TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

BISHWO PRAKASH ADHIKARI

CAMERA BASED OBJECT DETECTION FOR INDOOR SCENES

Master of Science thesis

Examiner: Prof. Heikki Huttunen
Examiner and topic approved by the
Faculty Council of the Faculty of
Computing and Electrical Engineering
on 28th March 2018

ABSTRACT

BISHWO PRAKASH ADHIKARI: Camera Based Object Detection for Indoor Scenes
Tampere University of Technology
Master of Science thesis, 56 pages
May 2018
Master's Degree Programme in Information Technology
Major: Data Engineering and Machine Learning
Examiner: Prof. Heikki Huttunen
Keywords: convolutional neural networks, deep learning, machine learning, object detection, TensorFlow Object Detection API

This master thesis describes a practical implementation of a deep learning framework for object detection on the self-collected multiclass dataset. The research work presents multiple perspectives of the data collection, labelling, preprocessing and training popular object detection architectures. The challenges in the collection of multiclass object detection dataset from the indoor premises and annotation process are presented with possible solutions. The performance evaluations of the trained object detectors are measured in terms of precision, recall, F_1-score, mAP and processing speed.

We experimented multiple object detection architectures that were available on the TensorFlow object detection model zoo. The multiclass dataset collected from the indoor premises were used to train and evaluate the performance of modern convolutional object detection models. We studied two scenarios, (a) pretrained object detection model and (b) fine-tuned detection model on the self-collected multiclass dataset. The performance of fine-tuned object detectors was better than the pretrained detectors. From our experiment, we found that region based convolutional neural network architectures have superior detection accuracy on our dataset. Faster region-based convolutional neural network (RCNN) architecture with residual networks features extractor has the best detection accuracy. Single shot multi-box detector (SSD) models are comparatively less precise in detection. However, they are faster in computation and easier to deploy in mobile and embedded devices. It is found that the region-based fully convolutional network (RFCN) is the suitable alternative for multi-class object detection considering the speed/accuracy trade-offs.

PREFACE

This master thesis research has been conducted at the Department of Signal Processing of the Tampere University of Technology (TUT). I appreciate the support of all people who helped me during my study and this thesis project. This thesis is dedicated to everyone who will find this information useful.

At first, I would like to express my deep gratitude to my supervisor Associate Professor Heikki Huttunen for allowing me to work at TUT Machine Learning Group and introducing me to this interesting research area. This would not be possible without his support and valuable advices throughout the process. I appreciate his role as mentor, supervisor and examiner. I would like to thank Jukka Peltomaki and other members of TUT Machine Learning Group for their support and valuable advices.

I would like to thank my parents, family, relatives and colleagues for supporting me throughout my studies and life in general. Many thanks to my girlfriend for her encouragement, support and faith on me.

20 May 2018, Tampere, Finland

Bishwo Prakash Adhikari

CONTENTS

FIGURES

TABLES

LIST OF ABBREVIATIONS AND SYMBOLS

ACC	Accuracy
AI	Artificial Intelligent
API	Application Programming Interface
CNN	Convolutional Neural Network
CV	Cross Validation
DNN	Deep Neural Network
FN	False Negative
FNR	False Negative Rate
FP	False Positive
FPR	False Positive Rate
FPS	Frames Per Second
GPU	Graphical Processing Unit
IoU	Intersection over Union
mAP	Mean Average Precision
OpenCV	Open Computer Vision
RCNN	Recurrent Convolutional Neural Network
RFCN	Region-based Fully Convolutional Network
SSD	Single Shot Detector
TF	Tensorflow
TN	True Negative
TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate
TUT	Tampere University of Technology
XML	Extensible Markup Language

B_{gt}	Ground-truth bounding box
B_p	Predicted bounding box

1. INTRODUCTION

Human can easily recognize various objects around our surrounding. However, it is still a tedious job for a computer to recognize common objects correctly. For example, it is easy for a mature human to differentiate apples and oranges but for very children, it is difficult to distinguish which one is an apple and which one is an orange. When we teach them several times with multiple examples of apples and oranges, they will start to learn the object category structure and able to know the difference. The same principle is applied to the computer. We teach a computer using a large numbers of object images (data) then it will start to learn the structure and predict the result based on its learning.

Human can precisely complete the challenging tasks that need intelligence, while the robots can do risky and tedious tasks well. The interaction between a human and a robot is profitable as together they can do complex, repetitive tasks applying intelligence. The interaction between humans and automated robots in working places can be risky for work and workers because of the possible collision with the robot and surrounding object. To reduce the possible damage, robot systems must recognize humans, objects and get their locations precisely all the time. The automated system requires an intelligent way to recognize, localize and track an object in real-time. This is a challenging task to achieve the good results in, regarding that the work environment might be crowded, many objects might be occluding each other, there might be a dissimilarity in illuminations, view angles and the same object might appear in varied sizes [16].

The machine learning algorithm that is capable of localizing and classifying the objects from image and video frames is an interesting topic in artificial intelligence (AI), especially in computer vision. The long-time research in AI focuses on how to make a computer work like a human [19]. The power of AI and machine learning is automating the substantial number of computing tasks. The focus is to develop an algorithm that can teach the computer how to recognize and track an object like a human or better than a human. The demand of the deep neural network (DNN) based approach for real-time object detection is increasing rapidly.

Camera-based object detection is one of the fastest growing research areas in the machine learning and computer vision [1]. The state-of-the-art technology is heavily used in the industries and the production line to detect objects, localize, track and inspect them in an automated and semi-automated environment. Object detection on image and video has been studied and implemented in various places such as computer vision, robotics, automation, construction and agriculture [9]. Well-functioning real-time object detection is the utmost goal for the object recognition, localization, object tracking, navigation and work safety in an automated environment.

An autonomous driving car is a well-known use case of real-time object detection using a Lidar sensor camera [3]. Another example is car-manufacturing facilities where humans and automated robots are working together. The real-time detection with the precise and robust performance is crucial in those scenarios. A small error might lead to a collision between humans and other objects. There is no room for a single false alarm, which could lead to human casualties and other damages. It is a hard job for a computer to detect objects in all environment regardless of the background, size, occlusion and lighting conditions. This is the major challenge in building a general-purpose object detector. The dataset that is used to train the model has significant effects on the trained detector performance. It might not detect the object which is collected from another environment.

The successful experiment of human detection in machinery working environment was the inspiration for this thesis project. We experimented with classic Histogram of Oriented Gradient (HOG), Convolutional Neural Network (CNN) implemented in Dlib library[1] and TensorFlow object detection application programming interface (API) for human detection. We compared the pretrained model and specially trained detector on the self-collected dataset from the real environment. It was found that the performance of the fine-tune model on own dataset was far better than the available pretrained models. From our experiment, we found that the object detection models available on TensorFlow object detection model zoo were better among the tested methods. Deep learning object detection models were better than the traditional HOG detector in detecting the human from the series of images/videos.

Object detection on the single class dataset is comparatively less challenging than in multiclass. We aimed to experiment with a more challenging and demanding task of object detection. We wanted to collect a multi-class object dataset from indoor premises, train existing object detection frameworks and test the performance of trained detectors on the self-collected dataset. In this thesis project, we focused more

[1] `http://dlib.net`

on collecting the multi-class dataset and implementing deep learning object detection models available on TensorFlow object detection model zoo. We experimented with the state-of-the-art of the real-time object detection on self-collected images and videos. Moreover, our goal was to know which object detection framework is better in performance and computation in detecting objects from our multiclass dataset.

The structure of this thesis is as follows. Chapter 2 introduces the terms and techniques used in machine learning, deep learning and neural networks. We discuss the concepts that are needed to understand this thesis contents. In chapter 3, the concept of object detection, popular deep learning object detection architectures, feature extractors and challenges for automatic object detection are described. In chapter 4, we provide information of the data collection, annotation, preprocessing, purposed object detection models together with the environmental requirements to train and evaluate purposed object detection models. Chapter 5 contains the experimental results to demonstrate how well fine-trained detectors perform on detecting objects and predicting their locations. The comparison of detectors on different evaluation subsets are discussed there. In chapter 6, we summarize our experimental findings and present possible improvement proposals for future.

2. METHODS

This chapter describes the theoretical background of machine learning, neural network, and deep learning. We discuss common assessment matrices that are used in machine learning and for the performance assessment of object detection model.

2.1 Machine Learning

Machine learning is a branch of AI where an algorithm learns from data and results a model from that data. Machine learning algorithm is able to learn from the data rather than be explicitly programmed [11]. It can be used in various computing tasks where designing and developing an explicit algorithm with good performance is difficult or infeasible [2]. Machine learning uses representation learning to learn the representation of the input object (data) automatically. The learned representation is then used to predict the result of new unseen data.

These days computer related tasks, businesses and systems are advanced by the machine learning techniques [2]. The automatic detection of spam in email, recommendation systems for the online stores, fraud-detection systems for the banks and insurance agencies, content filtering, classification, object recognition and fault detection systems are using machine learning techniques to solve problems automatically. Based on the learning principle, machine learning algorithms are divided into supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning.

Supervised Learning

Supervised learning is the machine learning system that is trained under supervision, learning with a teacher. The algorithm is trained using the labelled data and desired outputs. The supervised learning algorithm builds the model based on labelled data and tries to predict on the new dataset. For the input variables ($\boldsymbol{x} = \boldsymbol{x_1}, \boldsymbol{x_2}, .., \boldsymbol{x_n}$) and output variables or labels ($\boldsymbol{y} = \boldsymbol{y_1}, \boldsymbol{y_2}, .., \boldsymbol{y_n}$) the algorithm uses a mapping function from the input to the output as $\boldsymbol{y} = f(\boldsymbol{x})$. The idea is to approximate

the mapping function $f(\boldsymbol{x})$ well enough that when there is new input data $\boldsymbol{x'}$, the model can predict the output labels $\boldsymbol{y'}$ of that data.

Classification and regression are the popular examples of supervised learning. Classification is the process of predicting class category of input data. Identifying cats and dogs from the dataset containing cats and dogs is an example of classification. In regression, the prediction of a continuous value is done based on the trained data. The prediction of house prices based on house features is a typical example of regression. Regression is identical to the classification task except for its output, which is the numeric value rather than the categorical label. Decision trees, linear regression, logistic regression, Naive Bayes classifier, nearest neighbour, neural network and Support Vector Machines (SVMs) are examples of supervised machine learning algorithm [11].

Unsupervised Learning

Machine learning systems that are trained without supervision or using the unlabelled data is known as unsupervised learning. In this learning, we only have the input data with no corresponding output variables (labels). The goal is to design or find the pattern of the data to learn more about the data. Clustering, visualization and dimensionality reduction are common examples of unsupervised machine learning algorithms. Clustering is the method of making clusters of data based on the similarity measure. The similar featured data are kept inside the same cluster while different featured are kept separately. The process of representing the unlabelled complex data into 2D or 3D visual representation is known as visualization. Dimensionality reduction is the technique of reducing the dimension of the data without losing the important information. During this process, correlated features are merged into one feature. Association rule learning is another example of unsupervised learning where the goal is to find the interesting relations between attributes of data. [11]

Semi-supervised Learning

Semi-supervised learning is the mixture of both supervised and unsupervised learning. It is used to solve the problem where the dataset is partially labelled or missing some of the labels. The collection of the large-scale dataset, annotating/labelling is time consuming and expensive process. Including unlabelled dataset helps to collect the large enough dataset within reasonable time and cost. Semi-supervised learning is the best option to learn from partially labelled data. [11]

Reinforcement Learning

Reinforcement learning is quite different than above-mentioned learning principles. In reinforcement learning, an agent learns by interacting with the environment to perform its task [11]. An agent notices the environment, takes some action to interact with the environment and obtains the reward (positive or negative) based on its actions. An agent learns from own experience and collects the training examples through trial and error during its attempts to complete the task. This process is continued until the agent gets the maximum reward and completes the task. Markov Decision Processes and Q-learning are the examples of reinforcement learning. Reinforcement learning has been implemented in many deep learning models. [11]

Overfitting

The main challenge for the machine learning algorithm is to perform well on unseen data. The ability to perform well on previously unseen data or data other than training set is called *generalization*. The generalization error or test error is the expected value of the error measure on new data. For the good performance of an algorithm, the generalization error must be as low as possible. The training error can be calculated from the training set. The factor that determines the performance of machine learning algorithm is its aptitude to make the training error small and generalization gap small. Generalization gap is the distance between the training error and test error [18].

Underfitting and Overfitting terms are used to describe the machine learning challenges. Underfitting occurs when the model has high training error. Underfitting is the case when the model is not training well or model is too easy to learn patterns of input data. Overfitting happens when the generalization gap is high. Overfitting is the case when the model performs well on the input (train) data but poorly generalize on unseen data. *Regularization* is a technique used to make modification on the learning algorithm to mitigate the test error but not the training error. Regularization tries to construct the model structure as simple as possible which can avoid the effect of overfitting [18].

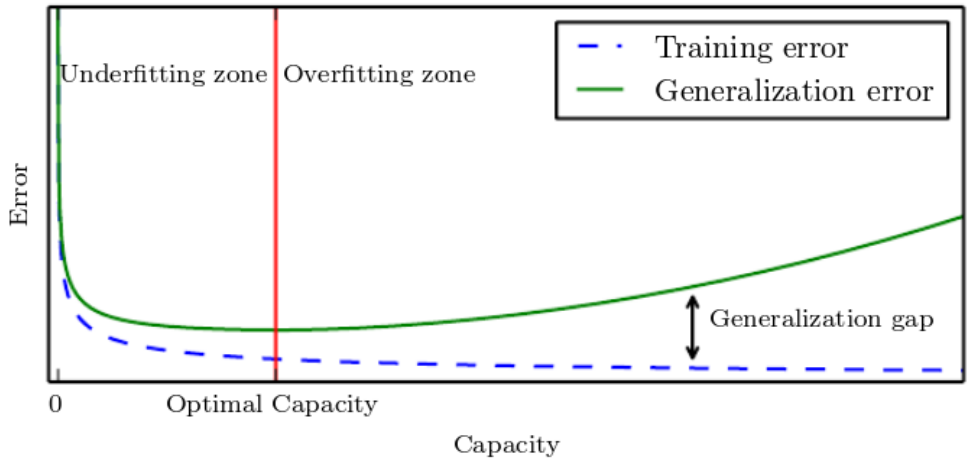

Figure 2.1 *The relationship between model capacity and error. [18].*

The behaviour of training and test (generalization) error is different at the different level of model capacity as shown in Figure 2.1. Capacity is the ability of the model to fit the wide range of functions. In the beginning, both the training and generalization errors are high. This area is known as the underfitting zone. When the capacity increases, the training error decreases but the generalization error starts increasing. At the overfitting zone, the training error is lower but the generalization error is higher, making the generalization gap bigger. The optimal capacity is the boundary line to distinguish the underfitting zone and overfitting zone. [18]

Model Performance Evaluation and Cross Validation

To understand how well our machine learning model will generalize on the new data, we need to try it with different instances of data than the data used to train the model. The goal is to split the dataset into two disjoint subsets, training and testing subsets. The model is trained with training subset and evaluate with the testing subset. Model performance is calculated based on the training error and testing error calculated using those subsets. The common practice in machine learning is to use 80 percent of total data to train the model and 20 percent to test/evaluate the model. Cross-validation (CV) is a technique to partition the dataset into disjoint training and testing subset. Majority of data, the training subsets, is used to train the machine learning model and the remaining data is used for model evaluation. K-fold CV and stratified k-fold CV are common practices in machine learning. In k-fold CV the dataset is divided into k subsets (folds) and each time one-fold is reserved for the testing and the remaining k-1 folds are combined to form the training set.

