|  |
| --- |
| University of Namibia, School of Computing |
| Object detection and classification using machine learning techniques |
| Research Paper  Name: Chisulo Mukabe  Student Number: 201406062  Supervisor: Prof. William Sverdlik |

|  |
| --- |
| Chisulo Mukabe  11-21-2017 |

Table of Contents

Declaration……………………………………………………………………………………………………………………………………………… i

Acknowledgements……………………………………………………………………………………………………….…………………………ii

Table of Figures…………………………………………………………………………………………………………..………….….…………. iii

Table of Acronyms and Abbreviations………………………………………………………………….……………………..…………. v

[Chapter 1: Introduction 3](#_Toc499017563)

[Abstract 3](#_Toc499017564)

[Haar Cascades 4](#_Toc499017565)

[Neural Network 6](#_Toc499017566)

[Motivation 7](#_Toc499017567)

[Chapter 2: Problem Statement 8](#_Toc499017568)

[Chapter 3: Research Question and Objectives 9](#_Toc499017569)

[3.1 Research Questions 9](#_Toc499017570)

[3.2 Research Objectives 9](#_Toc499017571)

[Chapter 4: Literature Review 10](#_Toc499017572)

[A comparative study of neural network algorithms applied to optical character recognition (Smagt, 1990) 10](#_Toc499017573)

[Neural Network Based Handwritten Hindi Character Recognition System (Singh, Dutta, & Singh, 2009) 10](#_Toc499017574)

[Eye Movement as Navigator for Disabled Person (Utaminingrum, Fauzi, Sari, Primaswara, & Adinugroho, 2016) 11](#_Toc499017575)

[A fully automatic hand gesture recognition system for human-robot interaction (Nguyen, Pham, Dong, Nguyen, & Tran, 2011) 11](#_Toc499017576)

[Model Generation for Video-Based Object Recognition (Noor, Mirza, Sheikh, Shah, 2006) 12](#_Toc499017577)

[A Pedestrian and Vehicle Rapid Identification Model Based on Convolutional Neural Networks (Wang and Xu) 12](#_Toc499017578)

[Going Deeper with Convolutions (Szegedy, et al., 2015) 13](#_Toc499017579)

[Chapter 5: Research Methodology and Process 14](#_Toc499017580)

[Experiments 14](#_Toc499017581)

[Training 14](#_Toc499017582)

[ImageGrab.py 18](#_Toc499017583)

[Training the Haar Cascade 24](#_Toc499017584)

[Retraining the Neural Network 32](#_Toc499017585)

[Testing 36](#_Toc499017586)

[Design and Creation 37](#_Toc499017587)

[How the Robot Works 39](#_Toc499017588)

[Chapter 6: Research Findings and Discussions 41](#_Toc499017589)

[Haar Cascade Findings 41](#_Toc499017590)

[Training results 42](#_Toc499017591)

[Training graphs 43](#_Toc499017592)

[Go Sign 44](#_Toc499017593)

[Stop Sign 46](#_Toc499017594)

[Left Sign 46](#_Toc499017595)

[Right Cascade 49](#_Toc499017596)

[Neural Network Findings 51](#_Toc499017597)

[Chapter 7: Summary and Conclusion 56](#_Toc499017598)

[Training and Testing 56](#_Toc499017599)

[Strengths and Limitations 57](#_Toc499017600)

[Chapter 8: Recommendations 57](#_Toc499017601)

[Robot Navigation 58](#_Toc499017602)

[Object-focused video Recording 58](#_Toc499017603)

[Surveillance and security 58](#_Toc499017604)

[Alternative input device 59](#_Toc499017605)

[Taxonomy 59](#_Toc499017606)

[Quality Control 59](#_Toc499017607)

[Translation Application 59](#_Toc499017608)

[References 60](#_Toc499017609)

[Appendices 61](#_Toc499017610)

# Declaration

I, Chisulo Mukabe, declare that this report is a presentation of my original research work and is submitted in partial fulfilment of the requirements for the Bachelor of Science Degree in Computer Science. I further declare that this report and programs, and robots developed during the research has not been used before at any institution of higher education to acquire a qualification.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Chisulo Mukabe |  | Date |
| *Researcher* |  |  |
|  |  |  |
|  |  |  |
| William Sverdlik |  | Date |
| *Research Project Supervisor* |  |  |
|  |  |  |
|  |  |  |
| Gabriel T Nhinda |  | Date |
| *Research Project Coordinator* |  |  |

# Acknowledgments

I’d like to thank everyone who has been there with me throughout this incredible journey. I would specifically like to thank my parents Yohane and Ann Mukabe, for the unconditional love and support you have given me in my life. To my supervisor Professor William Sverdlik, for the guidance during this research paper, and also for everything you have taught me and helping me realize my passion. I am truly grateful. To my family for the encouragement especially when things were difficult, and to my friends and my lecturers who have been like family.

# Table of figures

|  |  |
| --- | --- |
| Fig 1.1 | Examples of the rectangle features use in Haar-cascades. |
| Fig 1.2 | The Perceptron. |
| Fig 1.3 | Neural Network and Convolutional Neural Network |
| Fig 5.1 | Image net home page |
| Fig 5.2 | Class and class images |
| Fig 5.3 | Class download and URLs |
| Fig 5.4 | ImageGrab.py |
| Fig 5.5 | Store raw images. |
| Fig 5.6 | Downloading images |
| Fig 5.7 | Find Uglies |
| Fig5.7.1 | Placeholder image (53.jpg) |
| Fig 5.8 | Remove broken |
| Fig 5.9 | Edit Local |
| Fig 5.10 | Create neg file |
| Fig 5.11 | File structure of cascade workspace |
| Fig 5.12 | Create Samples: creates positive image dataset |
| Fig 5.13 | Create Samples: Creates the positive vector file |
| Fig 5.14 | Train Cascade command |
| Fig 5.15 | Train Cascade |
| Fig 5.16 | Cascade file |
| Fig 5.17 | Cloning git repository |
| Fig 5.18 | Dataset classes |
| Fig 5.19 | Starting tensorboard |
| Fig 5.20 | Retrain parameters |
| Fig 5.21 | Training Beginning: Creating Bottlenecks |
| Fig 5.22 | Training complete: Final test accuracy = 92% |
| Fig 5.23 | Testing the network |
| Fig 5.24 | Sketches and images of the robot |
| Fig 5.25 | Fritzing diagram |
| Fig 5.26 | Object detected (left) and coordinates of the object (right). |
| Fig 6.1 | data recorded on Haar Cascade training |
| Fig 6.2 | Graphs of training the cascades |
| Fig 6.3 | Graphs of accuracy and cross entropy of neural network being trained |
| Fig 6.4 | Additional graphs of the neural network training |

# Table of Acronyms and Abbreviations

|  |  |
| --- | --- |
| AI | Artificial Intelligence |
| NN | Neural Network |
| ANN | Artificial Neural Network |
| CNN | Convolutional Neural Network |
| SIFT | Scale-Invariant Feature Transformation |
| HOG | Histogram of Oriented Gradients |
| V3 | Version 3 |
| URL | Universal Resource Locator |
| Synset | Synonym Set |
| HR | Hit Rate |
| MR | Miss Rate |
| FA | False Alarm |

Chapter 1: Introduction

# Abstract

Object recognition and object detection are sub fields of computer vision, the task of giving computers the ability to perceive and respond to the world around them. This is a very useful technology and has many different applications in many different disciplines. Examples of applications include but are not limited to: Use in security by using facial recognition; use in the medical field for classification of cancer types (malignant or benign) or detecting sickle cells in the blood; or it can be used as a research tool to automatically record data (e.g. count the number of trucks that use a particular highway); or used in agriculture or industrialization for quality control; or for entertainment purposes in games that can detect and track a users’ movement to control a character in a game. And there are many more examples, in which it can be used.

However, just as they are many applications, there are also many methods of implementing these systems, such as Convolutional neural networks, Haar cascades, Scale-Invariant Feature Transformations (SIFT), Histogram of Oriented Gradients (HOG) and several others including combinations of these. Each technique may have different setup procedures and different applications where one may work better than the other. This paper aims to measure and give a comparison of some of these techniques for object detection and tracking. The methods used in this paper are, Haar Cascades and Neural Networks.

The purpose of this paper is to learn more about these machine learning techniques by comparing the way they work and the way they are implemented, by doing so one can also understand contexts in which to use these systems. The artifacts used in this paper included a robot that can track an object and an application that could classify an image based on a set of objects it was trained on, that is the application can determine what the picture is of based on a set of images it was trained on.

# Haar Cascades

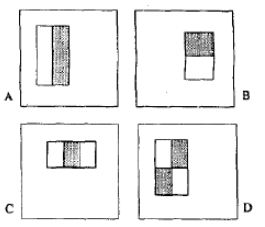
Haar Cascades is the more traditional, but still effective method for object detection. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. (Face Detection using Haar Cascades).

The feature selection is done with the classifier training using Ada-boost and integral images. There are three kinds of features generally used in which the value of a two-rectangular feature is the difference sum of the pixels within two regions. Where as in the three-rectangular features are computed by taking the sum of two outside rectangles and then subtracted with the sum in a center rectangle. With four-rectangles feature computes the difference between diagonal pairs of rectangles. (Gagan, 2012)

An example is extracted from the OpenCV Face Detection using Haar Cascades documentation.

Take for instance a classifier is being trained for facial detection. Initially, the algorithm needs a lot of positive and negative images. Then features need to be extracted from it.

The diagram below shows the haar features that are used.



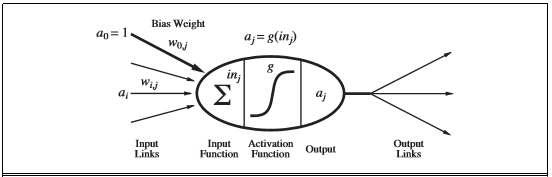
*Fig 1.1. Examples of the rectangle features use in Haar-cascades.*

A and B are edge features, while C is a line feature and D is a Four-rectangle feature. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from the sum of pixels under the black rectangle. However, a small 24x24 image could produce over 160000 features which would require a large amount of computational power to calculate each feature. However, among all these features, most of them are irrelevant. In order to get the best features, a technique called Adaboost is applied. For each feature, it finds the best threshold which will classify the faces to positive and negative. There will be errors of misclassifications and therefore, we pick the features with minimum error rate.

As mentioned earlier, the training process requires a set of positive and negative images for training and a set of features are selected using Ada-boost for training the classifier (Gagan, 2012). The idea of using Ada-boosting is that the system learns a single simple classifier, adjusting the weight of data where errors were made. Then a second simple classifier is learned on the weighted classifier, and the data is reweighted on a combination of the 1st and 2nd classifier. The process continues until the final classifier which is a combination of all the previous classifiers. Using Ada-boosting makes detection faster as an object that does not pass the first low-level classifiers will be dis-regarded and will not go through the other classifiers. This however poses limitation for objects that change shape.

# Neural Network

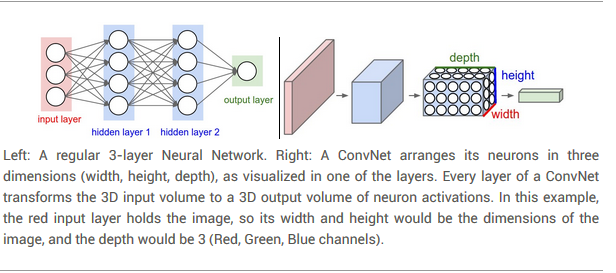
A neural network or artificial neural network is an implementation for machine learning that mimics the way biological neurons work (Jha, 2007). Though it is not a new concept, neural network research was limited by the lack of data and computing power at the time of its conception, both of which are readily available today. The network is made up of neurons, the equivalent of a neuron in the biological brain. These artificial neurons are capable of taking in various inputs and returning a certain output, similar to how neurons can receive various electrical signals and sends an output signal based on the different impulses. It achieves this by multiplying each input value by a value called a weight or bias then calculates the weighted sum of the inputs to represent the total strength of the input. A step function is then applied on the sum to determine its output. This output can be fed into other neurons, forming the neural network. The network can also be made up of just one neuron, in this case it’s called a perceptron. The diagram below extracted from Artificial Intelligence: A Modern Approach shows the perceptron, with certain number of inputs and an output. (Russell & Peter, 2010)



*Fig 1.2. The Perceptron.*

A combination of these artificial neurons is what make up a neural network. These networks usually have three layers, an input layer, a hidden layer and an output layer. Each layer consists of one or more neurons that receive data, perform some calculations on it and output it on to the next layer or as the final output.

The neural network used in this paper is a Convolutional Neural Network (CNN), which are very similar to ordinary neural networks. The difference lies in the fact that CNN architectures make the explicit assumption that the inputs are images (Karpathy, n.d.). The diagram from the Stanford Computer Science class: Convolutional Neural Networks for Visual Recognition, shows a neural network and a convolutional neural network.



*Fig 1.3. Neural Network and Convolutional Neural Network*

# Motivation

Artificial Intelligence is a topic that is getting a lot of attention in technological advances and as mentioned earlier has many applications in many different fields. The motivations for computer vision specifically with object detection and tracking, is simply because of the many different applications it can be applied to, from domestic to industrial applications. At the same time, it is better to understand the technology before implementing it in real world scenarios, this way it is possible to know which methods work best in different scenarios which can make the difference in areas such as efficiency to whether the entire system works or not.

Another, more personal motivation is that the field of interest is particularly new to the researcher and this provides the opportunity to research and gain a deeper understanding in the field of artificial intelligence and computer vision.

Chapter 2: Problem Statement

The fields of object recognition/detection and AI in general hold a lot of promise in many different and versatile fields. They can perform certain tasks faster and better than humans can which can result in faster, safer, more cost-efficient means of performing tasks.

However, despite the great promise these systems could bring, there is very little evidence to show its use in Namibia or Southern Africa as a whole. It’s research and development has been limited to a few African countries such as Kenya, Nigeria and South Africa and fewer are looking into vision systems.

Therefore, this research will aim to implement an object detection/classification system, to gain a better understanding of the process and to help discover areas of use. This will be achieved by comparing the training, and testing of the two mentioned techniques.

