

**Capstone Project: Visual Awareness System**

**AY20/21 Term 1**

**Mid-Project Report 1**

**Done by:**

**Ng Jian Ming**

Table of Contents

[Introduction 3](#_Toc52482111)

[Background 3](#_Toc52482112)

[Motivation 3](#_Toc52482113)

[Literature Review 3](#_Toc52482114)

[Literature 1: Hierarchical Models: Intrinsic Separability in High Dimensions (Lin, 2020) 3](#_Toc52482115)

[Literature 2: SLAM – Loop Closing with Visually Salient Features (Cesar Cadena, 2017) 4](#_Toc52482116)

[Literature 3: Deep Learning Features at Scale for Visual Place Recognition (Zetao Chen, 2017) 4](#_Toc52482117)

[The approach 5](#_Toc52482118)

[Shell Model 5](#_Toc52482119)

[Implementation 6](#_Toc52482120)

[Overview 6](#_Toc52482121)

[ResNet50 (Ma, 2019) 6](#_Toc52482122)

[MobileNetV2 (Tsang, 2019) 7](#_Toc52482123)

[VGG16 8](#_Toc52482124)

[Shell Model System 8](#_Toc52482125)

[ShellFamily Training 9](#_Toc52482126)

[ShellModel Parameter Calculations 10](#_Toc52482127)

[Scaling the System 10](#_Toc52482128)

[Creation of Application 10](#_Toc52482129)

[Current Results 12](#_Toc52482130)

[Roadmap 15](#_Toc52482131)

[Conclusion 15](#_Toc52482132)

[References 16](#_Toc52482133)

# Introduction

This report will detail the motivations for the project, literature reviews in which was important and constructive to the project itself along with what was attempted so far. has currently been done so far. It will go in detail the mathematical understanding as well as the challenges faced during the project and the roadmap for the project before the next report.

# Background

Image domain tasks have gained a lot of traction in the recent years. Much progress has been made from a wide variety of domain, from image classification to more challenging task like localization/segmentation of objects in the area. This can be attributed to the rise of deep learning, which is such a powerful tool that led to huge progress in the domain of AI, especially so with the Convolution Neural Network (CNN) architecture for the domain of image related tasks. The ability to exploit spatial related features efficiently in this network was what helped deep learning in image domain task to gain much popularity.

# Motivation

Although CNN has great ability to be able to perform well in many image-related tasks benchmarks like ImageNet (Stanford Vision Lab, 2016) which achieve performance exceeding humans, however there are still many challenges out there that has yet to achieve even a reasonable level of performance. Especially in the domain of self-driving cars where provision of such information from the model is a crucial task given the safety requirements, such tasks remain a huge obstacle for commercialization.

Self-driving cars face many challenges for it to work fully as a system. One of these challenges would be finding what is known as a loop closure in the context of a task called Simultaneous Localization and Mapping (SLAM) (Wolfram Burgard, n.d.). In summary, SLAM is the task of determining whether a vehicle after traveling for an arbitrary period, has reached back to its original location or not. SLAM plays a crucial role in self-driving cars due to drifting of the vehicle’s sensors’ over time. While there are a wide variety of ways in which researchers have approach SLAM like using feature matching algorithms with RANSAC in visual SLAM (Guangcong Zhang, 2015), or graph SLAM, deep learning has slowly became part of the research in finding ways to help improve the system’s performance of such task. Given the huge challenge, this project will not be tackling the whole task but a subtask of it, which is determining whether 2 objects are the same using the semantically rich features extracted from a Convolution Neural Network model. Determining whether 2 objects are the same will play a crucial role in helping the system to figure out if they are in the same area as before after the vehicle’s journey.

# Literature Review

## Literature 1: Hierarchical Models: Intrinsic Separability in High Dimensions (Lin, 2020)

The paper elaborates on how high dimension data are considered as ‘curse’ in the realm of mathematical paradigm and using multiple generative distributions this ‘curse’ can become a ‘blessing’ instead where it can be exploited. It adopts a hierarchical modelling approach with the concepts of shell model functioning as a once-class learner and using a series of shells a multi-class classifier can be formed. More details will be elaborated on the later section as the project involves the use of this architecture itself.

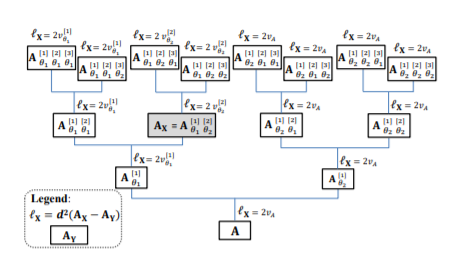


Figure : Hierarchical-model representation as distribution of everything (Lin, 2020)

## Literature 2: SLAM – Loop Closing with Visually Salient Features (Cesar Cadena, 2017)

This paper elaborates the idea of SLAM itself, what has been done and achieved, along with the challenges as well as the possible future of this task. It details the fundamental concepts of SLAM as well as how it scales up to bigger environments before highlighting how the approach toward SLAM has evolved over time in terms of the approaches used to tackle this issue. This paper played a crucial role in narrowing the project task towards the visual place recognition problem detailed in the paper. It highlights how loop closure done in the past was very brute force as it tries to perform a match against all previously detected features before evolving into a Bag-of-Word visual features concept and more. It also highlights the open problems faced by this subtask itself.

## Literature 3: Deep Learning Features at Scale for Visual Place Recognition (Zetao Chen, 2017)

This paper details how they converted the problem of Visual Place Recognition (which is an important subtask in SLAM for loop closure) to a classification problem and tries to map out what is important for the model to determine if 2 places are the same. They developed their very own dataset with images of the same place changing with weather and seasons and use a CNN that has weights initialized by CaffeNet called HybridNet and those not initialized by CaffeNet called AMOSNet. They realised that unlike ImageNet weights, their model layers tend to fire off more scene type patches like buildings in comparison to object type blobs, which led to the conclusion that their model training results in a model that is more robust to viewpoint and appearance changes than simple using a model pretrained on ImageNet and finetuned.

