

CAR BRAND DETECTION USING DEEP LEARNING

A PROJECT REPORT

Submitted by

MARIA NIVITUS. J

18-PCA-016

In partial fulfillment for the award of the degree of

Master of Computer Applications



Loyola College (Autonomous) Chennai - 34

DECEMBER-2020

BONAFIDE CERTIFICATE

This is to certify that the Mini-Project work titled “**Car Brand Detection using Deep Learning**” is being submitted to the Department of Computer Applications (MCA), Loyola College (Autonomous), Chennai-34 by **MARIA NIVITUS.J, 18-PCA-016** for the partial fulfillment for the award of degree of **MASTER OF COMPUTER APPLICATIONS** is a bonafide record of work carried out by him, under my guidance and supervision.

Head of the Department

Dr. C. Muthu

Project Guide

Prof. Antony S. Alexander

Submitted for the Software Development Lab held on **03-DEC-2020**.

Internal Examiner

External Examiner

ABSTRACT

Car Brand detection or Classification and identification is an important task in the area of traffic control and management. Typically, to tackle this task, large datasets and domain-specific features are used to best fit the data. In our project, a person implement, train, and test several state-of-the-art classifiers trained on domain general datasets for the task of identifying the make and models of cars from various angles and different settings, with the added constraint of limited data and time. Generally speaking, visual fine-grained classification can be very challenging due to more subtle differences between classes, compared to basic recognition or coarse classification, such as on ImageNet. Recognizing the makes and models for cars is one such task. For humans, this is usually a fairly straightforward task, especially for car aficionados. Cars can usually be identified by human eye due to certain key aspects, such as logos, hood ornaments, or lettering. However, due to the visual complexity of cars, this has traditionally been a hard task for computers. The main challenge for fine-grained classification is unarguably the very fine differences between different classes. Typically, to learn these minute differences, a large dataset is needed. However, in a setting with limited time, computational power, or data, this is not feasible. Computational power, or data, this is not feasible. In our project, a person design, implement, and test a light weight end-to-end system that uses an out of the box deep learning framework to fine-tune pre-trained classifiers for a specific fine-grained classification test. Our approach is based on taking deep learning models trained on ImageNet, which typically have very general features, and changing as little as possible to fit our training data for classifying the Car models.

TABLE OF CONTENTS

CHAPTERS		CONTENT	PAGE NO.
		List of Tables	6
		List of Figures	7
Chapter-1	THE PROBLEM	1.0 Introduction	8
		1.1 Description of the Problem	8
		1.2 Objectives	8
		1.3 Description of the Existing Solution	9
		1.4 Evaluation of Existing Solution	9
		1.5 Description of other Possible Solutions	11
		1.6 Evaluation of the Possible Solution	12
		1.7 Conclusion	14
Chapter-2	DESIGN	2.0 Introduction	15
		2.1 Project Plan	15
		2.2 Description of Method of Solution	22
		2.3 Hardware Requirements	23
		2.4 Software Requirements	22
		2.5 Packages and Techniques	23
		2.6 Conclusion	25
Chapter-3	IMPLEMENTATION	3.0 Introduction	26
		3.1 Method of solution developed related to suggested solution	26
		3.2 Accurate Method of Solution	27
		3.3 Conclusion	27
Chapter-4	TESTING	4.0 Introduction	28
		4.1 Test Strategy Methods	28
		4.2 Test Conditions	31

Chapter-5	CONCLUSION	5.0 Introduction	53
		5.1 Evaluation	53
		5.2 Future Enhancements	56
		5.3 Technical Skills Learnt	57
		5.4 Conclusion	58
		Bibliography	59
		Appendix	60
	& DOCUMENTATION	4.3 Technical Documentation	34
		4.4 User Manual	47
		4.5 Conclusion	52

LIST OF TABLES

CHAPTER NO	TABLE NO.	TABLE NAME	PAGE NO.
2	2.1	Project plan	15
2	2.2	Hardware Requirements	22
2	2.3	Software Requirements	22
2	2.4	Transfer Learning Techniques	23
3	3.1	Unit Testing	29
3	3.3	Integration Testing	30
3	3.4	Usability Testing	31
3	3.5	Navigation Testing	33
5	5.1	UI/UX Design	54
5	5.2	Project Functionality	55
5	5.3	Data Source / Objectives	56
5	5.4	Technical Skills Learn	57

LIST OF FIGURES

CHAPTER NO	FIGURE NO.	FIGURE NAME	PAGE NO.
2	2.1	UML Diagram	18
2	2.4	Authentication form	21
2	2.6	VGG-16 Architecture	24
2	2.9	VGG-19 Architecture	25
4	4.1	IDE Setup	36
4	4.2	Python Installation	37
4	4.3	Setup Progress	38
4	4.5	Anaconda Setup	39
4	4.6	Anaconda Installation	40

4	4.14	Spyder IDE Setup	46
4	4.16	Neural Network Model	48
4	4.17	Flask UI Development	48
4	4.20	Model Demo	49

CHAPTER 1

INTRODUCTION

1.0 Introduction

The chapter deals with the car brand classification problem analysis stage of data AI development life cycle of the Data Science project. This chapter discuss about the problem identified in the problem statement, business problem statement, data collection, data preprocessing, train and test data splitting, model building, and deployment solution for the problem identified, proposed system, resource required.

1.1 Description of the problem

Car Brand Detection or Classification and identification is an important task in the area of traffic control and management. Typically, to tackle this task, large datasets and domain-specific features are used to best fit the data. In our project, a person implement, train, and test several state-of-the-art classifiers trained on domain general datasets for the task of identifying the make and models of cars from various angles and different settings, with the added constraint of limited data and time. Generally speaking, visual fine-grained classification can be very challenging due to more subtle differences between classes, compared to basic recognition or coarse classification, such as on ImageNet. Recognizing the makes and models for cars is one such task. For humans, this is usually a fairly straightforward task, especially for car aficionados.

1.2 Objectives

- The main objective of making this model for detecting the various kind of car brands in real world.
- This model is one of the part in Traffic surveillance camera for detecting cars.
- This model will help full of people who are going to buy car. It may identify original car brand instead or duplicates.
- It would be help to identify various kind of vehicles.
- This model will help full for car tracking using their car registration id itself.

1.3 Description of the exiting Solution

Vehicle Detection

This system introduces and improves basic deep learning components that are important in vehicle detection. The implementations and results demonstrate that deep learning techniques have great performance in vehicle detection. The proposed model improves the performance and saves testing time. By incorporating extended RPN, the model is more robust to handle the large variation of vehicle scales. In our study, a person examined the nighttime detection-related behaviors of our system. Our system can tolerate severely blurred objects in images taken under urban nighttime lighting. However, for extremely dark conditions with no vehicle headlights or taillights illuminating the scene, the system can recognize the outlines of objects only if the objects are clear.