Chapter 3: Research Question and Objectives

# 3.1 Research Questions

The research questions this study aims to answer are:

* How does the Neural Network, specifically the Inception v3 CNN compare with Haar Cascades in terms of:
  + Training i.e. time to train, improvement rate with increased training times.
  + Testing i.e. the ratio of correct detections/classifications to incorrect detections/classifications (true positives to false positives).
  + Performance i.e. how well they can detect and track the object and how well it classifies the object in controlled environments and natural environments. (Using a predefined heuristic).
* What limitations exist in the above-mentioned procedures. What general limitations can be found during this process

# 3.2 Research Objectives

In so answering the research questions, they will aid in accomplishing the research objective which is to:

Gain insight into the two machine learning algorithms, to gain a deeper understanding as well as finding contexts in which these methods would be suitable, based on the information gathered.

Chapter 4: Literature Review

# A comparative study of neural network algorithms applied to optical character recognition (Smagt, 1990)

In this paper, three simple general-purpose networks are tested for pattern classification on an optical character recognition problem. The feed-forward (multi-layer perceptron) network, the Hopfield network and a competitive learning network are compared. The input patterns are obtained by optically scanning images of printed digits and uppercase letters. The resulting data is used as input for the networks with two-state input nodes; for others, features are extracted by template matching and pixel counting. The classification capabilities or the networks are compared with a nearest neighbor algorithm applied to the same feature vectors. The results of this study found that the feed-forward network reaches the same recognition rates as the nearest neighbor algorithm, even when only a small percentage of possible connections is used. The Hopefield network performed less well, and a problem of overloading the network was present. Thee competitive learning network also reached results similar to the nearest neighbor algorithm when input patterns were clustered well. (Smagt, 1990)

The paper compared the classification rate of different types of neural networks. This paper on the other hand is comparing different methods, rather than different types of the same method. Regardless of this though, it still provides some useful insight as to how the research was carried out, including the different factors that had to be considered for each type of neural network.

# Neural Network Based Handwritten Hindi Character Recognition System (Singh, Dutta, & Singh, 2009)

The paper presents a means to recognize handwritten Hindi characters using neural networks, specifically a Multilayer perceptron network (MLP). Apart from just creating the system, the paper also presents a comparison of the outcome between a Global input and Gradient feature input. The result of this research produced a system that was capable of recognizing the characters with an 85% accuracy using the Global input, and one over 94% accuracy using the Gradient feature input. (Singh, Dutta, & Singh, 2009)

This paper shows some of the applications that object recognition and artificial intelligence can be applied to.

# Eye Movement as Navigator for Disabled Person (Utaminingrum, Fauzi, Sari, Primaswara, & Adinugroho, 2016)

Eyes is one of the human organs which is mostly still functions properly in disabled people when other parts of the body are disabled. The research proposed a new framework to recognize and detect eye movement for handling the position by considering the decision of both left and right eye. Haar cascades were used as the algorithm for observing the area of the eyes, then thresholding the image using morphology to obtain the focus of the eyes. The results found that the performance could reach over 80% in all datasets. (Utaminingrum, Fauzi, Sari, Primaswara, & Adinugroho, 2016)

The paper presents an example of a useful application of object detection but also illustrates the data preparation of the images such as resizing, morphing and transformations.

# A fully automatic hand gesture recognition system for human-robot interaction (Nguyen, Pham, Dong, Nguyen, & Tran, 2011)

The paper mainly focuses on Human-Robot Interaction (HRI) and uses Haar Cascades. The paper made two contributions. The first was a proposed framework for hand gesture recognition: hand vocabulary design, feature extraction for hand posture representation, hand posture classification, and hand gesture database construction. The second were that some experimentations were realized to evaluate the defined hand gesture set that can be used in general situations of HRI. The experiment results showed that the defined hand gesture set satisfied both criteria: intuitiveness and recognizability. (Nguyen, Pham, Dong, Nguyen, & Tran, 2011)

# Model Generation for Video-Based Object Recognition (Noor, Mirza, Sheikh, Shah, 2006)

This paper presents a novel approach to object recognition involving a sparse 2D model ad matching using video. The model is generated on the basis of geometry and image measurables only. They first identified the underlying topological structure of an image dataset containing different views of the objects and represented it as a neighborhood graph. The graph is then refined by identifying redundant images and removing them using morphing. This gives a smaller dataset, leading to reduced space requirements and faster matching. Lastly, they exploit motion continuity in video and extend their algorithm to perform matching based on video input and demonstrate that the results obtained using a video sequence are more robust than using a single image (Noor, Mirza, Sheikh, Jain, & Shah, 2006).

The results found that by using their algorithm, the dataset size could be roughly reduced to 60% via removal of redundant images through morphing. They also showed that video based matching provided significant improvement over single image based matching with a 40% increase of correct matches. The reduced dataset can prove to be beneficial to measure the performance improvement of the different techniques used in this study.

# A Pedestrian and Vehicle Rapid Identification Model Based on Convolutional Neural Networks (Wang and Xu)

In the dissertation, the authors propose a model titled Overfeat to improve the performance of Convolutional Neural Networks (CNN) used for vehicle and pedestrian identification. They state that since the image recognition in the field of intelligent transportation needs high real-time performance, it requires improving the speed of CNN. The experiments involved using both the traditional sliding window method and the Overfeat method for the recognition of pedestrians and cars on the road. They then compared the advantages and disadvantages of the two methods in terms of their recognition effect and speed. Their outcomes showed that the Overfeat model improved the real-time performance and had a lower recognition error rate at fewer target categories. (Wang & Xu, 2015)

# Going Deeper with Convolutions (Szegedy, et al., 2015)

The paper was on the proposal of a deep convolutional neural network architecture codenamed Inception which achieves the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014. The highlight of the architecture is the improved utilization of the computing resources inside the network. They were able to increase the depth and width of the network without increasing the computational budget. To optimize on quality, the architectural decisions were based on the Hebbian principle and the intuition of multi-scale processing. One particular incarnation used in the submission for the competition is called GoogLeNet, which is a 22 layers deep network, the quality of which was assessed in the context of classification and detection.

This paper provides a general understanding as well as a technical explanation of the neural network model used in this paper. It provides information about its architecture as well as its performance compared with other models. As the model was used on classification and detection, it will not be used for tracking purposes in this paper.

Chapter 5: Research Methodology and Process

The research methodology used in this paper focuses mainly on experiments. This is mostly due to the fact that most of the research involved making observations and comparisons between the different object detection methods and making conclusions about their performance as well as strengths and limitations. The data analysis included both qualitative and quantitative analysis. At the same time, a few principles of Design and Creation have also been used in the development of the artifact.

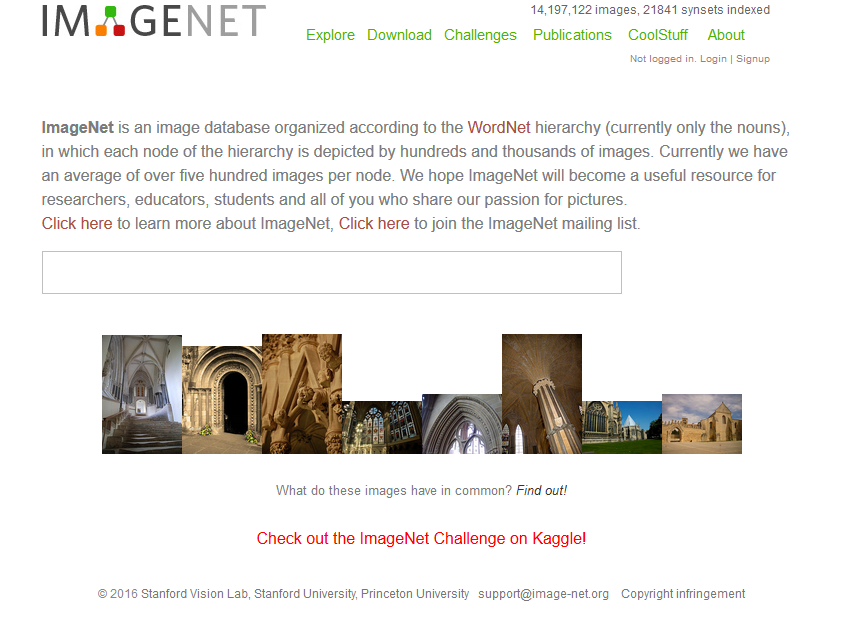
# Experiments

The experiments carried out in the research were aimed at addressing the questions posed in the research questions and objectives which are to compare the machine learning approaches on different parameters. These parameters being, Training, Testing and Performance. During each of these phases data was collected and observations were made.

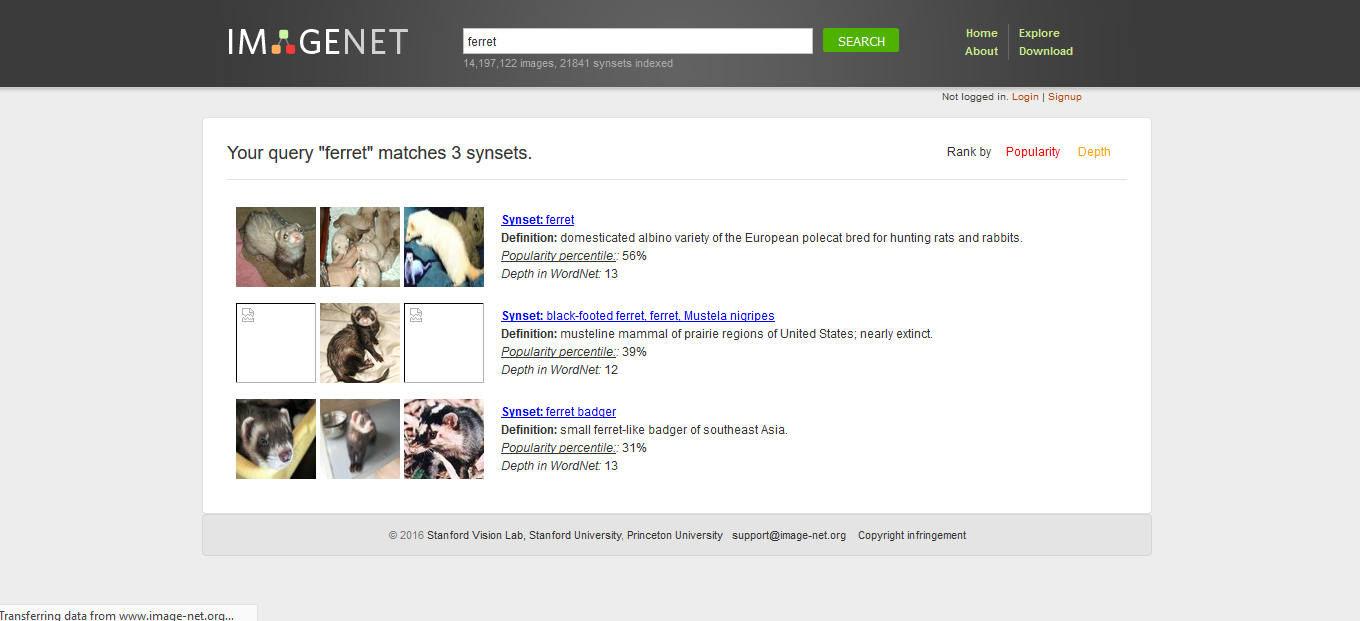
## Training

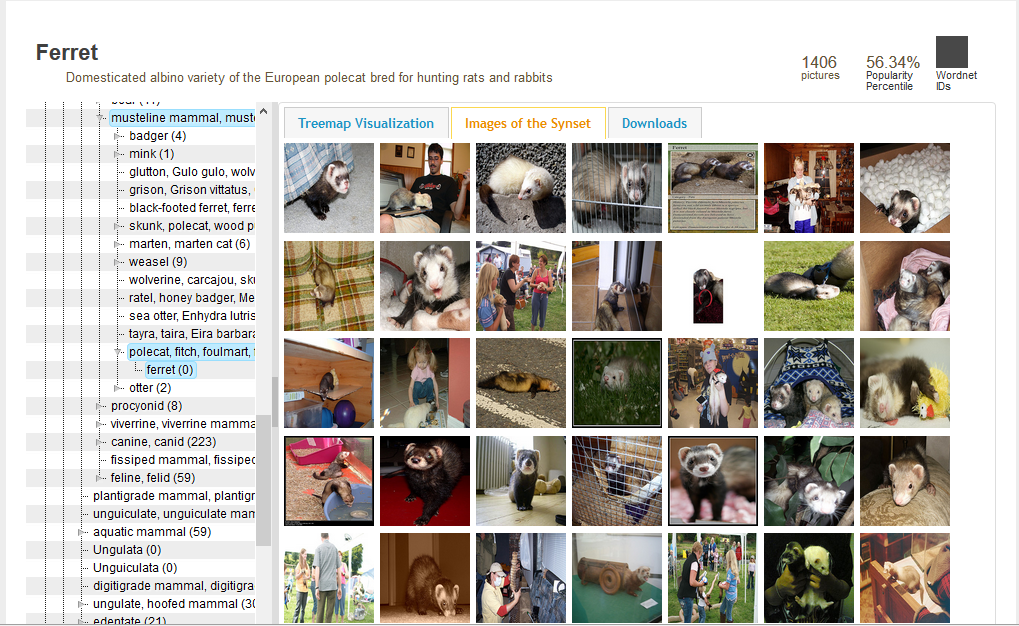
The training process involves feeding the AI system with labelled data to train it to detect different classes of objects. At this stage, the process was monitored to measure and compare the time taken to train the system given specified amounts of training data.

In order to train the systems, a dataset of images was required. This data set would be images of different objects that would represent the different labelled classes. Most of this dataset was acquired from an open source image database called ImageNet.org. It is an image database that is organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. Currently there is an average of over five hundred images per node (Stanford Vision Lab, Stanford university, Princeton University, n.d.). The images on the site are already categorized by objects and are available for download by providing the image URL. Various samples were downloaded using a script that parsed through the image URLs and downloaded them.

