Filter Selection in YOLOv3 for Object Tracking Efficiency Enhancement

Yun Chen

Master of Engineering

Electrical and Computer Engineering Department

McGill University

Montreal,Quebec

2019-10-29

A report submitted to the Faculty of Graduate Studies and Research in partial fulfilment of the requirements of the degree of Master of Engineering

DEDICATION

This document is dedicated to the graduate students of the McGill University.

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere thankfulness to my supervisor, Professor James J.Clark. He has been very thoughtful to me. He not only provided good suggestions for my master studies, but also helped me get such a good opportunity to do research in university and internship in the company. Without such experience, I would not have learnt so much. In the same time, he has given great support to my studies and research patiently although I have made lots of trouble during the process, which I felt so sorry about. Next, I would like to thank the company, Brisk Synergies, for their great help, especially Behnam Jamali, who discussed with me and gave lots of informative suggestions during my work. Furthermore, I would like to thank the Mitacs organization for giving me this opportunity to do research and also internship in the company. Finally, my gratitude goes into my parents who live in China, who have been supporting me recklessly, and tolerating my mistakes. It is their love that actually helps me gain most confidence and bravery to go on.

ABSTRACT

Object tracking is a very important task in computer vision. Many object tracking methods apply object detection firstly so that some data of object detection such as object classes and bounding boxes can be obtained. After that, object detection data are associated with other data about the objects such as optical flow so that object tracking can be performed. Lately, deep learning technique has made great improvements on the performance of object detection. One of the state-of-the-art deep learning object detection methods is YOLOv3. YOLOv3 is used for object detection to obtain object classes and bounding boxes. Then, other data are associated with the data from YOLOv3 to perform object tracking. Although YOLOv3 can achieve high object detection accuracy and speed, object tracking can still be influenced by some other factors such as occlusion and illumination. If the objects are occluded, the information will be lost in several frames especially when the objects are in the same class such as cars or pedestrians. As a result, it is hard to reconnect and differentiate them after the occluded objects occur again when they are from the same class. Apart from this, sometimes the lighting condition is poor so that objects are hard to be distinguished and detected so that object tracking accuracy drops. In order to solve this, each object can add new features to represent itself so that objects can be reconnected or recognized after occlusion or poor lighting conditions. A filter selection algorithm is proposed for creating these new features. The filter selection algorithm can choose filters with higher variations and high or medium means. These chosen filters are considered better to represent different objects because they

are more activated and have more variance than the random selected filters. The means and variances or the outputs of chosen filters can be distinctive features in object tracking. Furthermore, the filter selection algorithm is a general method to select better filters in convolutional neural networks so that important information is extracted from the redundant features in multiple convolutional neural networks layers. This method is not restricted to object tracking. It is applicable to other computer vision problems for feature extraction.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
1

1.1 Background

Multiple object tracking (MOT) is a very significant topic in computer vision. Various situations require multiple object tracking. By tracking different objects such as people, buses, and cars at the same time, significant information can be gathered to help traffic analysis. For example, MOT can provide information for motion states of objects and motion trajectories. Additionally, it can help monitor different situations of road intersections and examine car accident risks.

One of the most important applications in MOT is autonomous driving. An autonomous vehicle with an object tracking system will be safer because it can detect and track other vehicles autonomously. Based on the performance of other vehicles, the autonomous cars will have a better grasp of the surrounding environments. As a result, autonomous cars will make better decisions on the road. For instance, if a car in the front decreases the speed due to red lights, the autonomous car will get this information from tracking and also slows down to avoid sudden breaks.

Another application in MOT is athlete tracking in sports games such as soccer, basketball, and hockey. If the movements of athletes are tracked and visualized, coaches can make better analysis and improve their game strategy according to the tracking information. Due to tracking, audiences can also have a better understanding of the game if they can see the movements of each player apparently.

Road tracking is also very crucial for traffic surveillance. Sometimes government and urban planners want to manage the road situations in a more efficient way. In order to get a broader view so that more information can be gathered, fish-eye lenses are used. However, the distorted videos make it more difficult to track objects. Surveillance videos are usually captured at higher altitudes and larger pitches instead of on the ground. The road users in traffic surveillance videos are very different from what we see in daily life. What's more, weather conditions and lighting can affect traffic videos. If there is are heavy rain or low lighting condition, the objects might look blurry and noisy. In order to solve these, several approaches have been used [11][35] based on initializations of objects. These approaches need manual labeling of object states and they are set as inputs. Tracking is then performed on these initializations. However, it is hard to label all the objects because manual labeling is time-consuming. Besides, only a fixed number of objects can be labeled initially and then tracked. If some objects occur later, it is difficult to track them without initializations. As a result, it is not applicable in urban scenarios where many vehicles enter and exit the roads.

In general, road tracking can be classified into two main categories: highway traffic tracking and urban traffic tracking. It is easier to keep track of objects in highway traffic because vehicles can have similar speeds in highways and fewer direction changes. The types of road vehicles are also smaller than urban traffic. However, the scenarios are much more complicated in urban traffic tracking. Road users are prone to be influenced by other users. Sometimes the users make sudden turning at

intersections, sudden breaks or they jaywalk, making it more difficult to track the objects and predict their next motions.

Despite these difficulties, various algorithms have been proposed to perform object tracking. In highway traffic tracking, one important method is to use the Kanade-Lucas-Tomasi (KLT) tracker [2]. It extracts the features and groups them into discrete vehicles using a common motion constraint to perform vehicle tracking. The grouping constraint is based on similarities of KLT features. Correspondingly, the system is more robust to partial occlusion. Another popular technique is proposed by Hsieh et al [7]. By introducing a new "linearity" feature, the vehicles can be categorized into more categories. The line-based shadow algorithm uses a set of lines to eliminate unwanted shadows. A lane-dividing line detection algorithm is proposed so that vehicle occlusion problem caused by shadows can be easily solved, making the vehicle tracking and classification system more robust.

In urban traffic tracking, various efficient algorithms have also been proposed. There are lots of intersections in urban roads, so the structure of urban roads is more complicated. Due to the complicated road situations, collisions are more likely to happen. Saunier and Sayed [21] added newly detected features to current feature groups to track various types of road users. Objects are segmented based on their states of motion. As a result, smooth trajectories can be generated if the density of vehicles is low. However, it cannot perform well when the road is crowded because the road users of a similar speed will be merged. Road users are usually detected first before the tracking algorithm. An on-line discriminative classifier is trained to separate the object from the background. Then the classifier bootstraps itself

to extract positive or negative examples on the current frame using the state of a current tracker. However, when there is subtle inaccuracy in the tracker, the labeled training examples will cause huge drift. To overcome this drawback, Babenko et al [1] used Multiple Instance Learning (MIL) instead of traditional supervised learning to improve the results.

Lately, with the development of deep learning, more and more problems are solved and experiment accuracy has been greatly improved. Likewise, deep learning has also been used in object detection and tracking broadly. One popular approach is to use the pre-trained detector such as YOLO [18] to detect road users. More accurate data of bounding boxes and object classes are got from YOLO. Then, they are combined with other data such as optical flow to perform tracking.

1.2 Motivation

Deep learning technique provides more accurate detection of objects so that the tracking algorithm can get the spatial information in the form of object classes and bounding boxes. Notwithstanding, when data association is performed, it still suffers from occlusion and lost track problems. It is also common to lose track when cyclists or pedestrians are occluded by larger objects such as trucks or buses. When an occlusion occurs, the appearance information and corresponding features are lost. Even if the detection is correct, when the objects appear again, it is difficult to reconnect the trajectories of various objects especially when they are in the same class such as cars or pedestrians. Therefore, it is hard to know the states and motion of various objects. In other situations where there are changes in illumination, the tracking algorithm is not as accurate as it is in good lighting conditions. In other

situations where the objects are too distant, it is also hard to track them well. So the main reasons in the tracking problems are switches in objects and the decreasing of tracking accuracy.

To solve the problem, each object should have more distinction. If we can get some unique features that can represent the distinction of each object, object tracking efficiency will be enhanced. In the situations where the objects with the same class are moving close to each other and when the detection unit fails to detect in a few frames, the distinctive features will represent each object so that in new frames, we can discern which object is from which one in previous frames. In situations where lighting conditions are bad or the objects are too distant, unique features can be additional information in recognizing various objects.

In a word, if useful features are extracted, they can be added to the tracking algorithm to represent different objects within the same class or make the objects distinctive if they are affected by illumination or other partial occlusions. These unique features can be colors, shapes or any other important features that are not even apparent to human vision.

YOLOv3 (You Only Look Once version 3) [17] is a state-of-the-art real-time deep learning object detection method. The inputs can be videos or images, and the outputs are the class label and the bounding box of each object representing the spatial information. YOLOv3 contains a very deep network called Darknet-53 which is a 53-layer convolutional neural network. Each convolutional layer contains several filters and each layer outputs numbers as features. The outputs of the last convolutional layer of YOLOv3 are usually used as features in object tracking to provide

the object class and bounding box information. However, in urban traffic tracking, objects can be occluded or influenced by lighting conditions more, which leads to lost track problem or decline of tracking accuracy. How to reconnect the lost-track objects in the right way when they show up again or enhance the tracking accuracy using some distinctive features that can represent each object is very important to object tracking problem.

It is known that [33] different layers have disparate functions in convolutional neural networks. In lower layers, convolutional neural networks mainly learn detailed information such as colors or texture. In the middle layers, convolutional neural networks learn small patterns such as corners, tree leaves, etc. In higher layers, convolutional neural networks learn more high-level patterns such as cat, pedestrians, etc. Since the Darknet-53 is a very deep convolutional neural network, much potential information can be learned in previous layers compared to another convolutional neural network that contains fewer layers. Instead of just using the last layer outputs of object class and bounding boxes as the object tracking features, it is also possible that the information learned in middle or lower layers can be used in object tracking problem as extra features.

As we delve into the layers of Darknet-53, each convolutional layer contains 32 to 1024 filters and the number of output features in each layer is very huge. How to process such a huge amount of features in each layer and extract useful information in these filters so that they can differentiate various objects in the same class or represent the distinction of the lost-tracking object is a vital problem.

1.3 Project Objective

The objective of this project is to solve the problem of tracking failures in some frames of videos by extracting useful features that can represent the distinction of objects from YOLOv3. A filter selection algorithm is proposed based on YOLOv3. For each filter in each convolutional layer, the mean and variance of the activation level can be calculated. The filter with highest variance and high or medium variances are chosen. The chosen filters and their corresponding means and variances can be the characteristics of the objects. This algorithm can also be applied to other deep learning networks.

1.4 Outline of the Report

This report is organized as follows: In Chapter 2, a literature review is written to show relevant research work. Chapter 3 mainly covers the introduction of convolutional neural networks. In Chapter 4, understanding CNNs by filter activation level is illustrated. In Chapter 5, the filter selection algorithm which is the main algorithm of this project is derived. Chapter 5 shows the datasets and experiment process of this algorithm. Chapter 6 shows the results of the experiment. In Chapter 7, the filter selection algorithm is discussed. In chapter 8, a conclusion is made on this project and some possible future methods for improvements are proposed.

CHAPTER 2
Literature Review

Many object tracking methods perform object recognition and detection first, and combine the results with other data to perform object tracking. It is crucial to have a good performance on object recognition and detection. Traditional object recognition methods are built on handcrafted features and shallow trainable architectures. Typical methods include Scale Invariant Feature Transform (SIFT) [12] features, bag of words, etc. However, the performance can easily stagnate by constructing complex ensembles which combine multiple low-level image features with high-level context from object detectors and scene classifiers. With the development of deep learning, more and more deep learning approaches are used in object recognition and detection, which have enhanced the accuracy greatly.

2.1 Traditional Object Recognition

2.1.1 Scale Invariant Feature Transform (SIFT)

Many object features have been developed for object recognition and detection, but many of them are restricted to certain conditions. Some object recognition and detection methods are appearance-based methods such as eigenspace matching [14], color histograms [26], and receptive field histograms [22]. These approaches have a good performance on isolated images or images that have been pre-segmented. However, it is very hard to extend to other situations where the images are partially occluded or cluttered. Besides, many features do not work if the images are scaled,

translated, and rotated greatly. Some features are greatly influenced by changes in illumination and 3D projective transformation. Efficient recognition can be performed by dense local features [23] if the image descriptors are sampled at many repeatable locations. However, this approach still suffers from scaling and illumination changes. Based on these limitations, Scale Invariant Feature Transform (SIFT) is proposed to break these limitations.

SIFT features are a new class of features that are invariant to image scaling, translation, and rotation. Besides, they are partially invariant to 3D projection and changes of illumination. Based on the model of complex cortex behaviors, the features have similar response properties with the neurons in the inferior temporal cortex. SIFT algorithm extracts key points and computes the descriptors of images. Since it is very computationally expensive to calculate the Laplacian of Gaussian which is crucial for finding the key points in the image, an approximation method is used. The differences of Gaussian shown in Figure 2-1, the maxima, and minima are calculated. Then edges and low contrast regions are eliminated because they are bad key points. After that, for each key point, an orientation is calculated to ensure that the features are not influenced by rotation. These resulting SIFT keys are used in nearest-neighbors approach to identify possible objects in an image. As a result, images can be identified robustly.