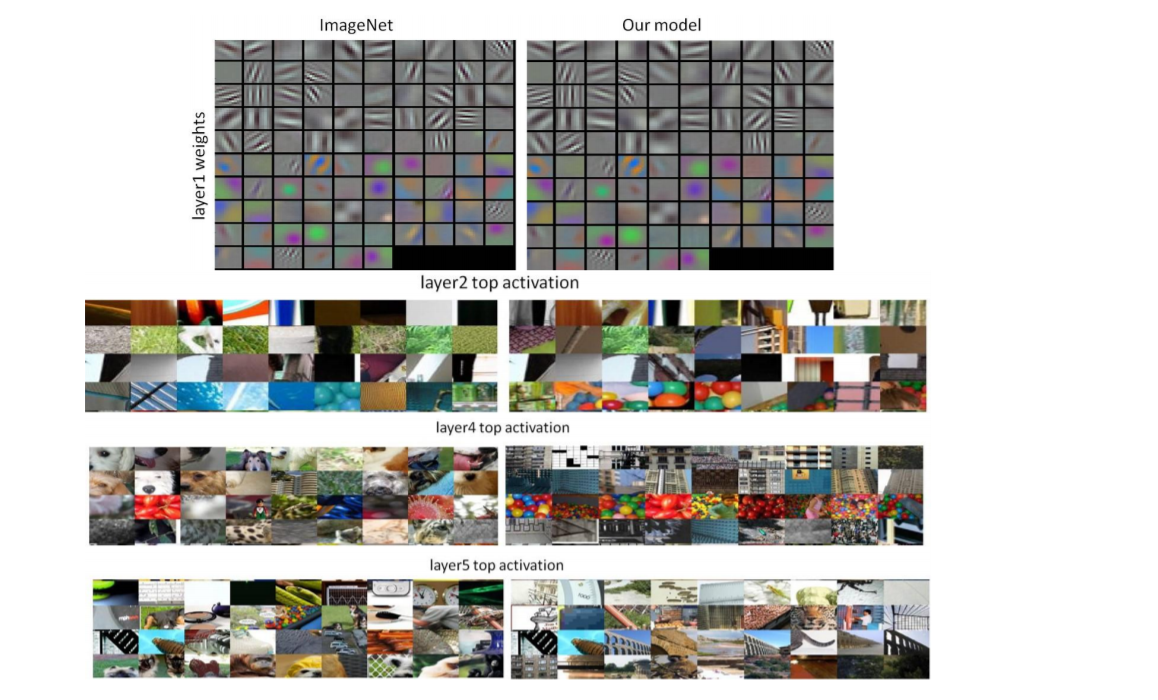


Figure : Visualisation of weights between ImageNet and Visual Place Recognition (Zetao Chen, 2017)

# The approach

The idea of building a visual awareness system comes from this: given that deep learning models are able to extract rich semantic features, since there are so many scenes that could possibly exist, are we able to instead leverage on these features to somewhat group similar objects/scenes together. It is known that the transfer learning in the context of deep learning has been very successful as the feature extraction filters learnt from training can easily be reusable, especially the earlier layers as they are often the building blocks of the system (earlier filters tend to capture edges and shapes of an image which are generalizable for all images). Hence, through a pretrained model, we will be using the rich feature outputs to develop an idea called the Shell Model. On top of it, unlike a typical deep learning system where retraining/finetuning is needed when a new class is introduced, this approach does not require any form of retraining, essentially also creating a model that can not only constantly learn but also scale effectively in terms of classes. Hence, the aim is to be able to create a never-ending learner, where not only current classes can constantly be updated but also new classes can be introduced.

# Shell Model

Referring to a hierarchical system, every child has some form of parent as its superset, which all eventually converge to the root which should contain everything that possibly exist in the world. Given that, the concept lies in which there is some form of hierarchical system and if there is such a system, two different class should look very different in the context of rich features. An example is this: a dog should look more like a cat than car and a Maltese (a dog breed) looks more like a Silky Terrier (another dog breed) than a Persian cat. As such, features that exist within the domain of the children should look like it (and hence as a vector representation of features, should be close to each other). We can derive a shell from this via finding the mean/median of these features which will represent the centrality of the shell and the standard deviation of the vectors of these features as a radius in this hypersphere of feature vectors, effectively forming a shell of one class. And in theory, given that a child exists within a parent, the child shell will also exist within the shell of a parent itself. With the idea of this shell model, what this indicates is we can constantly update our shells to improve its ability to recognise its own class while at the same time when any features do not lie within a certain threshold Euclidean distance from the shell it is able to indicate it is a new class and begin the creation/development of the shell itself.

# Implementation

## Overview

The implementation of this shell is done primarily with the use of the Tensorflow Library (Google Research, 2016) as well as Numpy library. Pretrained models from ImageNet will be used as the CNN model before generating the shell (current ones being tested are ResNet50, MobileNetV2 and VGG16). Without diving into too much detail about these architectures as it is not the focus, a high-level overview of each of these models will be provided.

## ResNet50 (Ma, 2019)

ResNet50 is one of the most used models in the industry today due to its special fundamental blocks called residual blocks that forms this model. It adds a skip-connection (also known as a highway) which can serve as a gate to determine how important a block is in the process of understanding features. Given this shortcut, it can scale deeper with less parameters and suffer lesser from the effects of vanishing gradients, which plays an important role as deeper models have richer features.