Stratified k- fold CV is the modification of k-fold CV where each k folds contain approximately the same portion of the sample of target class as in the whole dataset. Subsets created using stratified k-fold CV technique have equal representation of each class as in the original dataset. [11]

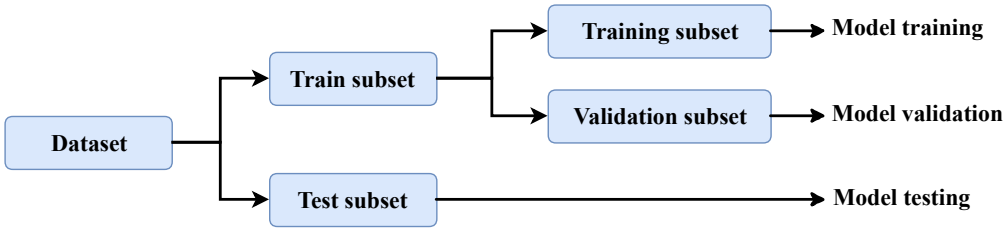

Figure 2.2 *A typical division of dataset in machine learning. Primarily the dataset is divided into train and test subsets. The train subset is further divided into two disjoint subsets: training and validation subset.*

Validation is the technique used in machine learning to monitor the training phase of the model. Validation dataset is used to check whether the trained model is overfitting or not. It evaluates the generalization error during the training process or after training is done. Validation set allows hyperparameter to update accordingly. It helps to minimize the overfitting and the generalization error. Often some portion of the training dataset is reserved to validate the model as shown in Figure 2.2. Common practice in machine learning is to use 20 percent of the train subset to validate the model and 80 percent for solely training the model. [18]

2.2 Deep Learning

Deep learning is the state-of-the-art technology and considered as the major player in the field of AI, machine learning and big data for its outstanding performance. Deep learning is a branch of machine learning. The traditional machine learning techniques strongly influence deep learning [2]. Deep learning is inspired by the concept of artificial neural network and usually consisting of a large number of neural networks (NN). The NN computing systems are inspired by the structure and function of the biological nervous system of an animal brain. Deep learning framework consists of multiple layers of simple modules, the majority of them are used for learning and many of these compute non-linear input-output mapping. The multi-layer model architecture learns the representation of data with multiple levels of abstraction.

Machine learning is good for the structured low dimensionality data and deep learning is used for unstructured, high dimensional data and perceptual problems

[2]. The limitation of linear model to solve high dimensional complex representation has led to the development of more complicated deep models. Machine learning uses single-level representation learning while deep learning uses combination of multiple processing layers to learn the best features needed to represent the data. The higher level of representation learning makes deep learning capable to solve complex task on high dimensional data such as image recognition, text processing and speech recognition [18].

In past, the challenges for the deep learning models were computing resources and large enough datasets to train the model. Easy accessibility of large datasets, availability of powerful computing devices and outperforming graphical processing unit (GPU) plays a significant role behind the popularity of deep learning methods. Modern deep learning models give impressive performance on computer vision, signal processing, speech/audio recognition and natural language processing tasks. [2]

Neural Network

A neural network (NN) is a highly parallel distributed processor that has a natural tendency of storing experimental knowledge and making it available for use. It is related to the function of an animal brain. The principle behind the NN is that knowledge is obtained by the network through a learning process, interconnection strengths (weights) are used to store the knowledge [21]. Neuron, also known as node or unit is the basic building block of artificial NN. The typical combination of neurons is known as a layer. A neuron receives multiple inputs ($\boldsymbol{x_i}$) from sources connected with it, multiplies each input by the weight of its connection ($\boldsymbol{w_i}$) and sums them together as shown in Figure 2.3. Often a bias is added to this sum. The sum calculated from the weighted connection is then processed via an activation function. The result is normalized and the output ($\boldsymbol{y}$) is produced. NN consisting of a large number of hidden layers is known as a deep neural network (DNN).

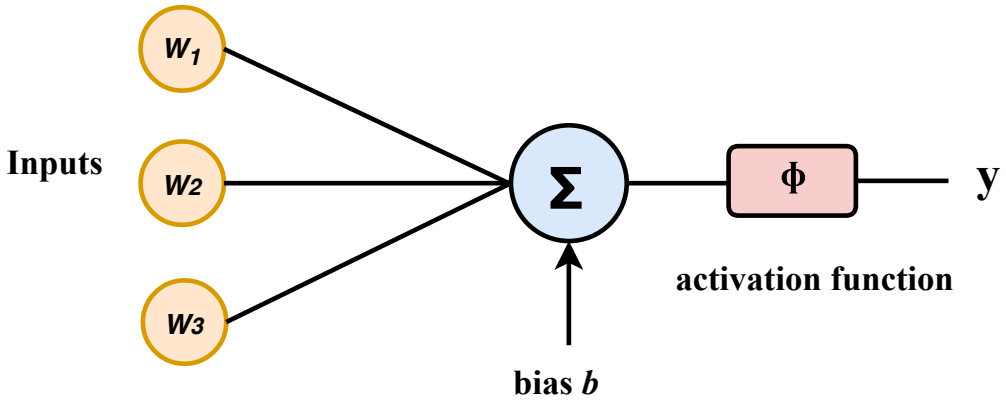

Figure 2.3 *A simple model of artificial neuron.*

Activation

An activation function is used to introduce a non-linearity to the linear activations produced by convolutional (neural) layers and it limits the output of a neuron. Rectified linear unit (ReLU) is used as an activation function in modern CNN architectures. ReLU only passes the positive value and the negative values are mapped to zero as shown in Figure 2.4. ReLU became a popular activation function over sigmoid and hyperbolic tangent (tanh). Meanwhile, the sigmoid function maps the input to values between 0 and 1. Sigmoid function is also known as logistic sigmoid. The tanh function maps input to the value between 1 and -1. ReLU allows fast and effective training of deep neural network on large and complex datasets [17].

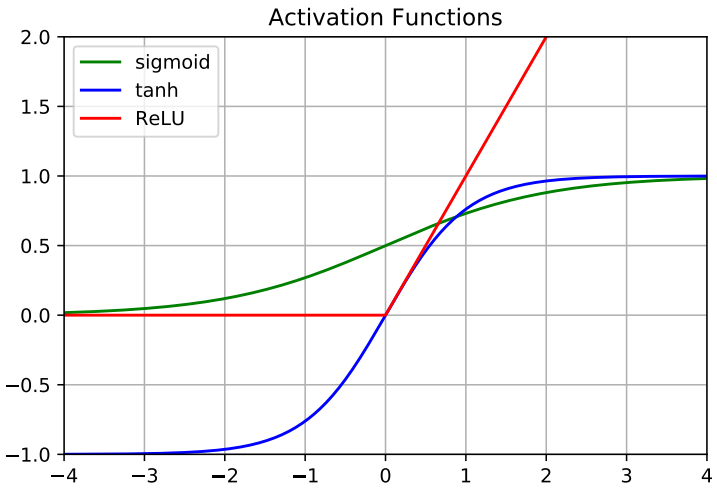

Figure 2.4 *The sigmoid (logistic), hyperbolic tangent (tanh) and rectified linear unit (ReLU) are common activation functions used in Neural Network.*

Subsampling

Subsampling shrinks the dimension of input by an integer factor. Subsampling is also known as pooling and is widely used in deep learning. Pooling layers reduce the dimension and resolution of input while preserving the most important information. Maxpooling, average pooling and L_2-norm pooling are examples of sampling technique used in machine learning. Maxpooling is the most commonly used subsampling technique where the output is computed as maximum value of input. As presented in Figure 2.5, a small window of dimension 2×2 with stride of 2 is sliding across the two dimensions of data and taking the maximum value from the window at each step. Here 4×4 input data is reduced along both width and height producing output of size 2×2. Pooling reduces the data size and improves the spatial invariance to reduce the number of parameters and computation complexity in network. [18]

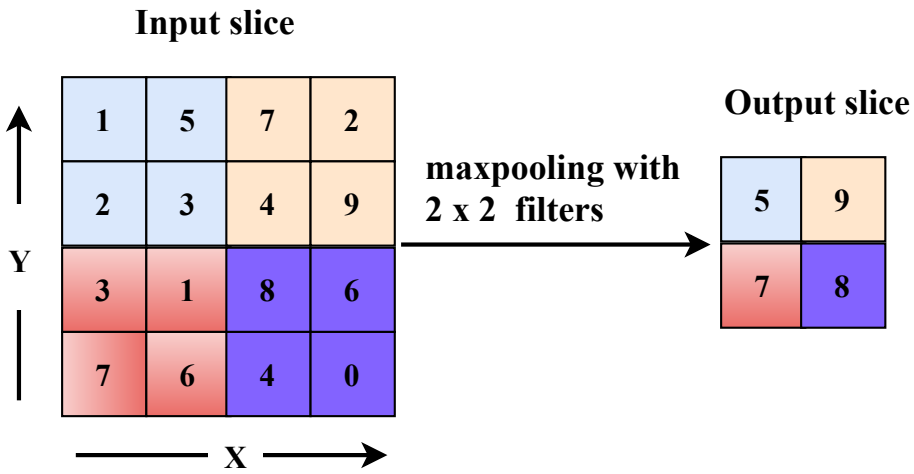

Figure 2.5 *The original image data in X and Y coordinates are down-sampled to half of its original dimension. The 2 × 2 maxpooling window with a stride of 2 is applied to the input slice of 4 × 4 matrix that reduces to 2 × 2 by taking maximum from each window frame. The pooling window size and sliding steps can be changed according to user/application need.*

Fully Connected Layer

The fully connected (dense) layer contains weights associated with every input-output pair. This layer combines inner product of weights and the input from every node of the previous layer and translates them into votes. The multidimensional spatial information received from the previous convolutional layers are converted into single feature vector that will help to predict the class probability. This is the

main block on deciding the class label by counting the vote. Usually one or two fully connected layers are connected to learn more sophisticated features from the network in order to make better prediction result. The flattening layer and dense layer are considered as the fully connected layer preceding the output layer. Flattening layer transforms the received multidimensional features into to one dimensional long feature vector. The dense layer reduces the one-dimensional feature vector size and normally makes same size as the number of class category in dataset.

Backpropagation

In NN, input data is passed via the network and the network produces the output. The error is calculated by comparing the output produced by the network and an actual output. The error is used to update the weights of the neurons in order to gradually decrease the error. Backpropagation algorithm is used to solve this issue in training. The goal of the backpropagation algorithm is to make the training error as small as possible. This is done by iteratively passing batches of data through the network and updating the weights. This mechanism is also known as stochastic gradient descent [18].

Backpropagation learning can be implemented in sequential mode or batch mode. In sequential mode, error adjustments are made to the free parameter of the network on one by one basis. Sequential mode is good for classification problems. In batch mode, adjustments are made to the free parameter of the network on an epoch by epoch basis, epochs consist of an entire set of training samples. Batch mode is good for nonlinear regression. Backpropagation algorithm is easy and efficient to implement. However, it is computationally slow for difficult (heavy) tasks. [21, 18]

Feedforward Neural Network

Feedforward neural networks, also known as deep forward networks or multilayer perceptron, are typical deep learning models. The number of layers in feedforward NN ranges from three to thousands. Feedforward neural networks play a vital role in machine learning/deep learning and have been used in many applications. The convolutional neural networks used for object detection are specialized version of the feedforward neural network. In the feedforward network models information from the input flow through the intermediate computation and then result in the final output. The aim of the model is to approximate some function that maps the input to its output. For example, a classifier, $y = f^*(\boldsymbol{x})$ maps input $\boldsymbol{x}$ to a category

y. The idea here is to design a mapping function $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$ and learn the value of the parameter $\boldsymbol{\theta}$ that approximates the best function. In the feedforward neural network, there are typically many different functions composed together which is known as a layer of the network. Functions are connected in a chain forming a deep model. [18]

Generally, the feedforward neural network consists of a large number of layers. The very first layer is known as an input layer and the final layer is called the output layer, which are shown in Figure 2.6. Layers between the input and the output are known as hidden layers. The information flow in feedforward is always in one direction (forward) **input layer** $\rightarrow$ **hidden layer** $\rightarrow$ **output layer**. In feedforward network the output of the model is never fed back into the network. The feedforward NN which includes feedback connection are known as recurrent neural networks (RNN). Convolutional neural network (CNN) is an example of feedforward neural network.

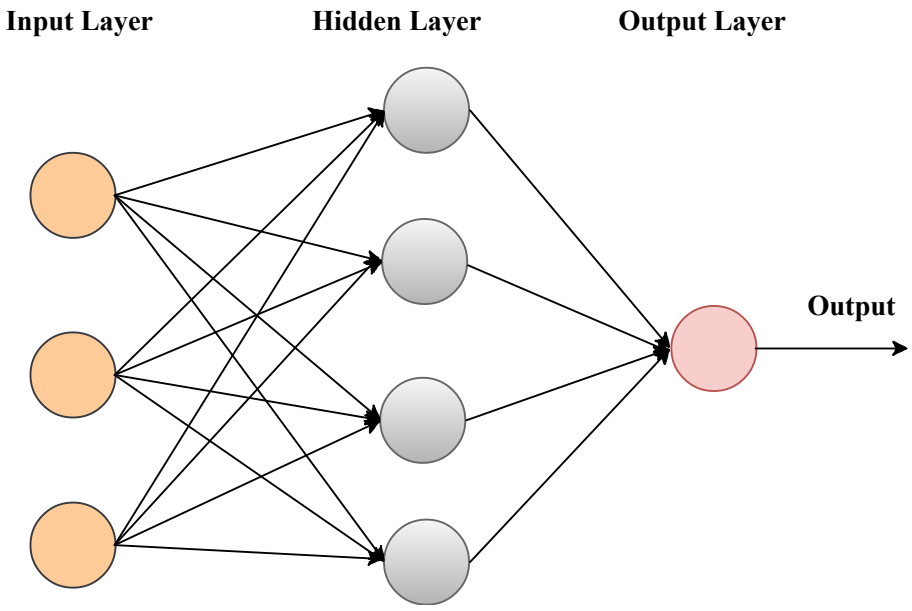

Figure 2.6 *Fully connected feedforward neural network with an input layer, two hidden layers and an output layer.*

The first layer, an input layer is used to prove the input data (features) to the network. The output layer is the final layer in the network that results in the prediction output. The activation function is used in this layer to get the desired output for the problem. Hidden layers are the main block of the model that produces the desired output based on the instruction provided by the learning algorithm. Hidden layer applies various functions to the input. Series of simple functions can

be cascaded to the hidden layers to compute highly complex functions. The number of hidden layers is often termed as the depth of neural network [20].