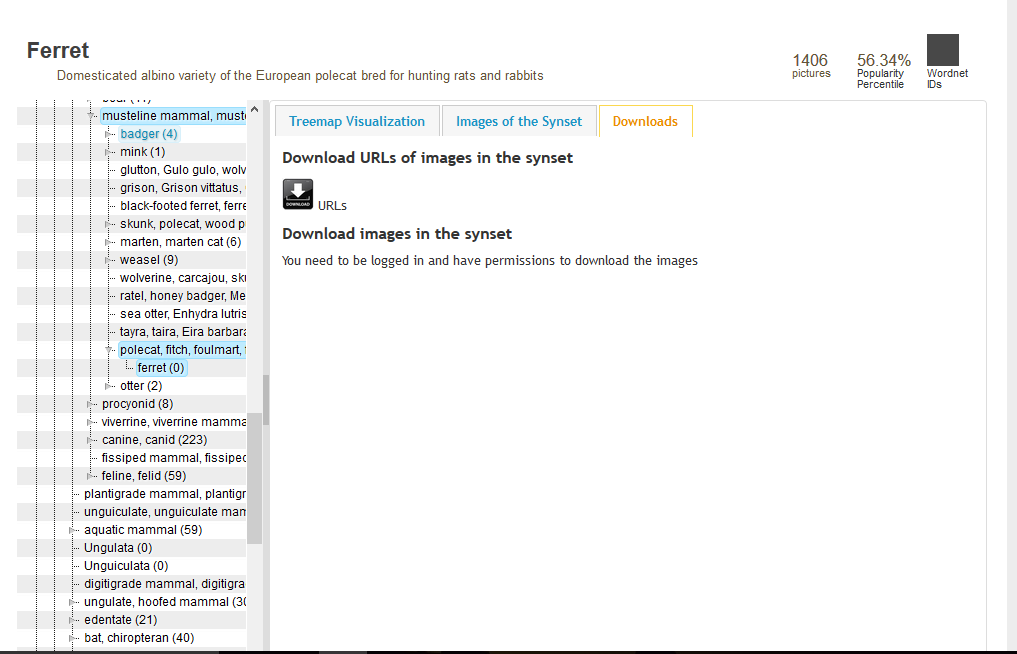


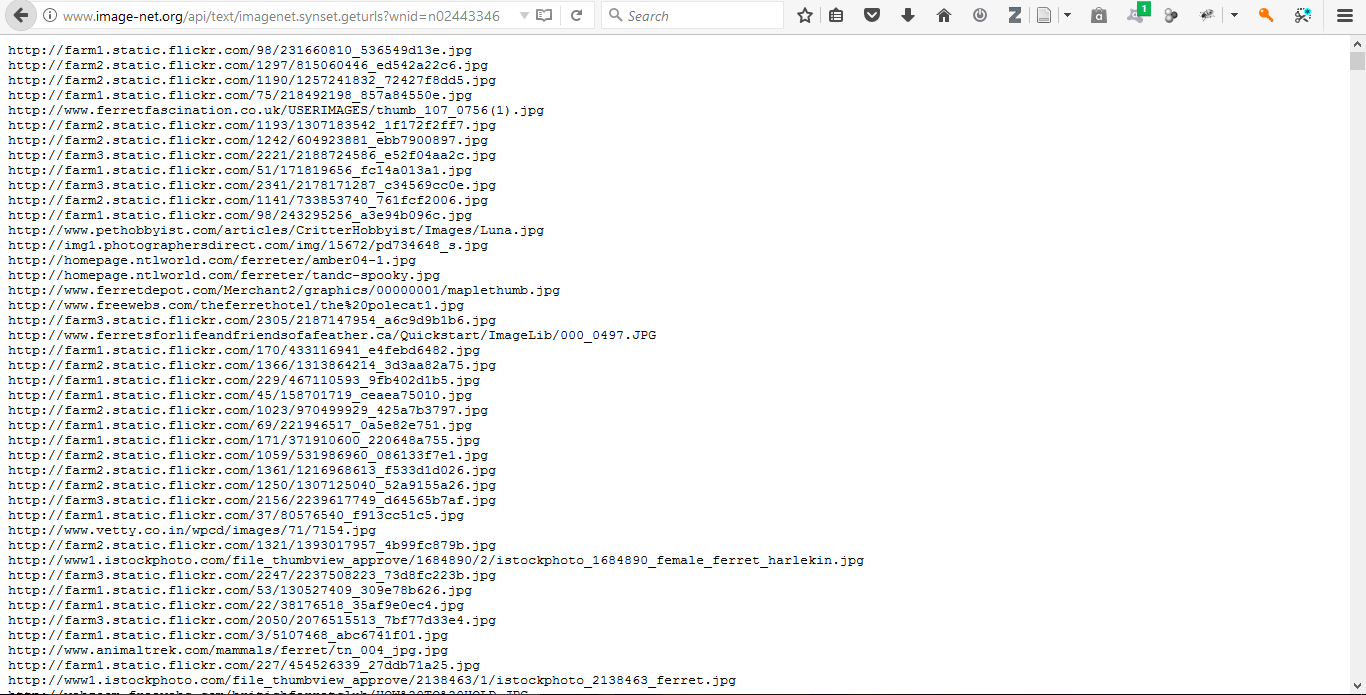
*Fig 5.1. Image net home page*





*Fig 5.2. Class and class images*



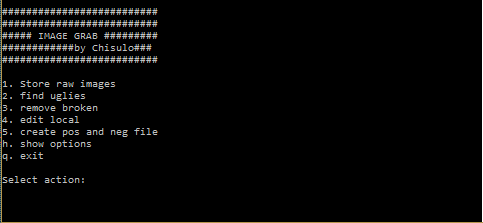


*Fig 5.3. Class download and URLs*

In order to download the images, each URL would have to be accessed, and with a node (also referred to as a synonym set or synset) of 500+ images, this would be a tedious task. A faster alternative would be to create a script that would automatically access the URLs and download the images. Any other preprocessing such as resizing and converting to grayscale can also be done here. More details about the Script is provided below.

### ImageGrab.py

ImageGrab.py is a python script that I created to automatically download and apply preprocessing to the images to be collected from imageNet.org. As shown in the diagram, the script allows the user to download the images, remove corrupted and unwanted images, edit images downloaded and create a positive and negative file that can be used by the Haar Cascade training.



*Fig 5.4. ImageGrab.py*

#### Store raw images

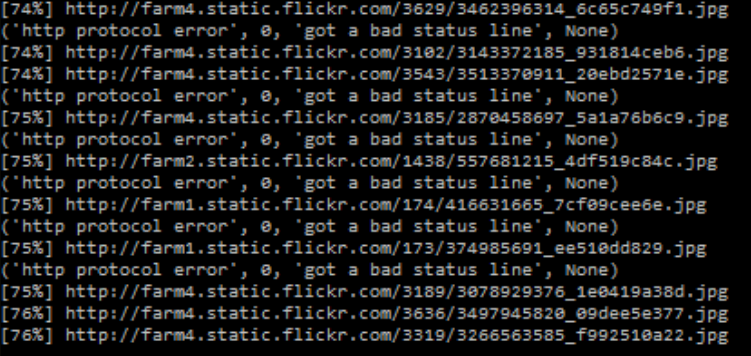
This option allows the user to download images from image net’s URL download. The user must enter the name of the folder where they wish to store the downloaded images, the URL of the webpage that shows the list of image URLs, then they are prompted whether they want to resize the images to a fixed size and whether they want to convert the images to grayscale. If they select to resize the images, they then enter the width and height the images should be converted to. After this, the script will then begin to download the images.

C:\Users\cmmuk_000\AppData\Local\Microsoft\Windows\INetCache\Content.Word\1folderName.png

C:\Users\cmmuk_000\AppData\Local\Microsoft\Windows\INetCache\Content.Word\1enterUrl.png

C:\Users\cmmuk_000\AppData\Local\Microsoft\Windows\INetCache\Content.Word\1imageTransform.png

*Fig 5.5. Store raw images.*



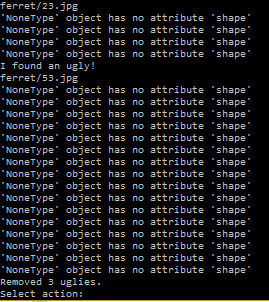
*Fig 5.6. Downloading images*

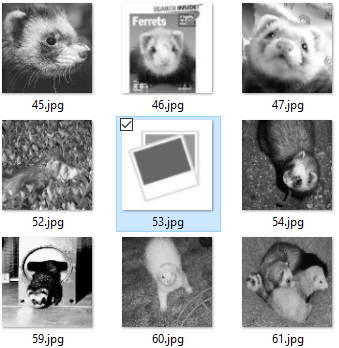
#### Find Uglies

This option allows the user to remove any image that may not be an actual member of the dataset. For instance, a site may no longer have the image that is being requested, but instead have a placeholder image. This image may not be part of the dataset and as such introduce noise to the dataset. In order to remove these type of images, the user needs to have a sample of the image(s) they wish to remove from the dataset. They must place all the images they wish to remove from the dataset in a folder named ‘uglies’ and this folder must be in the same location as the script. The user then only needs to enter the name of the folder they want to remove the unwanted images from.

C:\Users\cmmuk_000\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2remUglies.png

*Fig 5.7 Find Uglies*

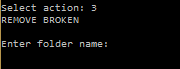




*Fig. 5.7.1 Placeholder image (53.jpg)*

#### Remove Broken

Similar to the ‘Find uglies’ option, this option searches through the downloaded images for images that may not have downloaded correctly and are corrupted. The user only needs to specify which folder they want to find and remove broken images.

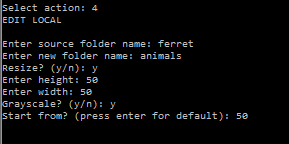


C:\Users\cmmuk_000\AppData\Local\Microsoft\Windows\INetCache\Content.Word\3remBrokeDone.png

*Fig 5.8 Remove broken*

#### Edit local

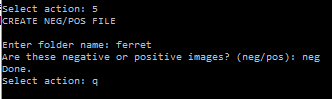
This option allows the user to resize and or change to grayscale images that have already been downloaded. This can be useful if the user needs to perform these changes on a dataset that has already been downloaded or if they wish to have a copy of the original dataset as well as the edited one. For this option, the user must enter the name of the folder which hold the images the wish to edit as well as the name of the folder they want to save the edited images to. If they wish to overwrite the original images, they should enter the same name as the source folder in the destination folder. Aside from being prompted whether they want to resize or convert to grayscale, they are also prompted to specify the starting point of the image names. Leaving this blank or entering 1 will mean that the images will be saved incrementally starting with 1.jpg, then 2.jpg and so forth. Entering any other number will start the image names at that position, i.e. if 40 is selected the images will be saved as 40.jpg, 41.jpg and so on. This can be useful when the user wants to combine images from multiple sources to one destination without overwriting previously edited images.



*Fig 5.9 Edit Local*

#### Create pos and neg file

This option is focused for the Haar Cascade training process. It creates a file that contains information about the images in the dataset. More details about this file will be explained in the section on training the Haar Cascade. The user must enter the name of the folder and whether the images in the folder are negative (background) or positive images. Positive images are the images that contain the object (class) the Haar Cascade must be trained on and negative images are images that do not contain the object (class) to be trained.





*Fig 5.10 Create neg file*

#### Show options

This option displays the list of commands

#### Exit

Ends the script.

### Training the Haar Cascade

Various classes of objects of objects were trained, including a tablet pc, a plush keychain, a cartoon character’s face, a person’s face and four navigation signs specifically, stop, go, left and right. In order to train the Haar Cascade, the following steps must be carried out:

* Collect “negative” or “background” images.
* Collect or create “positive” images.
* Create a positive vector file by stitching together the positives
* Train the Cascade.

#### Collecting the negative images

These are the images downloaded from imageNet.org. Once the images were downloaded and transformed (i.e. resized and changed to grayscale) a negative image file named bg.txt was created. This file basically contains information about the location of each image in the negative image folder.

Some notes on the negative images:

* Image size: resized to 100x100
* Image color: converted to grayscale

#### Collecting or creating the positive images

The images can either be collected or created. In the case of them being collected from imageNet.org for instance, a positive file would have to be created. Similarly, to the negative images this file would contain information about the location of each image in the folder but will also include the pixel location of the object to be trained on in the image. This can be a very tedious process as they may have to be done manually for each image in the folder. This can be simplified if the positive images only contain the object to be trained and if they cover the entire image. In this research this was achieved by collecting at least one positive image that covered the entire image and superimposing it onto background images to create a large positive image set. The drawback to using this method is that the system will be trained to recognize only that object, e.g. if trained to detect a tablet pc, it will only recognize the specific tablet pc. However, in cases specificity is required, this would be an advantage.

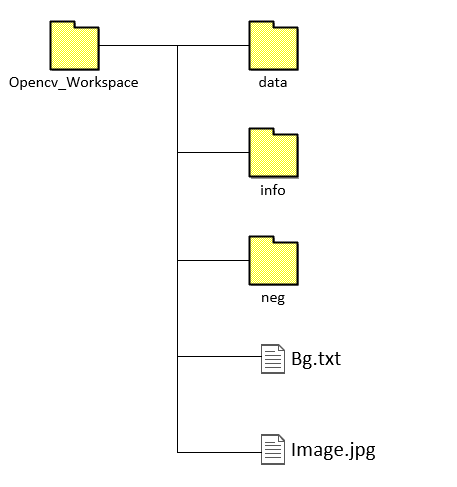
Notes on the positive images:

* Image size: resized to 50x50
* Image color: BGR

#### Creating a positive vector file

Once all the files were ready, a vector file is needed to join the positive images together for the training process. This was done on a linux machine that had opencv installed on it. The following shows the file structure and commands that were run to create the positive vector files.

File Structure



*Fig 5.11 File structure of cascade workspace*

Data – This is the directory that stores the output of the training process which are the cascade files.

Info – This directory contains the positive images that are created by superimposing a single positive image on a background image in random locations as well as creating the positive file ‘info.lst’ that contains the location of the image and the position of the object in the image.