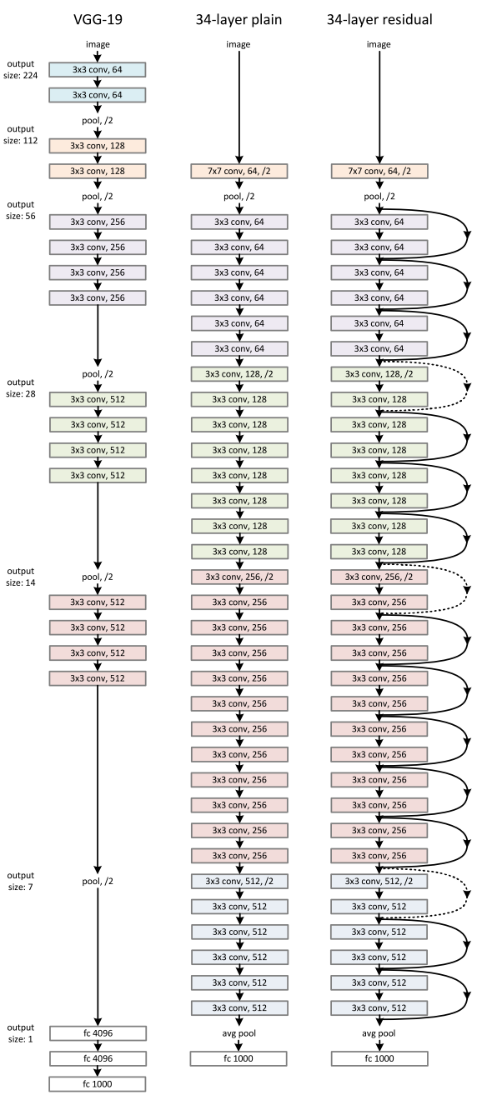


Figure : Example of ResNet Archtecture (Ma, 2019)

## MobileNetV2 (Tsang, 2019)

This model was developed in mind to be a lightweight model. It leverages on something called depthwise convolution which applies a single convolution filter per input channel. On top of it, it uses another concept called pointwise convolution which is a 1 x 1 convolution which constructs new features via linear combinations of its input channels. And lastly, exclusive to MobileNetV2 and not found in MobileNetV1 is the inclusion of the residual block which is a similar concept found is ResNet itself.

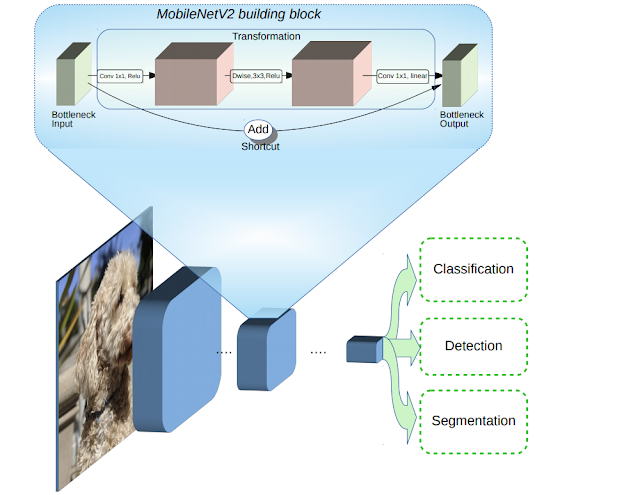


Figure : MobileNetV2 Architecture (Tsang, 2019)

## VGG16 (Karen Simonyan, 2015)

VGG16 can be said to be the oldest of the 3 models used. The idea was simply to scale the model as deep as possible back then. With that idea, it became one of them most famous models due to its great performance on ImageNet competition and hence solidifying the idea that deeper models works better due to richer features.



Figure : VGG16 Architecture (Neurohive, 2018)

## Shell Model System

To allow the system to scale, the shell model will consist of a class called ShellModel which is the shell of one class and all these shells will belong to another class called ShellFamily which encapsulates all the models that are currently present.