Convolutional Neural Network

Convolutional neural network (CNN) is an artificial neural network consisting of multiple layers also known as neurons. A typical structure of the CNN contains convolution, nonlinearity, subsampling repeatedly connected with fully connected layers. Modern CNN framework consists of a large number of layers (deep layers) containing convolutional and subsampling layers followed by one or more fully connected layers.

Convolution is a mathematical operation on two functions that produces a third function, which is the integral of the product of the two functions with one of them flipped. Convolution is the main building block of the CNN. It is considered as sliding a filter along its width and height. The convolution filter with desired kernel size is applied in input data resulting in the feature maps. Convolution is done at each point of input without overlap. Convolution filters are widely used in image processing, digital signal processing.

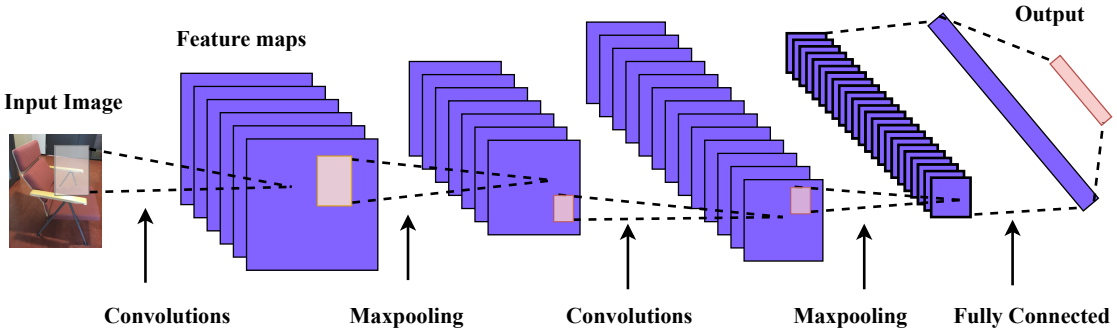

Figure 2.7 *A multi-layer convolutional neural network. An input image is passed through 2 convolution layers followed by maxpooling layers and a fully connected layer and the output layer.*

The input image is put through the series of convolution and pooling operations followed by the fully connected layer to produce the output. The convolution filter of fixed size window is applied to each point of image avoiding overlap. The results obtained from these filters are known as feature maps. Maxpooling is applied on each position of these features maps to reduce the dimensionality. The convolution filter is applied again followed by the maxpooling. These processes can be done repetitively many times. Convolutional layers are typically linear, hence, they might

not be able to express possible nonlinearity [48]. The activation function is applied to the output of the convolutional layer to solve the issue of non-linearity. As shown in Figure 2.7, the second last layer, fully connected layer, also known as the dense, layer is used to stack multidimensional output data into a single list. The output layer gets the information from the fully connected layer about the output class associated with the score (frequency) of each class category. [27]

2.3 Assessment Criteria

In machine learning, accuracies measurements are important in order to know how well the machine has learned and how well it performs on unseen data. There are wide ranges of assessment matrices available to measure learning algorithm performance. The performance assessment of the object detection model is done by checking how well the model recognizes the object and how precisely it localizes that object. The Intersection over Union (IoU) measure, also known as bounding box overlap, is popular among other assessment approaches [6]. The performance measurement of the object detection model is quantitative and explains to us how many objects are detected correctly and how many objects are predicted wrongly (false alarm).

The overall quality of the object detection model is calculated in terms of *IoU*, *precision*, *recall* and *F_1-score*. For a large scale multi-class dataset, the *mean average precision (mAP)* is used to measure the performance of the method in the whole dataset [7]. Apart from the accuracy measure, computation complexity of the model is of major concern. The performance of the trained detector on unseen data, training time of the model and processing speed are major factors that need to check while selecting the object detection model from the list of available models.

In this thesis, we consider an IoU, precision, recall, F_1-score, mAP and frames per second(FPS) as performance assessment criteria for experimented models. IoU, precision, recall, F_1-score and mAP are used to measure the correctness of the trained detector while the FPS is used to measure the inference speed of the detector.

2.3.1 Definition of a Detection

The performance assessment of the object detection that uses the bounding box localization method is based on the calculation of the intersection over union (IoU). An IoU is calculated as the ratio of the area of overlap and area of the union. The area of overlap is the total common area or overlap area (i.e. intersection) between

the predicted bounding box (B_p) and the ground truth bounding box (B_{gt}). The area of the union is the total area covered by both (B_p) and (B_{gt}).

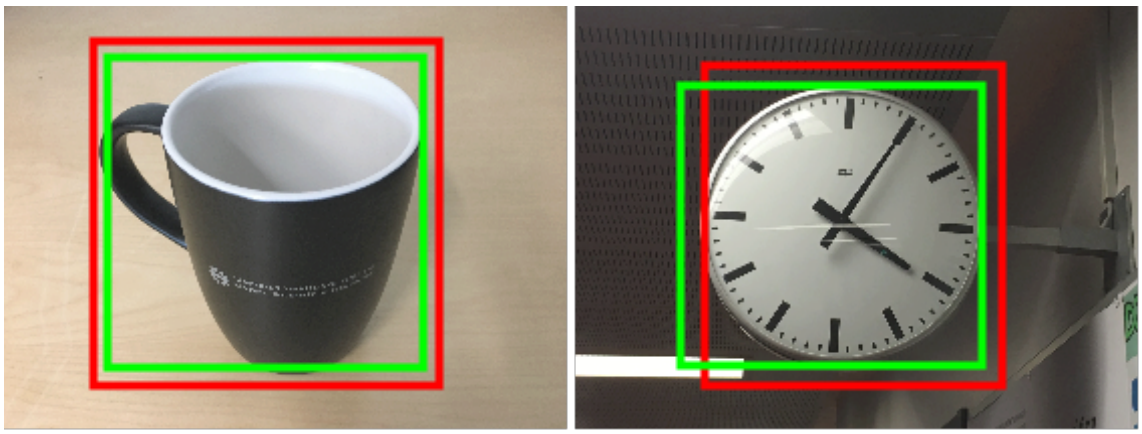

Figure 2.8 *Examples of detection of cup and clock with ground truth bounding box drawn in red and predicted bounding box in green. In the left image, the IoU is 0.82 and in the right image, the IoU is 0.73.*

Figure 2.8 demonstrates the examples of the predicted bounding box and the ground truth bounding box drawn in the cup and clock classes. The B_{gt} drawn in red colour is from the annotation file and the B_p in green colour is predicted from the trained detector. Unlike classification problems, object detection accuracy calculation is rather complex. The exact match of the (X, Y) coordinates of the ground truth box and predicted box is extremely rare [39]. For this reason, the object detection performance assessment metric is defined in such a way that the more the B_p overlap with the B_{gt} the better is the model performance.

An IoU can be calculated when we have the labelled dataset containing ground truth bounding boxes for objects and received prediction for bounding boxes for objects from the trained object detector. The formula to calculate the IoU is given in an equation 2.1. In the numerator, we compute the common area of (B_p) and (B_{gt}) which is the number of pixels covered by both boxes. In the denominator, we compute total area covered by both boxes.

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \in [0, 1] \tag{2.1}$$

An IoU is simply the ratio between these two areas. For the correct detection/prediction, the IoU must be greater than the detection threshold value. In object recognition tasks, 0.5 is used as the most commonly acceptable threshold

value above which it is considered as the correct detection [6].

2.3.2 Accuracy Metrics

In machine learning, precision is the fraction of retrieved items over all items that are present. In object detection case, precision is the sum of the correctly detected object divided by the total population of the object that is detected by the detector. Precision takes all the detected object into account.

$$precision = \frac{(relevant\ \ objects) \cap (retrieved\ \ objects)}{retrieved\ \ objects} \quad \in [0, 1] \qquad (2.2)$$

To understand the terms used in performance measurement, it is wise to consider the binary classification problem. The output of the classifier is positive or negative based on whether it is classified correctly or not. The detection of the single object can be considered as classification task. The clear and concise way to understand the idea behind the binary classification accuracy measure is using the confusion matrix, it might be confusing at the beginning as the name suggest.

Table 2.1 visualizes the confusion matrix also known as contingency table. The true class conditions are the ground truth of the data while predicted class conditions are the prediction made by the classifier. The model prediction result will be one of four outcomes, *true positives* (TP), *false positives* (FP), *true negatives* (TN), and *false negatives* (FN). Each instance holds two parts, the first part is whether the prediction is true or false and the second part is the predicted class (positive or negative). If the true class conditions and predicted class conditions are matched, it is considered as TP. If both the real class and prediction class are negative than the result is considered as TN. If the ground truth of the class is positive but the prediction is negative than this instance is called FP. If the prediction is positive but the ground truth is negative than that instance is called FN.

Table 2.1 *The* 2×2 *confusion matrix.*

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

The diagonal from the upper left to lower right filled with green colour is the correct classification. While another diagonal in red colour is the false classification.

A good classifier must have more instances in the green diagonal than the opposite side, red diagonal. The population of positive class (P) is the sum of all positive instances in data, calculated as, $P = TP+FN$. The population of negative class (N) is the amount of real negative instances in data, $N = TN + FP$. Total population is sum of positive and negative classes, $Total(T) = P+N = TP+FN+FP+TN$. The amount of false prediction is, $(F) = FN + FP$.

Table 2.2 *Calculation of various performance metrics based on the confusion matrix.*

Term	**Calculation**
True Positive Rate (TPR), Sensitivity, Recall	TPR $= \frac{TP}{P} = \frac{TP}{TP+FN}$
True Negative Rate (TNR)	TNR $= \frac{TN}{N} = \frac{TN}{TN+FP}$
Positive Predictive Value (PPV), Precision	PPV $= \frac{TP}{TP+FP}$
Negative Predictive Value (NPV)	NPV $= \frac{TN}{TN+FN}$
Accuracy (ACC)	ACC $= \frac{TP+TN}{P+N}$

Many performance measurement matrices can be calculated based on this contingency table. Table 2.2 contains the list of most commonly used matrices and their calculations. The calculation of TPR, *sensitivity* and *recall* are same. Also, PPV and *precision* calculation are same. An *accuracy* (ACC) is the measure of the fraction of correctly predicted instances over total population of instances. The high ACC value is always good for the classification model but is not sufficient to prove that the model is performing well [46]. Sometime the result might mislead. For better performance assessments, we need to consider more accuracy matrices than a single one.

Recall

Recall is the fraction of the relevant items that are successfully retrieved. In the object detection case, recall is the sum of the correctly detected object divided by the sum of actual objects. Recall take only the truly detected object into account. It is also known as the *sensitivity* of the model.

$$recall = \frac{(relevant\ \ objects) \cap (retrieved\ \ objects)}{relevant\ \ objects} \qquad \in [0, 1] \tag{2.3}$$

F_1-score

Precision focuses on accuracy of the model while recall focuses on the robustness of the model. Therefore, calculating the precision and recall alone is not enough

to measure the performance of the model while calculating both gives the better perception of the model performance. F-score combines both *precision(PPV)* and *recall (sensitivity, TPR)* into the single measurement. F-score is the measure of the accuracy of a classifier also known as F-measure. The formula to F-score based on the confusion matrix is given in the Equation 2.4.

$$F_\beta = (1+\beta^2)\frac{PPV \times TPR}{(\beta^2)PPV + TPR} \quad \beta \in [0, \infty) \tag{2.4}$$

This can be represented as:

$$F_\beta = (1+\beta^2)\frac{precision \times recall}{(\beta^2)precision + recall} \quad \beta \in [0, \infty) \tag{2.5}$$

where $0 \leq \beta \leq +\infty \quad and \quad 0 \leq F_\beta \leq 1$.

For $\beta < 1$, F-score is more *precision* oriented while for $\beta > 1$, it is more oriented towards *recall*. For $\beta = 1$, *precision* and *recall* are weighted equally, this case is known as F_1-score and given by Equation 2.6. The value of β manages significance of recall over precision. Different value of β can be used based on which metric is important and by what amount. For example, if recall is less important than precision, $\beta < 1$ is used and if recall is more important than precision, $\beta > 1$ is used in the Equation 2.5 [46].

$$F_1 = 2\frac{precision \times recall}{precision + recall} \quad \in [0, 1] \tag{2.6}$$

F_1-score is the weighted average of the precision and recall. Equation 2.6 can be represented as the form of harmonic mean. The value of F_1 is bounded between 0 and 1. This indicates that the F_1-score is higher only when both precision and recall values are higher. In multi-class case, it is always worthwhile to calculate the F_1-score as the harmonic mean of average precision and average recall than calculating the mean of individual F_1-scores [8].

2.3.3 Mean Average Precision

The mean average precision (mAP) has been widely used to compare the overall accuracies of object detections models on multi-class dataset, since its first introduction in Pascal VOC challenge 2007 [7]. The mAP is a single metric that gives the idea about the object detection model performance on the whole dataset. The mAP is not an absolute accuracy metric but it serves as the relative metric. At first, the

average precision (AP) of each class based on different recall values are calculated and then mean of these AP are computed to get mAP of the whole dataset. Different techniques are used to calculate the AP. In Pascal VOC, the precision values over 11 values of recall (0.0,0.1 ...,1.0) is used. While in COCO, precision values are computed over 101 different recall values. The calculation steps for mAP based on Pascal VOC is as follow :

1. At first precision and recall are calculated based on the IoU. An IoU greater than the threshold value (usually, 0.5) is considered as true detection.