2.1.2 Bag of Visual Words

Bag of Visual Words (BoVW) is a method to retrieve information adapted from the concept of the bag of words (BoW) on natural language processing. In the bag of words, the number of each word appeared is calculated. The frequency of each

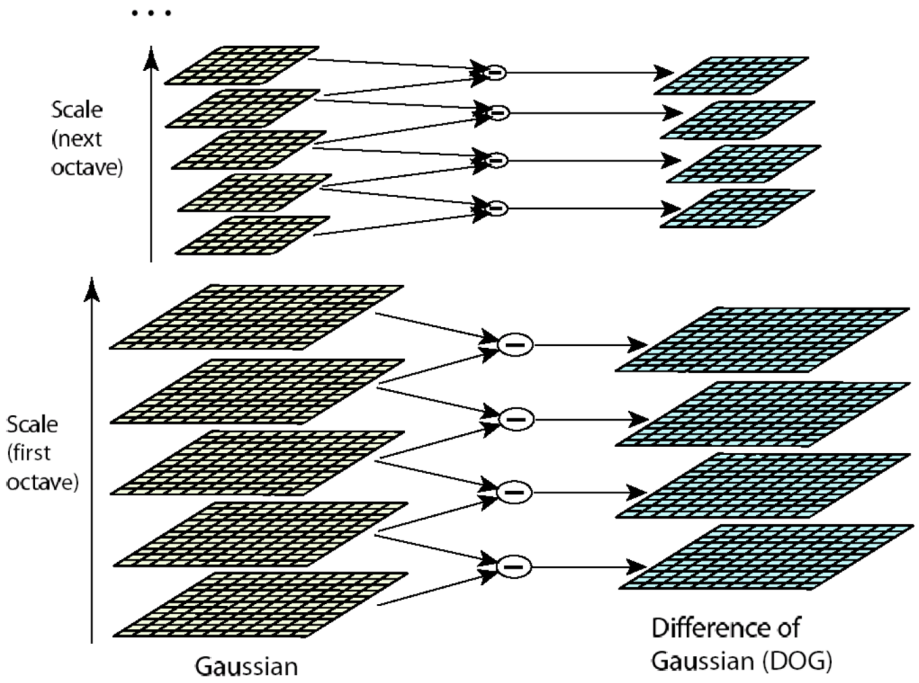

Figure 2–1: Structure of differences of Gaussian in SIFT [12].

word is used to know the keywords of the document and a frequency histogram is derived. As a result, every document is a bag of words.

Likewise, the same concept can be applied to images. Instead of words, each image feature is considered and calculated as the "words". Each image is represented as by a set of features which contain key points and descriptors. Key points are the points that stand out in an image and are not influenced by rotation, shrinking or expanding of images. Descriptors are the descriptions of key points. The key points and descriptors are gathered to form the vocabularies. Each image is represented by

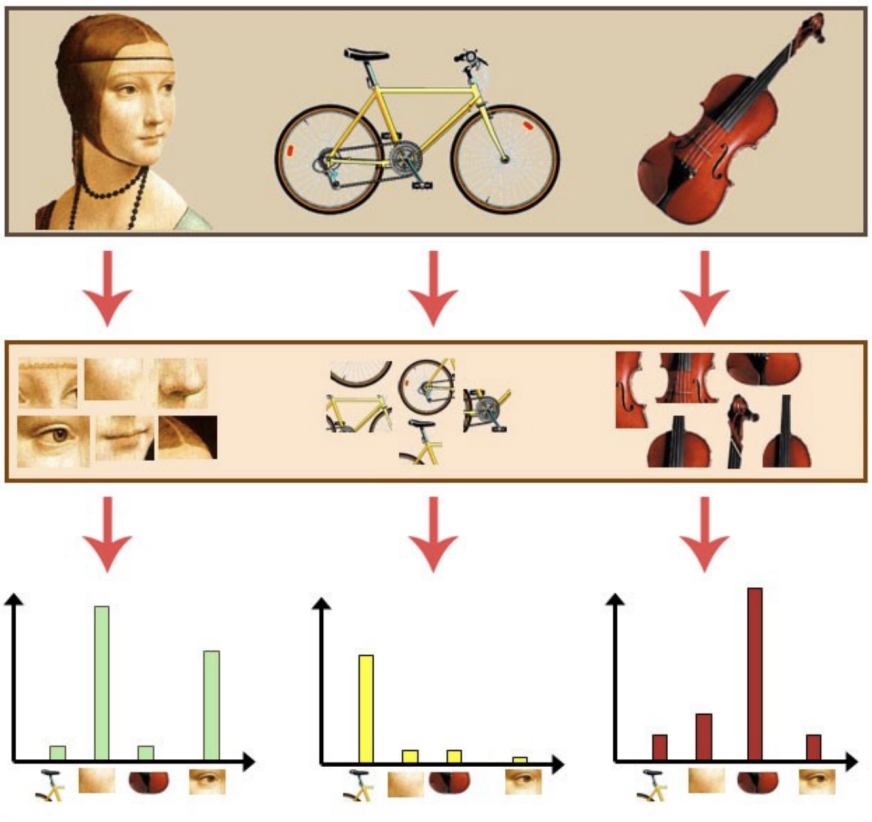

Figure 2–2: Representation of visual words and the histograms [13].

the frequency histogram of features in the image. The category of the image can be predicted by the histogram of frequency shown in Figure 2-2.

Bag of visual words is based on vector quantization of affine invariant descriptors of image patches. In this way, image recognition can be robust to image rotation.

2.2 Deep Learning Object Recognition and Detection

Generally, object recognition and detection can be solved by features of machine learning approaches such as SIFT, HOG [3], Haar-like features [30], etc. However,

deep learning approaches perform much better lately with the larger datasets collected such as ImageNet [4] and more powerful models designed. By varying the depth and breadth of the convolutional neural networks, strong assumptions can be made about the nature of the images.

2.2.1 Regions with CNNs

Regions with CNNs (R-CNN) [6] is a deep learning approach that has dramatically enhanced the object detection accuracy by combining region proposals with CNN features.

Firstly, many category-independent region proposals which are bounding boxes are generated by Selective Search [29]. Selective Search looks through each image at windows of varied sizes. At each size of the window, adjacent pixels are grouped by texture, color or intensity to identify objects.

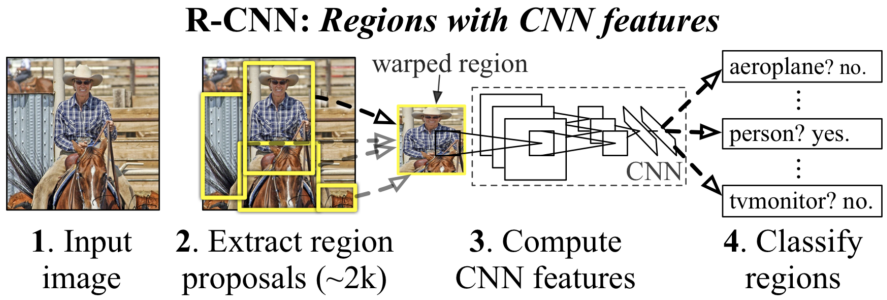

Figure 2–3: After the region proposals are generated, they are passed through a modified AlexNext to see whether they are valid regions [6].

After that, the regions are warped to a standard square and passed through five convolutional layers and two fully connected layers, which is a modified AlexNet [9].

As a result, a fixed length of 4096-dimensional feature vector from each region is generated.

At last, a class-specific linear SVM is used to classify whether each proposed region is an object and what the object is. The region box is run through a linear regression so that tighter coordinates are generated for the object.

R-CNN works well on object detection, but the speed is really slow. For each image, each region is passed through the AlexNet and there is around 2000 forward pass, which is very time-consuming. Three models need to be trained separately. The feature generator of CNNs, the classifier that classifies the class of each object, and the regression model that tighten the bounding boxes need to be trained, which makes it hard to train.

2.2.2 Fast R-CNN

To speed up R-CNN, a few changes have been made to improve the model, which leads to Fast R-CNN [5].

For each region proposed, the CNN is run for once. The forward pass is performed around 2000 times in R-CNN. So the same CNN computation is run again and again in R-CNN, which is not efficient. To improve this, Region of Interest Pooling (RoIPool) is proposed in each image and the forward pass is shared in proposed regions so that in each image, forward pass need to be run for one time instead of 2000.

Apart from this, CNN, classifier and bounding box regressor are combined in a single network instead of three to reduce the complexity. The SVM classifier is

replaced by a softmax layer after CNN for classification. A linear regression layer is added parallel to the softmax layer to get bounding boxes.

By modifying on R-CNN, Fast R-CNN can perform object recognition and detection in a faster way.

2.2.3 Faster R-CNN

Although improvements have been made on Fast R-CNN to make it faster, proposing regions by Selective Search is still the main bottleneck. Faster R-CNN [20] is thus designed to make the region proposal step almost cost-free so that object detection can be performed in real-time.

As shown in Figure 2-4, one single network is used in Faster R-CNN for region proposals and classification. As a result, just one CNN is trained and region proposals can be obtained efficiently due to the observation that the convolutional feature maps used by Fast R-CNN can also be used to generate region proposals.

By creating a Region Proposal Network which adds a fully convolutional network on top of the features of CNN, a sliding window is passed over the CNN feature map. In every window, k potential bounding boxes and scores for how good each bounding box should be are generated using anchor boxes. So for every anchor, a bounding box and a score are generated. Then the most likely bounding box for an object is passed to Fast R-CNN to perform classification and a tightened bounding box. Therefore, object detection can be performed faster by Faster R-CNN.

2.2.4 You Only Look Once

Instead of looking at regions which have higher chances of containing an object used in R-CNN family, You Only Look Once (YOLO) [19] looks at the entire image

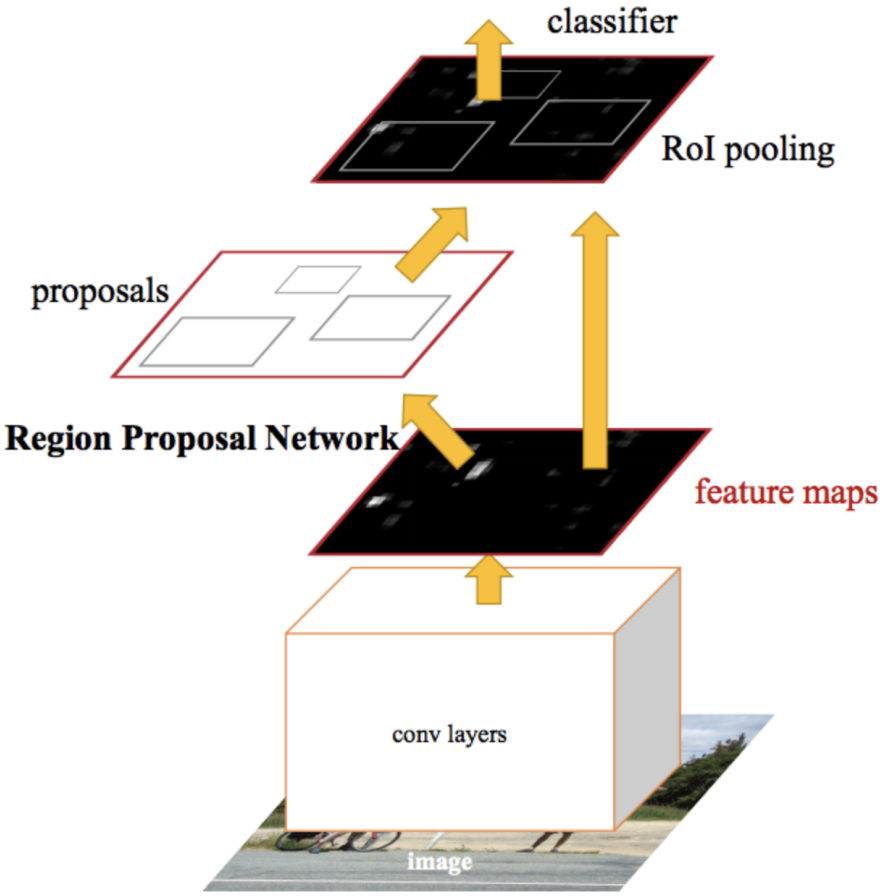

Figure 2–4: One single network is used in Faster R-CNN for region proposals and classification [20].

in a single instance and predicts the bounding box coordinates and class probabilities for these boxes shown in Figure 2-5. It performs detection at a very fast speed so that object detection is real-time at 45 frames per second.

YOLO can learn the objects in a general way. When YOLO looks at the entire image during training, contextual information is encoded and general representations

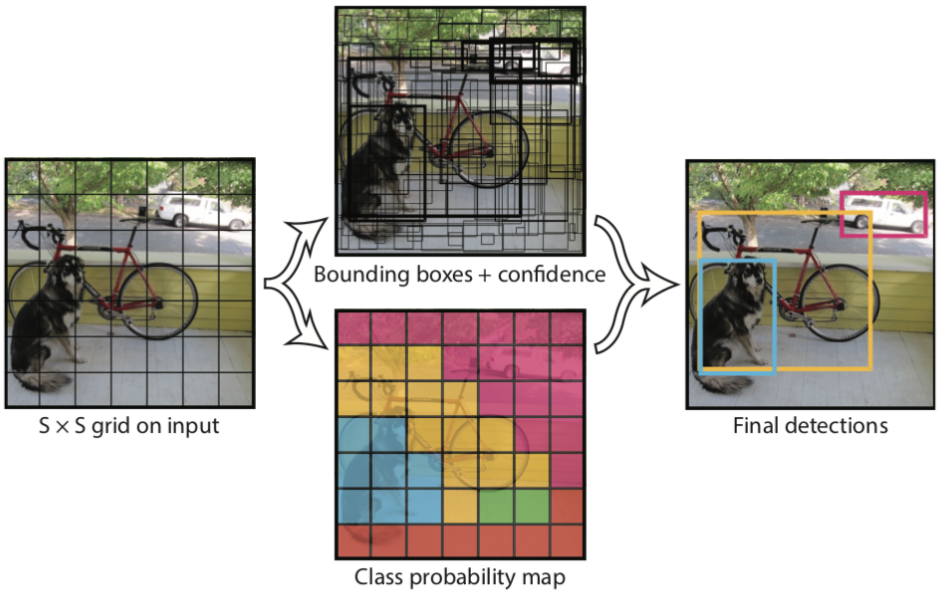

Figure 2–5: YOLO divides the image into $S \times S$ grids. Within each grid, B bounding boxes, confidence for those boxes, and class probabilities C are predicted [19].

of objects can be learned, which also makes YOLO more generalizable especially on artwork.

Although YOLO is one of the fastest approaches in real-time object detection, it struggles in detecting small objects that appear in groups, such as flocks of birds because YOLO has strong spatial constraints on bounding box predictions. Incorrect localization is the main error since it uses coarse features to predict bounding boxes and the loss function treats errors the same in small bounding boxes versus large bounding boxes.

Based on the limitations of YOLO, modifications are made on YOLO so that YOLOv2 [16] and YOLOv3 [17] are proposed later, which will be explained in detail in Chapter 3.

2.2.5 Single Shot Detector

Single Shot Detector (SSD) [10] is another real-time object detection approach that has a good balance between speed and accuracy. As the name states, SSD performs object localization and classification in a single forward pass of the network inspired by MultiBox [28] for bounding box regression shown in Figure 2-6.

