolute value of sum of correlation of all the filters. This is clearly observed in last rows of Table 6.4 where the accuracy drop is less than 5% for test dataset even after selecting the the optimized model trained for 35 epochs. It is noticed that there is huge margin between the number of filters selected for correlations and number of filters which are eventually removed. This could be attributed to repetition of filter in different filter pairs which are correlated.

Results for retraining the pruned model from scratch are tabulated in Table 6.5. An increase in accuracy is observed for test data set after removing 689 filters. The predictions of the original model and pruned model are plotted in Fig. 6.2. The red bounding box represents the groundtruth and blue bounding box represents the predicted bounding box. The faces are redacted for privacy purposes. It is clearly seen from the Fig 6.2 that the predictions are very promising with very high confidence.

Base Trained model and pruned model was tested on new Test data-set containing 2761 images. Accuracy on the base model was 81.85% and on the pruned model 67%.
Figure 6.2 Results
Table 6.4 Results of Correlation based One Shot Filter Pruning and Fine Tuning

<table>
<thead>
<tr>
<th>Optimization Iterations</th>
<th>Fine Tune Iterations</th>
<th>No. Filters Removed (Correlated)</th>
<th>Validation Set Accuracy</th>
<th>Test Set Accuracy</th>
<th>No of Basenet Filters</th>
<th>Percentage Model Pruned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basenet - 56</td>
<td>0</td>
<td>0</td>
<td>93.78%</td>
<td>82.35%</td>
<td>55914</td>
<td>0%</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>14(14)</td>
<td>93.64%</td>
<td>81.02%</td>
<td>55788</td>
<td>0.002%</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>70(70)</td>
<td>95.55%</td>
<td>82.72%</td>
<td>55284</td>
<td>0.01%</td>
</tr>
<tr>
<td>15</td>
<td>6</td>
<td>70(70)</td>
<td>93.41%</td>
<td>81.03%</td>
<td>55284</td>
<td>0.01%</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>174(224)</td>
<td>93.53%</td>
<td>78.31%</td>
<td>54348</td>
<td>0.03%</td>
</tr>
<tr>
<td>50</td>
<td>18</td>
<td>309(448)</td>
<td>93.45%</td>
<td>81.65%</td>
<td>53133</td>
<td>0.05%</td>
</tr>
<tr>
<td>50</td>
<td>1</td>
<td>689(1586)</td>
<td>91.94%</td>
<td>69.38%</td>
<td>49713</td>
<td>11.09%</td>
</tr>
<tr>
<td>50</td>
<td>13</td>
<td>689(1586)</td>
<td>93.46%</td>
<td>78.85%</td>
<td>49713</td>
<td>11.09%</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>685(1586)</td>
<td>93.57%</td>
<td>75.27%</td>
<td>49749</td>
<td>11.02%</td>
</tr>
<tr>
<td>100</td>
<td>13</td>
<td>685(1586)</td>
<td>93.08%</td>
<td>74.85%</td>
<td>49749</td>
<td>11.02%</td>
</tr>
<tr>
<td>35</td>
<td>11</td>
<td>689 (1586)</td>
<td>95.02 %</td>
<td>77.77 %</td>
<td>49713</td>
<td>11.09%</td>
</tr>
</tbody>
</table>

Table 6.5 Results of Correlation based One Shot Filter Pruning when Retrained from Scratch

<table>
<thead>
<tr>
<th>Optimization Iterations</th>
<th>Retrain Iterations</th>
<th>No. Filters Removed (Correlated)</th>
<th>Validation Set Accuracy</th>
<th>Test Set Accuracy</th>
<th>No of Basenet Filters</th>
<th>Percentage Model Pruned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basenet-56</td>
<td>0</td>
<td>0</td>
<td>93.78%</td>
<td>82.35%</td>
<td>55914</td>
<td>0%</td>
</tr>
<tr>
<td>35</td>
<td>47</td>
<td>689(1586)</td>
<td>93.52%</td>
<td>82.52%</td>
<td>49713</td>
<td>11.09%</td>
</tr>
<tr>
<td>19</td>
<td>50</td>
<td>893(2016)</td>
<td>95.37%</td>
<td>79.36%</td>
<td>47877</td>
<td>14.37%</td>
</tr>
<tr>
<td>13</td>
<td>35</td>
<td>2580</td>
<td>91.00%</td>
<td>77.29 %</td>
<td>32694</td>
<td>41.52%</td>
</tr>
</tbody>
</table>
Table 6.6 Results of Correlation based One shot Filter Pruning when Retrained from Scratch for Pascal VOC

<table>
<thead>
<tr>
<th>Optimization Iterations</th>
<th>Retrain Iterations</th>
<th>No. Filters Removed</th>
<th>Test Set Accuracy</th>
<th>No of Basenet Filters</th>
<th>Percentage Model Pruned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basenet-300</td>
<td>0</td>
<td>0</td>
<td>79.29%</td>
<td>55914</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>300</td>
<td>3458</td>
<td>72.42%</td>
<td>24792</td>
<td>55.66%</td>
</tr>
</tbody>
</table>
This research is focused on reducing the model parameters for single-shot refinement neural network. Filters from all the convolutional layer in the VGG-16 base network pruned. Before pruning, filter pairs are correlated so that the information loss after pruning the filter is minimum. Experiment was conducted with one-shot pruning and it is believed that iterative pruning can yield better results. Also, it empirically observed that the pruned architecture is of more significance than the weights of the trained model. Retraining the pruned architecture using the initial weights, results into a higher performing model in comparison over pruning and fine-tuning the trained model.
BIBLIOGRAPHY