2. AP is calculated by averaging the maximum precision value at different levels of recall. This is normally taken from the precision-recall curve. The precision values are collected from different recall (11) values ranging from 0.0 , 0.1, ...,1.0. Equations 2.7 - 2.9 show the AP calculation in Pascal VOC [25]. In the COCO dataset, AP is calculated from the precision-recall curve drawn using 101 equally spaced recall values.

$$AP = \frac{1}{11} \times (AP_r(0.0) + AP_r(0.1) + ... + AP_r(1.0)) \tag{2.7}$$

$$AP = \frac{1}{11} \sum_{r \in (0.0,0.1,...,1.0)} AP_r$$

$$AP = \frac{1}{11} \sum_{r \in (0.0,0.1,...,1.0)} p_{interp}(r) \tag{2.8}$$

where,

$$p_{interp}(r) = \max_{r' \geq r} p(r')$$

The $\boldsymbol{p_{interp}(r)}$ is the optimal precision for recall values above $\boldsymbol{r}$.

The simplified equation for AP calculation is:

$$AP = \frac{1}{11} \times (p_{interp}(0) + p_{interp}(0.1) + .. + p_{interp}(1.0)) \tag{2.9}$$

3. Calculate the AP for all object classes present in the dataset.

4. Finally, calculate the mean average precision (mAP) by taking the mean of AP over all the object classes.

The mAP value gives the overall performance of the detector over the dataset, while the AP tells about the performance on a single class. It is obvious that some

classes have higher AP and some classes have moderately low AP. This is due to the number of training samples and (or) quality of samples for each class. The AP result provides the information about whether we need to add/change the training samples in some class or not.

2.3.4 Detection Speed

Computers integrated with high-end GPUs process graphical contents faster than basic computers that only have CPUs or less powered graphics card [32]. Frames per second (FPS) is the frequency of consecutive images, also known as frames, appearing on display. FPS is the widely used metric in broadcasting, film, image and video processing, game, and computer graphics. Computation complexity of the model is one of the main concerns in real-time object detection. There is no clear answer to how much inference speed is considered real-time detection. However, a model is said to be real-time if, its output is faster than or as fast as to the input [32]. The higher the FPS, the better the model performance, but a value less than 1 FPS is considered to be a slow performing model. The inference speed of 1 FPS means every second a single frame is displayed and video of 1 minute contains 60 consecutive frames.

The processing time calculation gives the general information about the computation complexity of the algorithm. Deep learning frameworks are complex in nature and computationally heavy. Hence, it is hard to achieve real-time inference on less powerful devices. We considered the FPS as the measure of the computational complexity of the model. As we are running all models in the same working environment, the FPS results give better understanding of computation cost of experimented models. We compare the computation complexity of experimented object detection detectors during training and evaluation on the test dataset.

3. OBJECT DETECTION

In this chapter, we discuss the concept of object detection and its applications in modern computer vision. We discuss the convolutional neural networks based object detection architectures and their challenges. Object detection meta-architectures and feature extractors that are used in this thesis project are discussed here.

Background

In modern computer vision, object detection is considered together with the object recognition (classification), localization, tracking, and extracting information from the object. These processes are tightly related to the object detection. In classification, the aim is to find out the object class or recognize what the object is. Localization finds the location of the object/s in the image or localizes the object inside the frame. Object tracking in video or live recording is a way to get the movement and status of the object. The goal of object detection system is to classify and find the location of all objects that are present in an image. The input for the detector is an image containing the object and the output is a list of the bounding box. Figure 3.1 visualizes the tasks related to object detection. [40]

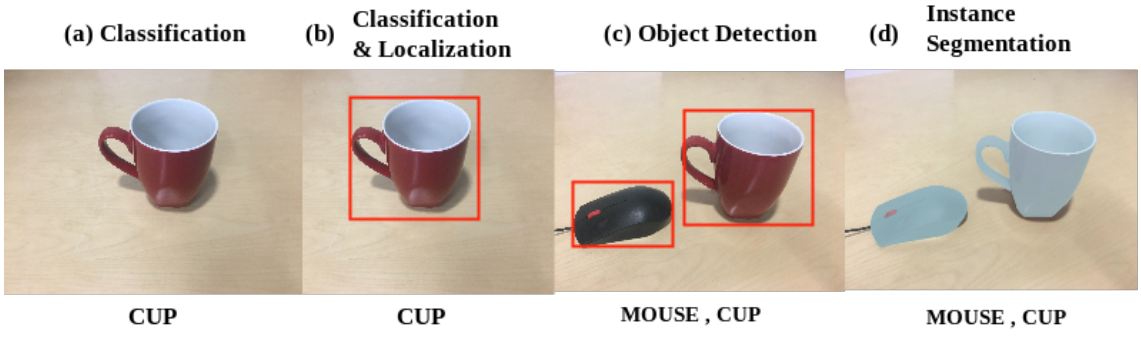

Figure 3.1 *The collection of computer vision tasks related to an object, (a) the classification (class label) of object in an image, (b) class label of the object and its location in an image, (c) class and location of each object in an image, and (d) the precise pixels of each object in an image.*

Object detection was based on extracting the feature descriptors from the images before the deep learning approaches became prevalent. HOG and scale-invariant feature transforms (SIFT) [33] with support vector machines (SVMs) classification were used for automatic object detection. Deep CNN based models outperform the classical object detection models and are considered as the best performing methods [12]. Faster region-based convolutional neural network(RCNN) [37], region-based fully convolutional network (RFCN) [5] and single shot detector (SSD) [31] have been used as major meta-architectures in object detection models. These meta-architectures consist of the deep CNN together with the feature extractors. Inception [43], MobileNet [23], NAS [47], ResNet [22] and VGG [42] are popular feature extractors that can be implemented in above mentioned meta-architectures depending on application type and its use in different case.

Object detection is the fast-growing field of research in machine learning and computer vision. The number of problems that can be solved using object detection algorithms is increasing rapidly. Camera-based real-time object detection is on top priority for face detection, object counting, visual/image search, landmark recognition, satellite image analysis, autonomous driving, drone and agriculture productions [9]. Detectron [13], Dlib [28], RetinaNet [29], TensorFlow [44] and You only look once (YOLO) [36] are popular platforms that are used for object detection. Deep convolutional network frameworks are implemented in all modern object detection platforms/libraries.

Challenges

Object detection is known to be a challenging task in computer vision. Large number of training datasets with big number of images from each class category is needed for better learning and generalization. The collection of a large dataset with varieties of objects, scalability of data, the computational complexity of the deep model and the robustness of detector are the main challenges of object detection [35].

Robustness

Robustness refers to the challenges of the detector in detecting the object of different appearances. The intra-class and inter-class difference are big challenges for automatic object detection [35]. The intra-class difference is the difference between the same class objects, while the inter-class difference is the difference between dissimilar class objects. The variation in objects and variation in images are challenging

too. As shown in Figure 3.2, a chair can be of different colour, shape, size and texture. The environments where the images of the chairs are captured are different (background, lighting, and view-angle). The similar appearing object of the different class category is another challenge. For example, the inter-class differences of moving chair, sofa chair, armchair and four leg chair are quite small and often considered as common class chairs.

Figure 3.2 *Examples of the chair class in our dataset. These are 8 different types of chairs representing single class label (chair). They are captured in different backgrounds, lightings and angles.*

Large datasets

The detection model must be complex enough to solve the robustness problem. This necessity motivates to have large dataset and deeper models. The necessity of large-scale datasets to train and test the object detector has been solved by available benchmark datasets. However, these datasets were collected on the specific environment which might not work well in another environment. To make the environment specific object detector, we need to collect a large number of samples from the specific environment for each class category object.

Scalability and Complexity

Another challenge in object detection model is the scalability of high dimensional data. High dimensional data is hard to represent, difficult to train the model and learn the required information. The feature extractors help to solve these issues by extracting only the required features from the data. Most of the modern object detection models use the regional CNN filters to extract the object maps from the input image. CNN architectures are computationally heavy to perform. The hardware needed for heavy computation is another challenge in object detection. Easy availability of powerful computing devices, high-end GPUs, parallel computing and cloud-based computing services help to solve this challenge.

3.1 Faster Region-based Convolutional Neural Network

A Region-based convolutional neural network (CNN) has three series RCNN [14], fast RCNN [14] and faster RCNN. Faster RCNN is the latest architecture in the RCNN series and one of the widely used architecture in object detection [37]. Faster RCNN is composed of fully deep convolutional networks that propose regions and the fast RCNN detector, that uses the proposed regions to detect the object. To understand the concept of faster RCNN it is wise to go through the principle behind the RCNN. RCNN is a three-stage process. At first, a selective search algorithm is used to scan the objects in input image known as region proposal. The convolutional neural network is run on top of each of these region proposals. The third stage is to feed CNN output into support vector machine (SVM) to classify the object region and a linear regression is used to tightened the bounding box of the object. [37]

In faster RCNN, slow selective search algorithm implemented in RCNN is replaced with a fast-neural network. The detection in faster RCNN is a two-stage process. At first, a convolutional filter is run through the entire image resulting in the feature maps. The region proposal network (RPN) is applied on those featured maps and output the set of object proposal boxes with scores. Image feature maps achieved from the convolutional filter are used to predict the object proposal with scores. Object proposal network is proposing boxes based on the intersection over union (IoU) between the purposed object and the ground-truth object. In the next stage, object proposal extracts the area of the object from the feature maps. This is done by region of interest (RoI) pooling layer. The extracted features are applied to all layers of feature extractors to get the prediction probability of the class and the bounding box for each region proposal. The final stage is to classify the object and its location (bounding box) found in the image. As shown in Figure 3.3, entire

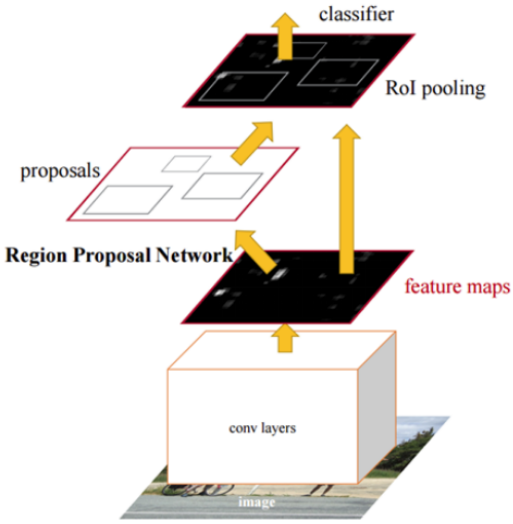

Figure 3.3 *Single unified network for object detection implemented in faster region-based convolutional network model [38].*

operation occurs on the unified single network, that enables the system to apply convolutional filters together with the detection network. It is proven that faster RCNN meta-architecture with a good feature extractor has excellent performance on complicated object recognition and classification challenges [9].

3.2 Region-based Fully Convolutional Network

The region-based fully convolutional network (RFCN) uses two-stage object detection strategy consisting of regional proposal and region classification. RFCN is identical to faster RCNN but instead of cropping features from the predicted regional proposals, features are taken from the last layer of features preceding the prediction. This technique helps to minimize the quantity of memory applied in region computation. It was mentioned that RFCN together with the ResNet101 feature extractor has competitive performance compared to faster RCNN [9, 5].

The architecture of RFCN is shown in Figure 3.4. At first, the convolutional filter is applied over the input image. The fully convolutional layer is added to generate a score of positive-sensitive score maps (feature maps). The fully convolutional regional proposal network (RPN) generates the regions of interest (RoI) from the previous layer output (feature maps). Each generated RoI is divided into subregions (bins) as scored maps. The score bank of each bin is compared with the corresponding position of the object. The process is repeated for all the bins and for all the classes presented in the image. When these bins have an object matched with

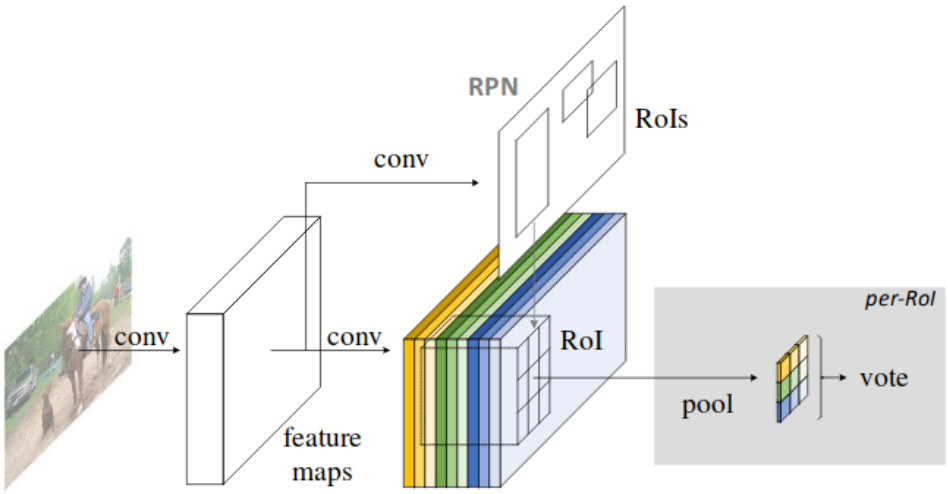

Figure 3.4 *Regional based fully convolutional network meta-architecture. Input image is gone through the convolution and RPN proposes the possible regions of objects. RoI pooling is applied to RPN outputs and finally classifier classify the object [5].*

enough of sub-region of the object, the average score is calculated per class. At the final stage, softmax pooling is used to classify the object class based on the previous vote from the RoI layer. RFCN is fully convolutional and shares the computation throughout the networks which make this model faster than the faster RCNN model [5].

3.3 Single Shot MultiBox Detector

A single shot multi-box detector (SSD) meta-architecture solves the object detection problem by using a single pass of a feedforward convolutional network. Unlike other models, this does not generate region proposals nor do resampling of image segments thus saving computational time [9, 31]. This network handles objects of different sizes by using features maps from different convolutional layers as input to the classifier. This network produces a large number of bounding boxes with the scores of object class in those boxes. Non-maximum suppression is used to eliminate boxes below a certain threshold so that only the boxes with higher confidence values proceed for classification. SSD meta-architecture allows end-to-end training and improving the speed of the detector. This meta-architecture does everything in one shot, thus, it is faster than other meta-architectures but it lags the detection accuracy.