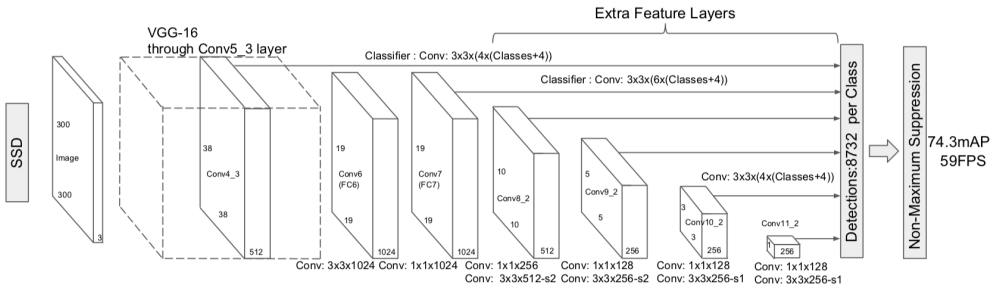

Figure 2–6: Structure of Single Shot Detector [10].

VGG-16 [25] without the fully-connected layer is used as the base network. Transfer learning and some auxiliary convolutional layers are then applied [32] to improve the results. By making improvements on MultiBox, predictions are closer to the ground truth. Therefore, a multi-scale convolutional bounding box outputs which are attached to multiple feature maps at the top of the network are used so that the space of possible box shapes can be modeled more easily.

2.3 CNN Visualization

To understand why these object detection models work well, some approaches are used, most of which are through visualization.

2.3.1 Weights Visualization

By intuition, it is reasonable to look at the features learned in first-layer filters, which are the linear weights in the input-to-first layer weight matrix. It is very easy to understand inputs like images or waveforms by weights visualization. The filters take the shape of stroke detectors if the neural networks are trained on digit data. If neural networks are trained on natural patches of images, the filters will take the shape of edge detectors. Generally, if there are some interpretable patterns in the weights visualization, it means that the weights are well trained.

Some good examples of weights visualization are shown in Figure 2-7 and Figure 2-8. The weights shown in the first layer are mainly edges and color blobs and the weights shown in the second layer are not very interpretable but smooth. The visualizations of the first layer and the second layer are color and grayscale respectively because AlexNet contains two streams of processing, one of which develops the high-frequency grayscale features and the other develops low-frequency color features.

However, this method can only visualize the first-layer filters well, due to their linearity. For the second or higher layers, visualization becomes harder. Additionally, the visualization of weights is hard to understand if the trained data are not digital or natural images.

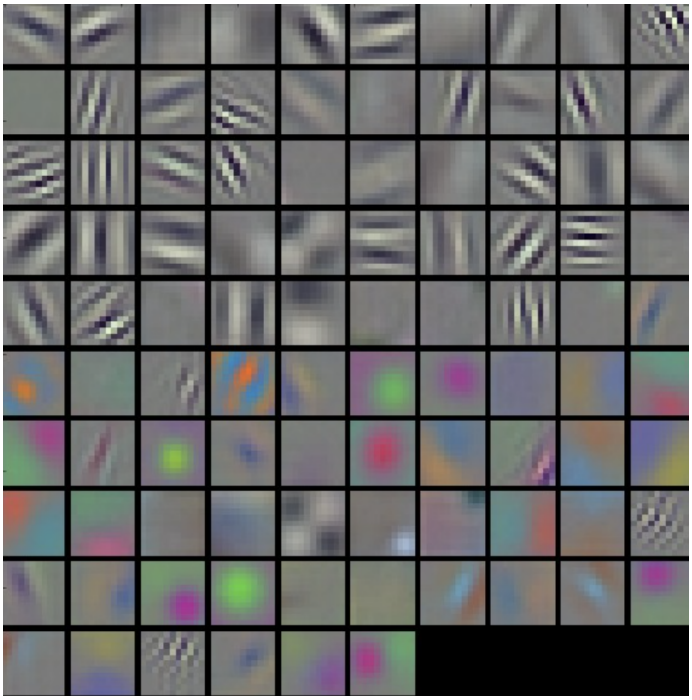

Figure 2–7: Typical filter looking on the first layer of a well trained AlexNet [15].

2.3.2 Activation Visualization

Except for weights visualization, another straight-forward visualization method is to look at the direct output of each filter during the forward pass, which is the activation level. However, the outputs are mostly small numbers. Thus, in visualization images, there are lots of blobby shapes. As the layers go deeper, the visualization becomes harder to interpret because the outputs become very sparse, some of which are near zero. As a result, it is easy to get black activation maps which are not good for analysis. Some examples are shown in Figure 2-9 and Figure 2-10 in which each box shows the output of a filter. We can see that some of the filters show a whole black box, indicating that the outputs are very sparse and hard to be visualized.

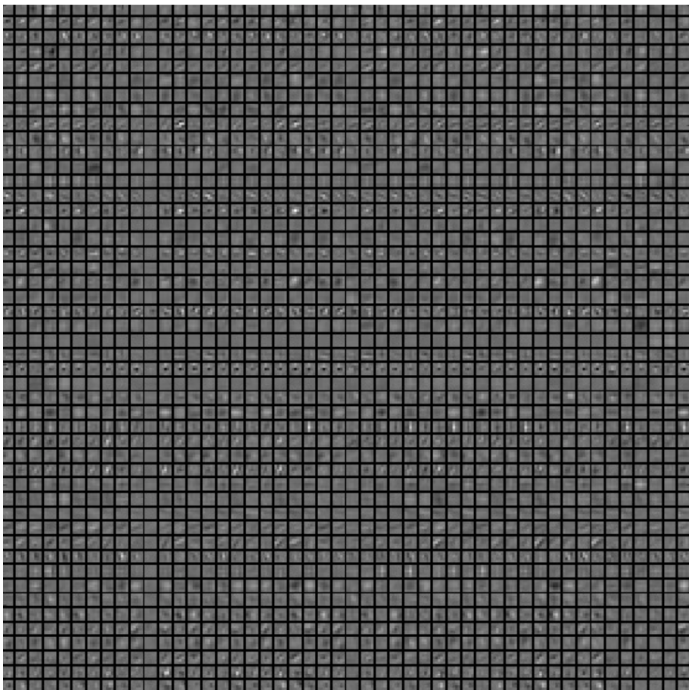

Figure 2–8: Typical filter looking on the second layer of a well trained AlexNet [15].

2.3.3 DeepDream

Based on the activation level, DeepDream is a computer program developed by Google to better understand the convolutional neural networks by generating psychedelic-looking images.

An image is fed into a pre-trained CNN. The image can be any image even random noise. A forward pass is done until a particular layer that is chosen by us. To get to know what has been learned on that layer, the activation level is maximized. The gradient of that particular layer is set equal and then gradient ascent is applied in that layer. To make the resulting image better to interpret, Gaussian blurring or use of octaves are implemented to make the image look smoother.

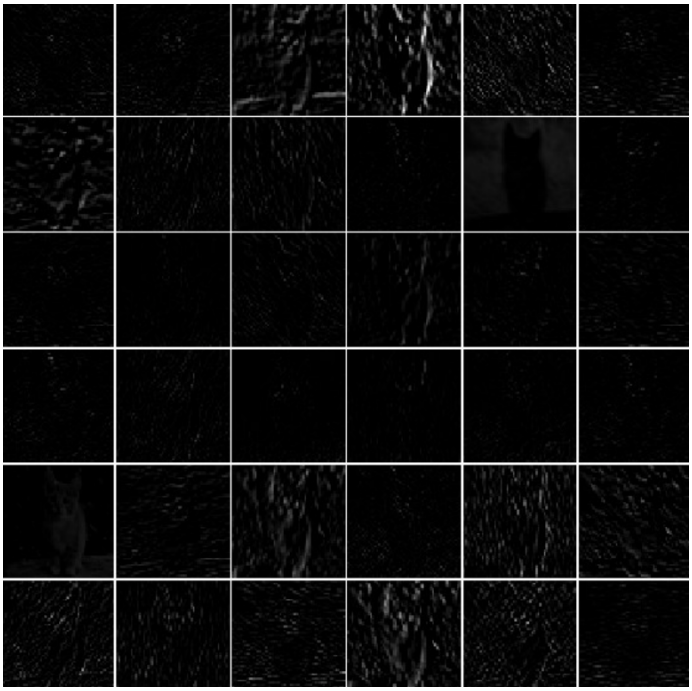

Figure 2–9: Typical first-layer output visualization of a trained AlexNet. Each box shows the output of a filter [15].

Correspondingly, the resulting image that represents what the particular layer has learned can be very psychedelic and the image usually contains a lot of repeated patterns. Through DeepDream, activation maximization is exploited to understand the potential patterns of what CNN has learned.

2.3.4 Deconvnet

Although DeepDream can help us understand CNN, the resulting image by gradient ascent usually contains many repeated patterns, which is psychedelic but not rigorous since there might be many repeated patterns in one image to make it hard to distinguish. Another novel visualization technique [33] is performed through Deconvolutional Network [34].

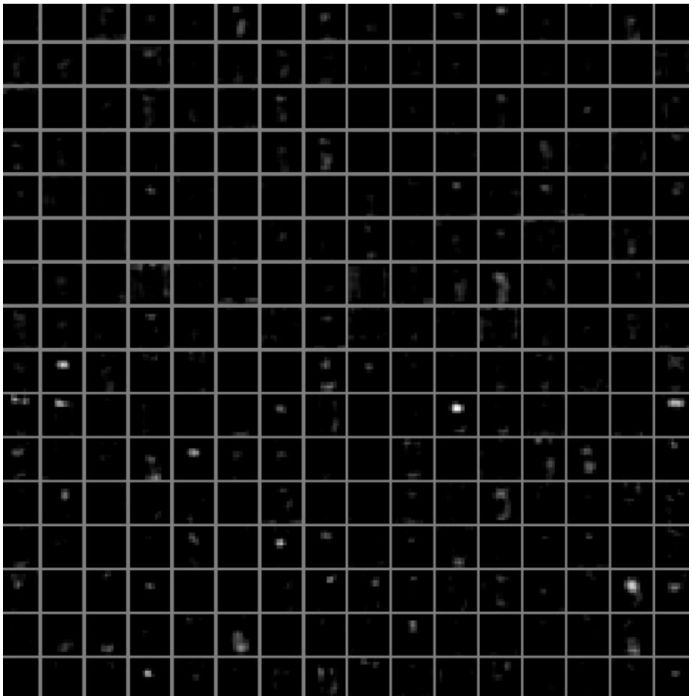

Figure 2–10: The 5th layer of a trained AlexNet looking at a picture of a cat [15].

A deconvnet is a convnet model that has the same component such as filtering, pooling but is processed in reverse. Instead of mapping pixels to features, a deconvnet does the opposite to examine the activation of a convnet.

An input image is fed into convnet first and features are computed through the layers. To examine the activation of the convnet, a deconvnet is added to each of the convnet's layers and the features are projected back to pixels shown in Figure 2-11.

The reconstruction process contains unpooling, rectification and filters, which are also repeated. In the deconvnet, switches are used which record the location of the local max in each of pooling region during pooling. After unpooling, the signal is passed by a relu non-linearity to ensure the positivity. Then, the transposed filters

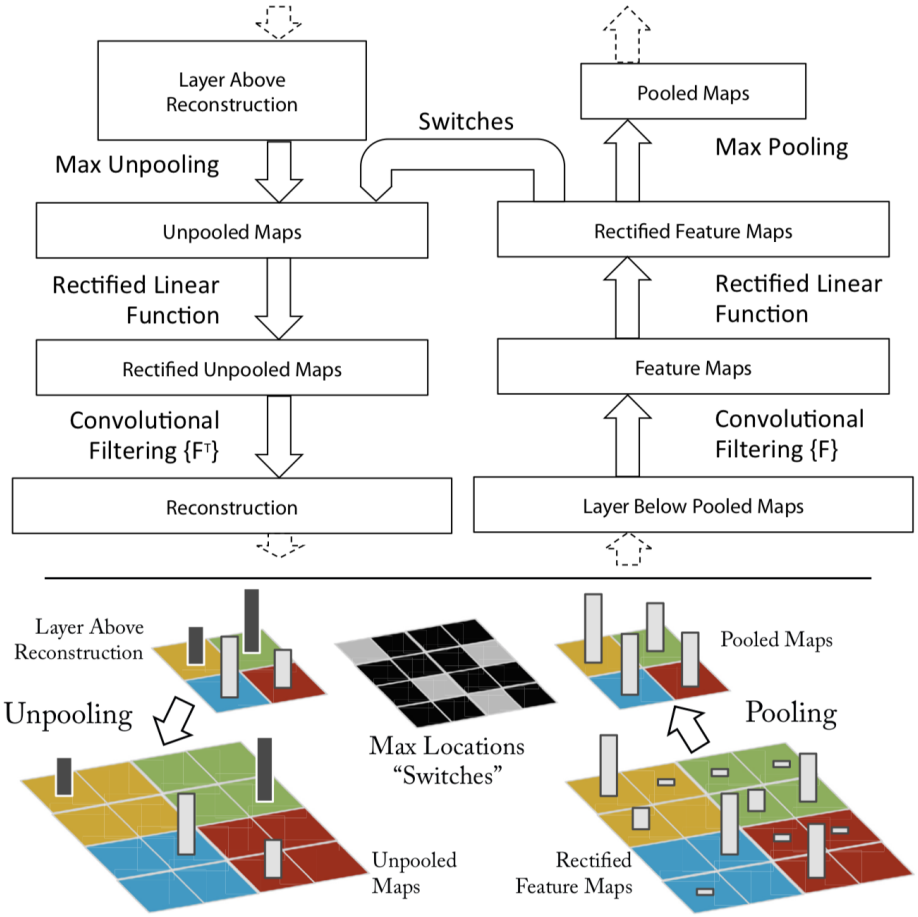

Figure 2–11: The left is a deconvnet layer, which is attached to the right convnet layer. The deconvnet will construct convnet features approximately from the layer beneath [34].

in convnet are applied. By deconvnet, different feature activations can be visualized to reveal the different structures that excite the feature map and also its invariance to input deformations.

2.3.5 Saliency Maps

Besides visualization by deconvnet, class saliency maps [24] are used to understand CNN classification. The areas of a given image are highlighted concerning the given class of image. Like DeepDream, it is also a gradient-based method.

The gradient of the output category for the input image is computed to show how output category value changes to a small change in input image pixels. These gradients can be used to highlight input regions that cause the most change in the output. Intuitively, salient image regions that most contribute towards the output will be highlighted. As a result, the saliency maps are very intuitive and easy to understand the CNNs.