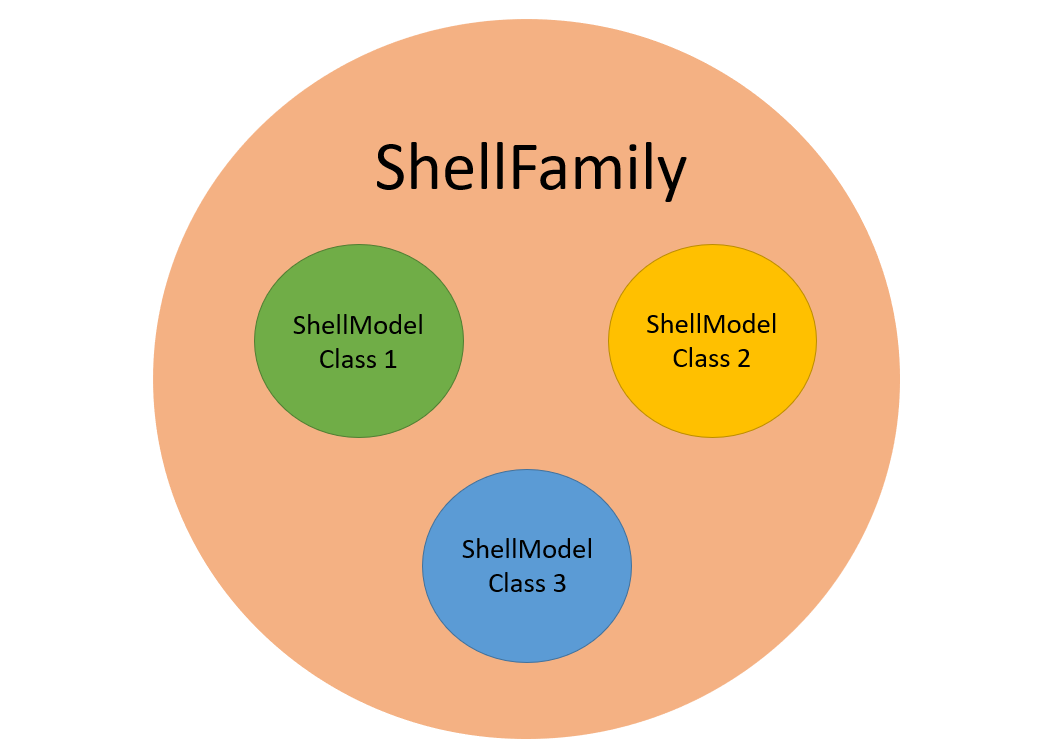


Figure : ShellFamily representation

Each of the ShellModel will contain the following attribute

|  |  |
| --- | --- |
| **Attribute** | **Meaning** |
| raw\_features | The extracted features from the CNN model |
| shell\_mean | The mean/median of the shell. Dimensions are based off the last layer of the CNN model |
| num\_instances | Number of images the shell has seen |
| noise\_mean | The mean/median of the Euclidean distance from the mean/median of the shell\_mean |
| noise\_std | The mean/median of the noise\_mean from the mean/median of the noise\_mean |

Table : Shell Model Parameters

Apart from that, the ShellFamily will contain an attribute called global\_mean which is the mean of the whole dataset (not exclusive to class) and an attribute called pre-processor which is the CNN model to be used for feature extraction. And lastly, to keep track of the number of instances existing in the ShellFamily, it contains an attribute called instances.

## ShellFamily Training

The training process begins with the extraction of the raw features via one of the CNN models selected as the preprocessor attribute. To ensure scalability of the process, a generator is used to feed the data via batches as extracting all the features at once is too memory intensive. Do note that the pixels from the images are preprocessed (or normalized/standardized) first before passing to the model via the use of ImageNet channel means and standard deviations. Within the batch, the images are separated by the groundtruth class and then passed through the preprocessor iteratively. For each class, after going through the preprocessor, the global\_mean attribute is recalculated by multiplying the mean with the number of instances and summing the new features from the output of that multiplication. The number of instances is updated. Lastly, the system checks if there is any shell that currently exist for this class. If no such shells exist, create a new shell while if it exists, update the raw\_features attribute in it the ShellModel class. Once all features are parsed, a fit is then initialized to all the shells in an iterative manner.

## ShellModel Parameter Calculations

The ShellModel parameters are calculated in the following manner. With all the raw\_features available, the raw output features are first normalized via the global\_mean of the dataset before any calculation of the remaining parameters. For shell\_mean, we take the mean of the normalized output features. This shell\_mean represents the centrality of the shell itself. To calculate noise\_mean and noise\_std, noise is first calculated via the subtraction of the normalized features from the shell\_mean followed by a unit vector normalization. The noise\_mean can then be calculated via the use of the median of the noise and lastly for noise\_std, it is the median of the absolute difference between the noise and the mean of the noise itself. And lastly, the num\_instances can also be updated from the first dimension of the features.

## Scaling the System

The above sub-sections highlight the fact if a new ShellFamily were to be created from scratch from a dataset. However, beyond this training process, to allow it to function as a never-ending learner, there are additional steps required. Firstly, the ShellFamily itself will be stored in a pickle file for ease of loading and updating. Next, given every new image that comes in, determine after going through the preprocessor and the normalization which shell it is closest to. To do so, the Euclidean distances are calculated from the difference between the new features and the different shell\_mean from each shell. The distance scores are then inversed to be negative and whichever has the highest score will be the class that new image be assigned to and parameter updates are performed similarly to the details mentioned in the ShellModel training section. Temporarily, there is no threshold included but future work will include this as there is a need be a limit in terms of closeness. If threshold is included, it will then be able to determine if the current image belongs to a specific class or should be a new shell (or a new class) as a system.

# Creation of Application

To create a visual system for understanding and scaling, the model will be served on a server with a frontend system. The package used to design this is called Dash by Plotly (Plotly). It will be served as a RESTAPI via Flask (Flask web development, one drop at a time, 2010) when the system is done. The planned flow of the system is designed as such.