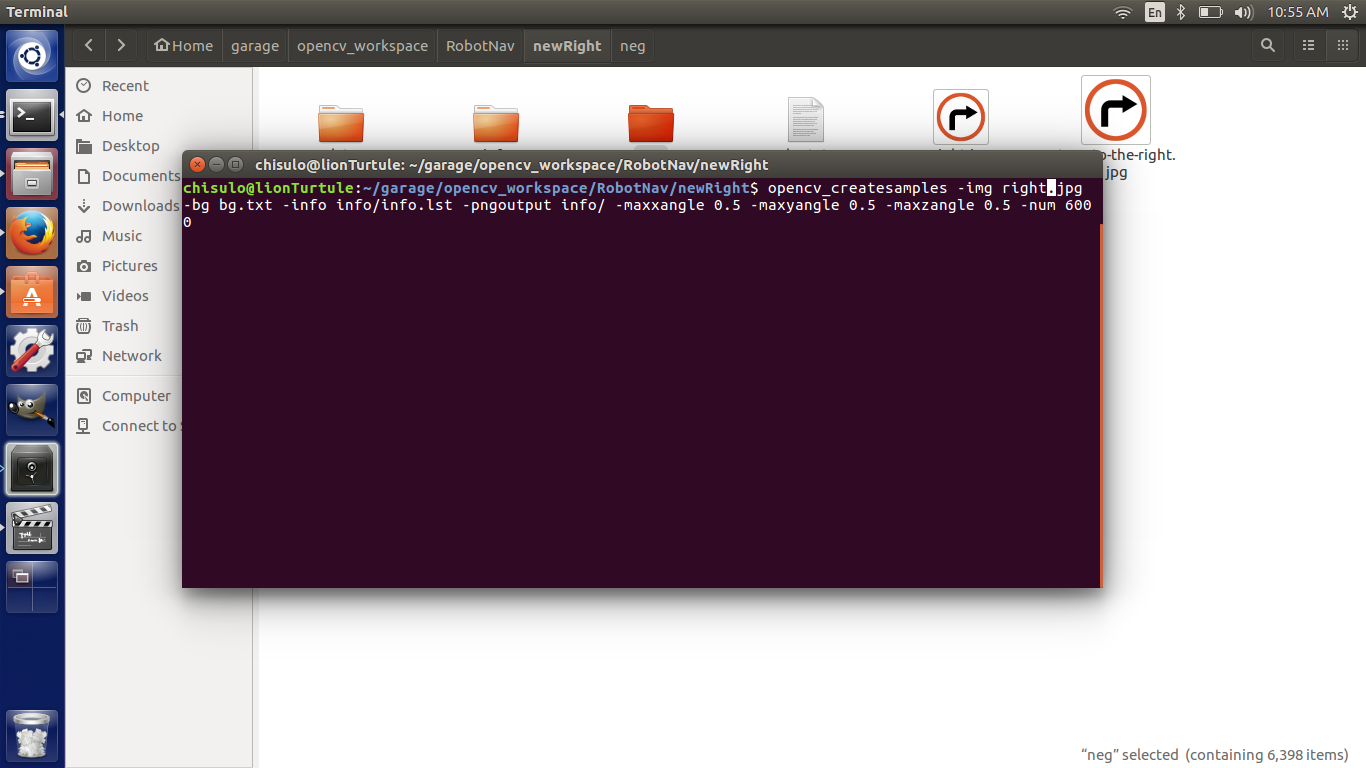
Neg – This directory contains all the negative images to be used for training the cascade.

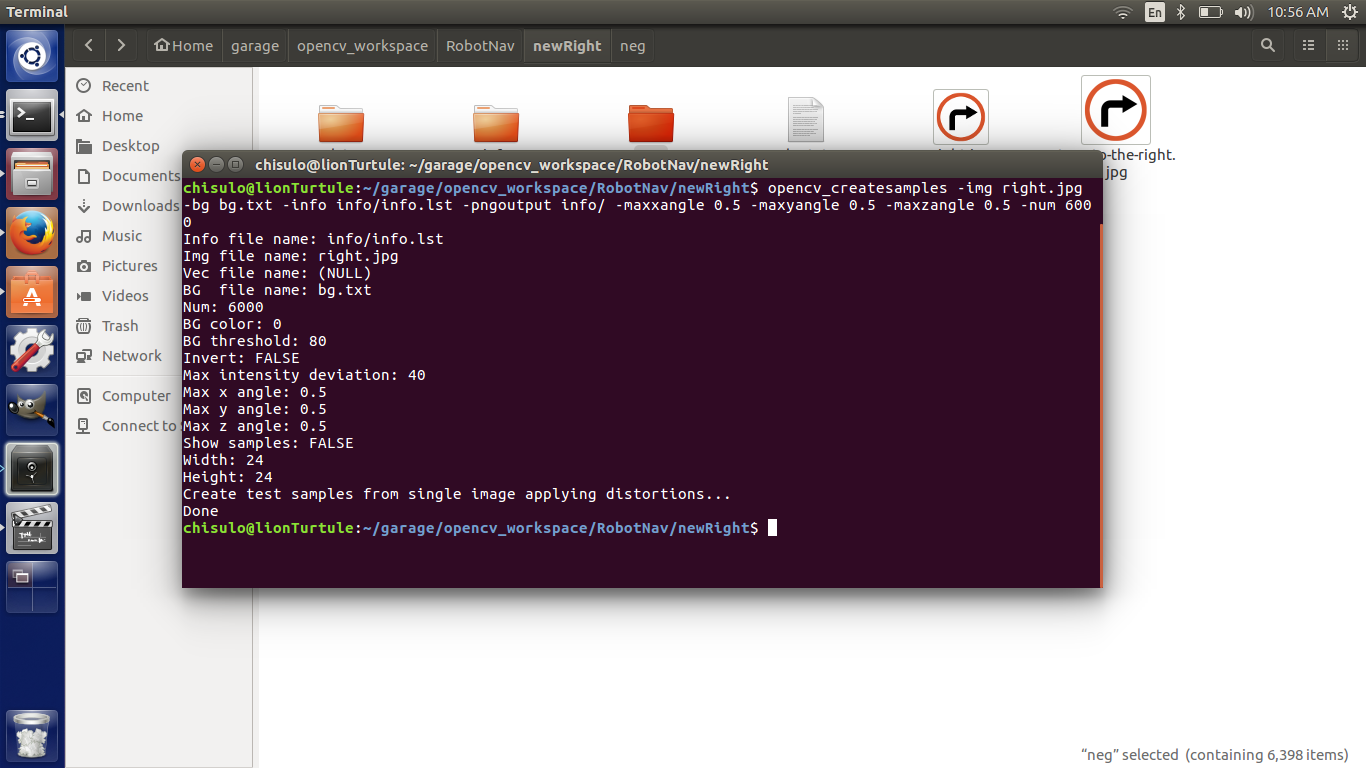
Bg.txt – This is the file that contains the location of the negative images.

Image.jpg – This is the single positive image that the cascade will be trained to detect.

Commands

The following commands were used to: (1) Create a positive image set by superimposing the positive image on background images in random locations, (2) create the positive vector file.





*Fig 5.12 Create Samples: creates positive image dataset*

The opencv\_createsamples takes the following parameters:

-img – the location of the positive image.

-bg – the location of the negative image file.

-info – the location to store the positive image file.

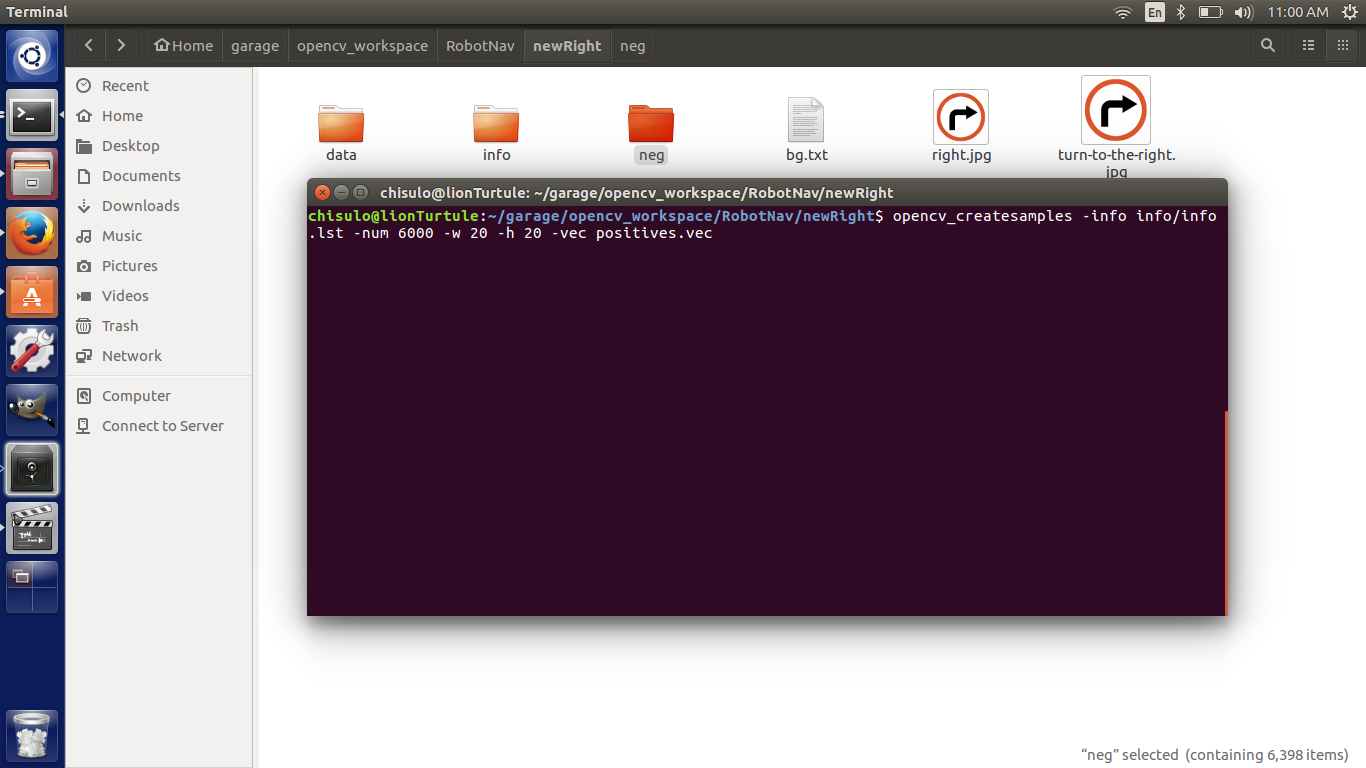
-pngoutput – the location to store the positive image dataset.

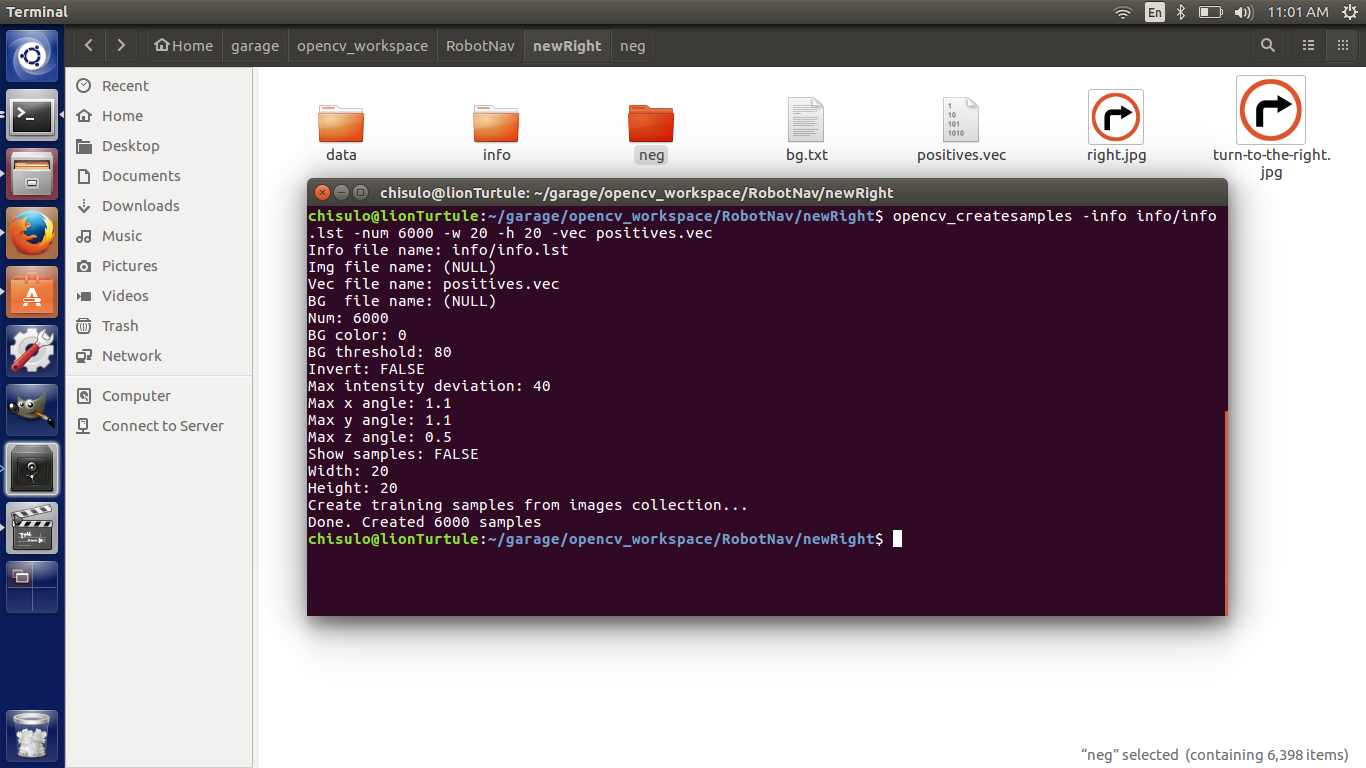
-maxxangle – the maximum degree the image should be transformed with respect to the x-axis.

-maxyangle – the maximum degree the image should be transformed with respect to the y-axis.

-maxzangle – the maximum degree the image should be transformed with respect to the z-axis.

-num – the number of images to be used in the training





*Fig 5.13 Create Samples: Creates the positive vector file*

The second opencv\_createsamples command is used to create the positive vector file and takes the following arguments:

-info – the location of the positive image file.

-num – the number of images being used in the training.

-w – The width of detection area.