Dog or Cat Classification

A person can be able to give the solution for this model. For allowing the more accurate prediction compare than previous model or system. The model is comprised of two main parts, the feature extractor part of the model that is made up of VGG blocks, and the classifier part of the model that is made up of fully connected layers and the output layer. A person can use the feature extraction part of the model and add a new classifier part of the model that is tailored to the dogs and cats dataset. Specifically, a person can hold the weights of all of the convolutional layers fixed during training, and only train new fully connected layers that will learn to interpret the features extracted from the model and make a binary classification.

Plant Disease Detection

In this system, a new approach of using deep learning method was explored in order to automatically classify and detect plant diseases from leaf images. The developed model was able to detect leaf presence and distinguish between healthy leaves and 13 different diseases, which can be visually diagnosed. An extension of this study will be on gathering images for enriching the database and improving accuracy of the model using different techniques of fine-tuning and augmentation. The main goal for the future work will be developing a complete system consisting of server side components containing a trained model and an application for smart mobile devices with features such as displaying recognized diseases in fruits, vegetables, and other plants, based on leaf images captured by the mobile phone camera.

1.4 Evaluation of Existing Solution

Vehicle Detection

The goal of this thesis is to build an advanced vehicle detector to recognize cars, pedestrians, cyclist, van, truck, tram, miscellaneous (e.g., trailers), and background based on the image dataset captured from the camera mounted on the front of the driving vehicles. A person simultaneously predict the scores and coordinates of the objects, which are combined into detection. In the field of object detection, a person reviewed two approaches, which are direct detection, including YOLO and SSD, and refined detection, including R-CNN and Faster R-CNN. The designs of these two object detection methods are different, but they achieve similar performance. Refined detection adopts a specialized feature extraction, and then feeds the output of feature extraction into the detector. Direct detection divides the raw image into a 7×7 grid such that each grid is a potential bounding box. Additionally, direct detection utilizes the grid cell to predict object locations and then feeds the information into the detector. Thus, the direct detection is faster than the refined detection, but slightly less accurate.

The work most related to our thesis is scale-aware RPN. They utilize a different RPN and feed the output into an XGBoost classifier. Their model achieves 84.81% AP on moderate difficulty level with testing time of 0.9 seconds per image by using a GTX TITAN X GPU with 12 GB memory. Our model achieves 68.77% AP on moderate difficulty level with time of 0.269 seconds per image by using a GTX 1080 with 11 GB memory. A TITAN GPU outperforms a 1080 GPU by 23% in training and testing. In vehicle detection, testing time is as important as accuracy. A person think it is necessary to trade-off between testing time and accuracy rather than sacrificing testing time to increase accuracy.

Dog or Cat Classification

In this model using Convolutional Layers a person train the model and test the model accuracy on test data. First used simple layer of 2 convolutional network and got an accuracy of 78%. Next used 8 layers Convolutional networks and got an accuracy of 92%. Next using pre-trained model VGG16 got an accuracy of 98%. Next using pre-trained model RESNET50 got an accuracy of 98.5%. In this section, a person evaluate the DefPM from perspectives of SVM kernels, body and head comparison and drawback analysis. Generally speaking, the Deformable Part Model is performing very well with head part, which achieves 92% accuracy on the Kaggle leaderboard. The Kaggle itself already provides a well evaluation of our work. On SVM Kernel Functions little difference between performances on kernel functions. The scores are well linear separable. Thus polynomial kernel works worst and linear SVM is an ideal choice for accuracy and convergence speed. In this model RGB works the best. Though the gain is relative tiny. Body DefPM vs Head DefPM Bodies are very deformable because of difference positions a pet can hold. Thus body is hard

to capture exactly and the bounding box are including more noise than head box. Head are undeform able and stable with much less noise included in bounding box and scores.

Plant Disease Detection

Plant disease detection and classification process, but still, this research field is lacking. In addition, there are still no commercial solutions on the market, except those dealing with plant species recognition based on the leaves images. In this system, a new approach of using deep learning method was explored in order to automatically classify and detect plant diseases from leaf images. The developed model was able to detect leaf presence and distinguish between healthy leaves and n different diseases, which can be visually diagnosed. The complete procedure was described, respectively, from collecting the images used for training and validation to image preprocessing and augmentation and finally the procedure of training the deep CNN and fine-tuning. Different tests were performed in order to check the performance of newly created model.

To get a sense of how our approaches will perform on new unseen data, and also to keep a track of if any of our approaches are overfitting, a person run all our experiments across a whole range of train-test set splits, namely 80–20 (80% of the whole dataset used for training, and 20% for testing), 60–40 (60% of the whole dataset used for training, and 40% for testing), 50–50 (50% of the whole dataset used for training, and 50% for testing), 40–60 (40% of the whole dataset used for training, and 60% for testing) and finally 20–80 (20% of the whole dataset used for training, and 80% for testing). It must be noted that in many cases, the plant dataset has multiple images of the same leaf (taken from different orientations), and person have the mappings of such cases for 41,112 images out of the 54,306 images.

1.5 Description of Other Possible Solutions

3D Vehicle Detection

Convolutional neural networks perform well and are innovative in the field of image analysis, including object detection, instance segmentation, and object tracking. Combined with large datasets, CNN outperforms traditional machine learning techniques. In vehicle detection, person have proposed a good detection model for two-dimensional vehicle detection, which means a person only compute the two-dimensional bounding box of specific objects. People are getting more interested in three-dimensional vehicle detection, which is a great technique for autonomous driving. A popular method in 3D vehicle detection is first computing the 2D bounding box by YOLO or Faster R-CNN, and then determining the dimension and orientation of the objects that are used for the determination of the location of 3D objects.

Dog or Cat 3D Classification

Graph object convolutional neural networks (Graph-CNNs) extend traditional CNNs to handle data that is supported on a graph. Major challenges when working with data on graphs are that the support set (the vertices of the graph) do not typically have a natural ordering, and in general, the topology of the graph is not regular (i.e., vertices do not all have the same number of neighbors). Thus, Graph-CNNs have huge potential to deal with 3D point cloud data which has been obtained from sampling a manifold. In this paper a person develop a GraphCNN for classifying 3D point cloud data, called PointGCN1. The architecture combines localized graph convolutions with two types of graph downsampling operations (also known as pooling). By the effective exploration of the point cloud local structure using the Graph-CNN, the proposed architecture achieves competitive performance on the 3D object classification benchmark ModelNet, and our architecture is more stable than competing schemes.