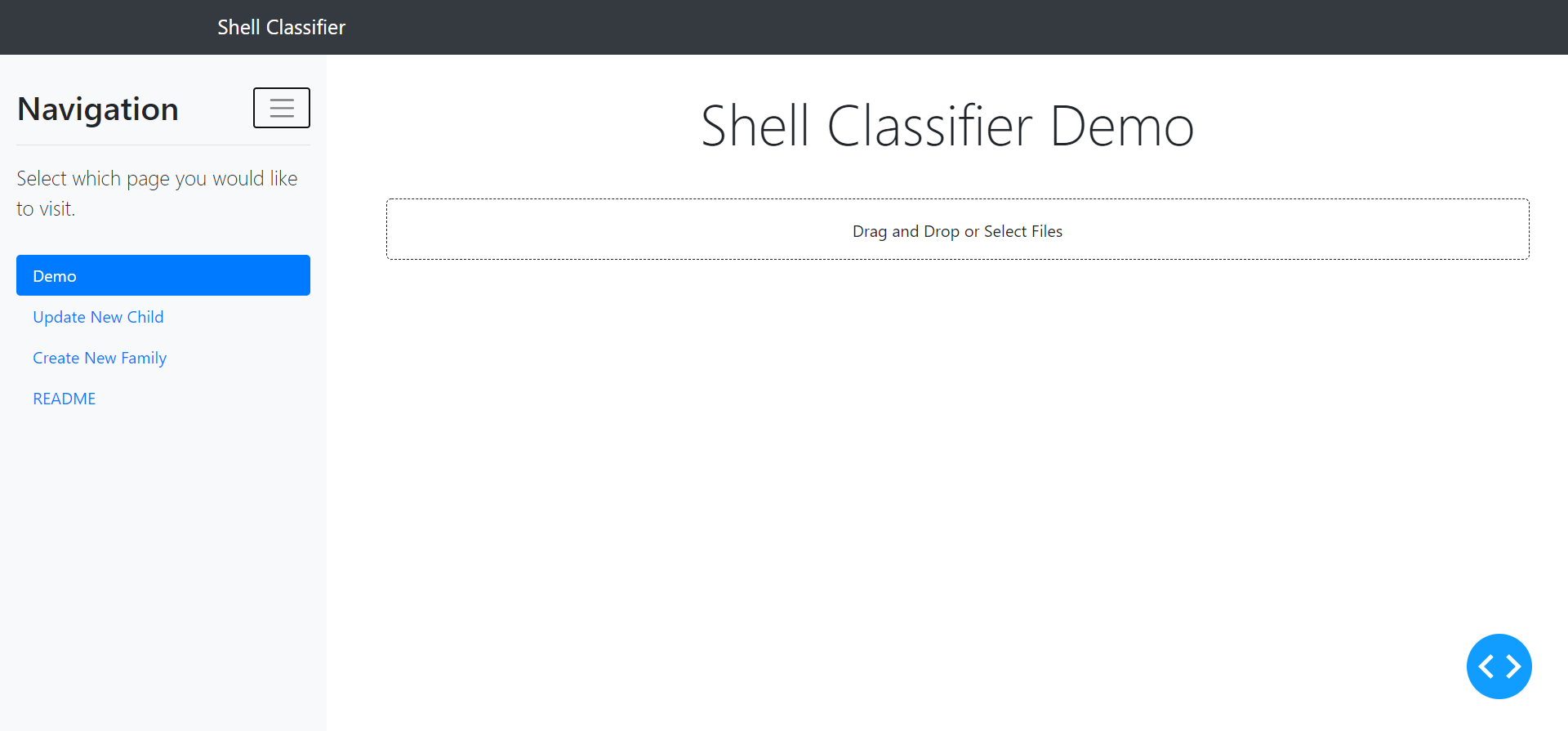


Figure : Main layout for ShellFamily Application

Firstly, the user will choose to upload a photo via the drag and drop box shown in the main content of the page itself. Upon the addition of the file, the application will indicate which class is it based off the distance from the shell. A PCA plot will be shown to provide understanding to how close the uploaded image features are from the different shell means.

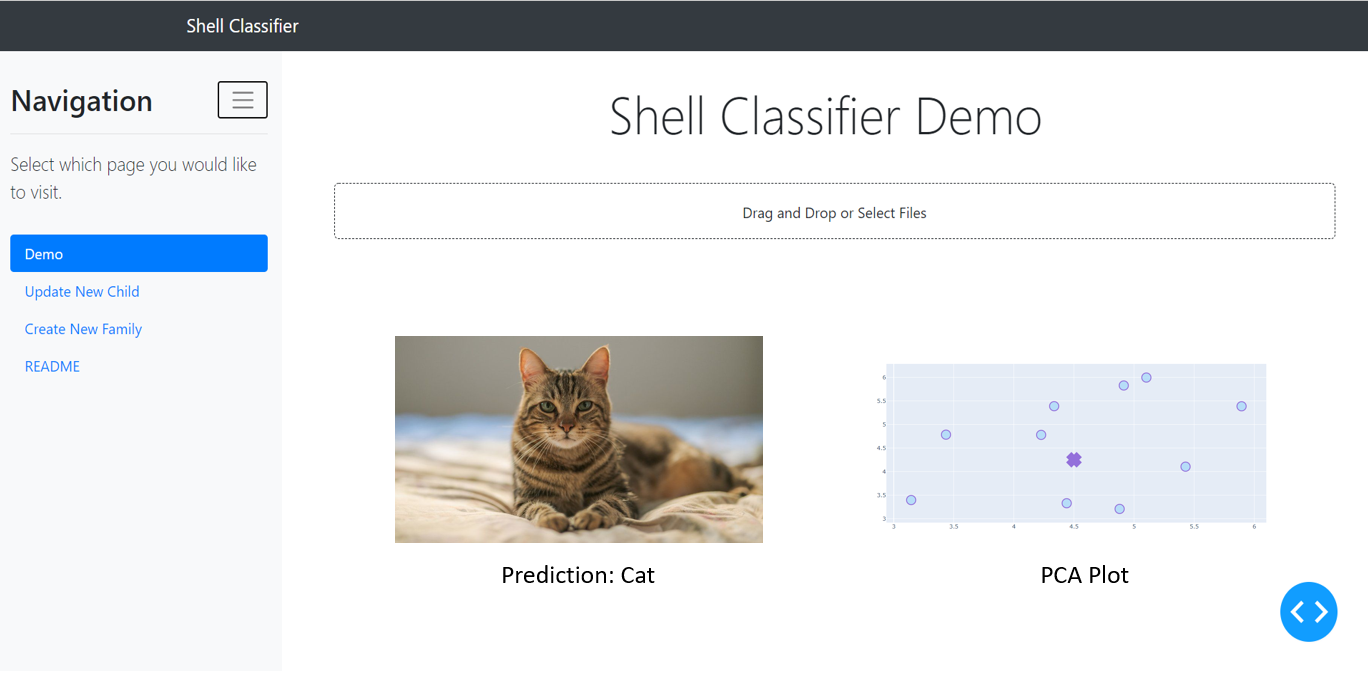


Figure : ShellFamily Application sample output for images that exist within ShellFamily

However, if it does not belong to any shell based off the threshold, an input box will appear requesting the user to define the new class. Once entered, this will be saved to the model and now the model will be able to pick up the new class.