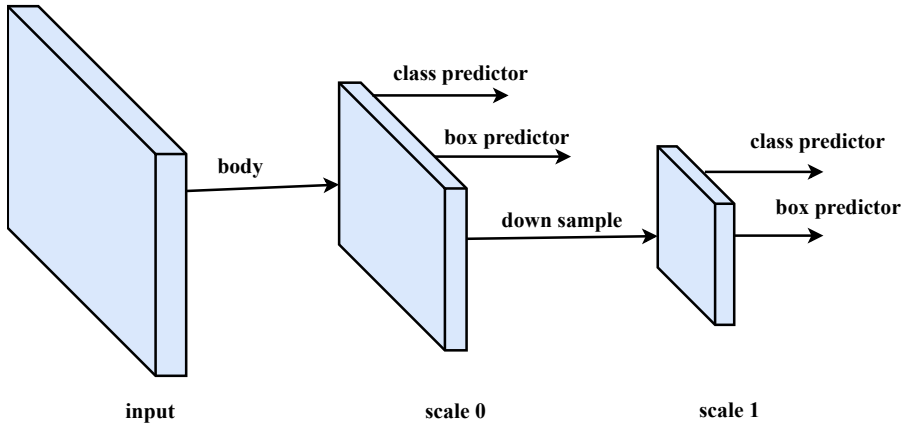

Figure 3.5 *SSD layer is series of small convolutional layer that is added on top of base layer [15].*

The SSD layer architecture is built on top of a feedforward CNN that results a fixed-size collection of bounding boxes and object class instances present in those boxes. The input image is passed through a series of convolutional layers and down-sampled via the SSD layer shown in Figure 3.5. This SSD layer is linked to the output of the last convolutional layer of the base model. Multiple sets of feature maps at different scales are achieved from the convolutional layers with the prediction of object classes (from class predictor) and set of bounding boxes (from box predictor). The predicted boxes are compared with the ground truth of the object and the best one with higher IoU is selected together with the higher probability score from class predictor.

3.4 Feature Extractors

The feature extractor is the major building block of the object detection model that is used to extract the features of objects from the data. The object detection model structure is composed of detection meta-architecture, feature extractor and classifier as shown in Figure 3.6. The input image is passed through the feature extractor that extracts features from the image. The extracted features are then forwarded to the classifier that classifies the class and the location of the object in the input image.

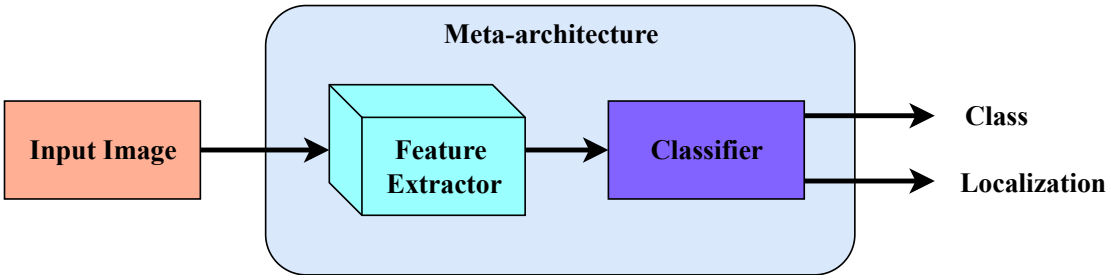

Figure 3.6 *Object detection model architecture is composed of the feature extractor and classifier in meta-data architecture [9].*

The feature extractor is a deep architecture that aims to increase the accuracy while reducing the computational complexity. **AlexNet, Inception, MobileNet, NAS, ResNet and VGG** are some popular feature extractors that can be implemented in object detection meta-structures. We used the Inception, MobileNet, ResNet and NAS feature extractors implemented in above-mentioned meta-architecture.

Inception

The inception module works as multiple convolution filters that are applied to the same input together with pooling and concatenation to get the result. Inception architecture allows the model to gain advantage from the multi-level feature extraction.

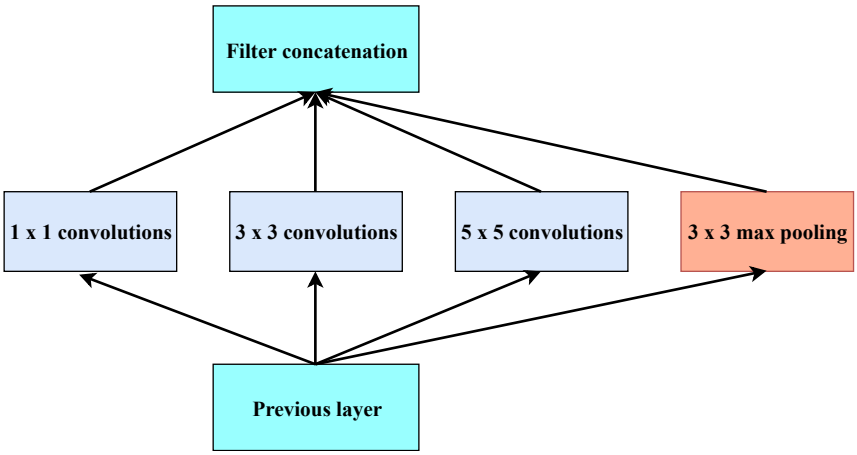

Figure 3.7 *Inception module, naive version [43].*

Inception module uses a combination of compositions of different convolution filters. At first, the 1×1 convolution is followed by the various size of convolution

filters (3×3 convolutions and 5×5 convolutions) and maxpooling operation. The output from these filters and pooling are concatenated to get the final result as shown in Figure 3.7. Inception network is the combination of numbers of this inception module. It is believed that getting multiple features from multiple filters improves the performance of the network.

Mobile Network

Mobile network (MobileNet) is a lightweight deep neural network that is efficient for mobile and embedded devices. The principle behind this architecture is the division of the standard convolutional filter into two convolution filters, depthwise convolution and pointwise convolution (1×1 convolution). The computation complexity of the standard convolutional filter is higher than the combined computation complexity of depthwise and pointwise convolutions.

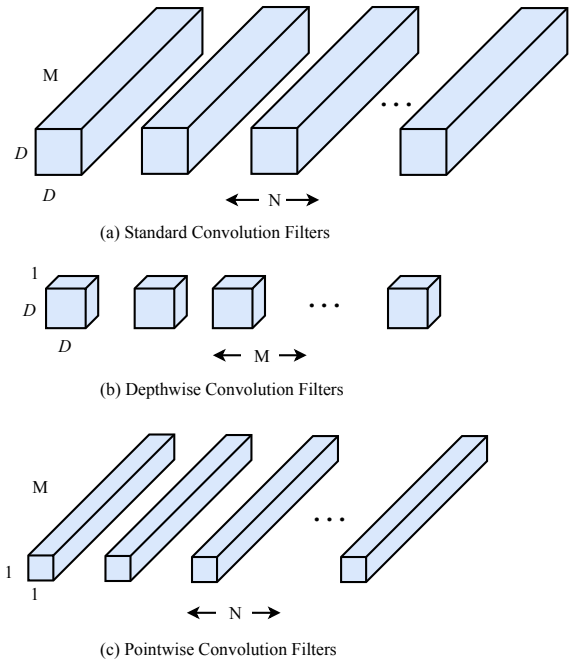

Figure 3.8 *MobileNet feature extractor is based on the separation of the standard convolutional filters into depthwise convolution and pointwise convolution [23].*

The computation cost of the convolution depends on the input network (M), size of the output network (N), feature map size ($D_F \times D_F$) and the kernel size ($D_K \times D_K$). The computation complexity of the standard convolutional filter shown in Figure 3.8 (a) is higher than the total computation cost of the depthwise (b) and the pointwise (c) convolution filters. This division is optimized for the computation

speed. The reduction in accuracy is rather small in comparison to the to the standard one. These are the parameters for balancing between speed and accuracy. [37]

Residual Network

Residual Network (ResNet) is based on the residual learning principle. The idea is to learn a residual instead of the features. The deep residual network consists of 152 residual blocks [22]. ResNet is the record breaker a single architecture for classification, detection, and localization tasks. It surpasses a human on ILSVRC 2015, ImageNet and COCO 2015 competitions with an incredible accuracy and is known as the best CNN architecture [26]. The increase in depth of the deep NN increases the accuracies until some point and after that, it starts decreasing. Residual learning tries to solve this issue of accuracy degradation in NN [22].

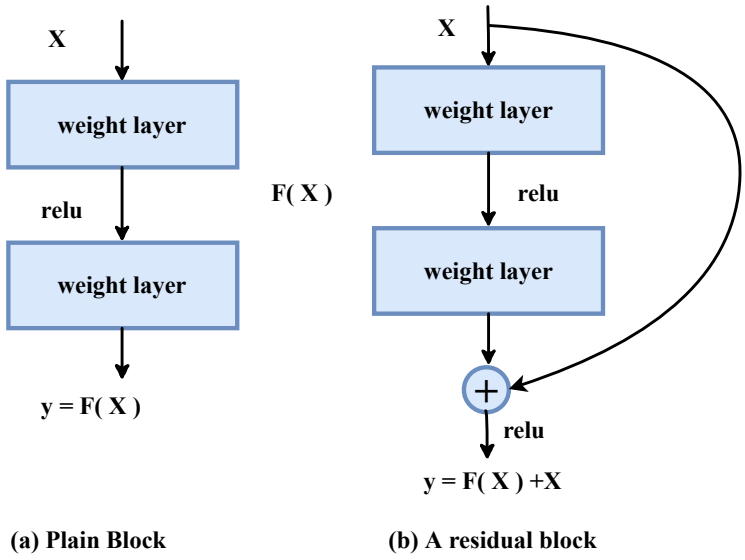

Figure 3.9 *F(X) is the residual function of non-linear CNN layers. In the plain block (a), it is difficult to get identity mapping by pushing the residual function to zero. It is easier to get identity mapping in the residual block (b) than in the plain block (a) [26].*

The idea behind the ResNet architecture is to use a residual function instead of direct mapping of input-output. The residual function is $F(x) = H(x) - x$ where $F(x)$ is the stacked non-linear layers, $H(x)$ is mapping function and x is identity function. This residual function can be re-framed to $H(x) = F(x) + x$. It is easier to get the residuals to zero than to fit an identity (input = output) mapping using stacks of non-linear CNN layers as the function.

4. IMPLEMENTATION

This chapter presents information about data collection, annotation and preprocessing needed to train the object detection models. The environmental setup needed to train and test the object detection models from TensorFlow object detection model zoo is discussed here. Data collection, annotation, preprocessing and the formation of the train-test dataset are of major concern in object detection.

4.1 Dataset

Data is the core necessity to train, test and validate any kind of machine learning task. Especially in supervised machine learning, we need labelled data that will determine what our algorithm will learn and predict as a result. The training data is the major factor that influences the model behaviour and performance on unseen data. The data processing stage includes the collection of the datasets, the annotation and formation of the train and test subsets. This section describes the methods used for data collection, annotation and preprocessing to achieve our experimental goal.

4.1.1 Data Collection

The data used in this project were collected during this project. The first stage of data collection was recording videos in the university premises including corridors, labs, meeting rooms and offices. From the series of recorded videos, interesting frames were extracted. The deep learning implementation needs way more training data than the basic machine learning algorithms. During the collection of our object detection dataset, we faced some challenges. One of the major challenges was the collection of common objects suitable for the dataset. The quality and clarity of images and objects was an issue itself. This challenge was solved by using the better quality video recorder. The collection of varied sets of images of the same object was another challenging task during data collection. Manually annotating all objects and their instances with the corresponding class labels was another challenge.

We extracted 1100 frames from the series of recorded videos inside TUT premises. In each captured frame there were one or many objects of single/multi-class instance. In addition to self-collected data, 40 images of remote controls and 25 images of exit signs were downloaded from Google image search.

Figure 4.1 *Sample images with different objects from our dataset. These were captured in various indoor premises at TUT.*

Table 4.1 *List of the objects in our dataset based on the class category.*

Category	**Object Instance**						
Natural	Human	Plant	Flower				
Furniture	Box	Boxes	Chair	Door	Drawer	Sofa	Table
Electronic	Clock	Printer	Remote	Screen	Socket	Switch	
Other	Bike	Board	Cup	Exit	Fire ext.	Picture	Trash bin

Our dataset contains 23 classes of interesting objects found at TUT indoor premises such as human, small plants, furnitures, safety symbols (fire extinguisher and exit) and electronics goods. Figure 4.1 presents some representative images from our dataset. Table 4.1 lists class label of the objects in our dataset. These images were taken from indoor TUT premises on diverse backgrounds, lighting conditions, sizes and viewpoints.

4.1.2 Data Annotation

Data annotation is the process of labelling the data for supervised machine learning method. In case of object detection, an annotation is a process of localizing the object inside the given frame and labelling it. Bounding box approach and the pixel-wise object segmentation are two methods used to annotate the object on the image frame. We used the bounding box annotation approach where the bounding box is drawn locating the object, a rectangular box is drawn around the object boundary making object inside the drawn box. Dlib image annotation tools Imglab [1] was used to annotate the images. The information of the images and corresponding boxes were stored in a file and saved in extensible markup language (XML) `.xml` format. The XML file format created using the Imglab tool contains the image file with the information of rectangular box around the object and the corresponding class label of the object/s.