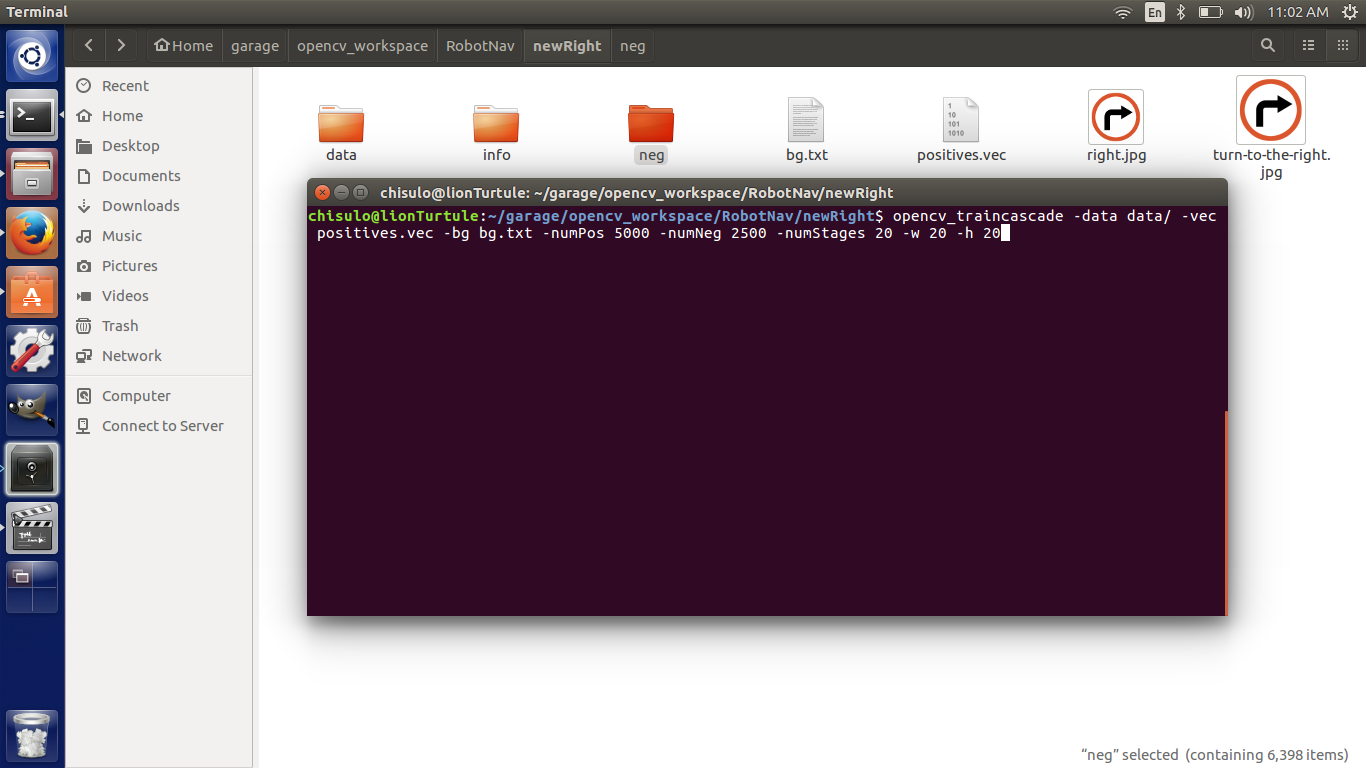
-h – The height of the detection area.

-vec – The location of the vector file.

Note on detection width and height: The larger the size, the larger the training time.

#### Train the Cascade

Once all the files are ready, the cascade can be trained. The images below show the training process. During this time, the time taken, Hit and Miss rate were recorded.



*Fig 5.14 Train Cascade command*

The opencv\_traincascade takes the following arguments:

-data – the location that the cascades should be stored when training is complete.

-vec – the location of the vector file.

-bg – the location of the negative image file

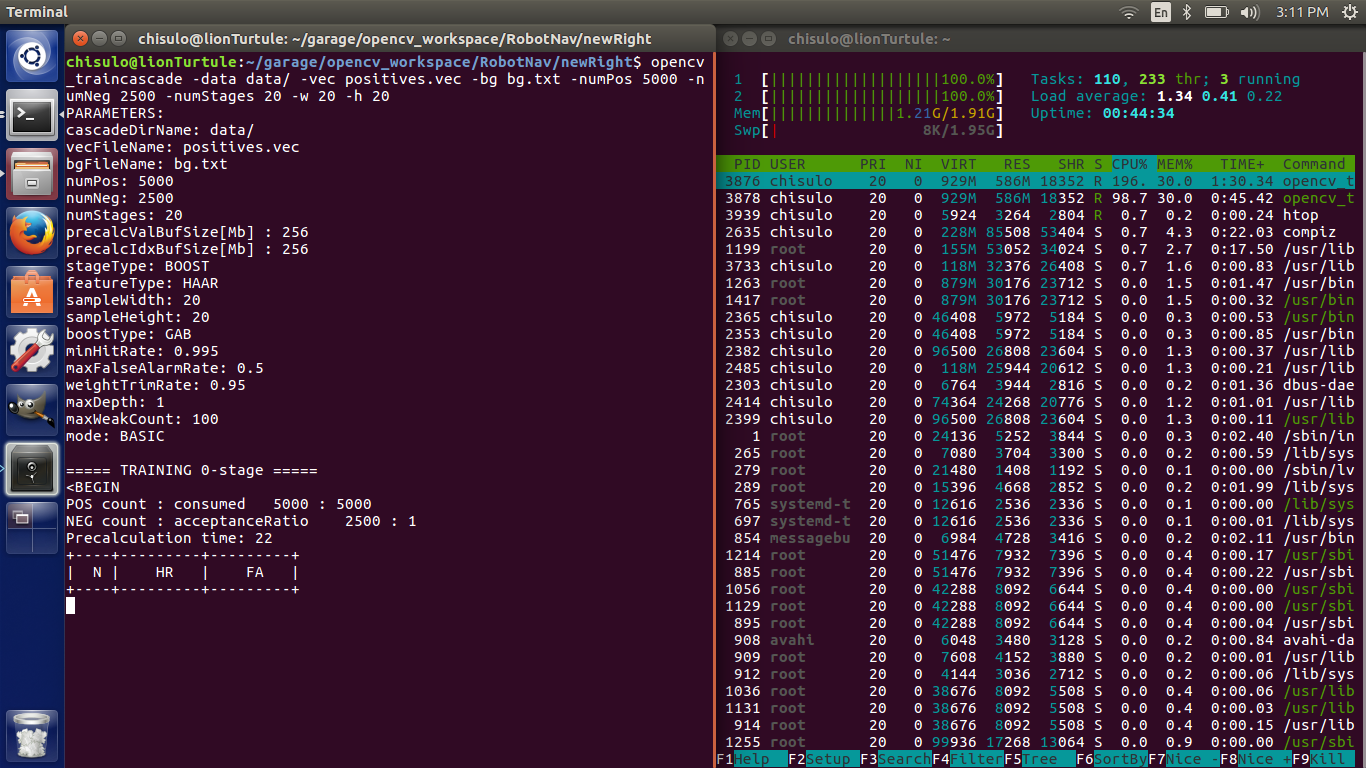
-numPos – Number of positive images that should be used in the training.

-numNeg – Number of negative images that should be used in the training.

-numStages – The number of training stages that should be done (from 0 – 19).

-w – The width of the detection area.

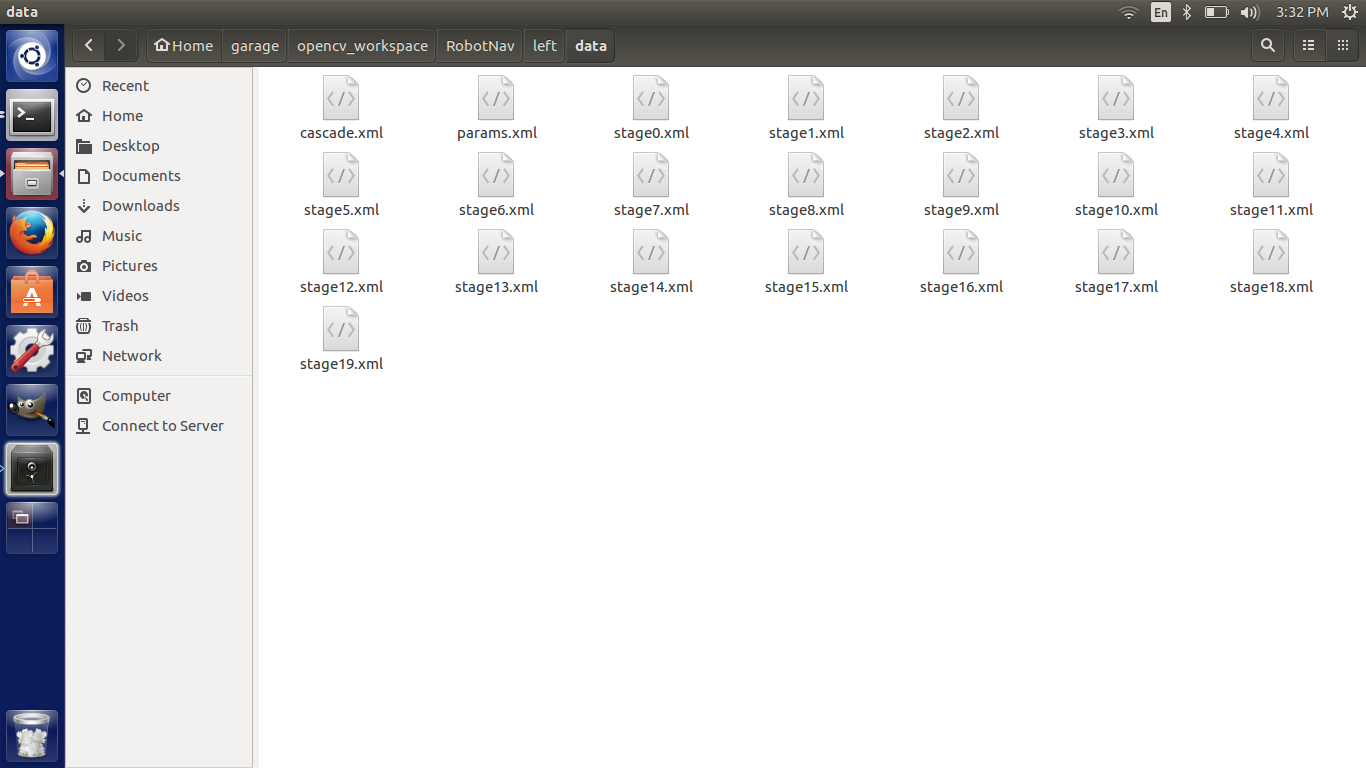
-h – The height of the detection area.



*Fig 5.15 Train Cascade*

The training begins by displaying the various parameters that will be used in the training process as well as information about the specific training stage, including the number of iterations, the Hit rate and the False Alarm or Miss rate which have been denoted as N, HR and FA respectively.

Once the training process is complete the cascade file is saved to location specified in .xml format.



*Fig 5.16 Cascade file*

### Retraining the Neural Network

The following requirements were needed to use the inception model:

* TensorFlow: an open source library for numerical computation, specializing in machine learning applications. (Google Inc, 2017)

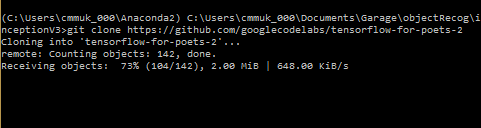
The following steps were carried out in order to retrain the Neural Network:

* Download the inception Network
* Collect and label dataset
* Retrain the network using the data collected.

#### Download the Inception Network

The inception network is a deep convolutional neural network architecture that achieves the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (Szegedy, et al., 2015).

It was cloned from the git repository <https://github.com/googlecodelabs/tensorflow-for-poets-2> .

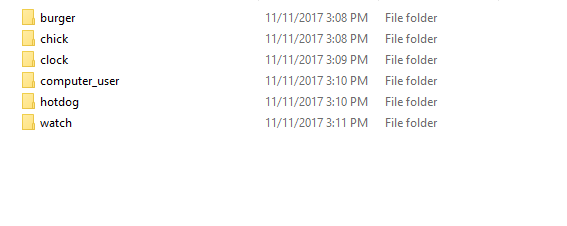


*Fig 5.17 Cloning git repository*

#### Collect and label dataset

The next step was to get a dataset of objects which were already acquired from imagenet.org. Once the images are downloaded. The folders they are in must be named as the name of the class of objects one wishes to train the model on e.g. different types of vehicles, such as truck, bus, minivan etc.

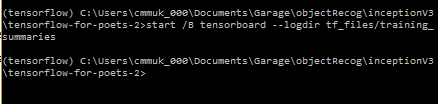
The dataset used in this training were the following classes of objects: burger, chick, clock, person using a computer, hotdog, and watch.



*Fig 5.18 Dataset classes*

#### Retrain the network

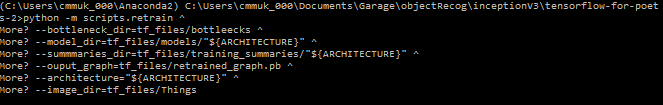
In order to monitor the training process, Tensorflow comes with a monitoring program called tensorboard, that tracks the process and plots it on a graph as well as provide other monitoring functions. To start tensorboard, run this command.



*Fig 5.19 Starting tensorboard*

The ‘start /B’ allow the command to run in the background on a windows machine. A similar effect can be achieved by using the ‘&’ sign at the end of the command.

After this, the retraining can begin by running the retraining script located in the repository.



*Fig 5.20 retrain parameters*

The retrain script takes several parameters. These are:

--bottleneck\_dir – This is the directory in which the bottlenecks are stored.

--model\_dir – This is the directory in which the model is stored.