Plant Disease Detection using Hyperspectral

In this system explained DL approaches for the detection of plant diseases. Moreover, many visualization techniques/mappings were summarized to recognize the symptoms of diseases. Although much significant progress was observed during the last three to four years, there are still some research gaps. In most of the researches (as described in the previous sections), the PlantVillage dataset was used to evaluate the accuracy and performance of the respective DL models/architectures. Although this dataset has a lot of images of several plant species with their diseases, system has a simple/plain background. However, for a practical scenario, the real environment should be considered. Hyperspectral/multispectral imaging is an emerging technology and has been used in many areas of research. Therefore, it should be used with the efficient DL architectures to detect the plants' diseases even before their symptoms are clearly apparent.

1.6 Evaluation of the possible Solution

3D Vehicle Detection

Vehicle 3D extents and trajectories are critical cues for predicting the future location of vehicles and planning future agent ego-motion based on those predictions. In this paper, a person propose a novel online framework for 3D vehicle detection and tracking from monocular videos. The framework can not only associate detections of vehicles in motion over time, but also estimate their complete 3D bounding box information from a sequence of 2D images captured on a moving platform. Our method leverages 3D box depth-ordering matching for robust instance association and utilizes 3D trajectory prediction for re-identification of occluded vehicles. A person also design a motion learning module based on an LSTM for more accurate long-term

motion extrapolation. Our experiments on simulation, KITTI, and Argoverse datasets show that our 3D tracking pipeline offers robust data association and tracking. On Argoverse, our image-based method is significantly better for tracking 3D vehicles within 30 meters than the LiDAR-centric baseline methods. The training of deep-learning-based 3D object detectors requires large datasets with 3D bounding box labels for supervision that have to be generated by hand-labeling. A person propose a network architecture and training procedure for learning monocular 3D object detection without 3D bounding box labels. By representing the objects as triangular meshes and employing differentiable shape rendering, a person define loss functions based on depth maps, segmentation masks, and ego- and object-motion, which are generated by pre-trained, off-the-shelf networks. A person evaluate the proposed algorithm on the real-world KITTI dataset and achieve promising performance in comparison to state-of-the-art methods requiring 3D bounding box labels for training and superior performance to conventional baseline methods.

Dog or Cat 3D Classification

3D Object detection is a crucial component of sensor-based perception systems for advanced driver assistance systems (ADAS) and for autonomous driving. Despite remarkable advancements in object detection using data from cameras, designing a LIDAR-based object detection system for real-world driving applications is a very challenging problem. Monocular cameras have been the most common sensor technology for object detection. Specifically, high-resolution color cameras are the primary choice to detect traffic signs, license plates, pedestrians, cars, and so on. However, passive vision systems suffer from disadvantages such as night vision incapability, inability of direct depth perception and illumination variations, which limits their use in realistic driving scenarios. By contrast, 3D-LIDARs are robust against the aforementioned problems at the cost of having a higher price and having moving parts.

Plant Disease Detection using Hyperspectral Imaging

This model/system can explores how imaging techniques are being developed with a focus on deployment for crop monitoring methods. Imaging applications are discussed in relation to both field and glasshouse-based plants, and techniques are sectioned into ‘healthy and diseased plant classification’ with an emphasis on classification accuracy, early detection of stress, and disease severity. A central focus of the review is the use of hyperspectral imaging and how this is being utilized to find additional information about plant health, and the ability to predict onset of disease. A summary of techniques used to detect biotic and abiotic stress in plants is presented, including the level of accuracy associated with each method. This system proposes the use of hyperspectral imaging (VNIR and SWIR) and machine learning techniques for the detection of the Tomato

Spotted Wilt Virus (TSWV) in capsicum plants. Discriminatory features are extracted using the full spectrum, a variety of vegetation indices, and probabilistic topic models. These features are used to train classifiers for discriminating between leaves obtained from healthy and inoculated plants. The results show excellent discrimination based on the full spectrum and comparable results based on data-driven probabilistic topic models and the domain vegetation indices. Additionally our results show increasing classification performance as the dimensionality of the features increase.

1.7 Conclusion

In this chapter a person proposed a CNN model for all kind of image classification. The model has two fast localized graph convolutional layers and point cloud data specific designed pooling layer using global pooling or multi-resolution pooling. Our proposed approach is demonstrated to be competitive on the 2D & 3D point-cloud classification benchmark dataset which makes the model converge faster and also improves the robustness as illustrated via the standard deviation of the model performance.

CHAPTER 2

DESIGN

2.0 Introduction

Project design is an early phase of the project where a project's key features, structure, criteria for success, and major deliverables are all planned out. The point is to develop one or more designs which can be used to achieve the desired project goals.

2.1 Project Plan

Phases	Date	Time	Objectives
1.Business Understanding	05/09/20	One Hour	Understand the Business problems
2.Data Understanding	07/09/20	One Hour	This involves the collection of all the available data
3.Data Preparation	17/09/20	Ten Hour	Selecting the relevant data, integrating the data by merging the data sets, cleaning it, treating the missing values.
4.EDA	01/10/20	Five Hour	Feature is explored graphically using bar-graphs.
5.Data Modelling	05/10/20	Five Hour	A model takes the prepared data as input and provides the desired output.
6.Model Evaluation	20/10/20	Six Hour	A model is evaluated for checking if it is ready to be deployed.

Table 2.1

1. Business Understanding

The entire cycle revolves around the business goal. What will you solve if you do not have a precise problem? It is extremely important to understand the business objective clearly because that will be your final goal of the analysis. After proper understanding only a person can set the specific goal of analysis that is in sync with the business objective. You need to know if the client wants to reduce credit loss, or if they want to predict the price of a commodity, etc.

2. Data Understanding

After business understanding, the next step is data understanding. This involves the collection of all the available data. Here you need to closely work with the business team as they are actually aware of what data is present, what data could be used for this business problem and other information. This step involves describing the data, their structure, their relevance, their data type. Explore the data using graphical plots. Basically, extracting any information that you can get about the data by just exploring the data.

3. Data Preparation

Next comes the data preparation stage. This includes steps like selecting the relevant data, integrating the data by merging the data sets, cleaning it, treating the missing values by either removing them or imputing them, treating erroneous data by removing them, also check for outliers using box plots and handle them. Constructing new data, derive new features from existing ones. Format the data into the desired structure, remove unwanted columns and features. Data preparation is the most time consuming yet arguably the most important step in the entire life cycle. Your model will be as good as your data.

4. Exploratory Data Analysis

This step involves getting some idea about the solution and factors affecting it, before building the actual model. Distribution of data within different variables of a feature is explored graphically using bar-graphs, Relations between different features is captured through graphical representations like scatter plots and heat maps. Many other data visualization techniques are extensively used to explore every feature individually, and by combining them with other features.