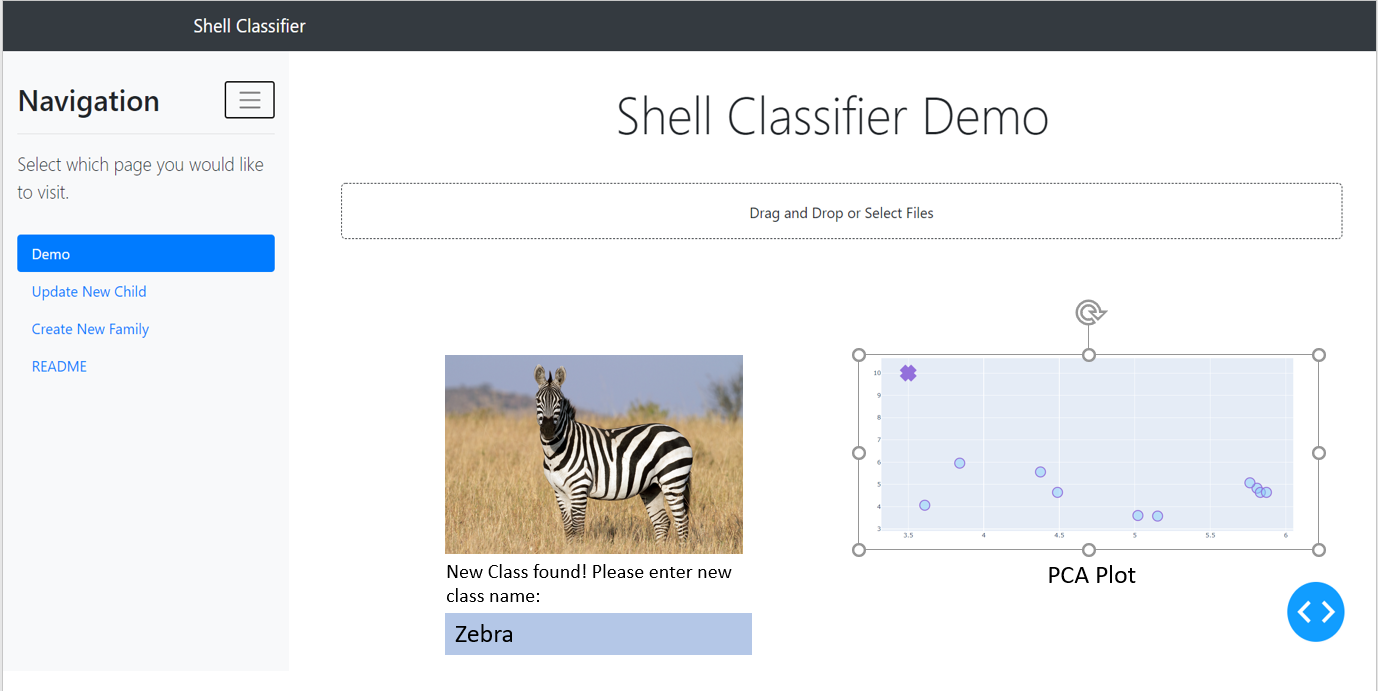


Figure : ShallFamily Application layout when new class is found

There will be other pages in which provides additional function. For the Update New Child Page, it will have a text box allowing users to select a folder from their directory to generate a new child. Ideally, the folder should contain more than 1 images to generate a decent shell model.

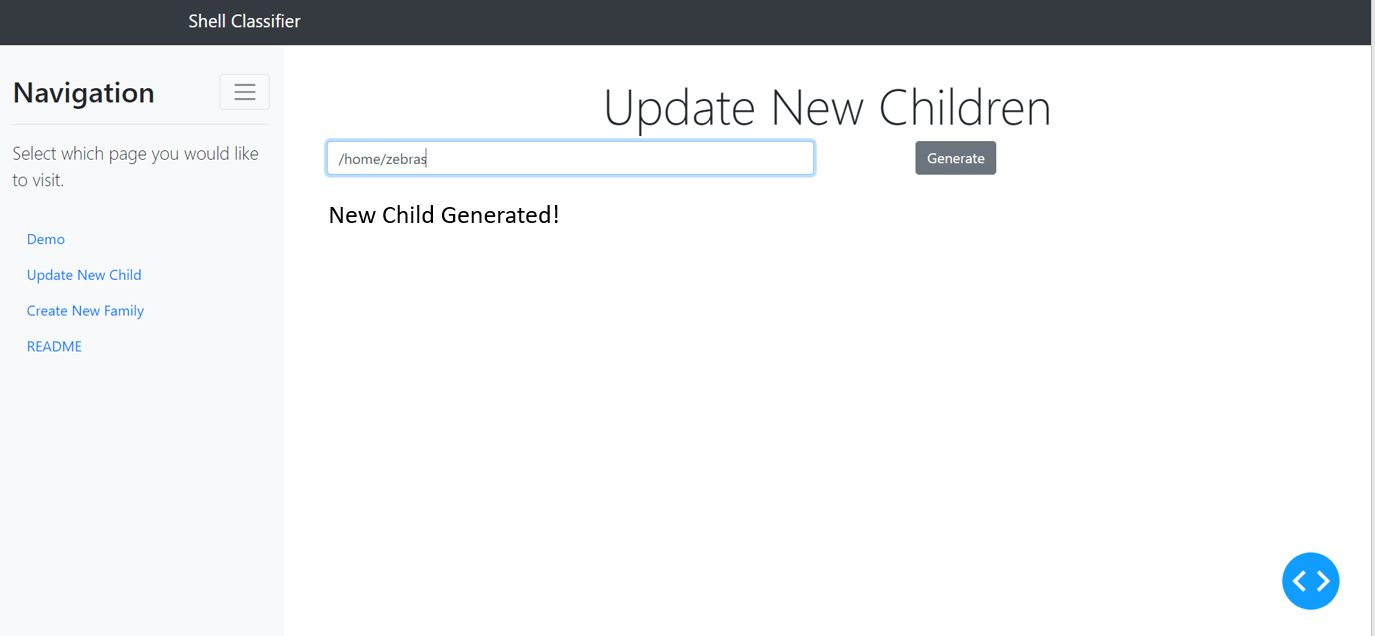


Figure : ShellFamily Application layout for introducing new class by user

Lastly, the Create New Family Page allows an admin to totally create from scratch a new ShellFamily and determine to replace the current one online or not. Currently, there is no exact planned template for this page.