2.4 Summary

In this chapter, several traditional and deep learning object recognition and detection algorithms have been reviewed. The typical traditional object recognition and detection approaches are machine-learning-based approaches, which extract different features of images such as SIFT features, bag of words, etc. Then classifier such as SVM is used to classify them according to features to perform object recognition.

However, with the development of deep learning, object recognition and detection accuracy and speed have been greatly improved. The main several state-of-the-art approaches are R-CNN, Fast R-CNN, Faster R-CNN, YOLO, SSD, etc.

To better utilize the potential information in CNNs, several visualization approaches have been reviewed to understand CNNs better. Although weights and activation levels can both been visualized, the visualizations are not interpretable

in deeper layers. However, other approaches such as deconvnet, saliency maps, and DeepDream are good ways to visualize and understand the CNNs.

CHAPTER 3
YOLOv3

To get the best features for object tracking, the tracking pipeline is studied at first. It turns out that the most popular method is tracking-by-detection. In each frame, an object detector is trained to classify and detect the objects. The outputs of the detector are bounding boxes and the class of each object. Then, data association is performed to connect the results of this frame to the next frame with other data such as optical flow. This tracking-by-detection step is iterated over the entire video. Thus, the tracking information of each object is gathered.

As a result, the object detection is a very crucial part of object tracking. The popular object detection method is YOLOv3, which contains very deep CNNs. Instead of just using the last layer outputs of object classes and bounding boxes, potential information can be learned in previous layers, which can be used as additional features to distinguish different objects of the same class or objects in poor lighting conditions or bad situations where the tracking accuracy declines due to occlusions. In this chapter, details of YOLOv3 are explained.

3.1 Architecture of YOLOv3

YOLOv3 is the state-of-the-art object detection algorithm with high accuracy and processing speed. As the name stated, it is the third version of YOLO. YOLOv3 has made some improvements on YOLOv2 and has increased accuracy on small object detection. The residual blocks, upsampling, and skipping connections which

are the latest computer vision techniques are used. Different scales of images are detected so that it does not miss small objects. The architecture is shown in Figure 3-1.

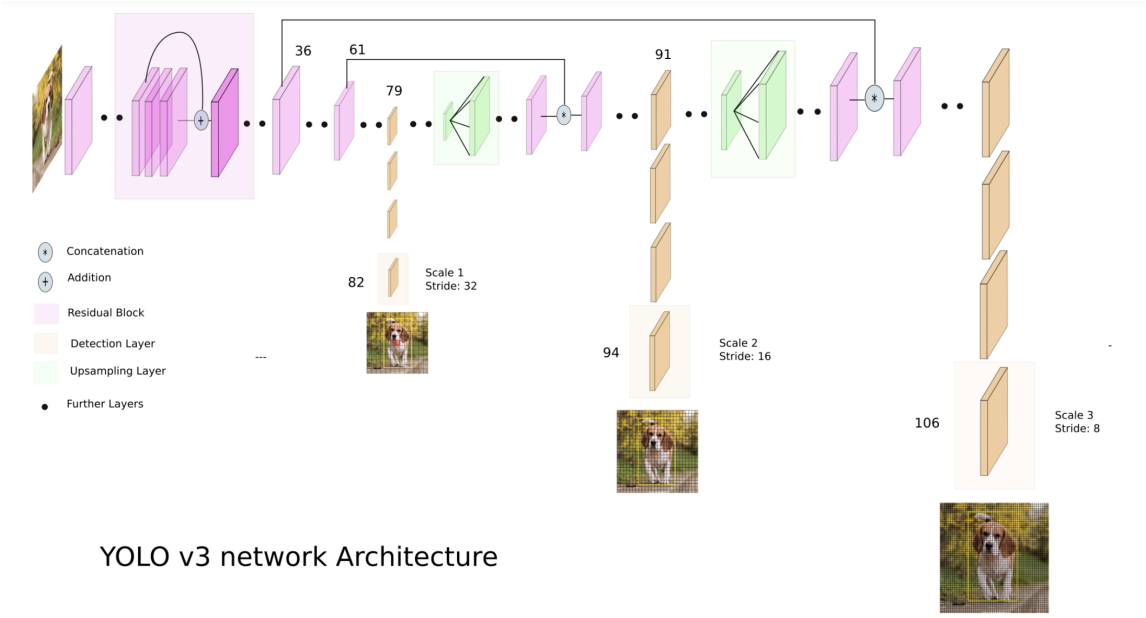

Figure 3–1: Structure of YOLOv3. Some latest computer vision techniques such as residual blocks, upsampling and skipping connections are used [17].

The feature extractor in YOLOv3 is called DarkNet-53. It is deeper than DarkNet-19 which is used in YOLOv2. YOLOv2 contains a 19-layer network supplemented with 11 more layers for object detection. With this 30-layer structure, YOLOv2 is already fast and accurate but not good at detecting small objects because of downsampling. When the layers go deeper, the features are less fine-grained. To overcome this, YOLOv2 has also used identity mapping which concatenates the feature maps from a previous layer to get low-level features. However, it does not contain the most staple elements in computer vision such as residual networks, skip

connections and upsampling. Nevertheless, YOLOv3 contains all of them, making great improvements in small objects detection.

3.2 Layers of Darknet-53

The feature extractor in YOLOv3 is DarkNet-53 which is a 53-layer convolutional neural network. Figure 3-2 shows the structure of DarkNet-53.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 3–2: The structure of layers in Darknet-53 [17].

As we can see, except for residual layers, average-pooling layer and fully-connected layer, the layers of Darknet-53 layers can be divided into 6 categories by the number

of filters that each layer contains. Shown in Figure 3-3, the number of layers that contain 32 filters is 2, the number of layers that contain 64 filters is 4, and the number of layers that contain 128 filters is 10. In addition, there are 17 layers that contain 256 filters, 13 layers that contain 512 filters in each layer, and 5 layers that contain 1024 filters. In a few words, there are at least 32 filters and at most 1024 filters in each layer.

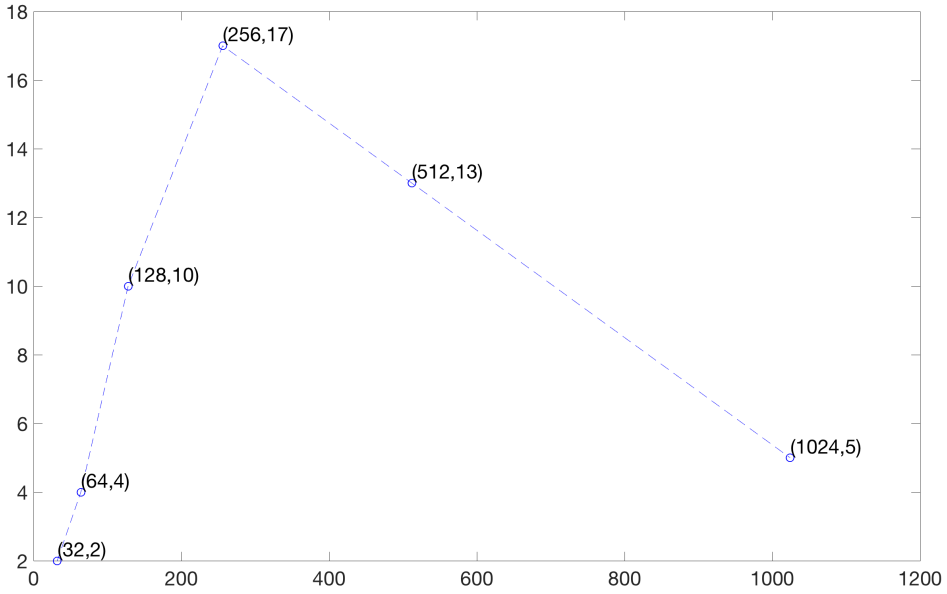

Figure 3–3: Number of layers in Darknet-53 that contain 32, 64, 128, 256, 512, 1024 filters respectively.

After calculating the number of filters that each layer contains, we can sum them all by layers and figure out the whole number of filters in Darknet-53. Through calculation, the number of filters in all layers of Darknet-53 together is 17728, which is very huge.

Shown in Figure 3-2, the number of output features in each filter varies from 8 $\times$ 8 which is 64 to 256 $\times$ 256 which is 65536. We can see that each filter outputs large amounts of features. Although each filter outputs a huge amount of features, they are usually seen as a 3-D tensor. Each filter outputs a 3-D feature tensor shown in Figure 3-1.

3.3 Filter Selection Problem

In a few words, the 17728 convolutional filters in Darknet-53 output 17728 feature tensors altogether. A lot of feature tensors are learned through the training process. Generally, the last-layer outputs of object classes and bounding boxes are used as features to represent each object in object tracking. Other features in previous layers are never used. However, through the learning process, Darknet-53 has learned much information about objects. The features in previous layers can be used to represent objects. It is obvious that such a huge number of feature tensors are redundant in object tracking. We need to take out some redundancy and extract more useful information in these filters so that they can be used to represent various objects.

If we want to use the feature tensors inside the layers of Darknet-53 as extra features for object tracking, it makes sense to just pick some of the filters instead of using them all. It is assumed that for various objects, different filters have varied responses. Some filters might response more to cars and some filters might response more to pedestrians. Making use of the distinction of filters can help us differentiate various objects in the same class or enhance object tracking accuracy while objects are in poor lighting conditions. How to pick up these filters in a more intelligent way

instead of picking them up randomly so that the selected filters and their output feature tensors can represent the objects more distinctively is very crucial. As a result, understanding convolutional neural networks is explored in Chapter 4 and a filter selection algorithm is derived in Chapter 5.

3.4 Summary

In this chapter, the structure of YOLOv3 is explained and the layers are analyzed. It turns out that although YOLOv3 can achieve high accuracy with high speed in object detection, it is hard to analyze the inside layers because each layer contains many filters, and there is a huge amount number of output features of each filter. Therefore, a filter selection problem is addressed to extract more useful information in YOLOv3 so that better object tracking efficiency can be obtained.

CHAPTER 4
Understanding CNNs by Filter Activation Level

To select useful filters that can represent the objects better instead of selecting them randomly, the first natural way is to understand the inside functions of CNNs in Darknet-53 and know what each filter is doing in each layer. By comparing the filters in different layers, some characteristics might be found.

However, there is no clear way to understand convolutional neural networks. CNNs in Darknet-53 in a whole is like a black box. The input data images are fed into Darknet-53 and the numbers of object classes and bounding boxes are the outputs we get. We can extract the outputs of filters in previous layers but they are just tensors. One popular method to understand them is by visualization. As stated in Chapter 2, various visualization techniques have been used to help us know the functions of previous layers in CNNs.

Generally, we know that each layer progressively extracts higher and higher level features of the image, until the final layer decides what the image means. For example, the first layer might looks at color, edges or corners. The intermediate layers might learn to see basic features like a door or leaf. The last few layers combine the former features into more general interpretations such as a tree or pedestrian.

4.1 Activation Level

One popular way of understanding CNNs is making use of the activation levels and maximizing them in several filters to see what the filter see in CNNs, which is

the general idea of DeepDream. The activation level means the output number of the filter.

Let θ denote our convolutional neural network parameters (weights and biases) and let $h_{ij}(\theta, \mathbf{x})$ be the activation of a given filter i from a given layer j in the network. $h_{ij}(\theta, \mathbf{x})$ is the function of both the input sample $\mathbf{x}$ and also parameters θ. For a trained YOLOv3, the parameters θ are fixed numbers. In Darknet-53, each activation level of a filter $h_{ij}(\theta, \mathbf{x})$ can be considered a 3-D tensor.

Through DeepDream, we know that the activation levels of various filters are very different. The activation level can be considered as the responsive level of what pattern each filter has learned.

4.2 Activation Maximization

The reasoning behind activation level of filter is that if a filter is maximally responding to a pattern, it means that this filter is mostly activated on such pattern and this pattern will be what the filter has learned.

As a result, the problem of finding the pattern that the filter learns can be viewed as an optimization problem. We are looking for

$$\mathbf{x}^* = arg \max_{\mathbf{x} s.t. \|\mathbf{x}\| = \rho} h_{i,j} = (\theta, \mathbf{x}) \tag{4.1}$$

Generally, this is a non-convex optimization problem. Nevertheless, at least a local minimum can be found. As a result, the *gradient ascent* is performed. The gradient of $h_{ij}(\theta, \mathbf{x})$ moving in the direction of $\mathbf{x}$ is computed.

In this way, two scenarios are possible. One possible outcome is that the same maximum is found when starting from random initializations. The second possible

outcome is that two or more local maxima are found if random initializations are performed. In both cases, we can get the characteristic of the unit by the maximum or set of local maxima. In the latter case, we can choose the one which maximizes the activation or just average the results. This technique applies to any network. It also involves a choice of hyperparameters such as learning rate and a stopping criterion like other gradient descent techniques.

4.3 DeepDream Algorithm Visualization

DeepDream [27] is based on the activation maximization method using a convolutional neural network to find or enhance the patterns in images. The DeepDream creates dream-like hallucinogenic appearance if the images are over-processed. Using DeepDream algorithm, we can better understand convolutional neural networks.

Based on the idea of activation maximization, for a trained network, there is some level of activation in each filter. It can also be run in reverse, adjusting the original image slightly so that a given output neuron yields a higher confidence score. For example, the neuron can be the one that learns faces or animals. The process is similar to back-propagation, instead, the weights are fixed and inputs are adjusted.

Figure 4-2 and Figure 4-3 are two examples of DeepDream outputs of an input traffic image shown in Figure 4-1. DeepDream utilizes Inception 5h model because it is easier to work with. It takes input images of any size and creates pretty pictures.

We can see in Figure 4-2 that there are a lot of repeated shapes of spirals. Compared to Figure 4-2, there are a lot of green and purple line-shape textures filled in Figure 4-3. It is because that the former output image is based on the maximization of layer inception_4a-3×3 while the latter output image is based on

Figure 4–1: DeepDream input traffic image.

Figure 4–2: DeepDream output which maximized the layer inception_4a-3×3.

the maximization of layer inception_4c. Not only the activation of different layers can be maximized, but certain filters can also be chosen to be maximized. As a result, we can know that the filters in layer inception_4a-3×3 mainly learns the pattern

Figure 4–3: DeepDream output which maximized the layer inception_4c.

of spiral shapes while the filters of layer inception_4c mainly learn more green and purple line-shape texture.