5. Data Modeling

Data modeling is the heart of data science. A model takes the prepared data as input and provides the desired output. This step includes choosing the appropriate type of model, whether the problem is a classification

problem, or a regression problem or a clustering problem. After choosing the model family, amongst the various algorithm amongst that family, a person need to carefully choose the algorithms to implement and implement them. A person need to tune the hyperparameters of each model to achieve the desired performance. A person also need to make sure there is a correct balance between performance and generalizability. A person do not want the model to learn the data and perform poorly on new data.

6. Model Evaluation

Here the model is evaluated for checking if it is ready to be deployed. The model is tested on an unseen data, evaluated on a carefully thought out set of evaluation metrics. A person also need to make sure that the model conforms to reality. If a person do not obtain a satisfactory result in the evaluation, a person must re-iterate the entire modeling process until the desired level of metrics is achieved. Any data science solution, a machine learning model, just like a human, should evolve, should be able to improve itself with new data, adapt to a new evaluation metric. A person can build multiple models for a certain phenomenon, but a lot of them may be imperfect. Model evaluation helps us choose and build a perfect model.

7. Model Deployment

The model after a rigorous evaluation is finally deployed in the desired format and channel. This is the final step in the data science life cycle. Each step in the data science life cycle explained above should be worked upon carefully. If any step is executed improperly, it will consequently affect the next step and the entire effort goes to waste. For example, if data is not collected properly, you'll lose information and you will not be building a perfect model. If data is not cleaned properly, the model will not work. If the model is not evaluated properly, it will fail in the real world. Right from Business understanding to model deployment, each step should be given proper attention, time and effort.

2.2 Description of Method of Solution

Architectural Design

The architectural diagram is a basic view of any system application to be developed. It is a foundation layout of the project works. With regard to that, the architectural design of the project of Car Brand Classification is depicted follows:

Use case Diagram

The use case view models functionality of the system as perceived by outside uses. A use case is a coherent unit of functionality expressed as a transaction among actors and the system.

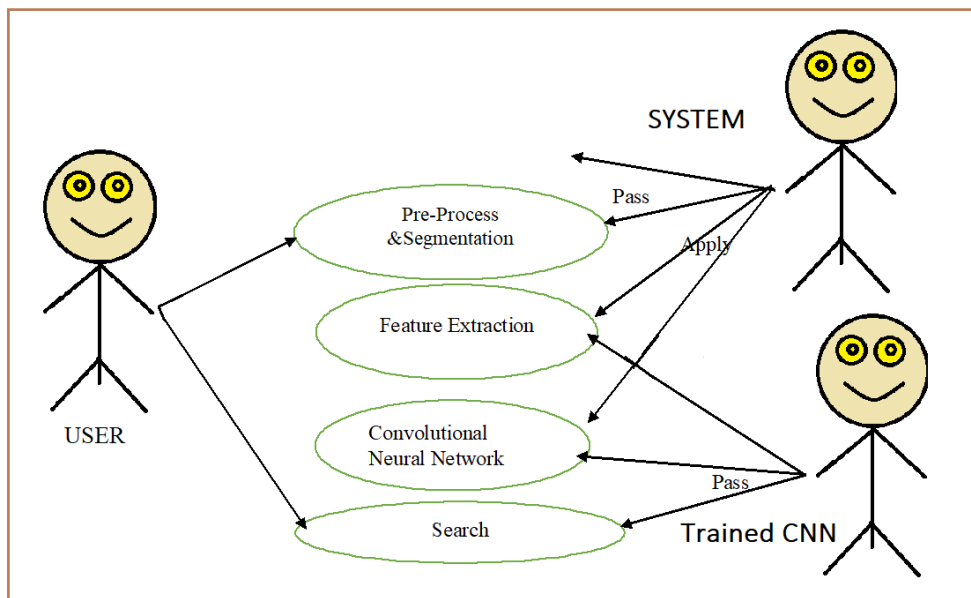


Figure 2.1

Sequence Diagram

A sequence diagram is a graphical view of a scenario that shows object interaction in a time- based sequence what happens first, what happens next. Sequence diagram establish the role of objects and helps provide essential information to determine class responsibilities and interfaces. This type of diagram is best used during early analysis phase in design because they are simple and easy to comprehend. Sequence diagram are normally associated with use cases.