# Current Results

The developed model was tested on a dataset called STL-dataset (Adam Coates, 2011) created by Stanford University for development of unsupervised feature learning, deep learning and self-learning algorithms. This dataset is like cifar-10 which is typically used for classification data with an exception that there are lesser labelled data but a very large set of unlabelled data examples that exist within it. The size of each images is 96 x 96 x 3.

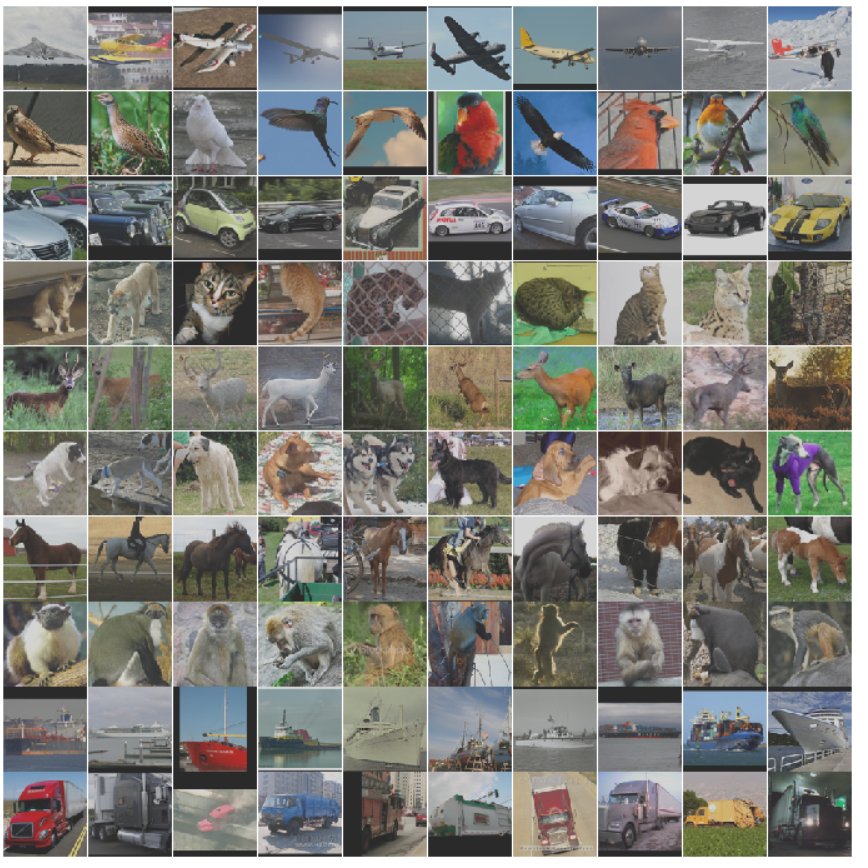


Figure : Examples of STL-10 dataset images (Adam Coates, 2011)

Using a custom train test split dataset with the same ratio (0.2) for testing purposes (4000 images for training and 1000 images for testing), the results are as shown below.