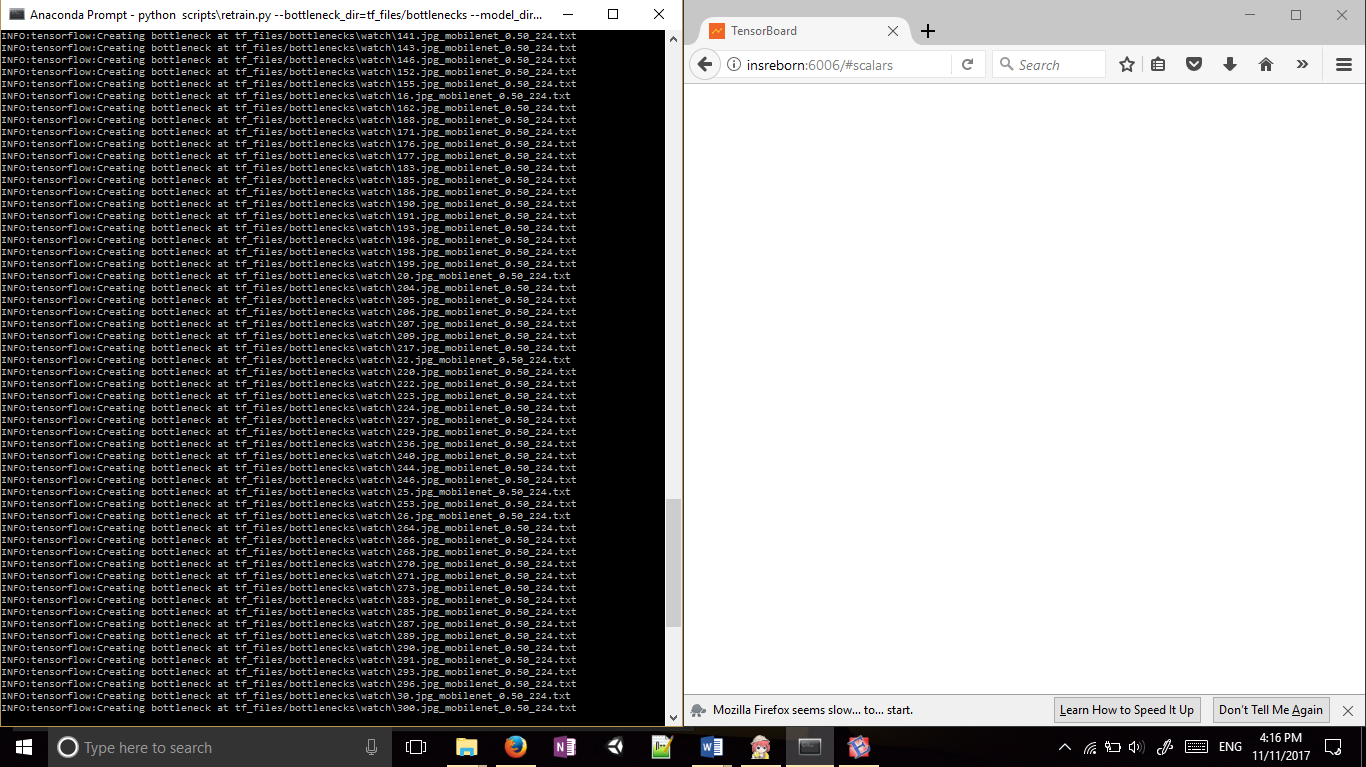
--summaries\_dir - This is the directory in which the training summaries are stored.

--output\_graph – This is the name of the graph that will be created when the training is complete.

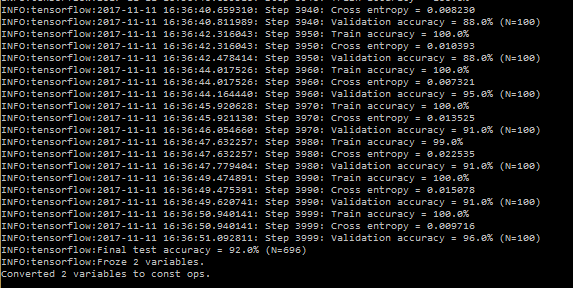
--output\_labels – This the name of the label file that will be created once the training is complete. It is a text file that stores the names of each of the classes.

--architecure – This is the architecture that is used to train the system.

--image\_dir – This is the directory that holds the image dataset.

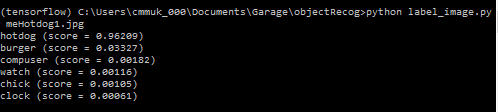


*Fig 5.21 Training Beginning: Creating Bottlenecks*



*Fig 5.22 Training complete: Final test accuracy = 92%*

Once the training is complete, the created model can be tested by using the label\_image.py script, which takes the picture you wish to test as a parameter.



*Fig 5.23 Testing the network*

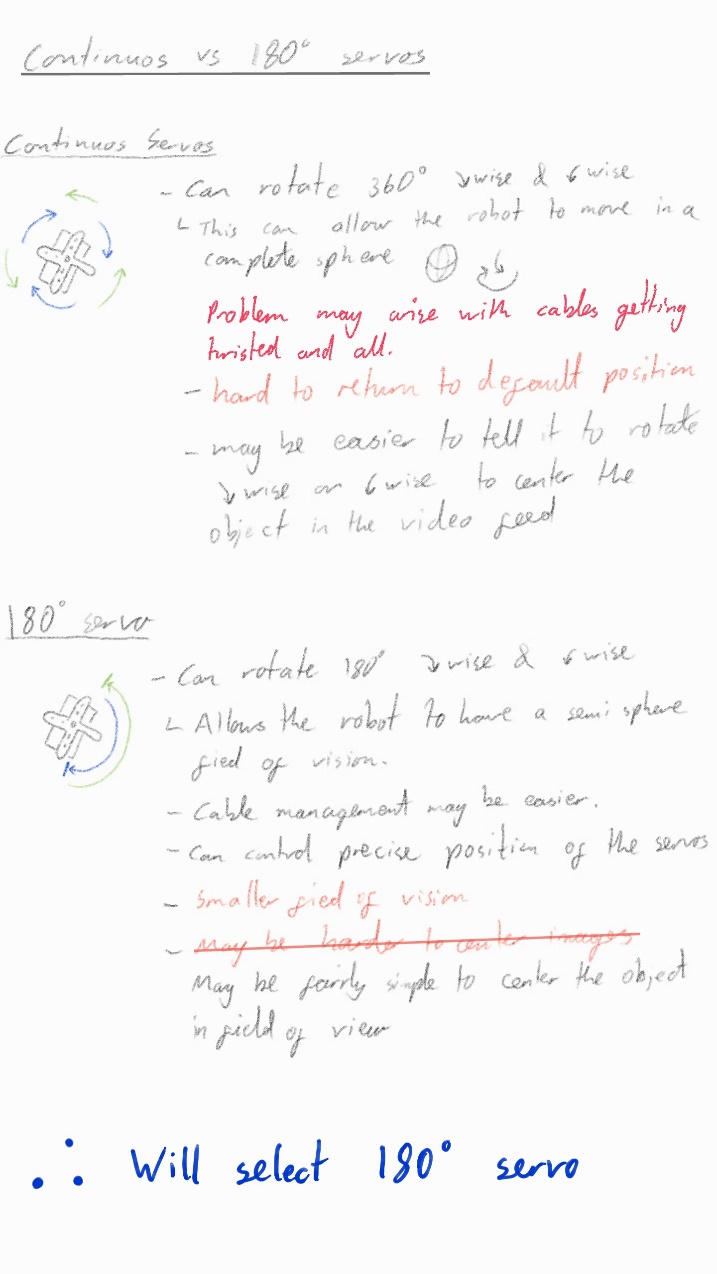
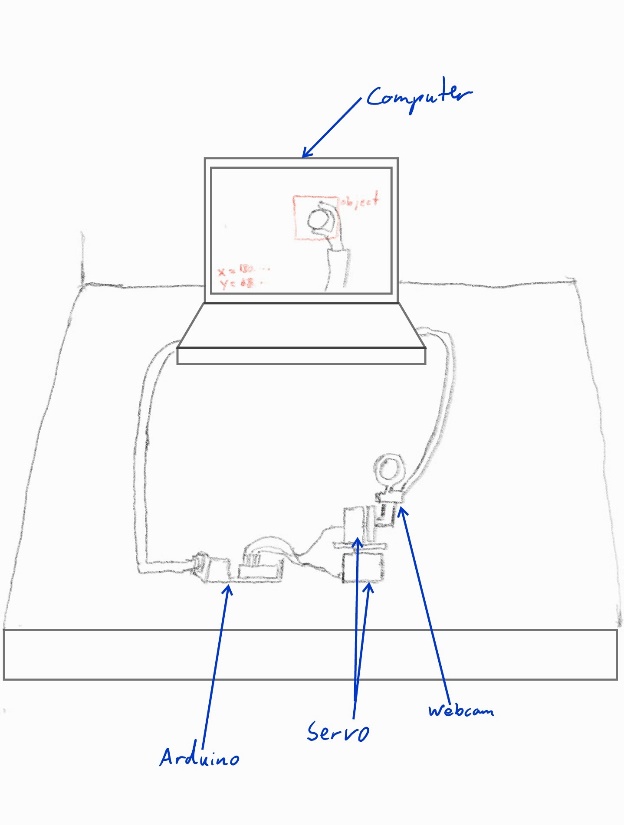
## Testing

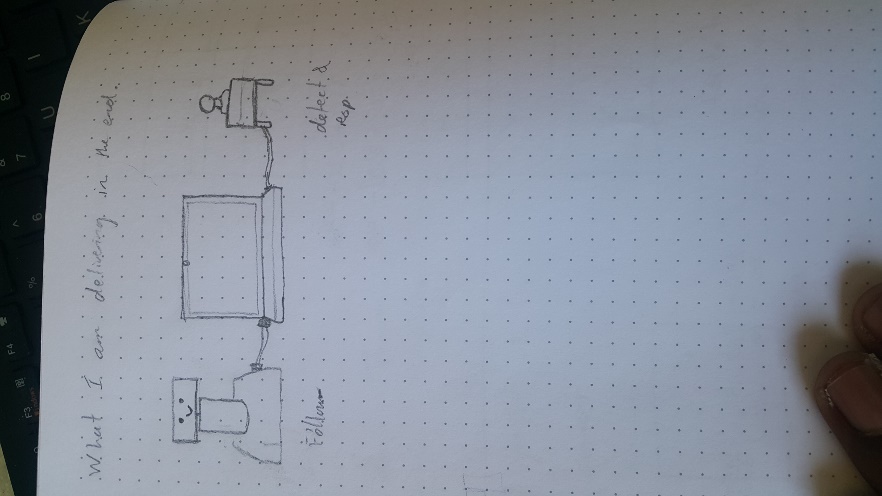
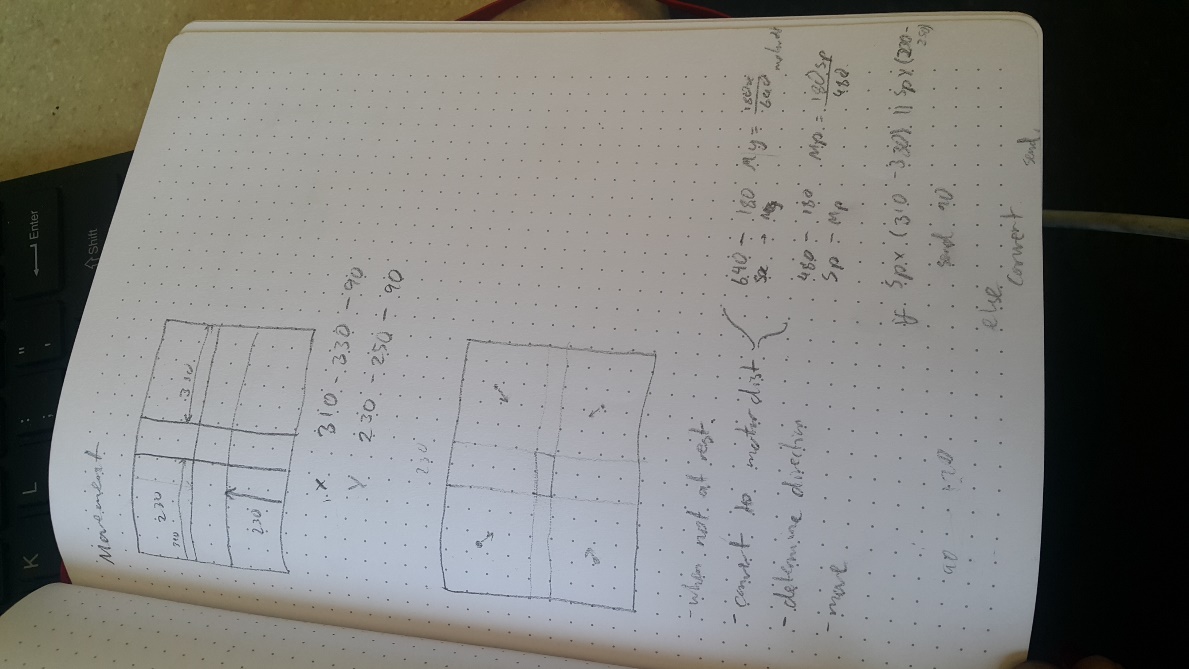
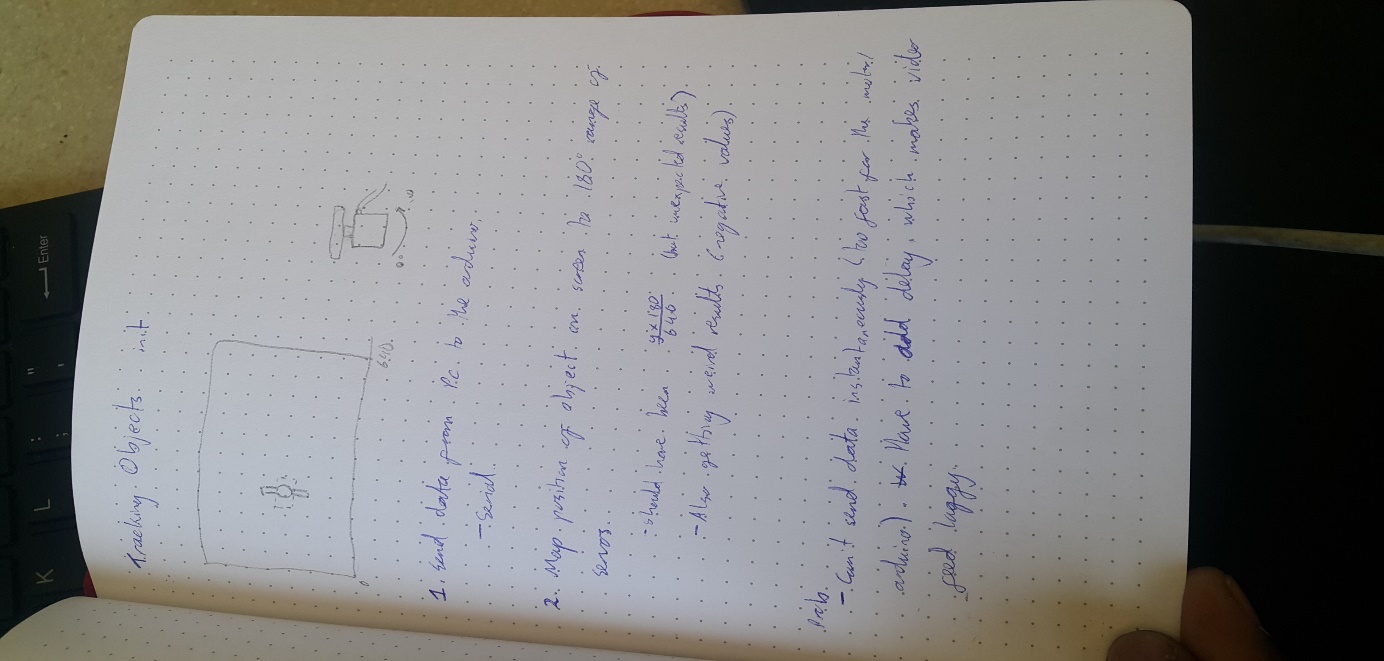
The testing of these systems involved determining how well they were able to detect/classify the objects they were trained on. For both systems, object detection and classification were measured by giving each system a set of test images, and the number of correct detections/classifications were recorded. Video data was also used in testing the Haar cascade, to measure how well the robot can track the image as it moves.