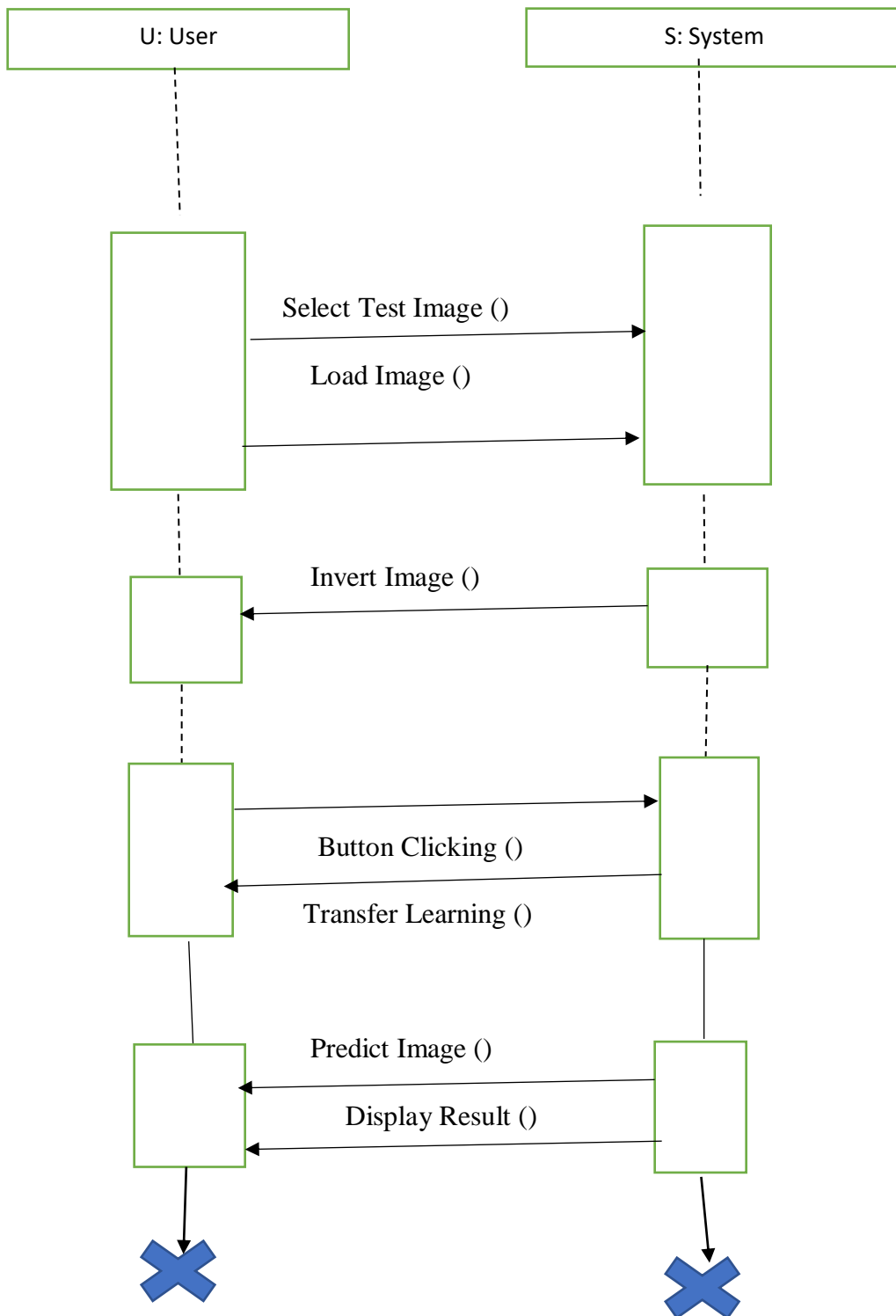


Figure 2.2

Data Flow Diagram

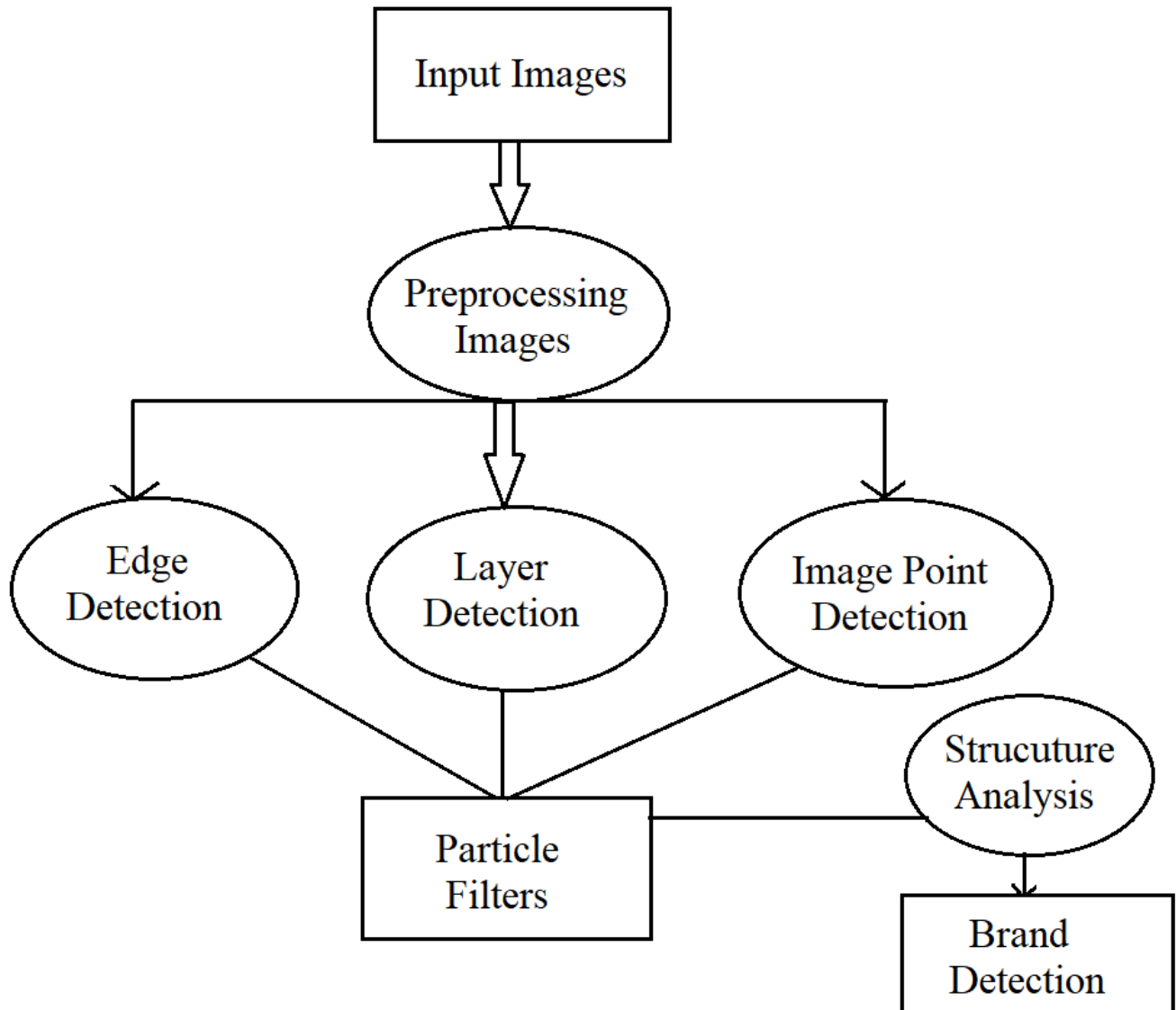
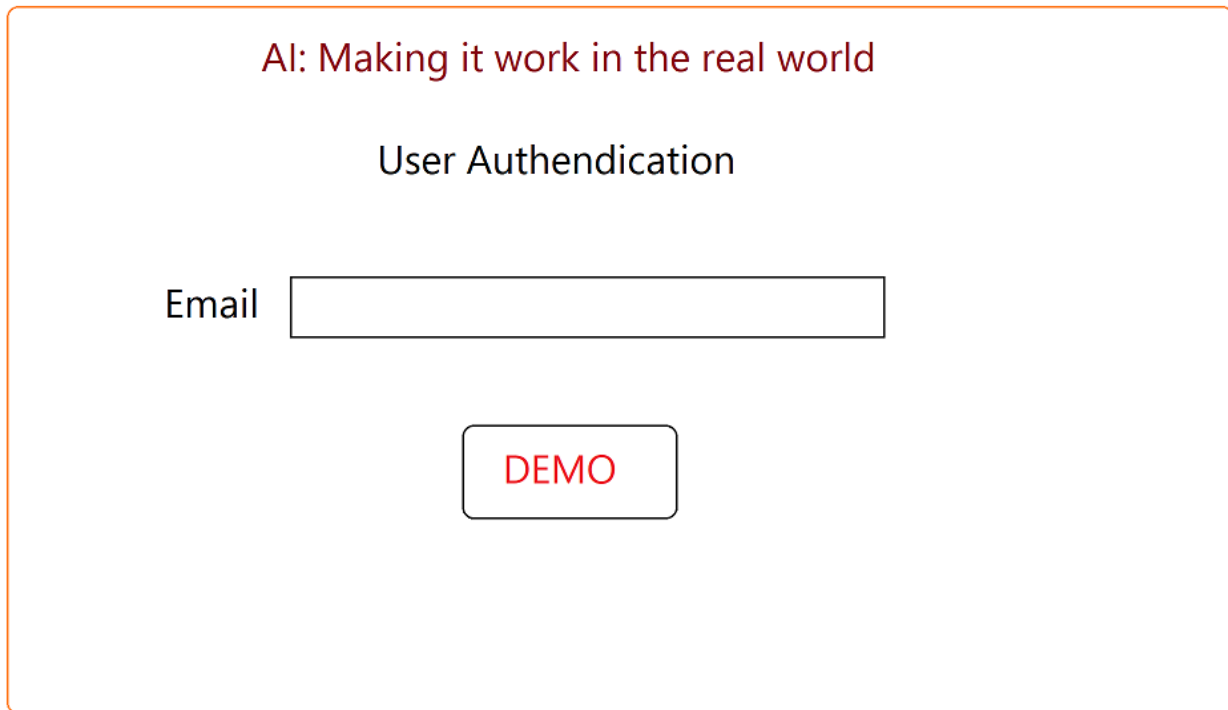