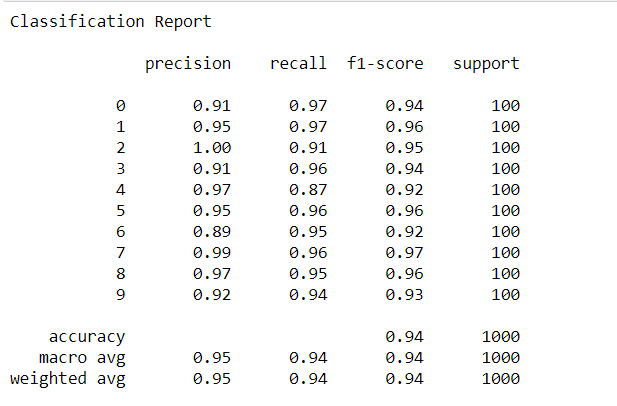


Figure : Classification report from sklearn on STL-10 dataset ShellFamily Model

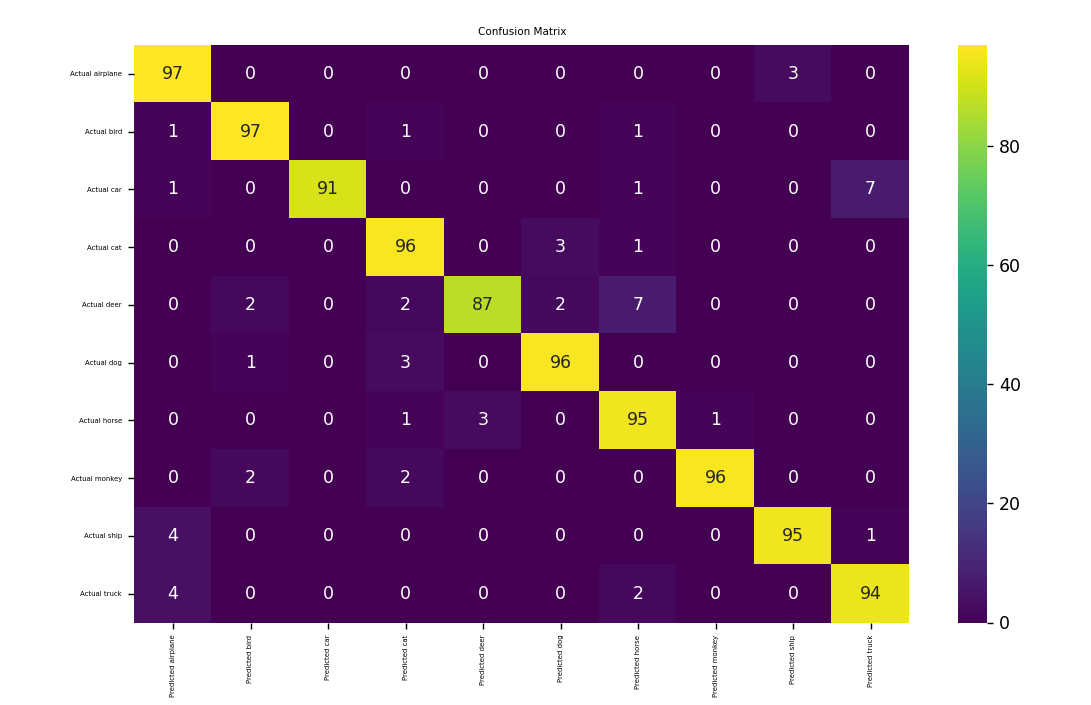


Figure : Confusion Matrix for ShellFamily model on STL-10 dataset

From the results, the ShellFamily system shows a strong ability in performing a classification. The ones that it predicts wrongly are classes that have similar attributes, for example deer being misclassified as horse (since both have 4 legs) or car being classified as trucks (both are vehicles).

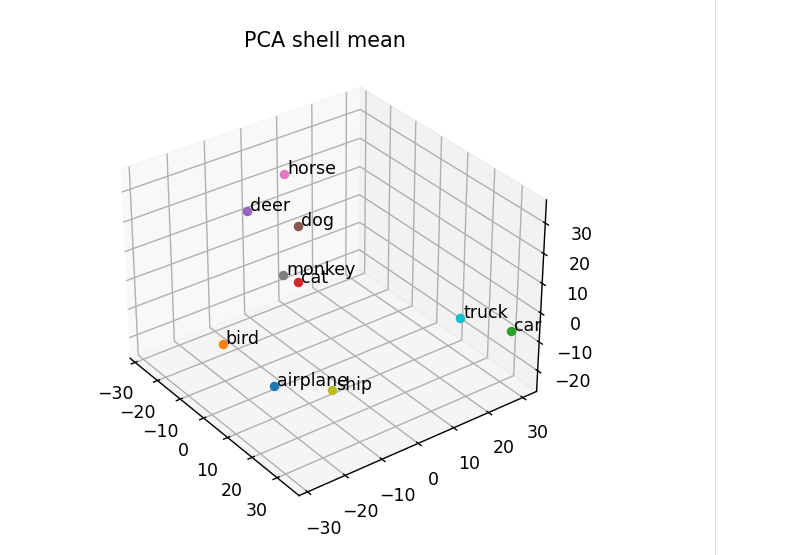


Figure : PCA of all classes' Shell Means into 3D vectors

The shell mean PCA figure also indicates something interesting. We can see a few distinct groups being present. Firstly, all the land animals are close to each other with bird being separated clearly despite being an animal too. Car and truck are close together while airplane and ship are close together. In some manner this reflects a certain parent class encapsulating the different classes mentioned here and while there is some error within the conversion to a lower dimension, nevertheless it is still a good representation to how the features from the pretrained CNN output are like as a cluster.

# Roadmap

The model performs sufficiently well on a dataset stl-dataset achieving approximately 94% accuracy in terms of classifying the different object classes. Template application has also been built where integration of the model’s source code is left along with some backend interactions to ensure that the model can keep growing in an effective manner. Not only that, some ability or plotting of scatterplot to indicate where the input image lies within the shell system for better understanding will be added. Lastly, improvements to the model will be made via developing/researching on methodologies to further improve the performance of this idea.

# Conclusion

Currently, apart from engineering works, more work is needed to find ways to improve not only the performance of the Shell Model itself but also to ensure that it is robust too. Some functionalities could be considered like the thresholding for new class to allow the system to function fully as a never-ending learner. This will be the main part along with some engineering section for the next few months after this report.

# References

Adam Coates, H. L. (2011). An Analysis of Single Layer Networks in Unsupervised Feature Learning. *AISTATS*. Retrieved from https://cs.stanford.edu/~acoates/stl10/

Cesar Cadena, L. C. (30 January, 2017). Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age. *IEEE Transactions on Robotics 32*. Retrieved from https://arxiv.org/pdf/1606.05830.pdf

Flask web development, one drop at a time. (2010). Retrieved from https://flask.palletsprojects.com/en/1.1.x/

Google Research. (16 March, 2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. Retrieved from https://arxiv.org/pdf/1603.04467.pdf

Guangcong Zhang, P. A. (2015). Good Features to Track for Visual SLAM. Retrieved from https://www.cv-foundation.org/openaccess/content\_cvpr\_2015/papers/Zhang\_Good\_Features\_to\_2015\_CVPR\_paper.pdf

Karen Simonyan, A. Z. (10 April, 2015). VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION. Retrieved from https://arxiv.org/pdf/1409.1556.pdf

Lin, W.-Y. (15 March, 2020). Hierarchical Models: Intrinsic Separability in High Dimensions. Retrieved from https://arxiv.org/pdf/2003.07770.pdf

Ma, E. (7 December, 2019). How does ResNet improve performance. Retrieved from https://medium.com/dataseries/how-does-resnet-improve-performance-caaa436f885b

Plotly. (n.d.). Dash by Plotly. Retrieved from https://dash.plotly.com/

Stanford Vision Lab. (2016). Retrieved from ImageNet: http://www.image-net.org/

Tsang, S.-h. (19 May, 2019). Review: MobileNetV2 — Light Weight Model (Image Classification). Retrieved from https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c

Wolfram Burgard, C. S. (n.d.). *Introduction to Mobile Robotics*. Retrieved from http://ais.informatik.uni-freiburg.de/teaching/ss12/robotics/slides/12-slam.pdf

Zetao Chen, A. J. (January, 2017). Deep Learning Features at Scale for Visual Place Recognition. Retrieved from https://www.researchgate.net/publication/312521420\_Deep\_Learning\_Features\_at\_Scale\_for\_Visual\_Place\_Recognition