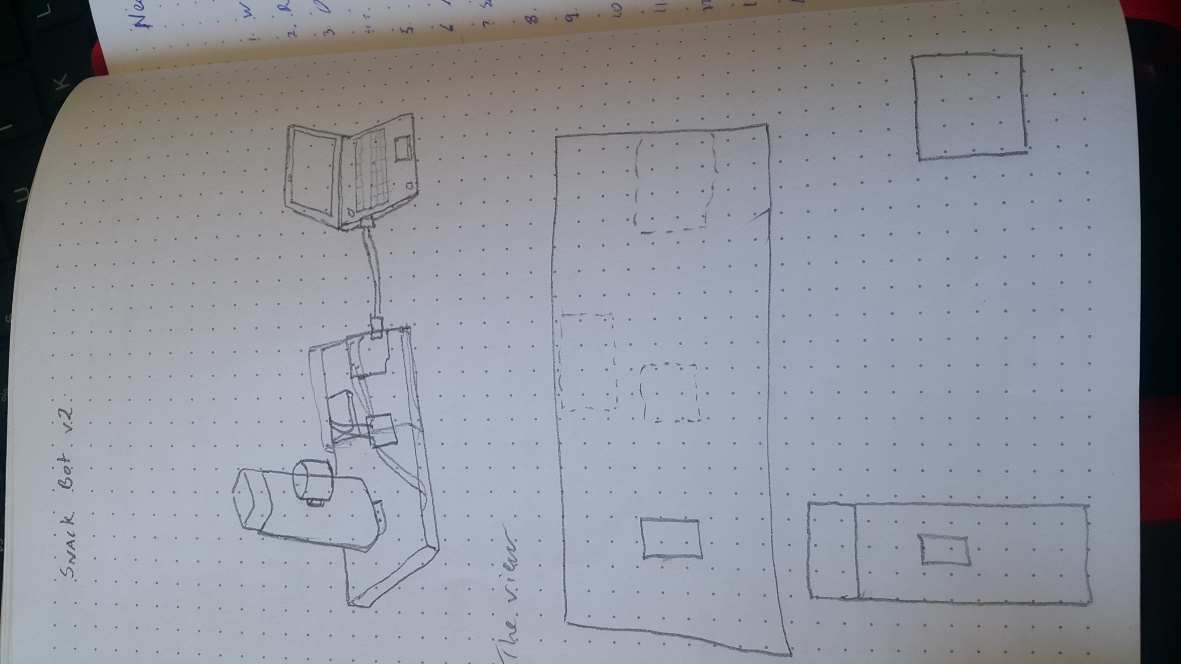
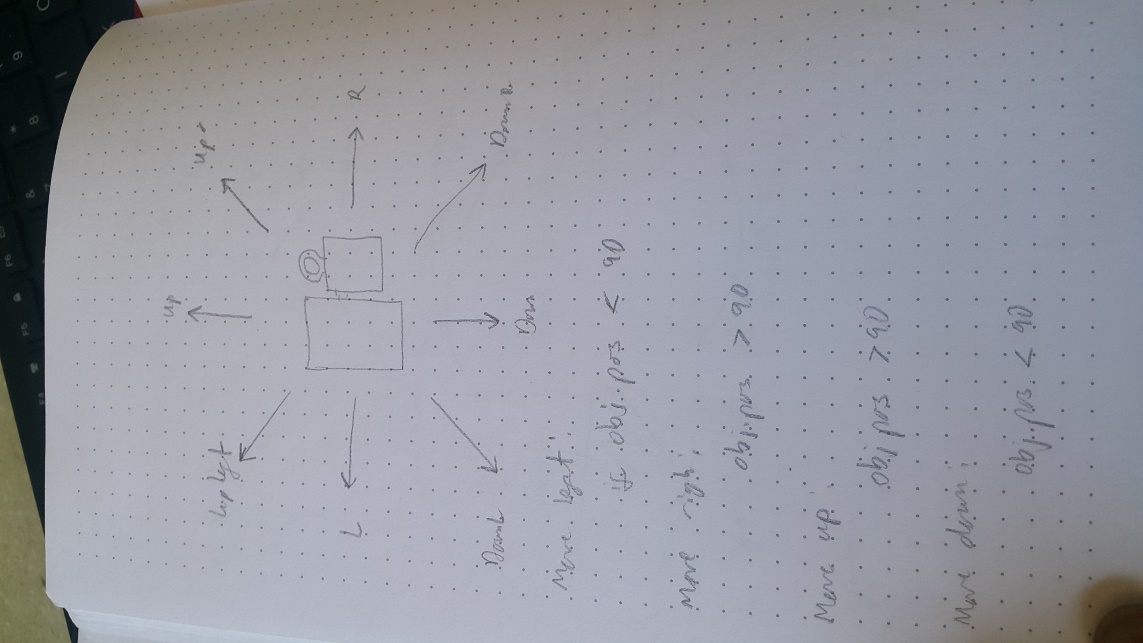
Further details on the training stage will be discussed in the following chapter.

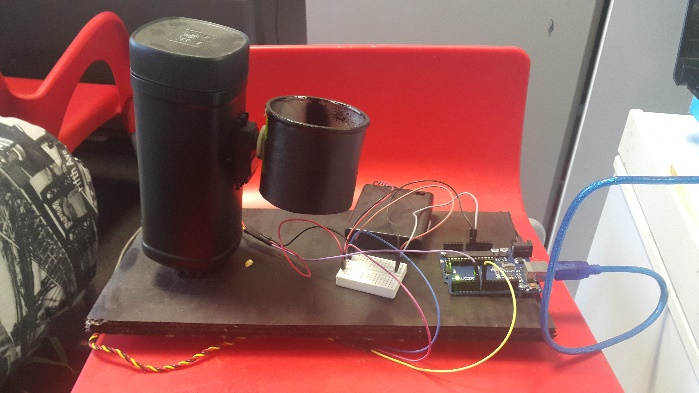
# Design and Creation

Apart from experiments, several design and creation principles were also adopted for the creation of the object tracking robot dubbed the Snack Bot and the object recognition robot dubbed Road Hog. In this process, an iterative process of suggestion, development and evaluate was carried out to improve the function and design of the robot. Provided below are sketches and images of the development of the robot.





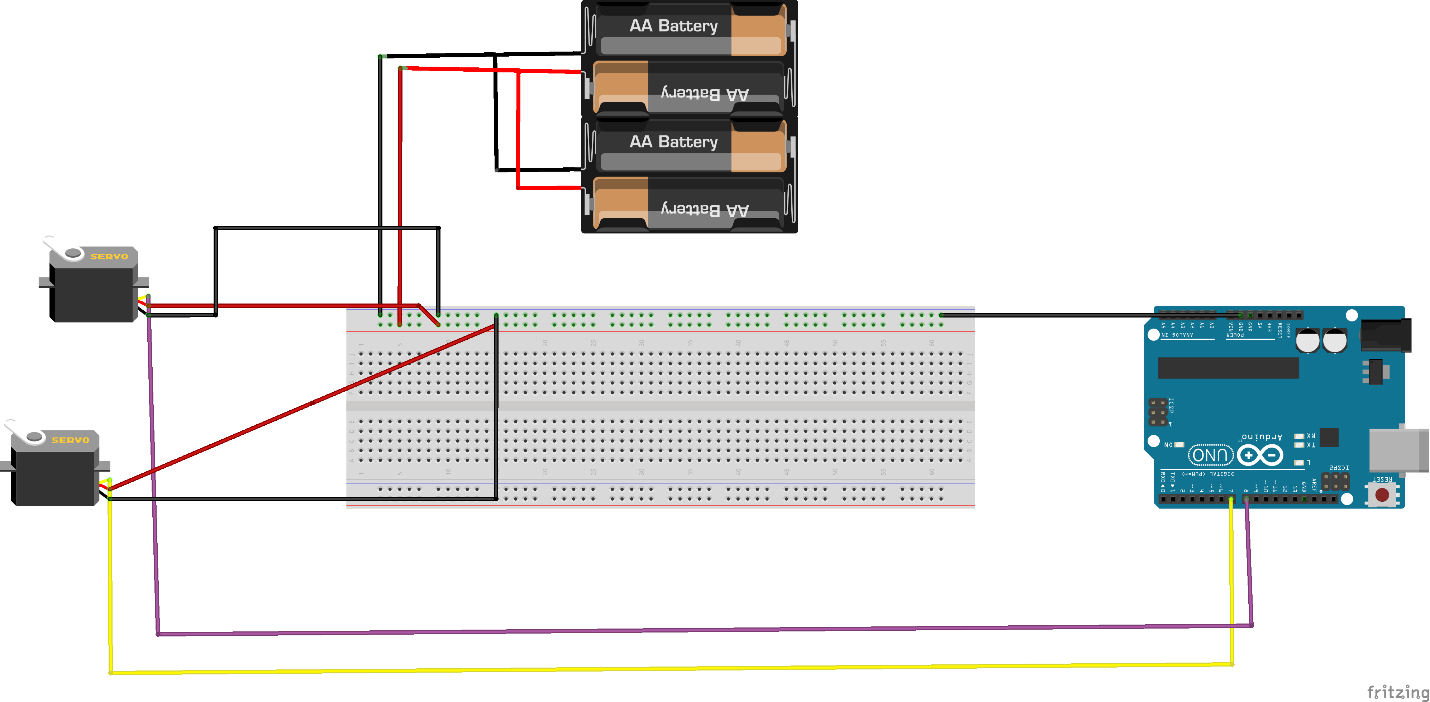




*Fig 5.24 Sketches and images of the robot*

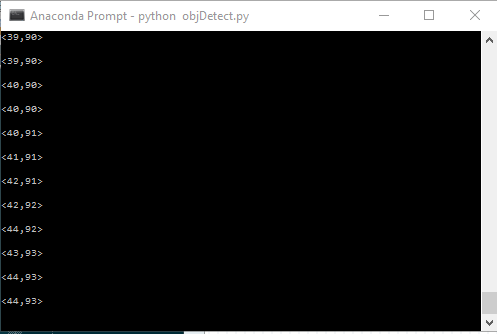
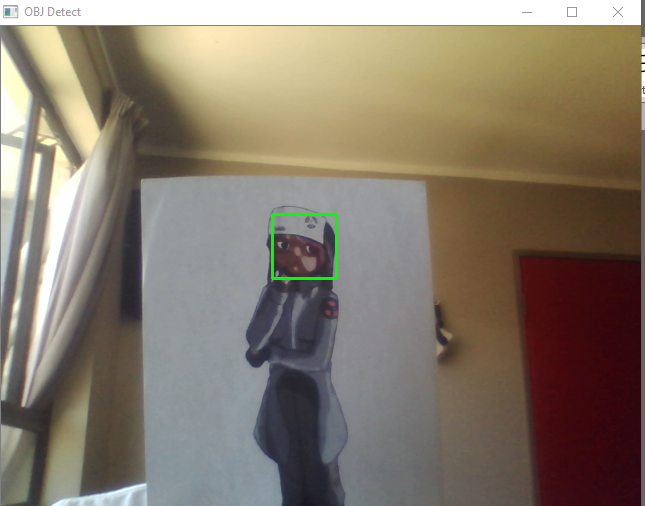
## How the Robot Works

The Snack Bot robot consists of an Arduino Uno microcontroller that is attached to two positional rotation servo motors that rotate on two axes, namely x and y axis respectively. It is connected to a computer through a serial cable which both power the microcontroller and send information between them. This architecture is illustrated below.



*Fig 5.25 Fritzing diagram*

The computer runs the python program that displays a view from the connected web camera and is able to detect objects trained using the Haar Cascade method. It draws a rectangle around the object and sends the coordinates of the object on the screen to the robot over the serial connection. Before the coordinates are sent, it will determine what location of the screen the object is in (upper-left, lower-right etc.) and convert it to coordinates the motors should move to (0-179 degrees).



*Fig 5.26 Object detected (left) and coordinates of the object (right).*

The Arduino runs a program that reads the data sent from the computer and translates the information to move the servo motors into the positions it reads.

Chapter 6: Research Findings and Discussions

This section will look at the findings of the training and testing process of both Haar cascade and Neural Network techniques, as well as look at the performance of the object tracking robot.

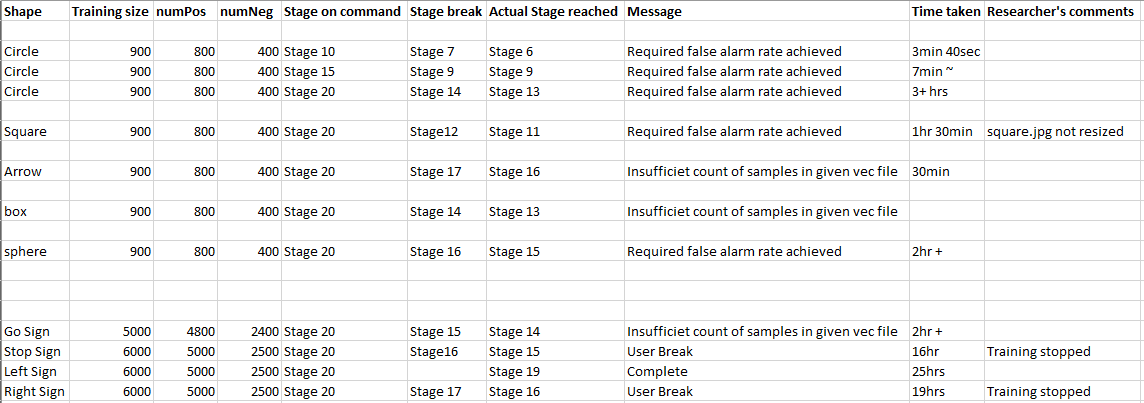
# Haar Cascade Findings

The results from the training and testing of the haar cascades are shown below. The following are a list of the cascades that were trained during this research:

* Tablet pc
* Face:
  + Human face
  + Cartoon face
* Plush key chain
* Navigation signs:
  + Right sign
  + Left sign
  + Stop
  + Go
* Geometric shapes:
  + Circle
  + Square
  + Arrow
  + Sphere
* Image of a cardboard box

## Training results

This is the data recorded on the training process of some of the objects in the training set.