Through DeepDream, we know that each filter and each layer have different activation responses on the same image by gradient ascent. As a result, a filter selection algorithm is derived based on the inspiration of DeepDream.

4.4 Summary

In this chapter, convolutional neural networks are analyzed so that we can better understand them by visualization. One typical method of DeepDream is explained. Besides, the concept of activation level and method of activation maximization in DeepDream are explained by examples. Based on the activation level, a filter selection algorithm is derived in the next chapter.

CHAPTER 5
Filter Selection Algorithm

According to the analysis of activation level, it is obvious that for one image, the activation level on different filters should be various. According to the visualization theory, some filters learn more patterns. The filters in the former layers mainly learn some colors and texture. The middle-layer filters mainly learn some parts of the object such as eyes, grass, wheels, etc. The final layers predominantly learn the overall class types.

If the input image is a car, the filters which learn the pattern of a wheel should be highly activated. For the same input image, different filters have various activation levels. If we can select the highly activated filters, they are the ones that learn more useful patterns than the less activated filters. Thus, the highly activated filters will be more distinctive to represent a specific object compared to less activated filters. However, highly activated filters cannot necessarily be interpreted visually. Apart from this, it is not sufficient if we only choose the filters with a higher activation level because the high activation level can only indicate that the filters learn more patterns and contain more information. We want to differentiate different objects of the same class such as two different cars. The activation level of the filter should have more variations. Therefore, if the filter has a higher variance in the activation level on the same class of objects, the filter might be suitable to differentiate objects in the same class.

5.1 Quantify the Activation

How to find the filters that have higher activations and also higher variations? A quantifiable way to measure the activation and variation levels should be designed.

In each layer, the output of each filter is a tensor. Each output element in the tensor represents the activation value. The value can be negative. The bigger the absolute value of the number is, the bigger the activation will be. In order to get the activation level of one filter, we first calculate the absolute value in each element of the tensor and then sum them all according to the total number of elements of one tensor. By dividing the total number of elements in the tensor, we can get the mean activation value of the tensor. Therefore, the mean activation value that we calculated of one filter's output tensor can represent the activation level of this filter.

5.2 Filter Selection

Since we want to choose the filter that has a higher mean and variance for objects of the same class, a filter selection algorithm is proposed. The pipeline can be shown in Figure 5-1.

As shown in the pipeline, firstly a filter is chosen to be calculated. The chosen filter can come from any layer in Darknet-53. For one input image, the activation level of the chosen filter is calculated. Then, this process is repeated through the whole dataset that contains different images of the same class. After that, the mean and variance of the activation level of this filter on the whole dataset are calculated. Therefore, the function of this filter can be quantified by the mean and variance of activation value based on one dataset. In the end, the whole process is repeated through different filters so that every filter can be quantified by the mean and variance

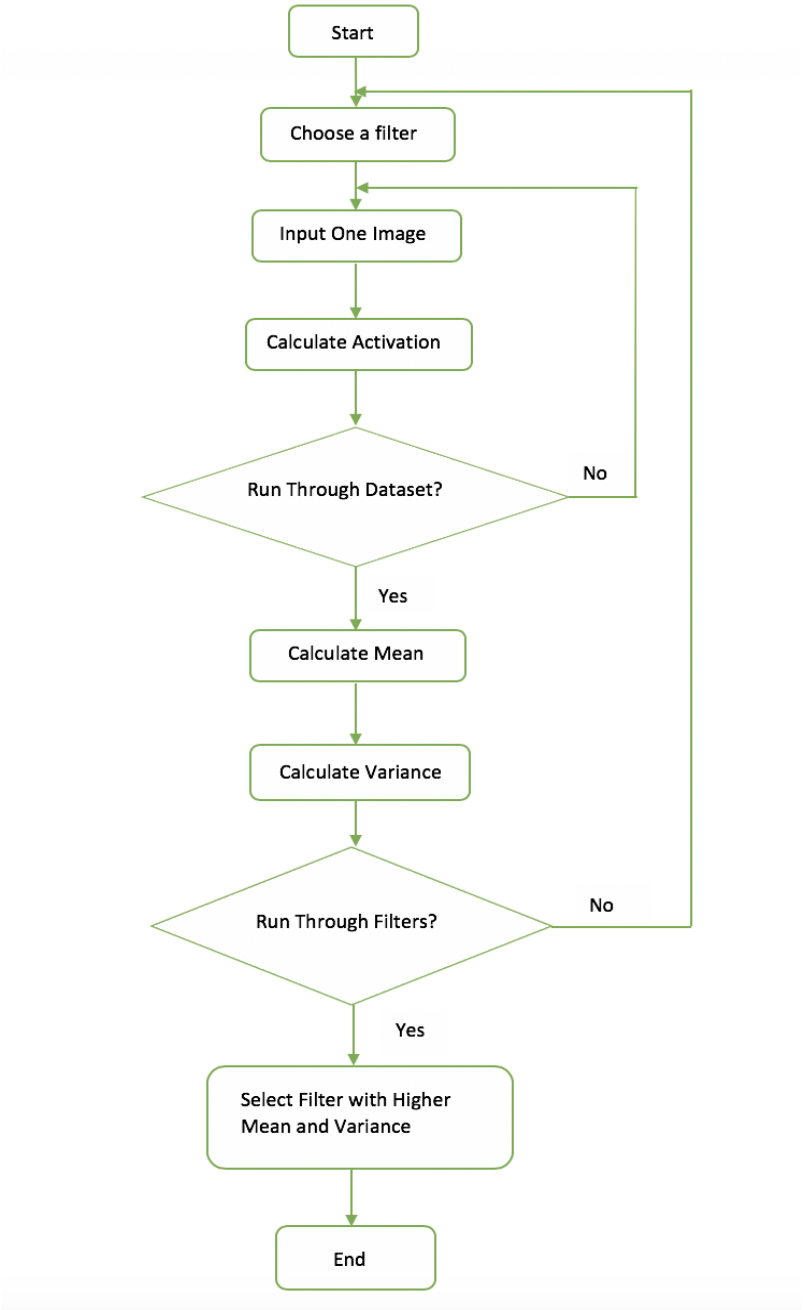

Figure 5–1: Filter Selection Algorithm Pipeline.

on datasets. After getting all the means and variances of different filters, we choose suitable filters with highest variances and high or medium means. When choosing the filters, variances should be considered first because it represents the difference in activation levels when the input images are different objects but in the same class. A filter with highest variance and high or medium mean is very suitable.

5.3 Testing Selected Filters

After selecting several filters with variances and medium or high means, these filters are considered to perform better in discriminating different objects in the same class or be more distinctive representations than randomly selected filters. To confirm this, a testing experiment should be done. Since we want to compare the selected filters and random filters, activation levels can be calculated on test image datasets based on the selected filters and random filters. If the variances of selected filters are higher than random selected filters and also the means are higher or similar, it means the selected filters work well.

5.4 Summary

In this chapter, the key filter selection algorithm is explained. The quantifying process of activation is explained. The filter selection algorithm is addressed based on calculating the means and variances on datasets. In order to prove that the filter selected by the algorithm is better, a testing method is proposed for comparison.

CHAPTER 6
Experiment

6.1 Experiment Design

The whole experiment can be divided into three parts. The first part is to pre-process the datasets, so that input image datasets are good for the experiment. As a result, two datasets on cars and pedestrians respectively are pre-processed and tested for comparison. Then, the filter selection algorithm is performed so that 9 filters are selected in three specific Darknet-53 convolutional layers. They are conv_1 layer, conv_30 layer, and conv_70 layer, representing the first, middle and last layer respectively in Darknet-53. In each layer, three filters with higher variances and high or middle level means are chosen as our useful filters. At last, other 9 random selected filters are chosen in the three layers of conv_1 layer, conv_30 layer, and conv_70. In each layer, three filters are randomly chosen. The randomly selected filters and the 9 useful filters selected by the algorithm are tested on testing datasets and compared.

6.2 Dataset and Preprocessing

6.2.1 Cars Dataset

To calculate one filter by the mean and variance of activation level, filter selection algorithm is run through image datasets. Two datasets of cars and pedestrians are pre-processed and tested in the experiment for comparison. One is the Cars dataset by Jonathan Krause [8] and the other dataset is Penn-Fudan pedestrian dataset [31].

The Cars dataset is good for 3D object representation of cars. It outperforms the state-of-art 2D car dataset for fine-grained categorization. Additionally, the 3D Cars dataset has demonstrated its efficacy in estimating the 3D geometry of images. The Cars dataset contains 16,185 images of 196 classes of cars. Two examples of the Cars dataset are shown in Figure 6-1 and Figure 6-2:

Figure 6–1: Image of blue car with background in Cars dataset [8].

It is obvious that although the dataset is focused on cars, the backgrounds in varied images are different. For instance, the background in Figure 6-1 contains a stony wall and a little car is behind the blue car, which is hard to recognize. The background in Figure 6-2 contains a house. Some small car-shaping objects are around the house. If the whole image is fed into the algorithm, the neural

Figure 6–2: Image of red car with background in Cars dataset [8].

networks might be activated on the backgrounds more. The selected filters might be activated on house-shaped features or stony-shaped texture instead of the front car. To prevent this, they are cropped according to the bounding box of the big car to guarantee a more accurate implementation of the algorithm. For the cropping process, the software I used is MATLAB_R2018b. The corresponding cropped cars are shown in Figure 6-3 and Figure 6-4.

Although car images are cropped so that only car parts are detained, this whole cropped Cars dataset still contains a large number of images. Since the filter selection algorithm takes around 6s to calculate mean activation level per filter per image, feeding the whole image dataset into the algorithm is too time-consuming. As a result, 99 images in the dataset are randomly chosen as the input of the algorithm

Figure 6–3: Cropped blue image.

Figure 6–4: Cropped red image.

for the filter selection process. After some filters with higher means and variances are selected, other 30 images in the cropped Cars dataset are chosen as a testing dataset of cars.

6.2.2 Pedestrian Dataset

To get more general filters that have higher means and variances, pedestrian images are run in filter selection and tested for comparison. The pedestrian dataset used is Penn-Fudan pedestrian dataset. There are 170 images with 345 labeled pedestrians, among which 96 images are taken from around the University of Pennsylvania, and other 74 are taken from around Fudan University. The Penn-Fudan pedestrian dataset contains images that are used for object detection which are mainly taken on campus or urban streets. Each image contains at least one pedestrian. Two examples are shown in Figure 6-5 and Figure 6-6. In order to get only the pedestrian part as our algorithm input, each pedestrian is cropped out according to the bounding box. The three cropped pedestrians are shown in Figure 6-7. Since we only want to know the filter response on the pedestrian part instead of the background part in the image, 99 pedestrians are cropped out to be used as the input image dataset for filter selection algorithm. Other 30 pedestrians are cropped out to be used as the image dataset for testing.

6.3 Filter Selection in YOLOv3

The YOLOv3 filter selection algorithm is based on structure of YOLOv3. Since using Keras version of YOLOv3 (Tensorflow backend) is more convenient, the algorithm is written by Keras.

I used NVIDIA Tesla K40c GPU computing processor. The video memory is 12GB. It takes about 6s to implement this algorithm on one image each filter. Since there are 99 images in each dataset, it takes about 10 minutes to calculate the mean and variance of each filter. Since there are 17728 filters altogether in Darknet-53,

Figure 6–5: One example with two pedestrians in Penn-Fudan pedestrian dataset [31].

it is not wise to calculate all these filters and choose suitable ones. But we can calculate the typical filters in different layers to get a more general idea. Therefore, we calculate 25 filters in the first, middle and last layer of Darknet-53 respectively. They are the conv_1 layer, conv_30 layer, and conv_70 layer. In each layer, the first 25 filters are calculated so that we get the means and variances of filters. In each layer, 3 filters are chosen. In a word, 75 filters are calculated in each dataset so that we can choose 9 filters from each dataset.

6.4 Testing Filters on Datasets

After choosing 9 useful filters out of 75 filters, 9 random filters are also chosen in the three layers. Both 9 useful filters selected by the algorithm and 9 random filters

Figure 6–6: One example with one pedestrian in Penn-Fudan pedestrian dataset [31].

are tested and compared on testing datasets of cars and pedestrians. The means and variances of the filters based on the testing dataset are compared.

6.5 Summary

In this chapter, the whole experiment process is explained. The datasets of cars and pedestrians are pre-processed in order to fit the experiment better. The filter selection algorithm is applied and tested on the two processed datasets. As a result, 9 useful filters are selected in YOLOv3 and compared with 9 randomly selected filters.

Figure 6–7: Three cropped pedestrians that are used as three pedestrian image inputs.

CHAPTER 7
Results

7.1 Filter Selection Algorithm

For the filter selection algorithm, the means and variances of 25 filters in each layer is shown in Figure 7-1, Figure 7-2, and Figure 7-3 respectively.

As we can see, the general means and variances in both car and pedestrian datasets have the same trend. For one filter, if the mean activation value is high in car dataset, the mean activation value is also high in pedestrian dataset. Likewise, if the variance of activation values is high in car dataset, the variance of activation values is also high in the pedestrian dataset. This indicates that some filters are more useful than other filters no matter what the input data classes are.

However, in general, the variances on the car dataset is bigger than on pedestrian dataset while the means in each filter on both car and pedestrian datasets are similar. It is reasonable because cars have more variations in details like shape, pose and color while different human pedestrian have less variations in details, which lead to the more variations on filter activation.

In each layer, we can see that some filters have very high variance with high or medium mean compared to other filters. They are the filters with number 7, 9 and 23 in conv_1 layer, number 8, 15 and 24 in conv_30 layer, and number 8, 13 and 16 in conv_70 layer. The 9 filters with highest variations and high or medium means are chosen from the three layers.