Figure 2.3

PSEUDO FORM DESIGN

Pseudo form designing means deciding the contents and layout of forms for the purpose of collecting and processing the required information economically and efficiently.

User Authentication Page



The image shows a user authentication page layout. At the top, the text "AI: Making it work in the real world" is displayed in a dark red font. Below this, the title "User Authendication" (note the typo) is centered in a black font. Underneath the title, the label "Email" is positioned to the left of a rectangular input field. At the bottom center of the form area, there is a button with the text "DEMO" in red, rounded corners, and a thin black border.

Figure 2.4

User Interface

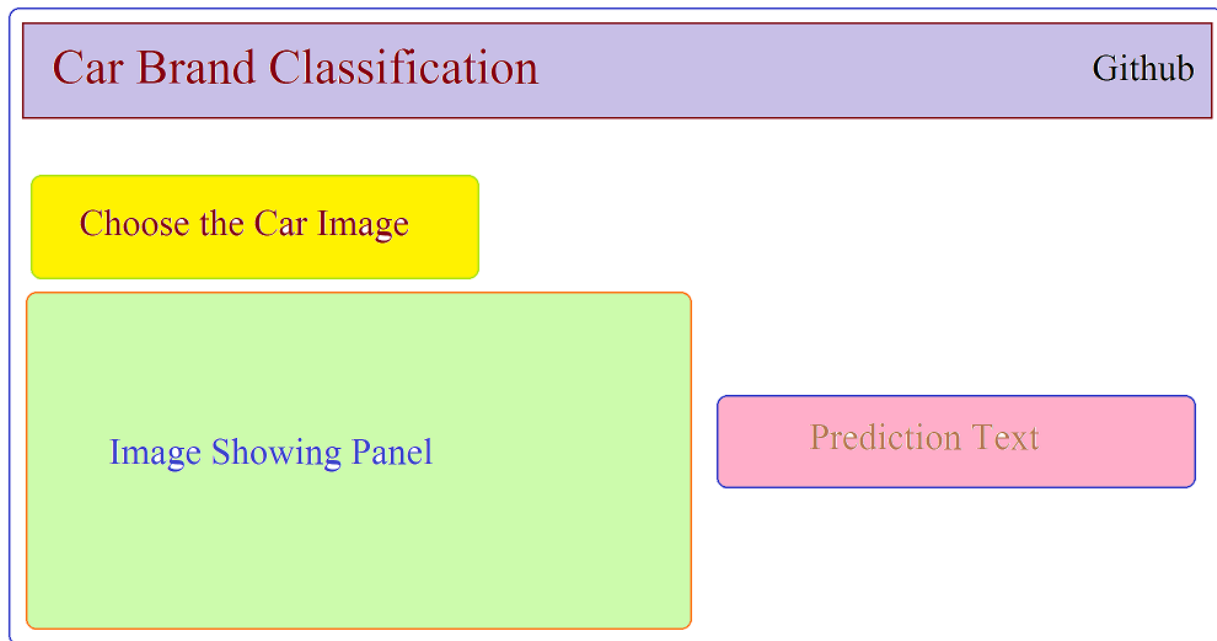


Figure 2.5

2.3 Hardware Requirements

Hardware Requirements

The system requirements include the hardware and software requirements. This may serve as the basis for contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. To be used efficiently, all computer software needs certain hardware components that are often used as guidelines. All the physical equipment's i.e. input devices, GPU, processor, and output device & inter connecting processor of the computer s called as hardware.

Processor	I5
Hard Disk	40GB
Ram	Min 8GB Max 32GB
Keyboard	Standard Keyboard
Mouse	Optical or Wired
GPU	NVIDIA GeForce GTX 1650

Table 2.2

2.4 Software Requirements

A set of instructions or program required to make hardware platform suitable for desired task is known as software.

Operating System	Windows 10
Front End	Python 3.7, HTML 5, CSS 3, Java Script, Flask 1.1.0
Back End	Firebase
Concepts and Algorithms	Deep Learning, Machine Learning, AI and State-of-Art-Algorithms.
Data	Car Images
Browser	Google Chrome 87.0.11209.109
IDE	Jupyter Notebook 6.1.5, Sypder 3.3.1

Table 2.3

2.5 Packages and Techniques

Tensorflow:

TensorFlow is an end-to-end open source platform for machine learning. System has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

Keras:

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

Transfer Learning Techniques

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
<u>VGG16</u>	528 MB	0.713	0.901	138,357,544	23
<u>VGG19</u>	549 MB	0.713	0.900	143,667,240	26
<u>ResNet50</u>	98 MB	0.749	0.921	25,636,712	-
<u>ResNet101</u>	171 MB	0.764	0.928	44,707,176	-
<u>ResNet152</u>	232 MB	0.766	0.931	60,419,944	-
<u>ResNet50V2</u>	98 MB	0.760	0.930	25,613,800	-
<u>InceptionV3</u>	92 MB	0.779	0.937	23,851,784	159

Table 2.4

VGG-16 | CNN model

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is an annual computer vision competition. Each year, teams compete on two tasks. The first is to detect objects within an image coming from 200 classes, which is called object localization. The second is to classify images, each labeled with one of 1000 categories, which is called image classification. VGG 16 was proposed by Karen Simonyan and Andrew Zisserman of the Visual Geometry Group Lab of Oxford University in 2014 in the paper “VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION”. This model won the 1st and 2nd place on the above categories in 2014 ILSVRC challenge.