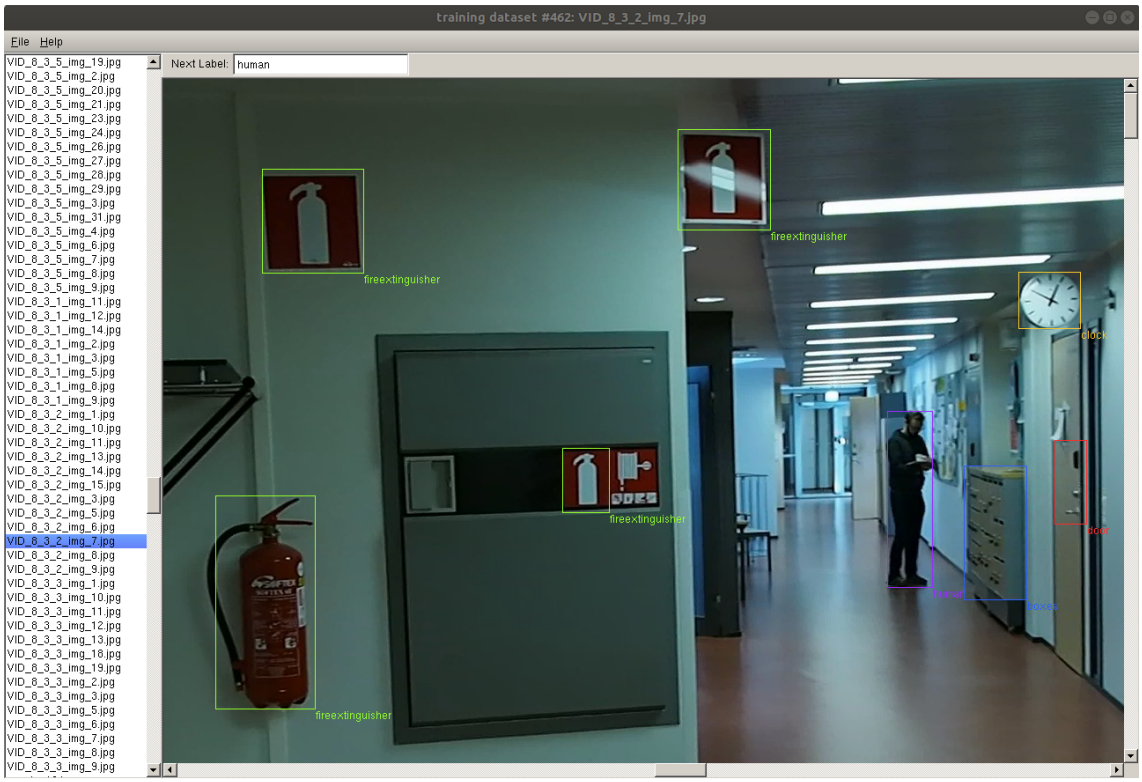

Figure 4.2 *Screen capture of Imglab graphical user interface(GUI) during multi-class object annotation.*

The screen capture of Imglab graphical user interface (GUI) is shown in Figure

[1] `https://github.com/davisking/dlib/tree/master/tools/imglab`

4.2. After we open the Imglab with the images, we can start annotating the desired objects. At first, we need to give the class label and can start drawing the box around the object. The rectangular bounding box is drawn around the object using the shift and the left key of the mouse starting from the top-left, following height and width of the object. When annotation process is done, it can be saved to the file using the save option under the file menu. The location of drawn ground-truth bounding box around the object is stored in the file and saved in the XML file format. The annotated XML file contains top-left corner of bounding box with height and width of the box and object class label. The XML file store the information of ground truth bounding box on top of the object in an image frame. Once the annotation is done, it can be changed or modify using the same technique.

The annotated XML file can be opened in the browser to visualize the object with recently drawn ground truth bounding box. This is a handy way to figure out whether there was a mistake during the annotation or not. More specifically when the annotation is done in an automated way, viewing the generated annotation file on the browser is a promising idea to observe the performance and check whether there is a need to modify the annotation or not.

Data Augmentation

The amount of labelled data is the main factor that affects behaviour of the trained model on new data. A large amount of annotated object samples are needed to train the deep and robust object detection model. Data augmentation helps to overcome the challenge of collecting a large amount of labelled dataset. Data augmentation is a technique to produce new data by transforming and manipulating the original data. It is widely used in image processing. The newly created data must hold the same class label as the original data. The wide range of data augmentation techniques are used for image data, scaling (resizing the images), flipping (horizontal/vertical flip), rotation, adding noise (salt/pepper noise), and transformation. [34]

As the majority of the objects in our dataset are symmetric in nature, we used a flipping technique to increase the size of labelled dataset. Imglab flip function was used for the data augmentation. The annotated images and objects were flipped horizontal left-right preserving the label of the object. We found that including flipped data on training improved the performance of the trained object detector.

Automatic Annotation

Data annotation for object detection is time and resource consuming. We need to collect and annotate thousands of images with many objects to make our detector model robust. It is wise to automate the process of annotation to reduce time and complexity constraints for data labelling. We tested an automated way to annotate the multi-class dataset. After we train the model on the manually annotated dataset and get the trained model graph, we can use it to automatically generate the annotation (XML) file from new images. Once the automatic annotation file is created, we need to check the annotation and correct wrong annotation box/object and label manually. The performance of the automatic annotation heavily relies on the trained detector, the dataset used to train the model and the data that need to annotate. The result will be good if the trained detector is well performing and input images are of good quality with clear objects.

4.1.3 Preprocessing

After annotating all the frames, we have annotated datasets ready for our experiments. The next step is to split the datasets into train and test/evaluation subsets. In the experiment, we randomly divide our dataset into 80-20 split. The relation between the training data and the performance of the detector is explained in the result section. Train-test split can be done in multiple ways. However, it is better for prediction if the distribution of classes on both subsets are balanced enough. After splitting the dataset into two subsets we need to change the annotated XML files to the file format that is supported by TensorFlow.

TensorFlow uses its native file format TFrecord (*.record*) to do the batch operation. Unlike other platforms which do most of the batch process directly from the images, TensorFlow uses a single file for the batch operation. In TFrecord file, images are converted into Numpy array. This format for processing the large dataset helps to mix-match dataset and network architecture as well as process the large dataset that does not fit into the memory. This format is the record-oriented binary format that is used for training and testing data in many TensorFlow applications. There are different options available for data preprocessing. The training set and testing split from the annotated dataset can be done before converting it to TFrecord (splitting XML file) or after using TFrecord function. [45]

The training and testing XML files are needed to be in TFrecord file format. The conversion from the Imglab XML format to the TFrecord format was done using

the TFrecord reading and writing principle shown in Figure 4.3. As shown in the figure, the first step is to convert the image and labels into the proper data type. Then, it is further converted into feature sets. The feature sets are converted into serialized data and finally written to TFrecord file format using TFRecordWriter function. The reading process, or extracting the data from the TFrecord file is the reverse way back to data.

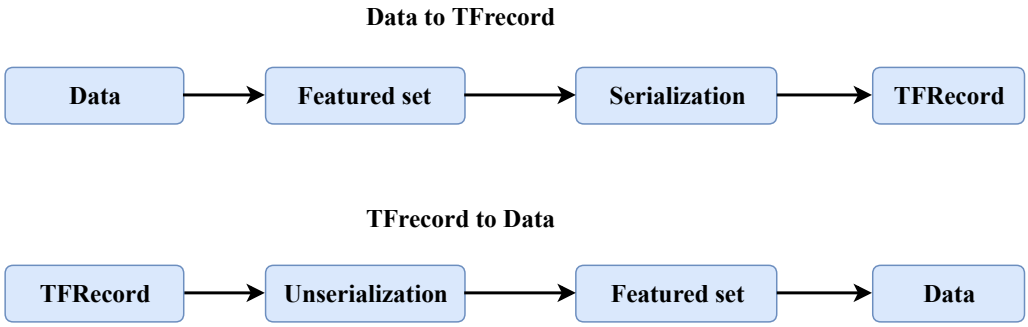

Figure 4.3 *The principle behind the data conversion to TFrecord file format and retrieving the information from the TFrecord file.*

Imglab XML format files were converted into TFrecord file format using the conversion script written in Python. TensorFlow uses the google protobuf text format (*.pbtxt*) to store the information of class label information.

4.2 TensorFlow Object Detection

TensorFlow is an open-source machine learning framework released in 2015 by Google [44]. Google Brain team originally developed it to conduct the machine learning and deep learning research and implemented in wide variety of Google applications such as speech recognition, search, Gmail, and photos. The library is implemented in C++ and has good Python bindings. It became the most popular platform among deep learning practitioner and researcher soon after it was released to the public. [44, 41]

In TensorFlow library the numerical computations are done with information stream (data flow) graphs. In these graphs, nodes mean mathematical operations and the edges mean the multidimensional data array also known as tensors that are communicated between edges. TensorFlow has its interactive visualization environment known as TensorBoard. This tool visualizes a flowchart of the data transformations, visualize summary logs over time, traces the performance of algorithm and computation graph before and during runtime. The interesting thing in TensorFlow

library is that computations are expressed as flowcharts, separating design from implementation. The same design can be implemented on powerful computing devices with large numbers of processors and on mobile devices. TensorFlow provides a wide range of functions and classes that allow users to build the desired models from the scratch. [44, 41]

TensorFlow Object Detection API

The TensorFlow object detection API is deep learning framework built on top of TensorFlow used for object detection. It is popular among the researcher community because of its easiness to build, train and deploy. There are various models of pretrained object detection models trained on the COCO (Common Objects in Context) dataset, Kitti datasets and open-image datasets. The COCO dataset consists of 328,000 images of 90 common objects classes with a total of 2.5 million labelled instances [30]. Kitti dataset is collected from roads in Germany, consisting of different classes on road environment [10].

TensorFlow model zoo provides numbers of pretrained models trained on COCO dataset and few models trained on other datasets. Among the available pretrained models, we experimented eight different models on the self-collected dataset. Table 4.2 contains models that were experimented from the list of available models. In the model zoo, the pretrained models are listed with their speed of execution, accuracy, and the type of output. We used the default configuration and pipeline provided in the model zoo and fine-tuned our training using provided pretrained models.

Table 4.2 *List of models that were downloaded from the TensorFlow object detection model zoo and experimented on our dataset. We found that these six models were the most interesting.*

Model Name	Speed(ms)	COCO mAP	Output
ssd_mobilenet_v1	30	21	Boxes
faster_rcnn_inception_v2	58	28	Boxes
faster_rcnn_resnet50	89	30	Boxes
faster_rcnn_resnet101	106	32	Boxes
rfcn_resnet101	92	30	Boxes
faster_rcnn_nas	1833	43	Boxes

The execution speed is measured in milliseconds(ms) in and the accuracies are measured in mean average precision (mAP). As we annotated the data using bounding box approach, our interest was on boxes output models rather than the masks

output. At the time of this experiment, there were 17 different pretrained models trained on COCO dataset, one pretrained model trained on Kitti dataset and two models trained on open-image dataset [4]. In addition to models that are shown in Table 4.2, we fine-tuned and evaluated *faster_rcnn_inception_resnet_v2* and *ssd_inception_v2* models from the model zoo.

4.3 Environment Requirements

The system requirements to work with object detection models may vary depending on the operating system and version of the libraries. Basic requirements to train and deploy the TensorFlow object detection API are described in this section. Following is the list of common libraries and tools that are necessary for the object detection using TensorFlow object detection API.

- Python, version 3.5 or higher is preferred
- Open Computer Vision (OpenCV) libraries
- Imglab annotation tools
- CUDA library
- CUDA Deep Neural Network (cuDNN) libraries
- TensorFlow Object Detection API libraries
- Pretrained object detection models from the model zoo
- Good GPU/computing source to train deep learning models

All of our experiments were done on Lenovo ThinkStation with an Intel Xeon CPU E5 2.10 GHz, 32 GBs of RAM and Ubuntu 17.10 operating system. Nvidia GeForce GTX 1080 Ti 11 GBs was used for training and evaluation. This GPU is considered as best performing GPU at the time of our experiment. More importantly, TensorFlow GPU version 1.6 was installed on the working machine and object detection API library was configured on top of it. Note that the FPS presented in this thesis is single frame processing speed of detector computed during the evaluation process.

5. EVALUATIONS

In this chapter, we discuss the results of our experiment based on precision, recall, F_1-score, computational speed and mean average precision. We test the relation between the number of training samples and the performance of the trained detector. In addition to automatic accuracies calculation, we present visual observation of fine-tuned detector performance and a discussion of our findings.

5.1 Results

We discuss the findings of our experiments in this section. The accuracy and computation complexity measure of trained and tested object detection models are compared. Automatic calculation of the assessment matrices of experimented models on single and multi-class cases are discussed.

To evaluate the performance of our detectors, we used three disjoint subsets from our dataset. TUT test dataset randomly selected from the dataset, a single class dataset containing cup images (cup test dataset) and small test dataset that is used only for testing purpose. In cup test dataset, we have 72 images containing 104 cup instances. The small test dataset contains 86 images of 210 multi-class instances. There are 367 images and 1210 object instances in TUT test dataset. The TUT test dataset might differ slightly as each random split produces different scenarios.

Single-class Comparison

The performance measures of six fine-tuned models evaluated on the single class dataset are shown in Table 5.1. The default configuration is used to train the model based on the provided fine-tune checkpoints. In this experiment, the models are trained with 80 percent of the total dataset containing 1455 sample images with 4266 object instances and tested on cup test dataset. The precision, recall and FPS of all experimented models are 100 percent on testing with single (cup) class dataset. However, there are significant differences in processing speed. The *ssd_mobilenet* model is the fastest one and nearly twice as fast as the rest of the models. In the

single-class case with clear and identical objects, SSD with MobileNet architecture is preferable in terms of computation. It is found that all of the trained models perform well in the simple single class object. The accuracy gets distinguishable in the multi-class object and for difficult objects.

Table 5.1 *Performance of models evaluated on the single class, cup test dataset.*

Model	**Precision**	**Recall**	**F_1-score**	**FPS**
ssd_mobilenet_v1	1.00	1.00	1.00	**20.19**
faster_rcnn_inception_v2	1.00	1.00	1.00	13.87
faster_rcnn_resnet50	1.00	1.00	1.00	10.91
rfcn_resnet101	1.00	1.00	1.00	10.03
faster_rcnn_resnet101	1.00	1.00	1.00	9.40
faster_rcnn_nas	1.00	1.00	1.00	1.09

The cup instances are easily detected by all the tested models without a single false prediction. The interclass variation of the cup is quite less and there is no such object in our dataset that looks similar to the structure of a cup. There are clearly recognizable cups in the dataset that helped to achieve the perfect result for all models trained detectors.

Multi-class Comparison

An automatic object detector needs to detect the wide range of objects in the varied environment with optimal accuracy. The performance of the trained detector model on the multi-class dataset is the major concern in the real use case of object detector. In this experiment, 80 percent of the total dataset containing 1455 sample images with 4266 objects were used to train the models and tested on TUT test dataset. All models were trained using the default configuration and provided fine-tuned checkpoints for 100k train steps. The result of the fine-tuned trained models detector on this test dataset with their accuracy measures are shown in Table 5.2.

Table 5.2 *Performance of models evaluated on the multi-class, TUT test dataset.*

Model	Precision	Recall	F_1-score	FPS
ssd_mobilenet_v1	**0.962**	0.719	0.823	**20.19**
faster_rcnn_inception_v2	0.936	0.922	0.929	13.87
faster_rcnn_resnet50	0.936	0.954	0.945	10.91
rfcn_resnet101	0.952	0.934	0.943	10.03
faster_rcnn_resnet101	0.943	**0.965**	**0.954**	9.40
faster_rcnn_nas	0.803	0.901	0.849	1.09

It is obvious that *ssd_mobilenet* is the fastest in training and testing. However, the accuracy measures are not the best among other models. Three models *faster_rcnn_resnet50*, *rfcn_resnet101* and *faster_rcnn_resnet101* have almost similar performance in terms of F_1-score. Faster RCNN with a ResNet50 architecture is slightly faster but less accurate than RFCN and faster RCNN architecture with ResNet101 feature extractor. ResNet101 contains more residual layers than ResNet50, hence it is computationally more complex than ResNet50 architecture. The performance of *faster_rcnn_nas* is poor in both: accuracies and computation speed. Only recall of this model detector is slightly better than SSD MobileNet with the cost of almost 20 times slower computation. Based on these results, we continue rest of our experiments using *faster_rcnn_resnet101* model.