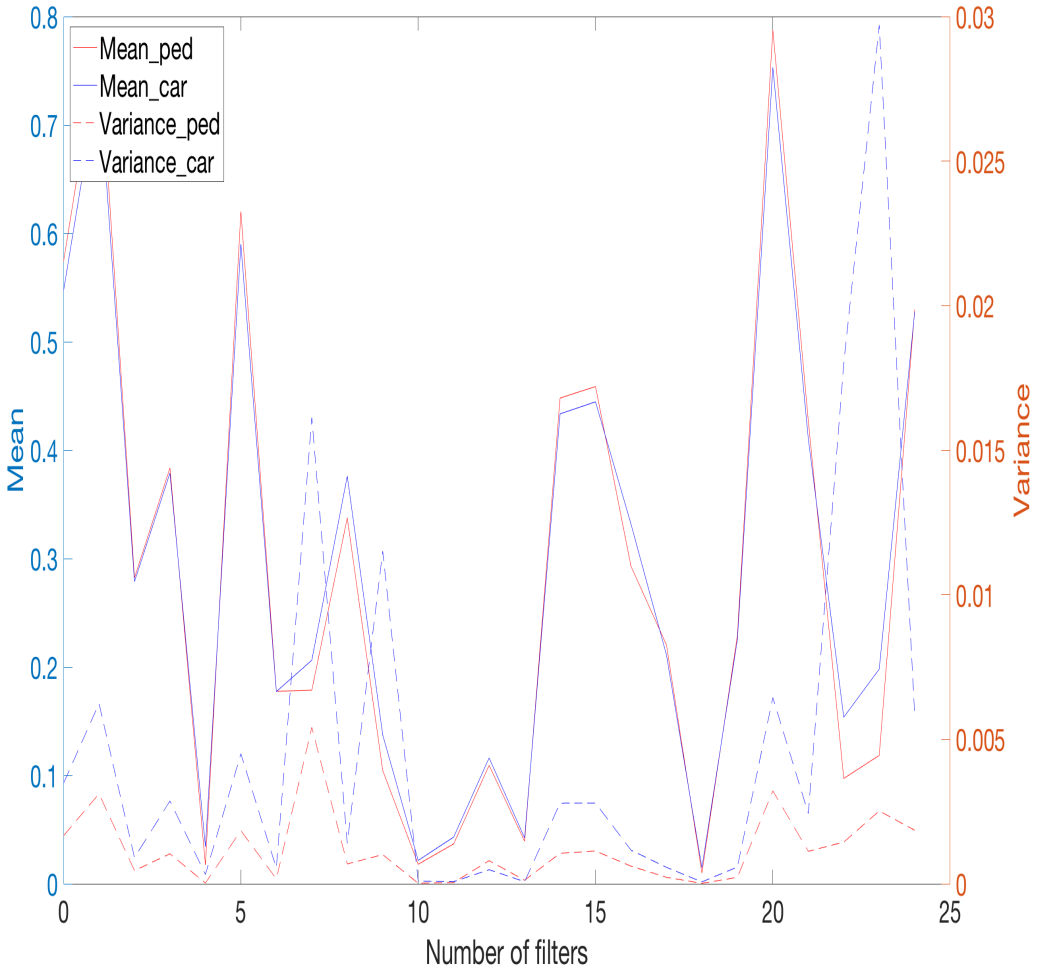

Figure 7–1: Means and variances of the 25 filters in layer conv_1.

7.2 Testing Filters

For testing, 3 filters are randomly chosen in layer conv_1, conv_30, and conv_70 respectively so that 9 filters are randomly chosen to compare with our selected filters from algorithm.

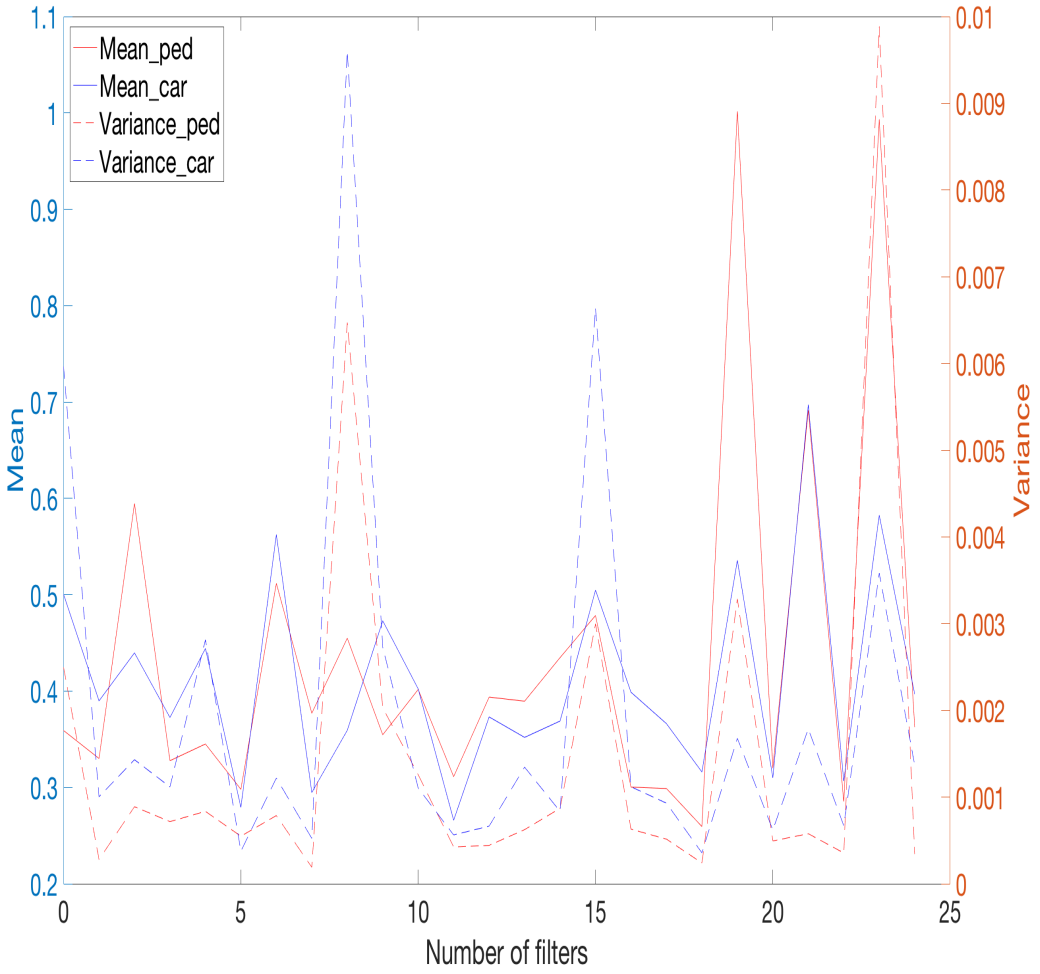

Figure 7–2: Means and variances of the 25 filters in layer conv_30.

In layer conv_1, the means and variances of both 3 selected useful filters by algorithm and the 3 randomly selected filters are shown in Figure 7-4 and Figure 7-7, representing car testing and pedestrian testing respectively. In layer conv_30, the means and variances of both 3 selected useful filters by algorithm and the 3 randomly selected filters are shown in Figure 7-5 and Figure 7-8,representing car testing and

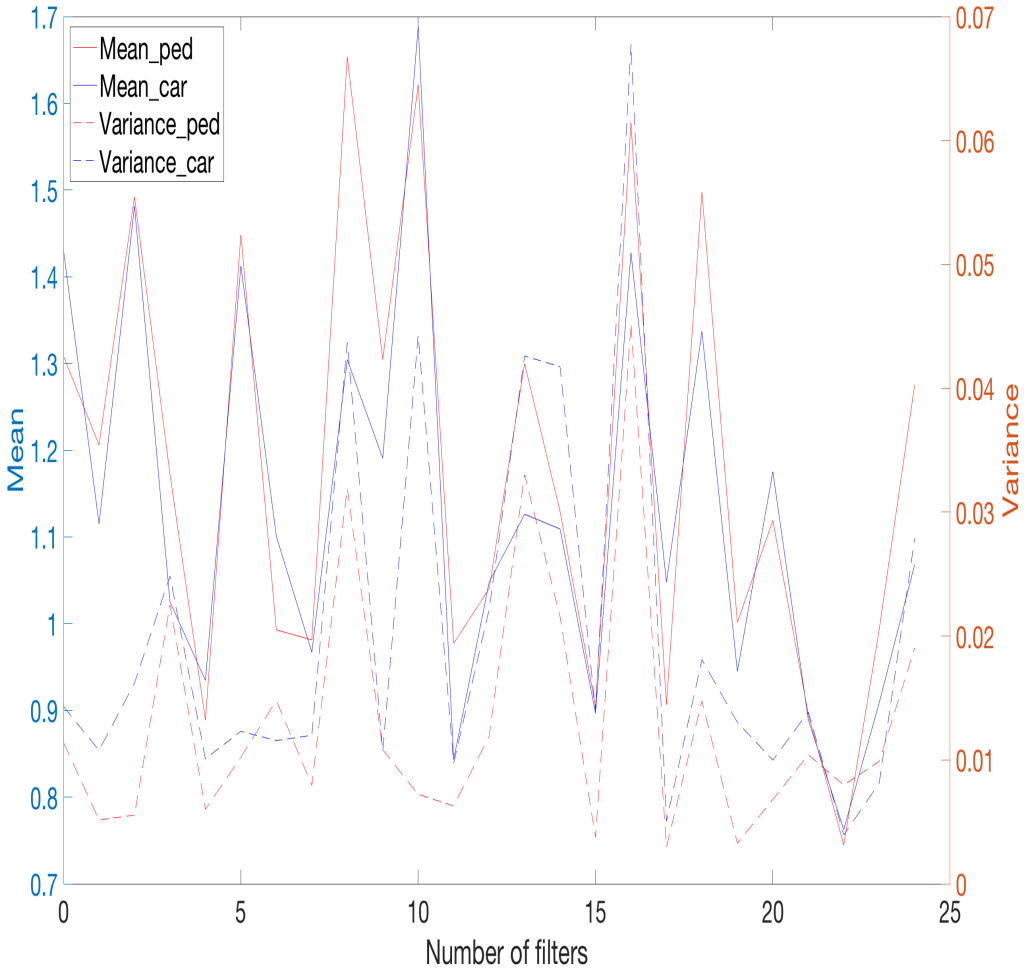

Figure 7–3: Means and variances of the 25 filters in layer conv_70.

pedestrian testing respectively. In layer conv_70, the means and variances of both 3 selected filters from algorithm and the 3 randomly selected filters are shown in Figure 7-6 and Figure 7-9, representing car testing and pedestrian testing respectively.

We can see that the variances of selected useful filters by algorithm are higher than filters chosen randomly, meaning that the selected filters have more variations

on different objects in the same class. The means of selected useful filters by algorithm are generally higher or similar compared to the filters chosen randomly. This shows that the selected useful filters are generally more or similarly activated with randomly-selected filters.

The algorithm-selected filters generally have higher or similar means compared to randomly-chosen filters, which indicates that the selected-useful filters are generally more sensitive and activated on objects. Since the selected filters have higher variance than randomly-selected filters, the selected useful filters will be more useful in object tracking because for the various objects in the same class, the selected useful filters will give different responses to the various objects while randomly-chosen filters will give similar responses to the various objects. The difference of the responses of selected filters is much bigger than randomly-chosen filters for the objects in the same class even if they have similar means of activation. Therefore, the selected filters can differentiate various objects in the same class better than randomly-chosen filters cannot.

In conclusion, these comparisons indicates that the filter selection algorithm can select more useful filters that contain more information.

7.3 Summary

In this chapter, the results of experiment are shown. 9 useful filters are selected by algorithm and also tested and compared with 9 other randomly selected filters. It turns out that the 9 selected filters have high variations and high or medium means so that they are good for discriminating different objects of the same class.

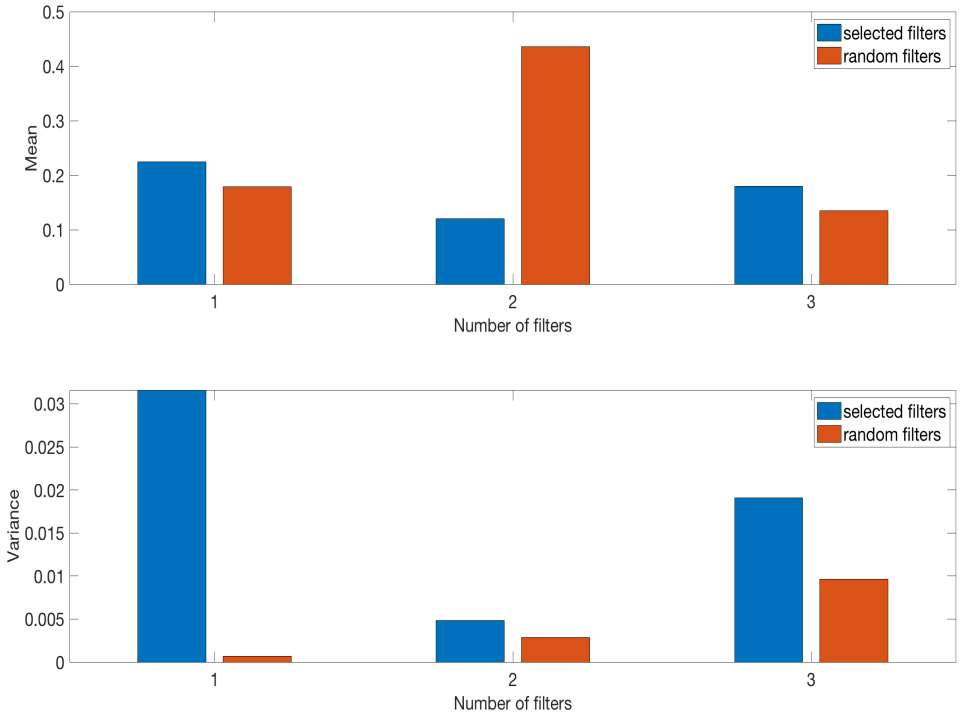

Figure 7–4: Means and variances on car testing dataset in layer conv_1.