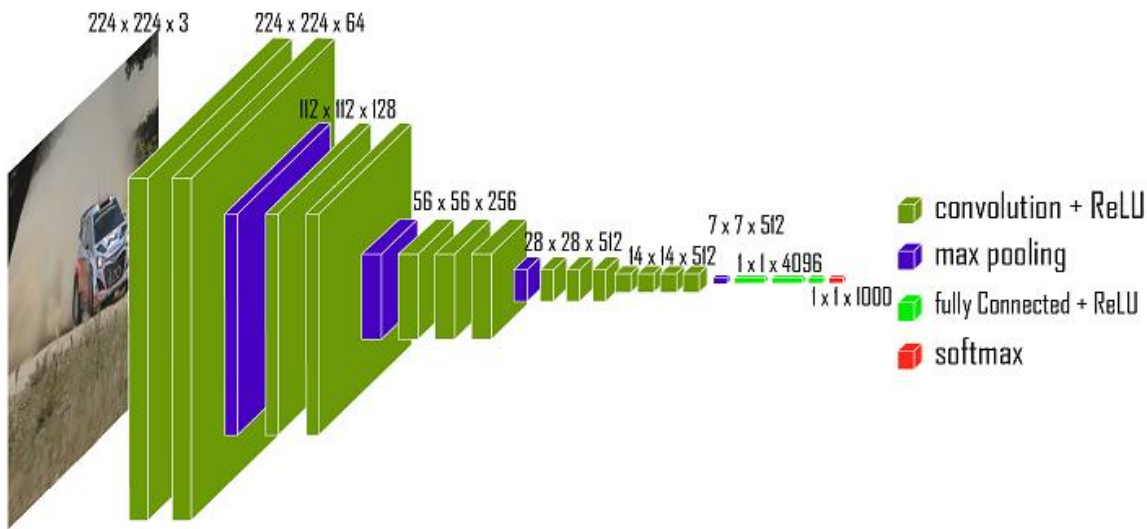


Figure 2.6

VGG-19 | CNN model

VGG-19 is a convolutional neural network that is **19** layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals.

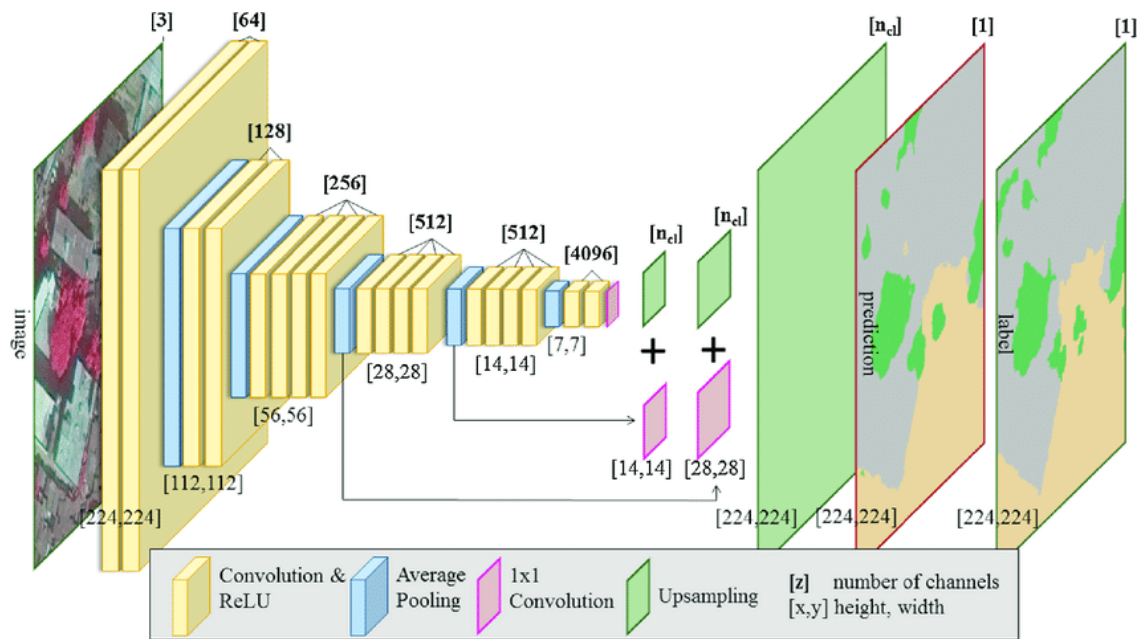


Figure 2.7

2.6 Conclusion

In this chapter a person proposed a design and user interface of image classification model. The design model has various kind of bases such as project plan, Description of Method of Solution such as DFD/System Flow/HIPO Chart / ER Diagram and system requirements hardware and software requirements. Over here person have seen about life cycle of data science projects and its components. As well person have seen about techniques and packages for implementing thins Deep Learning project.

CHAPTER 3

IMPLEMENTATION

3.0 Introduction

The implementation of data science projects in different areas of a company seeks to improve its processes and increase its value. The idea is to transform data into information to increased revenue, reduced costs, increase the business agility and also enhanced customer experience. Data science something sounds like the holy grail however require extraordinary efforts until to obtain the successful implementation to collect data, process and explore and transform to obtain KPIs, metrics and valuable and reliable insights about the business, and even more effort to develop the predictive analysis.

3.1 Method of Solution Developed related to the suggested solution

- Making this model to improve 3D model for detecting real world cars or objects.
- Connecting this model to IOT devices for identifying real time car detection.
- Using some concepts of Computer vision techniques for making machine vision model.
- Training more car brand images for detecting more brands in real world
- Try to make this model to mobile Application in future.
- Try to train a night time data images for detecting car images night time also.

3.1.1 Pseudocode for Image Classification in CNN:

I = Input Image, node = Root Node of the Tree

procedure ClassPredict(I, node)

count = # of children of node

if count = 0 then

label = class label of the node

return label

else

nextNode = EvaluateNode(I, node)

I returns the address of the child node of highest output neuron

return ClassPredict(I, nextNode)

end if

end procedure

3.2 Accurate Method of solution

The ability to reliably detect car vehicles offers significant advantages for asset management, resource allocation, brand classification, site safety, and traffic control. To identify the right technology for your vehicle detection application, many factors must be taken into consideration, including task, size of target, sensing range, sensor mounting, and whether the application is primarily indoor or outdoor.

3.3 Conclusion

In our future work, a person plan to improve the performance of our system by using alternative normalization methods. A person plan to focus on the images taken under extreme illumination conditions. A person plan to develop suitable metrics to appropriately measure the model performance for localizing objects, specifically for our nighttime data.

CHAPTER 4

Testing and Documentation

4.0 Introduction

This testing approach document is designed for Information and Technology Services upgrades to PeopleSoft. The document contains an overview of the testing activities to be performed when an upgrade or enhancement is made, or a module is added to an existing application. The emphasis is on testing critical business processes, while minimizing the time necessary for testing while also mitigating risks. It's important to note that reducing the amount of testing done in an upgrade increases the potential for problems after go-live. Management will need to determine how much risk is acceptable on an upgrade by upgrade basis.