*Fig 6.1. data recorded on Haar Cascade training*

Each column is described as follows:

Shape – This is the object that is being trained to detect.

Training size – This is the number of images being used in the training.

numPos – This is the number of positive images.

numNeg – This is the number of negative images

stage on command – This is the training stage that was set when beginning the training.

Stage break – The stage at which the training stopped.

Actual Stage Reached – last stage completed.

Message – Message given when training stopped

Time taken – Time taken to train.

Researcher’s comments – Any comments noted by researcher.

This table four main outcomes:

* Training stops due to false alarm rate achieved.
* Training stops due to insufficient count of samples in the given vec file.
* Training stop due to user interrupt and
* Training completed.

## Training graphs

The following graphs show the Hit rate (HR) and Miss Rate (MR) of the cascades during training. The HR are the rate of correct detections and must aim to be as high as possible out of a total of 1 while the MR must be as low as possible out 1. The graphs shown are for the navigation objects, namely the left, right, stop and go signs.

Notes on graphs:

* Omitted stages are due to incomplete or missing data.
* Precise data recordings have been attached in the ‘training graphs’ excel sheet.

### Go Sign

### Stop Sign

### Left Sign

### Right Cascade

*Fig 6.2 Graphs of training the cascades*

The following results were found after analysis of the graphs:

* The hit rate remains in the 0.9 range throughout the training, meaning the number of correct predictions remains high.
* The accuracy then depends on the Miss Rate or False Alarm Rate. The lower it becomes the more accurate the cascade is.
* The miss rates generally drop to ranges between 0.48 and 0.38.
* The number of iterations that are made in the training stages generally increase the higher the stage being trained.

# Neural Network Findings

Two different sets of objects were trained using the neural network. These are the ‘Things’ dataset consisting of the following objects:

* Burger
* Hotdog
* Watch
* Clock
* Chick
* Computer user

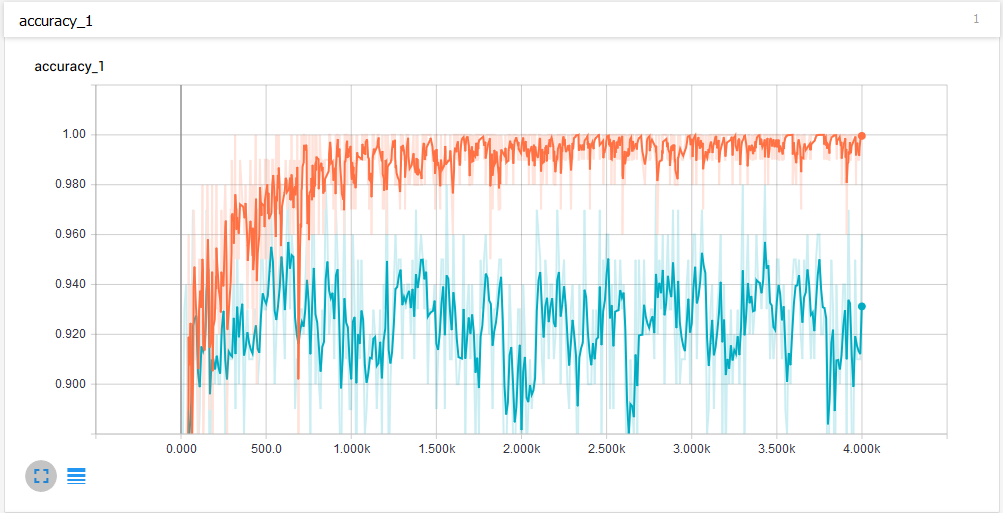
And the ‘People’ dataset which includes:

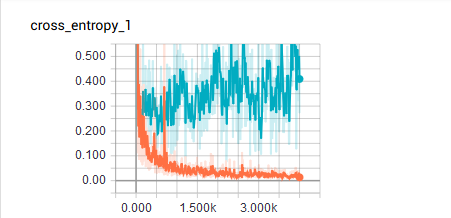
* Boy
* Girl
* Middle aged man
* Old man
* Old woman
* Young man
* Young woman

The ‘Things’ dataset produced a better model as compared to the ‘People’ dataset. This could be due to various reasons including:

* Smaller datasets for some of the ‘People’ classes
* Mixed classes (i.e. images of women present in girls’ folder)
* Similar classes (e.g. girl, woman, old woman)

For these reasons further data recording was carried out for the ‘Things’ dataset. The following graphs show different parameters recorded by tensorboard during the training of the neural network.



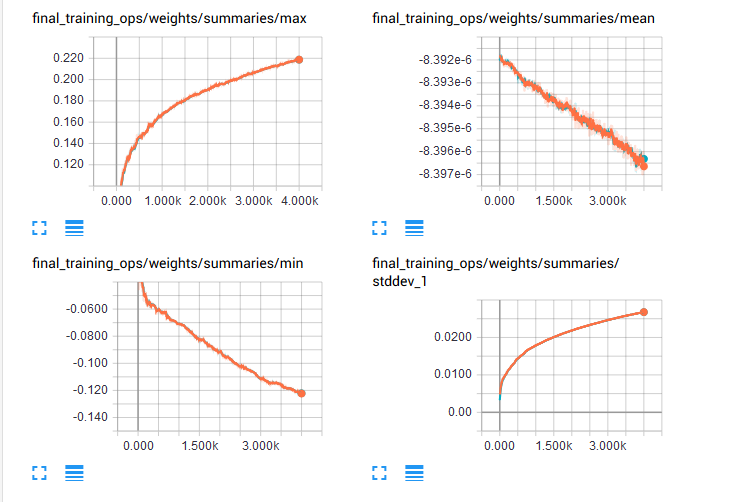
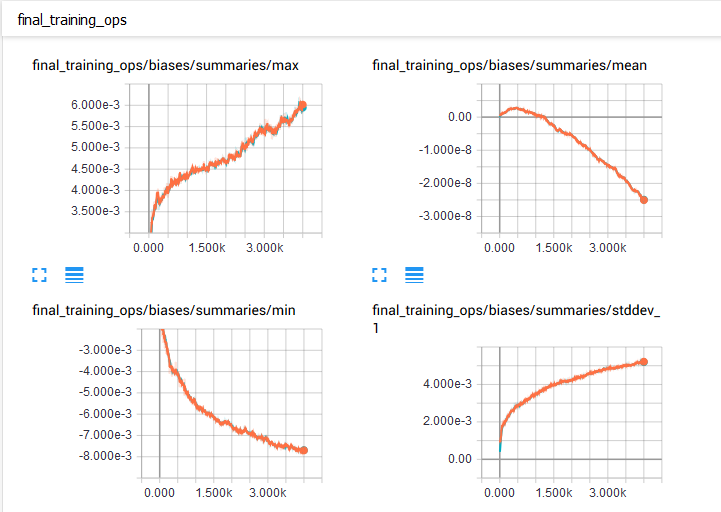


*Fig 6.3 Graphs of accuracy and cross entropy of neural network being trained*

In the two graphs above, the first graph labelled accuracy is similar to the Hit rate discussed in the training of the Haar cascades. The training accuracy (red line) shows the percentage of the images used in the current training batch that were labelled with the correct class. The validation accuracy (labelled blue) is the precision (percentage of correctly-labelled images) on a randomly-selected group of images from a different set. (higher numbers are better).

The cross\_entropy\_1 graph shows the cross entropy which is a loss function that gives an idea into how well the learning process is progressing (lower numbers are better).

The graphs below provide a visualization for the biases and the weights



*Fig 6.4 Additional graphs of the neural network training*

The neural network had a final accuracy of 0.92 or 92%. It was then tested on a number of test classes in which it was able to classify all of them correctly.

Chapter 7: Summary and Conclusion

In conclusion, the research was aimed at exploring different machine learning techniques of implementing object detection and classification, to compare them in terms of training and testing, their strengths and limitations, and provide sample applications.

# Training and Testing

For the sake of the conclusion, the ‘left’ navigation cascade shall be used to illustrate the Haar cascade method, and the neural network trained to classify the ‘Things’ dataset shall be used as it was the better model.

In terms of training, the Neural network was faster to train. However, it is important to note that the model was not built from scratch, but only had the final layer retrained. The model was retrained on a total of 3,454 images of different classes and was able to complete the training in under an hour (approximately 40 minutes) with a test accuracy of 92.0%. Haar cascade however can be trained at different stages, and generally the higher the stage, the more accurate the cascade. The left cascade completed all 20 stages of training, this however took over a day to train (25 hours), reaching a Hit Rate (HR) of 0.9952 or 99.52% and a False Alarm rate (FA) of 0.4744 or 47.44%. It has been noted that different computers were used in the training of the two systems and that the Haar cascade was trained on a slower machine, however, the large difference in time taken by both systems shows that the particular neural network model is faster than the Haar Cascades. However, it is also worth noting that the time taken to train the lowest stage of the Haar cascade was about 10 minutes, which is faster than the neural network but comes at the cost of accuracy.

# Strengths and Limitations

The strengths and limitations have been into tabular form and make comparisons on the two techniques’ properties.

|  |  |  |
| --- | --- | --- |
| Property | Haar Cascade | Inception (Neural Net) |
| Cascade/ Model Size | Small (~100KB trained on 1 object) | Larger (~6MB trained on 6 objects) |
| Training time | Long | Short |
| Type | Object detection | Object Classification |
| Object Specific? | Yes | No |
| Detection/ Classification time | Instant | Slow (~30s) |
| Able to use binding boxes? | Yes | No |

It is also noted that because the inception network is a classification neural network, it cannot be used for object tracking using the robot.

Sample applications will be discussed in the next chapter

Chapter 8: Recommendations

In this section, we discuss the further development as well as different applications of use of these systems. The list given is by no not a definitive list but rather just a sample of applications that this technology can be applied to.

# Robot Navigation

As shown in this research, object detection is a useful technology that can be applied to robot navigation. It can enable them to see and respond to various objects in its environment.

The robots themselves can be applied to various applications e.g.

* Entertainment such as autonomous racing robots
* Agriculture as monitoring or quality control device.
* Logistics as a delivery drone.
* Etc.

# Object-focused video Recording

This is a concept in which camera is mounted upon a set of motors or a drone and is able to detect a particular object, such that it is always at the center of focus even though the object moves. This can be a useful tool in video recording, documenting whether for recreation or professional use.

# Surveillance and security

A typical application if for video surveillance and security. The security cameras can be used for intruder-detection or facial recognition to ensure that unwanted people or objects do not breach an area

# Alternative input device

As described in the research paper by Utaminingrum et al. on using eye movement as a navigation method for disabled people, or using hand gestures as a form of input that can be used where the conventional keyboard and mouse would not be appropriate.

# Taxonomy

This can be used by researchers, biologists, students or any interested person, to classify different species of animals. This can be used of educational and recreational purposes.

# Quality Control

As mentioned earlier, this can be used for quality control devices in factories, companies etc. It can help ensure that damaged products are not used or to sort different products into different categories e.g. different sized eggs or different flavored drinks etc.

# Translation Application

This can also be used in a translation application, that can take a picture of an object and give the translation of the object as well as sample sentence in which the object can be used.

# References

*Face Detection using Haar Cascades.* (n.d.). Retrieved from opencv.org: http://docs.opencv.org/trunk/d7/d8b/tutorial\_py\_face\_detection.html

Gagan. (2012, October 14). *Explaining Haar Cascade.* Retrieved from www.singhgaganpreet.wordpress.com: https://singhgaganpreet.wordpress.com/tag/explaining-haar-cascade/

Google Inc. (2017, April). *TensorFlow For Poets*. Retrieved from Google Codelabs: https://codelabs.developers.google.com/codelabs/tensorflow-for-poets/?utm\_campaign=chrome\_series\_machinelearning\_063016&utm\_source=gdev&utm\_medium=yt-desc#0

Jha, G. K. (2007). Artificial Neural Networks and its Applications. 1.

Karpathy, A. (n.d.). *CS231n Convolutional Neural Networks for Visual Recognition*. Retrieved from CS231n Convolutional Neural Networks for Visual Recognition: http://cs231n.github.io/convolutional-networks/

Nguyen, T. T., Pham, H. N., Dong, T. V., Nguyen, S. V., & Tran, T. T. (2011, October 13). A fully automatic hand gesture recognition system for human-robot interaction. *SoICT '11 Proceedings of the Second Symposium on Information and Communication Technology*, pp. 112-119.

Noor, H., Mirza, S. H., Sheikh, Y., Jain, A., & Shah, M. (2006). Model Generation for Video-based Object Recognition.

Russell, S., & Peter, N. (2010). Artificial Neural Networks. In S. Russell, & N. Peter, *Artificial Intelligence A modern Approach 3rd Ed.* (p. 728). New Jersey: Prentice Hall.

Singh, D., Dutta, M., & Singh, S. H. (2009, January 9). Neural network based handwritten hindi character recognition system. *COMPUTE '09 Proceedings of the 2nd Bangalore Annual Compute Conference*.

Smagt, P. P. (1990). A comparative study of neural network algorithms applied to optical character recognition. *IEA/AIE '90 Proceedings of the 3rd international conference on Industrial and engineering applications of artificial intelligence and expert systems - Volume 2*, pp. 1037-1044.

Stanford Vision Lab, Stanford university, Princeton University. (n.d.). *ImageNet*. Retrieved from ImageNet: www.image-net.org

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., . . . Rabinovich, A. (2015). Going Deeper with Convolutions. *Computer Vision Foundation*.

Utaminingrum, F., Fauzi, A. M., Sari, A. Y., Primaswara, R., & Adinugroho, S. (2016, December 16). Eye Movement as Navigator for Disabled Person. *ICCIS '16 Proceedings of the 2016 International Conference on Communication and Information Systems* , pp. 1-5.

Wang, R., & Xu, Z. (2015). A Pedestrian And Vehicle Rapid Identification Model Based On Convolutional Neural Network.

# Appendices

|  |  |
| --- | --- |
| **TITLE** | **DESCRIPTION** |
| Training cascade.xlsx | Excel sheet on training of the Haar Cascade |
| Training graphs.xlsx | Excel sheet showing the graphs of the training of the navigation sign objects |