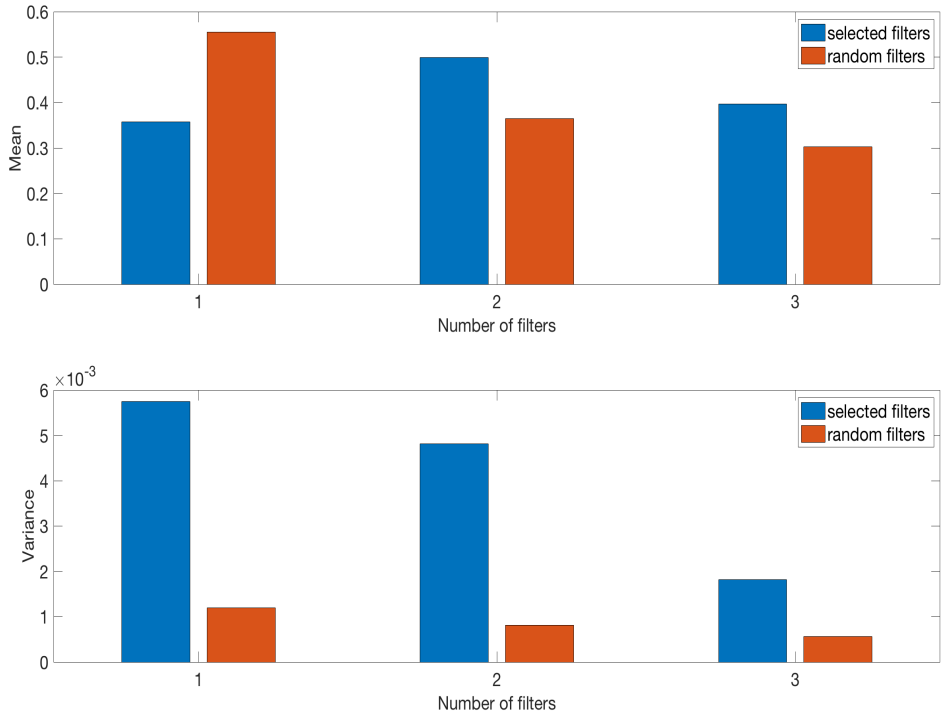

Figure 7–5: Means and variances on car testing dataset in layer conv_30.

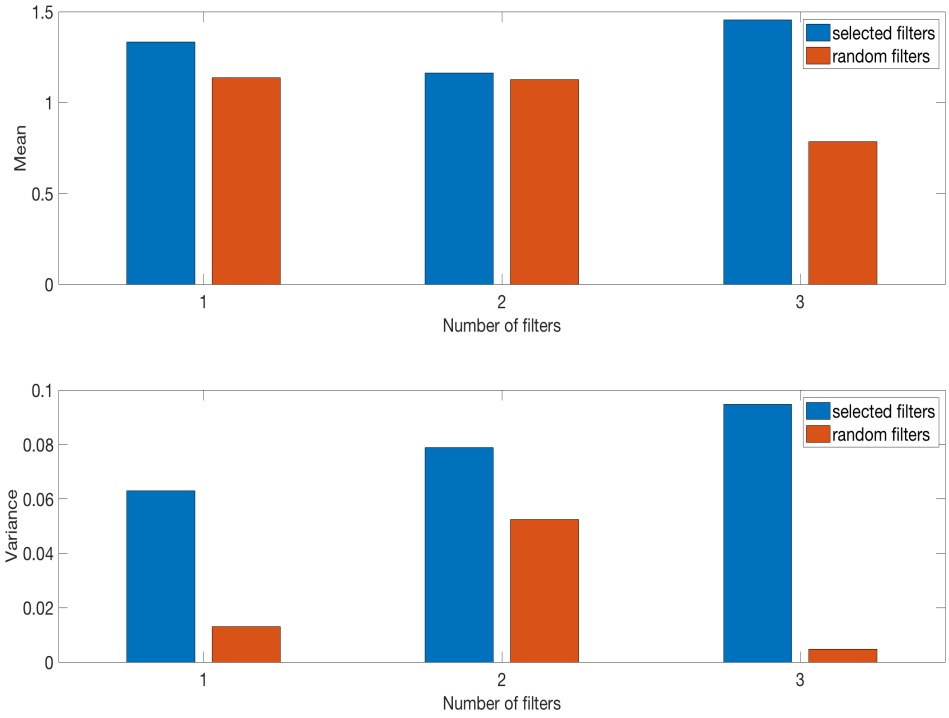

Figure 7–6: Means and variances on car testing dataset in layer conv_70.

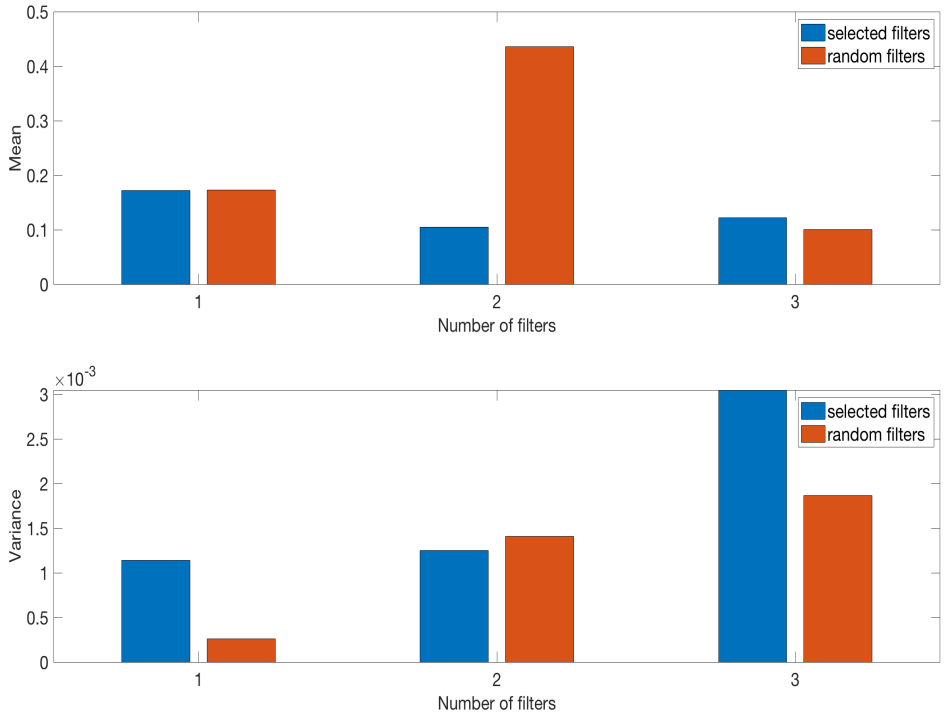

Figure 7–7: Means and variances on pedestrian testing dataset in layer conv_1.

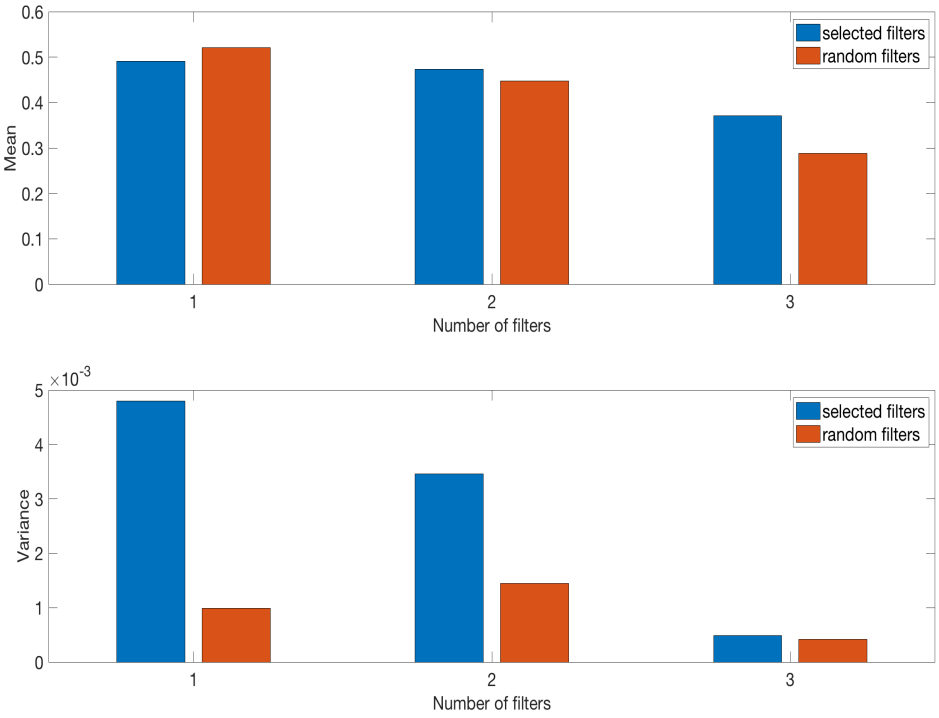

Figure 7–8: Means and variances on pedestrian dataset in layer conv_30.

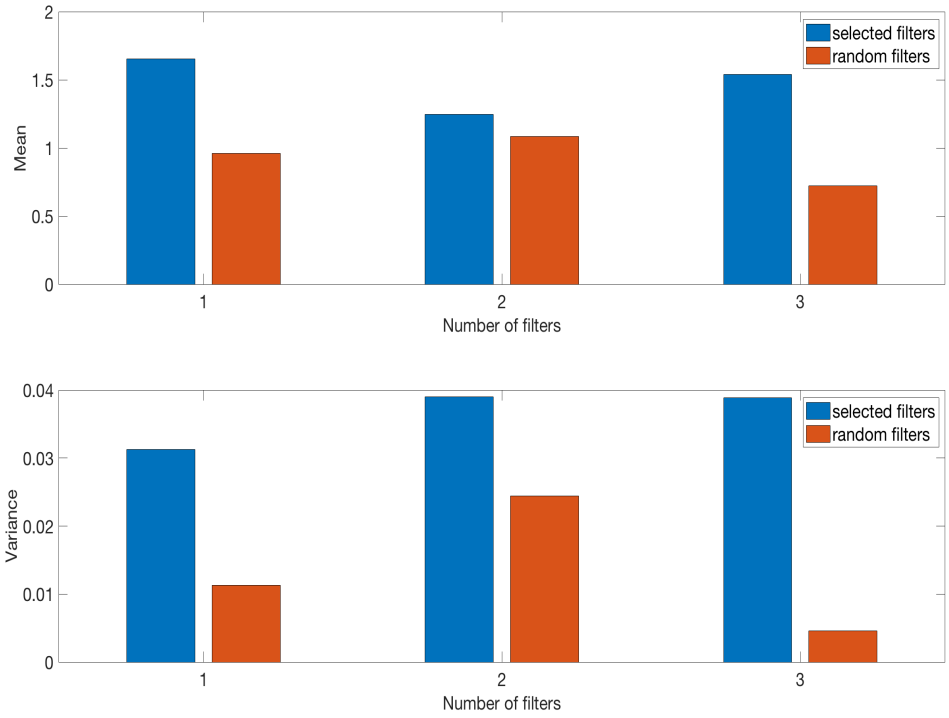

Figure 7–9: Means and variances on pedestrian testing dataset in layer conv_70.

CHAPTER 8
Discussion

Through testing the algorithm, we know that the means of selected useful filters are generally higher or similar with randomly-chosen filters, which tells us that algorithm-selected filters are generally more or similarly activated on objects than randomly-selected filters. Additionally, the selected useful filters have more variances than randomly-chosen filters on various objects in the same class, which indicates that these selected filters by the algorithm will give more different responses to objects in the same class. These high difference of responses makes the selected filters more useful in object tracking for differentiating various objects in the same class. They can also be regarded as more distinctive features to represent specific objects.

Since we use means and variances to represent the quality of filters, it is assumed that the distribution of filter activation levels on image datasets is Gaussian distribution. Figure 8-1, Figure 8-3, Figure 8-4 and Figure 8-6 are the 4 typical histograms of filter activation level on the 99 cropped car dataset. They can be roughly regarded as the Gaussian distribution. However, some other filters such as filter 23 in conv_1 shown in Figure 8-2, filter 0 in conv_70 shown in Figure 8-5 do not have very distinctive shapes of Gaussian distribution on the histograms. In such situation, means and variances might not be the best way to represent filters.

Since we choose the filters based on variances and means, how to embed the chosen filters and make use of them in object tracking is another question to be probed

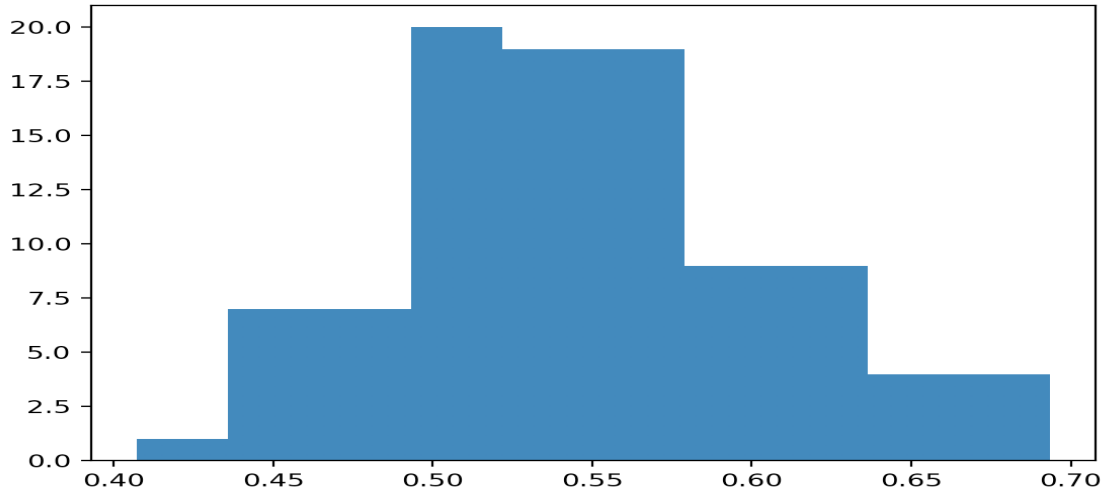

Figure 8–1: Histogram of activation level of filter 0 in layer conv_1 on 99 cropped car dataset.