We used the threshold value 0.5 for the calculation of these accuracy metrics. F_1-score gives the single accuracy measure as it is harmonic mean of precision and recall. The value of precision, recall and F_1-score calculated on single threshold value do not generalize the performance of the object detector well in all threshold values. Average precision, average recall calculated from the precision-recall curve give better accuracy measure. The mAP would be the suitable metric to calculate the overall performance of the detector on detecting multi-class objects in the dataset.

Mean Average Precision

The mAP is widely used in object detection to measure the performance of detection model. The higher the value of mAP the better the detection model is in detecting multi-class objects. It is the mean of average precision over all classes. In our experiment, we computed the AP overall found detections on the test dataset. As we used the random split to divide the dataset into train-test subset, it is most likely that every next frame from the same video is placed in the different subset. This results in high precision and recall as they are almost identical to each other. The

stratified k-fold cross-validation for train-test split might change this scenario.

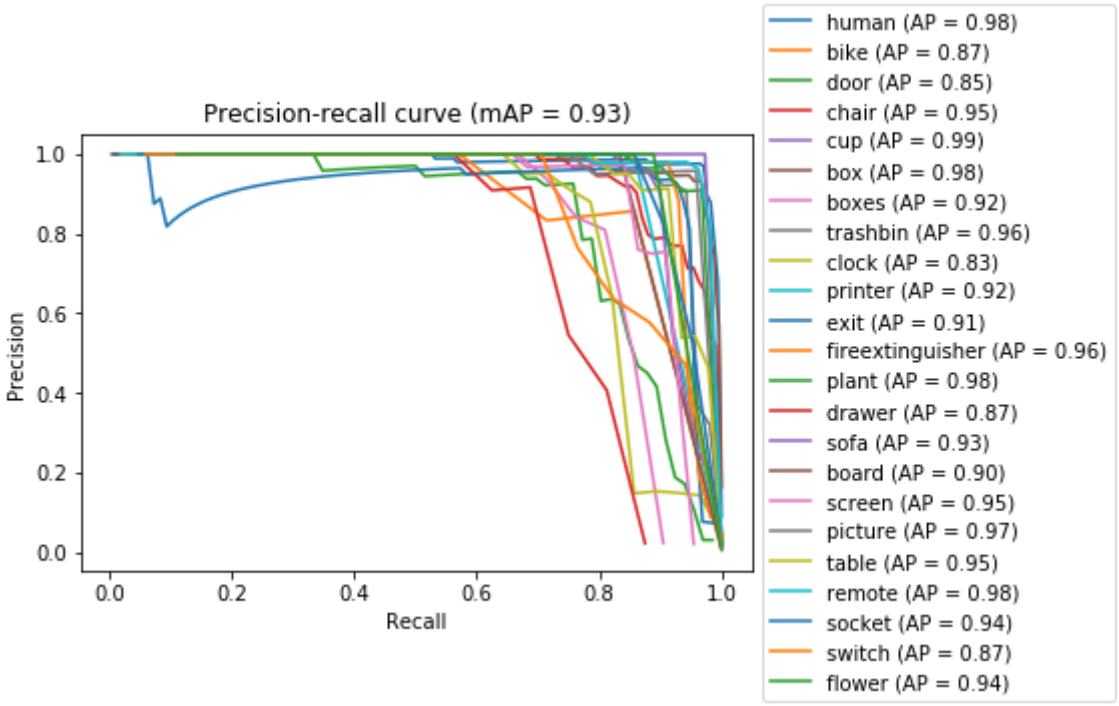

Figure 5.1 *The precision-recall curve with the mAP calculation in left and average precision value of each class shown in right. This calculation is based on the faster RCNN ResNet101 model detector.*

The precision-recall curve together with the average precision of each class is shown in Figure 5.1. It is found that the average precision of bike, clock, door, drawer and socket are quite less than other object classes. These five classes from our dataset are the most difficult for the trained detector. The reason behind the lower accuracy on clock class is that the model over learned the specific time shown on clock (location of the hour and minute hands) rather than structure of clock. While in the case of door and drawer, there are different type of handles and different backgrounds. In the case of switch, the background colour of the wall makes it difficult to detect. Our dataset does not have enough samples with intra-class and inter-class variations from these classes. Meanwhile, the human, cup, fire extinguisher, remote and trash bin classes are easy to detect. These classes have comparatively more samples also their structures are almost identical in every case.

Effect of Data Size

The amount of data needed for the robust object detection model is an important question. There is no clear answer to the question of how much training samples does

it require to train the object detection model to get a robust detector. We tried to solve this question by using our limited dataset. We divided the dataset randomly into the different portions of training and testing subsets. Each time randomly selected 50, 60, 70, 80, and 90 percent of total dataset were used as training set. Faster RCNN ResNet101 model is trained individually with this disjoint data subset for 100k train steps. The performance of the trained detector is evaluated on the new dataset which was never seen to the model before. The small test dataset was used to evaluate the performance of detector trained on each training subset.

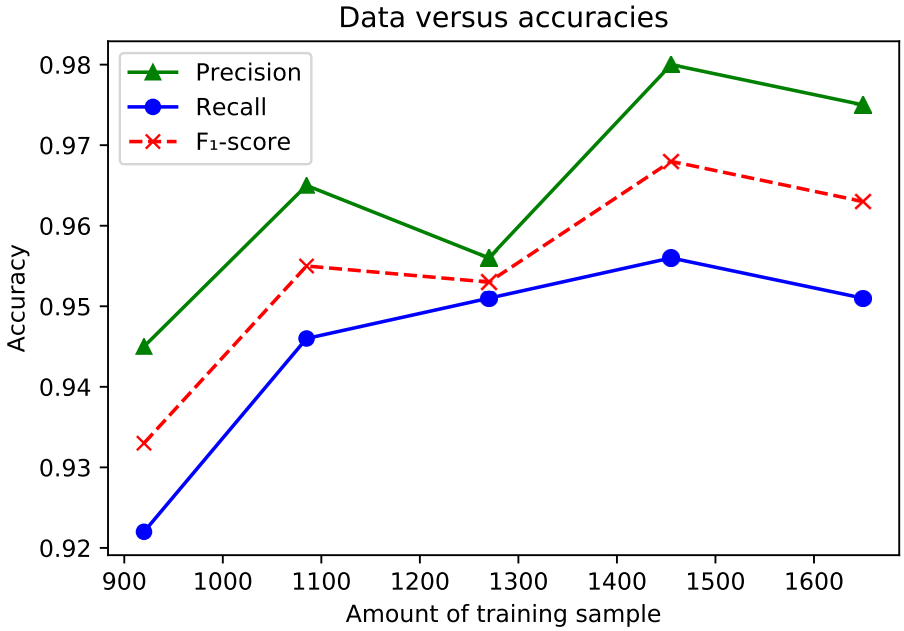

Figure 5.2 *Precision, recall and F_1-scores over the different amount training samples size.*

As shown in Figure 5.2, the number of images on training data have the significant effect on the performance of the detector. As we increase the size of the training samples, the precision, recall and F_1-score of the detector on unseen evaluate dataset increase. Even though the increase on accuracies is not smooth, we can generalize that the accuracies of detector trained on more samples are better compared to less training samples. As the density of objects in the dataset is not balanced, the random split generates the unbalanced subsets. The fluctuation of accuracies is due to unbalanced distribution of objects on the training dataset.

Visual Comparison

We describe manual experiment and analysis on the performance of the fine-tuned object detector in this section. Detection results are compared with the ground truth data from annotation file. We compared the various prediction cases with their ground truth labels.

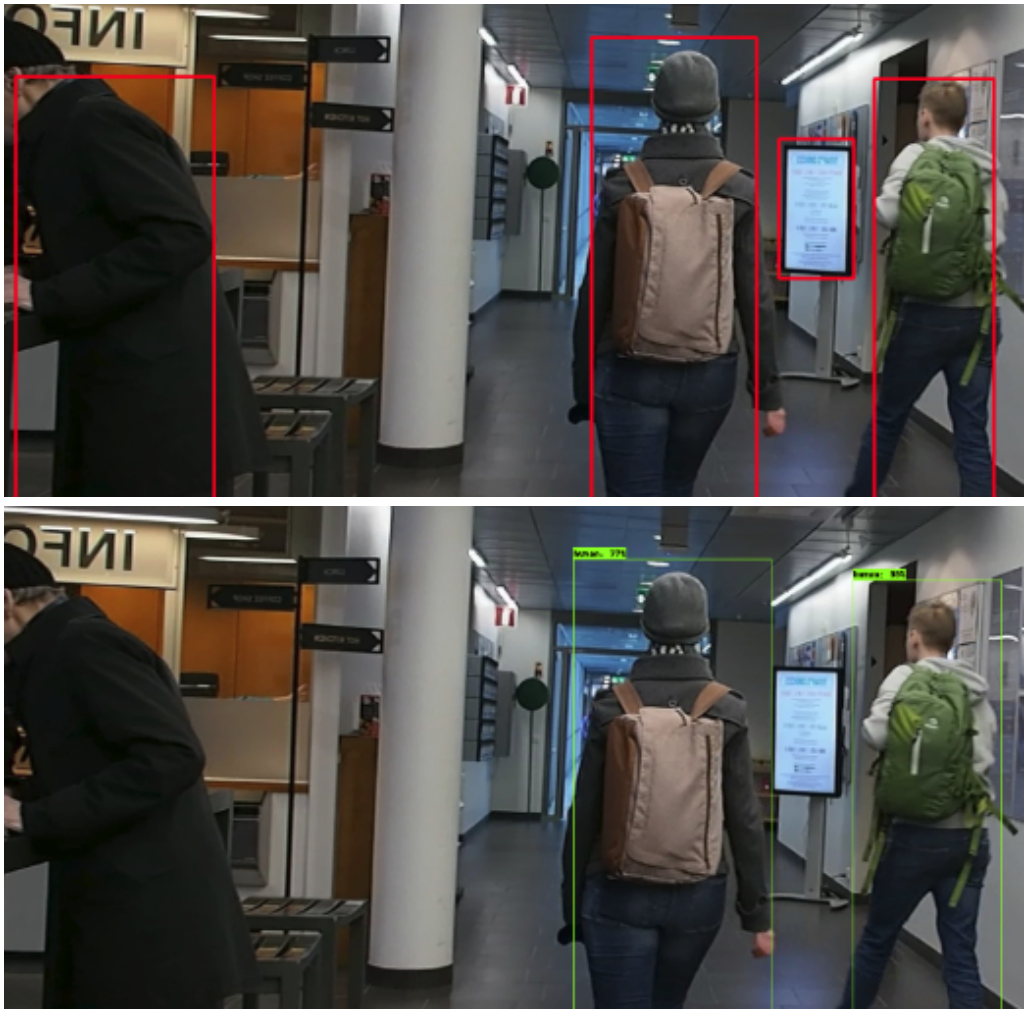

Figure 5.3 *The top image from the annotated dataset contains three humans and a screen in the red bounding box. The down image contains two humans predicted by the detector with the confidence of 99 percent.*

Overlapping objects are the major challenge for an automatic object detection. If there are many occluding object instances in training samples, the trained model on that dataset cannot learn the object map perfectly and hence, the trained detector can not generalized well on unseen images. It is obvious that the prediction of automatic object detection is not 100 percent correct in all cases. The false prediction or

prediction error is the case where trained detector found more or fewer objects than in the original annotated image. As shown in Figure 5.3, the number of detected objects (down image) are less than that are in ground truth annotation (upper image). The human on the left hand and the screen between two detected humans are not detected. This is due to the clarity of human and the screen. Changing the detection threshold value and image quality might change this scenario.

Figure 5.4 *The left image contains two annotated objects: a human and an exit sign. The right image is resulting image of the detector. It found two humans and an exit sign.*

Another case in false prediction is when there are more objects predicted than that are in ground truth. As shown in Figure 5.4, the detector detects the partial body of the human with the confidence of 99 percent that is not in ground truth label. This is one of the most needed ability of the detector to find the partial or occluded object. If the model is trained well with enough data, it outperforms a human as shown in Figure 5.3. Even though the person on the right corner is not clear and some part of him is missing, the detector is able to detect him with excellent confidence. It is more likely to have a human error during data annotation process. We can conclude that, if the well-performing object detection model is trained well with large samples of object instances, it will outperform a human in detecting difficult cases.

We implemented the detector on the mobile device and tested it on university premises. The performance of the SSD MobileNet model deployed on Samsung A3 is quite good. However, the generalization of the model and the confidence level on the correctly detected object is not good compared to detection from the videos. The detector inferences on recorded video have better performance than the deployed detector on a mobile device. The possible reason behind this is quality of the camera, background noise and the view angles. Figure 5.5 visualizes two examples of the detection from our mobile detector.

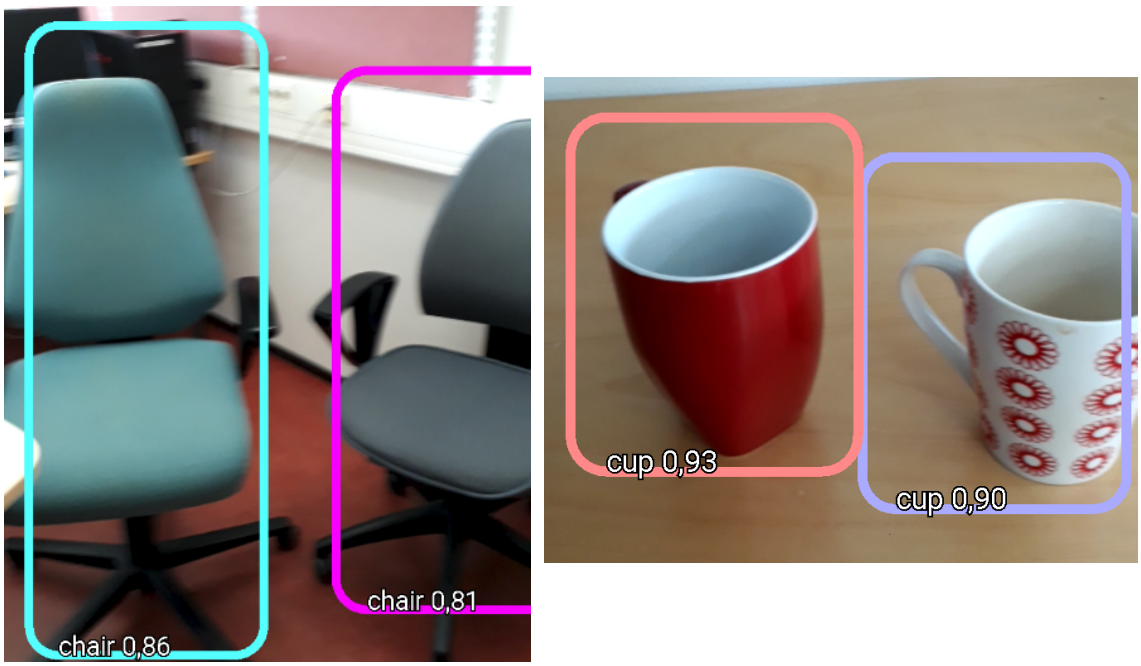

Figure 5.5 *Screenshots captured from the TUT mobile detector application.*

Prediction Error

The *false positive* and *false negative* cases are considered as the false prediction. It is one way to check how well the model detector generalizes on testing data. However, this is not the optimal way to compare the goodness of detection models. Note that the changes in threshold might give different results. The amount of failures shown in this section is obtained from the TUT small test dataset. Out of these 86 images, we wanted to know how many images will be predicted incorrectly from different model detectors.