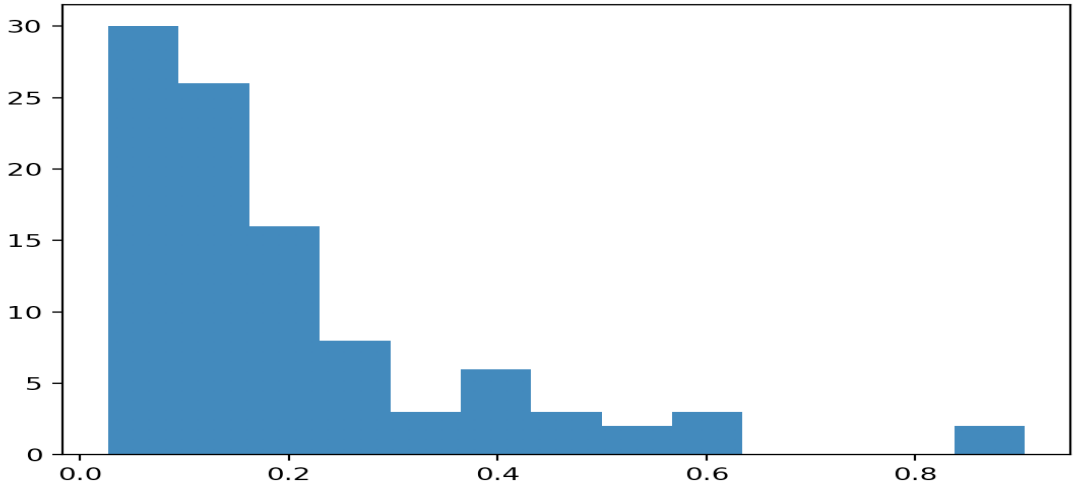

Figure 8–2: Histogram of activation level of filter 23 in layer conv_1 on 99 cropped car dataset.

in the future. The values of means and variances can be set as features representing the filters and extra distinctive features to differentiate various objects in the same

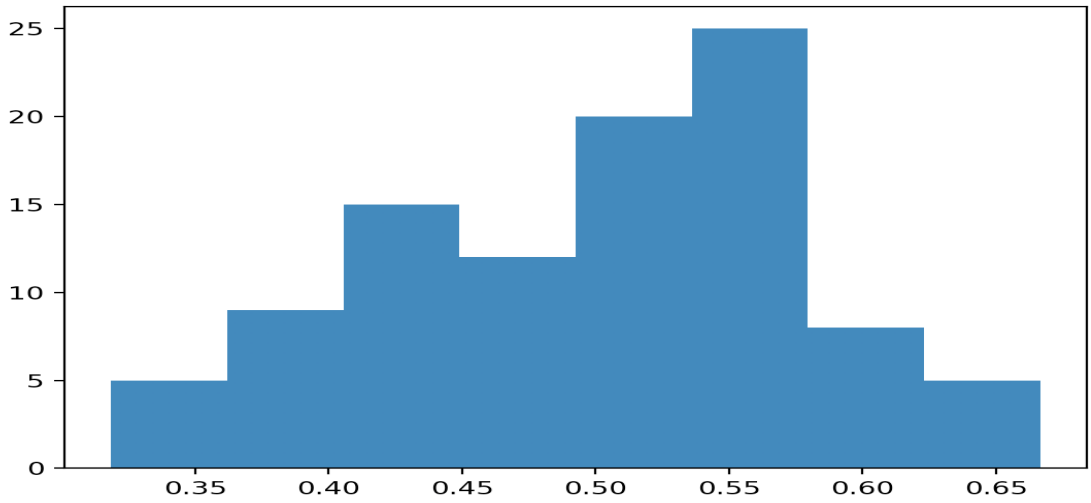

Figure 8–3: Histogram of activation level of filter 0 in layer conv_30 on 99 cropped car dataset.

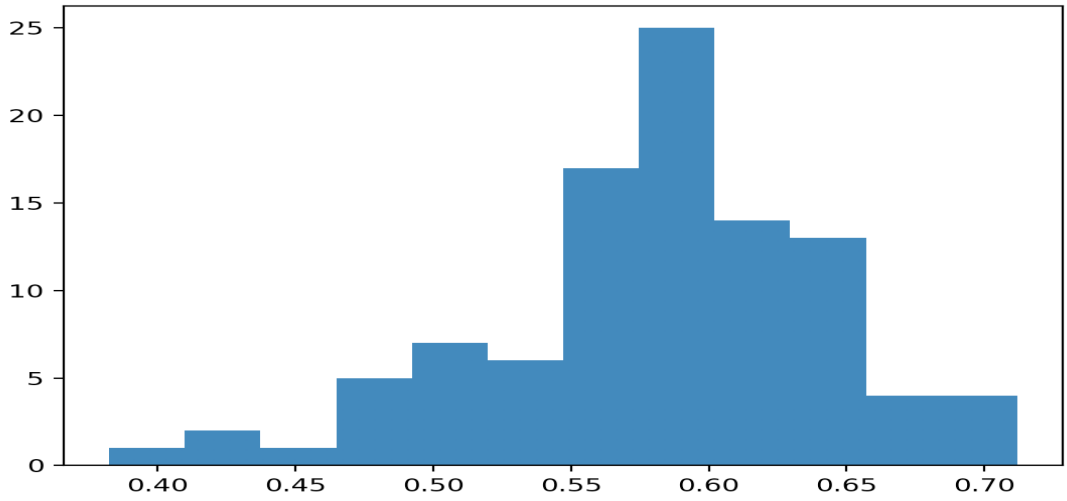

Figure 8–4: Histogram of activation level of filter 23 in layer conv_30 on 99 cropped car dataset.

class. Another kind of combination of mean and variance such as multiplication or adding the two can also be considered as useful features for object tracking.

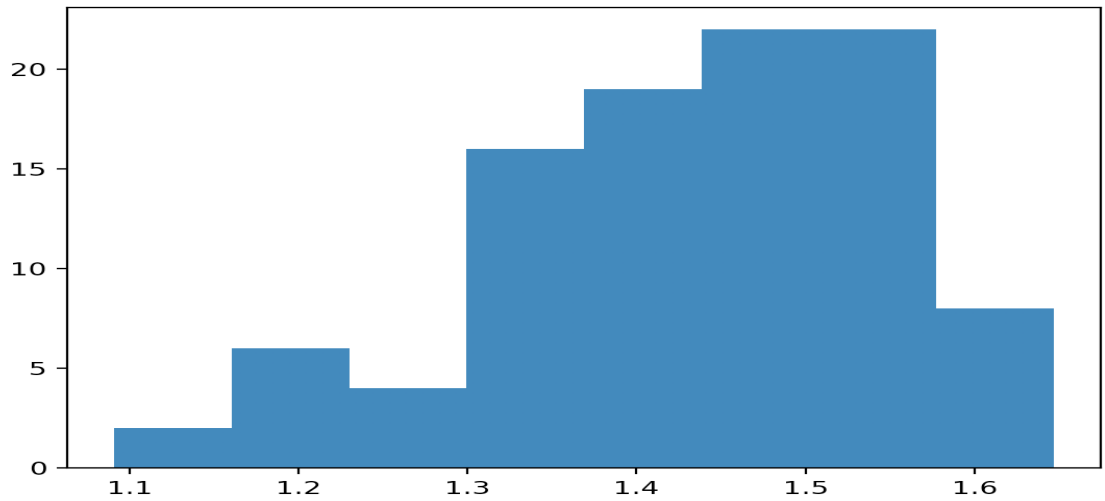

Figure 8–5: Histogram of activation level of filter 0 in layer conv_70 on 99 cropped car dataset.

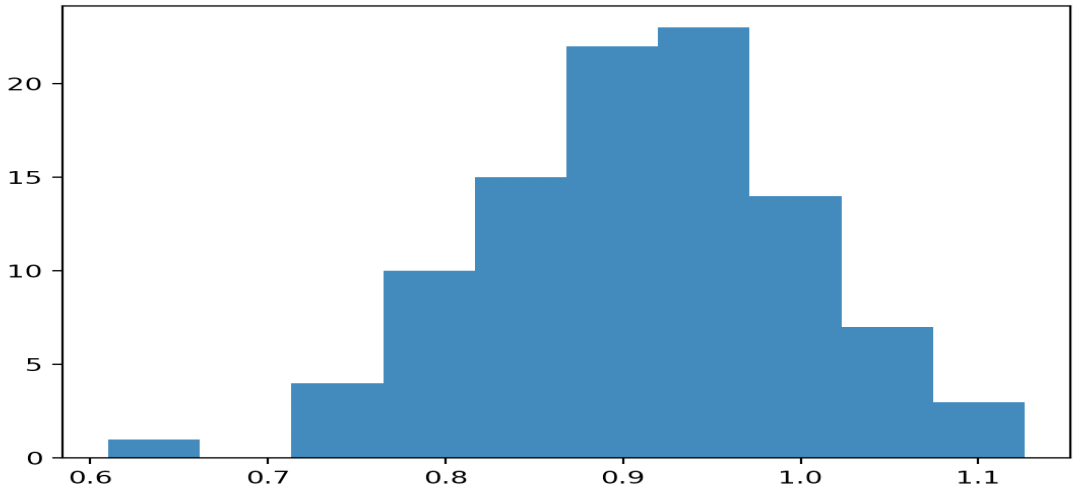

Figure 8–6: Histogram of activation level of filter 23 in layer conv_70 on 99 cropped car dataset.

What's more, since the calculation of mean and variance on activation levels of the whole dataset can be very time-consuming, two 99-image datasets are chosen

in filter selection and two 30-image datasets are chosen as testing datasets. How to make the dataset contain more images while the calculation time is shorter is also be a very crucial problem to be addressed so that our results are more general.

In a few words, to use the selected filters and their features, one way is just using the means and variances. Another way is to use the combination of means and variances such as adding them together or multiplying them. What's more, the output of each filter can be considered a feature vector if needed.

Although there are some limitations, the filter selection algorithm is a very generalized method to select filters in convolutional neural networks. The filter selection algorithm is not only restricted to YOLOv3, but it can also be applied to other CNNs so that the underlying information learned by CNNs is extracted and can be used instead of using the massive numbers in CNNs which are redundant and not efficient for object tracking. The selected filters can be used as special features to represent objects. This filter selection algorithm is also not restricted in object tracking problem and can be applied to other deep learning problems as a method of feature extraction.

8.1 Summary

In this chapter, the filter selection algorithm is discussed. Some applications and also limitations of the algorithm are proposed. How to break these limitations are considered valuable problems to be solved in the future. Except for the limitations, the algorithm is also very generalizable to other convolutional neural networks as a way of feature extraction.

CHAPTER 9
Conclusion

In this project, based on YOLOv3 and CNNs visualization theories, a filter selection algorithm is proposed and implemented on two image datasets to enhance object tracking efficiency. The results are compared and analyzed by testing the algorithm. The variances and means of activation level are compared through both selected useful filters by the algorithm and filters randomly chosen. It turns out that the filter selection algorithm can help us select more useful filters than just choosing filters randomly. The chosen useful filters have more variations on different objects in the same class so that they can differentiate objects better. The means and variances of filters are more distinctive as features than randomly selected filters.

The proposed filter selection algorithm is also generalizable to other convolutional neural networks. It is a feature extraction method of different objects of the same class for convolutional neural networks. It can also be applied to other problems as long as there is a need to extract features to represent various objects in deep learning.

Bibliography

[1] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. "Robust object tracking with online multiple instance learning". In: *IEEE transactions on pattern analysis and machine intelligence* 33.8 (2010), pp. 1619–1632.

[2] Benjamin Coifman et al. "A real-time computer vision system for vehicle tracking and traffic surveillance". In: *Transportation Research Part C: Emerging Technologies* 6.4 (1998), pp. 271–288.

[3] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *international Conference on computer vision & Pattern Recognition (CVPR'05)*. Vol. 1. IEEE Computer Society. 2005, pp. 886–893.

[4] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[5] Ross Girshick. "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[6] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[7] Jun-Wei Hsieh et al. “Automatic traffic surveillance system for vehicle tracking and classification”. In: *IEEE Transactions on Intelligent Transportation Systems* 7.2 (2006), pp. 175–187.

[8] Jonathan Krause et al. “3D Object Representations for Fine-Grained Categorization”. In: *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*. Sydney, Australia, 2013.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[10] Wei Liu et al. “Ssd: Single shot multibox detector”. In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[11] Ye Liu, Hui Li, and Yan Qiu Chen. “Automatic tracking of a large number of moving targets in 3d”. In: *European Conference on Computer Vision*. Springer. 2012, pp. 730–742.

[12] David G Lowe et al. “Object recognition from local scale-invariant features.” In: *iccv*. Vol. 99. 2. 1999, pp. 1150–1157.

[13] Hui-Lan Luo, Hui Wei, and Loi Lei Lai. “Creating efficient visual codebook ensembles for object categorization”. In: *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 41.2 (2010), pp. 238–253.

[14] Hiroshi Murase and Shree K Nayar. “Visual learning and recognition of 3-D objects from appearance”. In: *International journal of computer vision* 14.1 (1995), pp. 5–24.

[15] Kar Pathy. *Convolutional Neural Networks for Visual Recognition*. 2015. URL: http://cs231n.github.io/understanding-cnn/.

[16] Joseph Redmon and Ali Farhadi. "YOLO9000: better, faster, stronger". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271.

[17] Joseph Redmon and Ali Farhadi. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[18] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[19] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[20] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[21] Nicolas Saunier and Tarek Sayed. "A feature-based tracking algorithm for vehicles in intersections". In: *The 3rd Canadian Conference on Computer and Robot Vision (CRV'06)*. IEEE. 2006, pp. 59–59.

[22] Bernt Schiele and James L Crowley. "Object recognition using multidimensional receptive field histograms". In: *European Conference on Computer Vision*. Springer. 1996, pp. 610–619.

[23] Cordelia Schmid and Roger Mohr. "Local grayvalue invariants for image retrieval". In: *IEEE transactions on pattern analysis and machine intelligence* 19.5 (1997), pp. 530–535.

[24] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps". In: *arXiv preprint arXiv:1312.6034* (2013).

[25] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[26] Michael J Swain and Dana H Ballard. "Color indexing". In: *International journal of computer vision* 7.1 (1991), pp. 11–32.

[27] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[28] Christian Szegedy et al. "Scalable, high-quality object detection". In: *arXiv preprint arXiv:1412.1441* (2014).

[29] Jasper RR Uijlings et al. "Selective search for object recognition". In: *International journal of computer vision* 104.2 (2013), pp. 154–171.

[30] Paul Viola, Michael Jones, et al. "Rapid object detection using a boosted cascade of simple features". In: *CVPR (1)* 1 (2001), pp. 511–518.

[31] Liming Wang et al. "Object detection combining recognition and segmentation". In: *Asian conference on computer vision*. Springer. 2007, pp. 189–199.

[32] Jason Yosinski et al. "How transferable are features in deep neural networks?" In: *Advances in neural information processing systems*. 2014, pp. 3320–3328.

[33] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[34] Matthew D Zeiler, Graham W Taylor, Rob Fergus, et al. "Adaptive deconvolutional networks for mid and high level feature learning." In: *ICCV*. Vol. 1. 2. 2011, p. 6.

[35] Lu Zhang and Laurens van der Maaten. "Structure preserving object tracking". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013, pp. 1838–1845.