Based on our accuracy calculation given in Table 5.2, it is obvious that the amount of false prediction is lower in *rfcn_resnet101* and *faster_rcnn_resnet101*, and higher in *ssd_mobilenet*. The false prediction on the RCNN and RFCN architecture with

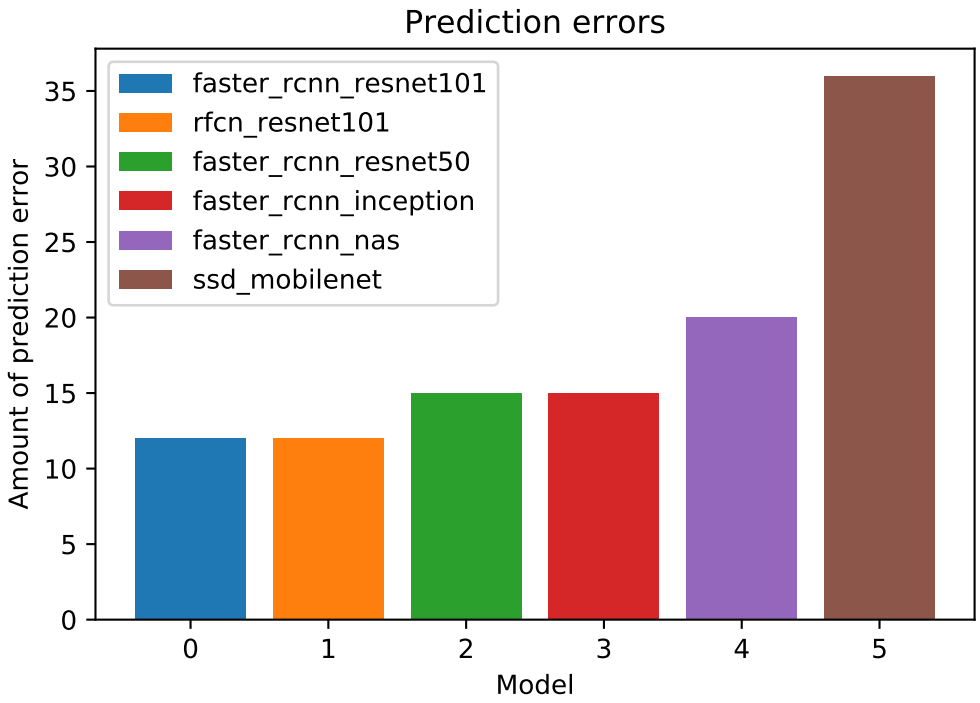

Figure 5.6 *The graph shows the total number of failure cases from each model detectors. There were 86 images on the test dataset. The minimum number of failures is 12 while, the maximum is 36.*

residual network feature extractors are lower in compared to other architectures. Among these failure cases, there are some cases where more objects than the annotated object (more B_p than B_{gt}) were found by the *faster_rcnn_resnet101* detector. While the *ssd_mobilenet* detector failed to detect many ground truth objects (less B_p than B_{gt}). The false predictions of different models are diverse when using the same threshold value for all. When using the detection threshold 0.5, the *faster_rcnn_resnet101* and *rfcn_resnet101* models gave the least false predictions (12 frames) while *ssd_mobilenet* model false prediction was three times higher than those (36 frames) as shown in Figure 5.6. The detection threshold can be changed to change the definition of correct detection that might decrease the amount of false detection. Also changing the training configuration might change the detection performance of models.

5.2 Discussion

Considering the accuracies and speed of detector, it is the difficult job to select one object detection model from the list of different models. In addition to the detection meta-architecture, the feature extractor plays the vital role in the object detection

performance. Regional based CNN commonly considered as outperforming object detection model architecture. We experimentally found that regional based convolutional networks are excellent for the object detection in terms of accuracies even though they are computationally complex than traditional methods. It was found that faster RCNN and RFCN architecture have the almost identical accuracies with the slight difference in computation cost. Faster RFCN with ResNet101 feature extractor is slightly faster than the faster RCNN ResNet101, while the RCNN architecture gives quite good accuracies. As shown in comparison Tables 5.1 and 5.2, SSD architecture is faster to train and have better inference time. Easy deployment ability on a mobile and embedded devices is the big plus point of this model.

The computation speed and the prediction accuracy are the central issues in object detection [24]. The detection of multiple objects in real-time is rather challenging and it is hard to get the perfect result. However, the single class detection was perfect and all experimented models have 100 percent performance accuracy on the easy single class object dataset. Camera-based automatic object detection is challenging due to the large number of objects, intra-class and inter-class variation and object variations. In our experiment, the difficult classes for the detection are chair, printer and sofa. As we discussed in chapter 3, there are eight different type of chairs in our dataset. There are not enough images from each chair type that resulted in less accuracy in chair class. The detection of the cup, exit, fire extinguisher and human were easy as these object instances were almost identical in all cases. Often the printer was detected with multiple classes, printer, screen and box. This was due to the parts of printer matched with the screen and box classes. We found that the data augmentation, especially the horizontal flipping of images increases the accuracies of the detectors by 3 to 4 percentages.

6. CONCLUSION

In this thesis project, we experimented with the popular object detection models from the TensorFlow object detection model zoo. Deep object detection models were trained and tested on the self-collected dataset. We collected the multiclass object dataset from the university premises, labelled it, trained object detection models on it and evaluated the performance using the most common accuracy metrics used in object detection.

We experimented faster RCNN, RFCN and SSD architectures integrated with Inception, MobileNet and NAS, ResNet50 and ResNet101 feature extractors. From this experiment, we found that the performance of fine-tuned models using pretrained configuration on the self-collected dataset is far better than directly using pretrained models. This is obvious because a large number of environment specific dataset is the main necessity to train one's own detector for better detection result. It is very important to do the annotation and preprocessing flawlessly. It was found that data augmentation slightly improves the performance and accuracy of the trained detector.

From our experiment, we conclude that region-based convolutional neural networks are better for object detection. The faster RCNN meta-architecture with ResNet101 feature extractor has superior accuracies in terms of precision, recall and F_1-score, considering its high computation cost. The SSD MobileNet and RFCN-ResNet101 models are faster but they stay below faster RCNN ResNet101 in terms of accuracies. The SSD MobileNet model is good to deploy on mobile and embedded devices while the accuracy is not at top level. Faster RCNN ResNet50 model is the suitable alternative for automatic object detection considering accuracy versus speed trade-offs. The computation complexity of the model is directly proportional to the number of object classes in the dataset, the bigger the number of classes, the more it takes to fine-tune the model.

Future Work

This research experiment is a step towards the real-time object detection of multi-class objects using a camera. The performance of the detector on datasets, demo videos and the mobile implementation are the inspiration to continue the research furthermore. The scope of object detection is wide, there is a broad road to continue from this stage. The future research can consider pixel-wise segmentation of object in dataset instead of bounding box annotation approach. This might give better performance accuracy than the current bounding box annotation approach.

The amount of data in our dataset is not enough and not balanced well to generalize all collected object classes. More images with intra-class and inter-class variation are needed to get better generalization result. Automatic annotation might be good practice for data annotation to reduce the time constraint during the collection of the labelled dataset. More data can be collected and trained to improve the current accuracy score of detectors. Other object detection architectures can be trained and evaluated using the collected dataset. The deployment of well-performing detection models to the mobile and embedded devices is another interesting thing to see in the future. Real-time object tracking, counting interesting objects, localization of object using indoor navigation can be investigated in the future. As accuracy and computational complexity are all time concerns for real-time object detection, we can shift our focus on it by modifying the detection model architecture. The collected dataset can be used in other research projects.

REFERENCES

[1] T. Bottger, P. Follmann, and M. Fauser, "Measuring the accuracy of object detectors and trackers," *CoRR*, vol. abs/1704.07293, 2017. [Online]. Available: http://arxiv.org/abs/1704.07293

[2] F. Chollet, *Deep Learning with Python*. Shelter Island, New York, United States: Manning Publication, Nov 2017.

[3] D. Coldewey. (2018) Here's how uber's self-driving cars are supposed to detect pedestrians. [Online]. Available: https://techcrunch.com/2018/03/19/heres-how-ubers-self-driving-cars-are-supposed-to-detect-pedestrians/

[4] T. Community, "Tensorflow detection model zoo," https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md, 2018.

[5] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: object detection via region-based fully convolutional networks," *CoRR*, vol. abs/1605.06409, 2016. [Online]. Available: http://arxiv.org/abs/1605.06409

[6] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, pp. 98–136, Jan 2015. [Online]. Available: https://doi.org/10.1007/s11263-014-0733-5

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun 2010. [Online]. Available: https://doi.org/10.1007/s11263-009-0275-4

[8] G. Forman and M. Scholz, "Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement," *SIGKDD Explor. Newsl.*, vol. 12, no. 1, pp. 49–57, Nov 2010. [Online]. Available: http://doi.acm.org/10.1145/1882471.1882479

[9] A. Fuentes, S. Yoon, S. C. Kim, and D. S. Park, "A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition," *Sensors*, vol. 17, no. 9, 2017. [Online]. Available: http://www.mdpi.com/1424-8220/17/9/2022

[10] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. [Online]. Available: http://www.cvlibs.net/datasets/kitti/

[11] A. Geron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media, Inc., 2017.

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, Jan 2016.

[13] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He, "Detectron," https://github.com/facebookresearch/detectron, 2018.

[14] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015. [Online]. Available: http://arxiv.org/abs/1504.08083

[15] Gluon. (2017) Object detection using convolutional neural networks. [Online]. Available: http://gluon.mxnet.io/chapter08_computer-vision/object-detection.html

[16] A. A. Godil, R. V. Bostelman, W. P. Shackleford, T. H. Hong, and M. O. Shneier, "Performance metrics for evaluating object and human detection and tracking systems," *NIST Interagency/Internal Report (NISTIR)*, 2014. [Online]. Available: https://dx.doi.org/10.6028/NIST.IR.7972

[17] V. V. Gomez, "Object detection for autonomous driving using deep learning," Dec 2015. [Online]. Available: http://www.iri.upc.edu/files/academic/thesis/98-Research-Plan.pdf

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[19] A. Gopnik, "Making ai more human," vol. 316, pp. 60–65, 06 2017.

[20] V. GUPTA. (2017) Understanding feedforward neural networks. [Online]. Available: https://www.learnopencv.com/understanding-feedforward-neural-networks/

[21] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed. New York: Pearson Prentice Hall, Nov 2009.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[23] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[24] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," *CoRR*, vol. abs/1611.10012, 2016. [Online]. Available: http://arxiv.org/abs/1611.10012

[25] J. Hui. (2018) map for object detection. [Online]. Available: https://medium.com/@jonathan_hui

[26] P. Jay. (2018) What problem does resnet solve? [Online]. Available: https://medium.com/@14prakash

[27] P. Kim, *MATLAB Deep Learning With Machine Learning, Neural Networks and Artificial Intelligence*. Apress, Berkeley, CA, 2017. [Online]. Available: https://doi.org/10.1007/978-1-4842-2845-6

[28] D. E. King, "Dlib-ml: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[29] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *CoRR*, vol. abs/1708.02002, 2017. [Online]. Available: http://arxiv.org/abs/1708.02002

[30] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *Computing Research Repository*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[31] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, and M. Sebe, Nicuand Welling, Eds. Cham: Springer International Publishing, 2016, pp. 21–37.

[32] S. Murray, "Real-time multiple object tracking - A study on the importance of speed," *CoRR*, vol. abs/1709.03572, 2017. [Online]. Available: http://arxiv.org/abs/1709.03572

[33] T. Nguyen1, E.-A. Park1, J. Han1, D.-C. Park1, and S.-Y. Min2, "Object detection using scale invariant feature transform," vol. 238, 2014.

[34] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *CoRR*, vol. abs/1712.04621, 2017. [Online]. Available: http://arxiv.org/abs/1712.04621

[35] C. B. Rasmussen, "R-fcn object detection ensemble based on object resolution and image quality," Master's thesis, Aalborg University, Aalborg, Denmark, Jun 2017.

[36] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[37] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, June 2017.

[38] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[39] A. Rosebrock. (2016) Intersection over union (iou) for object detection. [Online]. Available: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

[40] F. Shaikh. (2017) 10 advanced deep learning architectures data scientists should know! [Online]. Available: https://www.analyticsvidhya.com/blog/2017/08/10-advanced-deep-learning-architectures-data-scientists/

[41] N. Shukla and K. Fricklas, *Machine Learning with TensorFlow*. Shelter Island, New York, United States: Manning Publication, Jan 2018.

[42] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[43] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842

[44] G. B. Team. (2018) About tensorflow. [Online]. Available: https://www.tensorflow.org/

[45] Tensorflow. (2018) Programmer's guide. [Online]. Available: https://www.tensorflow.org/programmers_guide/datasets

[46] T. Tikkainen, "Cell detection from microscope image using histogram of oriented gradients," Master's thesis, Tampere University of Technology, Tampere, Finland, Nov 2014.

[47] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *CoRR*, vol. abs/1611.01578, 2016. [Online]. Available: http://arxiv.org/abs/1611.01578

[48] G. Zoumpourlis, A. Doumanoglou, N. Vretos, and P. Daras, "Non-linear convolution filters for cnn-based learning," *CoRR*, vol. abs/1708.07038, 2017. [Online]. Available: http://arxiv.org/abs/1708.07038