System testing is simply testing the system as a whole; it gets all the integrated modules of the various components from the integration testing phase and combines all the different parts into a system which is then tested. Testing is then done on the system as all the parts are now integrated into one system the testing phase will now have to be done on the system to check and remove any errors or bugs.

In the system testing process the system will be checked not only for errors but also to see if the system does what was intended, the system functionality and if it is what the end user expected.

There are various tests that need to be conducted again in the system testing which include;

- Test Plan
- Test Case
- Test Data

If the integration stage was done accurately then most of the test plan and test cases would already have been done and simple testing would only have to be done in order to ensure there are no bugs because this will be the final product.

4.1 Description of Testing Methods

There are loads of examples of system testing; below I will discuss some of the important types of systems testing that are done regularly,

- Usability testing – this is how well the user can access the different features in the system and how easy it is to use.
- Software testing – this is to check if graphically that the program looks how was intended and the GUI works as intended.
- Navigation testing – this would be to check if pages are navigate the next page.
- Accessibility – how easy is it for various users including users with disability to use the system.
- Reliability testing – to check that the system works for long period of time and does not constantly crash.

Below is a full list that is shown on Wikipedia of all the different types of system testing that is available.

Unit Testing

In computer programming, **unit testing** is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

Test Cases

Image pre-processing module

Test Case ID	Input description	Input data	Expected Results	Actual Results
W-1	1.Uplode an image	Upload=Upload File	Successfully Uploaded	Pass
W-2	2.Image processed	Upload File=processed	Image processed successfully	Pass

W-3	3.RGB Image method	Processed file=Grayscale	RGB Image processed successfully	Pass
-----	--------------------	--------------------------	----------------------------------	------

Table 3.1 - Unit Test case using Image pre-processing

Car Brand Detection Module

Test Case ID	Input description	Input data	Expected Results	Actual Results
W-4	1. Check image format	Valid image format MUST be provided to continue	Input image accepted and display	Pass
W-5	2. Application should allow selecting any type of image.	Application should allow selecting any type of image, which a person want.	After choose any image, image is selected	Pass
W-6	3. Add car images from upload folder images	Application should allow to select car image for training of character of An image, which a person want to add.	Car image should add successfully.	Pass
W-7	4.Start Training	Start training Car image.	Training should be done and accurate.	Pass

Table 3.2 - Unit Test case using Car Brand Detection.

Integration Testing

After Unit Testing, software components are clubbed together in large aggregates and tested, to verify the proper functioning, performance and reliability between units, and expose any defect in the interface. This process is known as Integration Testing.

Test Cases

Car Brand Detection Module

Test Case ID	Input description	Input data	Expected Results	Actual Results
W-8	1. Check image format	Valid image format must be provided to continue	Input image accepted and display	Pass
W-9	2. Application should allow selecting any type of image.	Application should allow selecting any type of image, which A person want.	After choose any image, image is selected	Pass
W-10	3. Add Image from uploads	Application should allow to select car image for training of car image of An image, which a person want to add.	Car image should add successfully.	Pass
W-11	4.Start Training	Start training car image.	Training should be done and accurate.	Pass

Table 3.3 - Integration Test case using Car Brand Detection

4.2 Test Conditions

System testing is simply testing the system as a whole; it gets all the integrated modules of the various components from the integration testing phase and combines all the different parts into a system which is then tested.

4.2.1 Usability Testing

Image pre-processing module

Test Case ID	Input description	Input data	Expected Results	Actual Results
--------------	-------------------	------------	------------------	----------------

W-1	1.Uplode an image	Upload=Upload File	Successfully Uploaded	Pass
W-2	2.Image processed	Upload File=processed	Image processed successfully	Pass
W-3	3.RGB Image method	Processed file=Grayscale	RGB Image processed successfully	Pass

Table 3.4 - Usability Test case using Image pre-processing

Car Brand Identifier Module

Test Case ID	Input description	Input data	Expected Results	Actual Results
W-4	1.Uplode an image	Upload=Upload File	Successfully Uploaded	Pass
W-5	2.Image processed	Upload File=processed	Image processed successfully	Pass
W-6	3. RGB Image method	Processed file=RGB Image	RGB Image processed successfully	Pass
W-7	4.Identified your Car Brand	Letter=file upload	Successfully Identified your Car Brand	Pass
W-8	5.Car brand Identified	File=Car Brand identified	Your car brand has been identified	Pass

Table No: 3.5 - Usability Test case using Car Brand Detection

4.2.2 Software Testing

Image to Brand Converted

Test Case ID	Input description	Input data	Expected Results	Actual Results
---------------------	--------------------------	-------------------	-------------------------	-----------------------

W-9	1.Validate search button	Click Button=File Entered	Search button should be enabled	Pass
W-10	2.Uplode button	Upload File=processed	Upload successfully	Pass
W-11	3.OCR_Software	Accepting the image	Successfully accepted	Pass
W-12	4.Pytesseract_Software	Accepting the image	Successfully accepted	Pass

Table 3.6 - Software Test case using Image to Brand

Image pre-processing

Test Case ID	Input description	Input data	Expected Results	Actual Results
W-13	1.OCR_Software	Accepting the image	Successfully accepted	Pass
W-14	2.Pytesseract_Software	Accepting the image	Successfully accepted	Pass

Table 3.7 - Software Test case using Image pre-processing

4.2.3 Navigation Testing

Car Brand Identifier

Test Case ID	Input description	Input data	Expected Results	Actual Results
W-15	1.Search for query that generates at least 2 pages of output	Query=()	1. Search results in 2 or more pages of output. 2. Top-most images off top of screen and next images appears at the bottom of screen, except when near bottom of list.	Pass

W-16	1. Search for query that generates at least 2 pages of output. 2.Scroll down by one page	Query=()	1. Search results in two or more pages of output. 2. Current screen contents scrolls up and next page appears, except when at top of list or at bottom of list.	Pass
------	---	----------	--	------

Table 3.8 - Navigation Test case using Car Brand Detection Module

4.3 Technical Documentation

Documentation is the product of technical writing that refers to different documents with product-related data and information; system has the details about a technical product that is either under development or already in use. Create technical documentation with the help of other groups of professionals like SMEs, designers, developers, reviewers. Sometimes, product developers themselves create documentation for their products. Create technical documentation are to reduce customer tickets, the expenses on customer service and enable their support team to solve customer queries effectively. So, the main purpose of technical documentation is to help users achieve their goal using the product. It can be in the form of a printed or online manual, video, and the like.

4.3.1 Application Deployment

Software Environment

Anaconda IDE

Anaconda is a conditional free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and Travis Oliphant in 2012. As an Anaconda, Inc. product, it is also known as Anaconda Distribution or Anaconda Individual Edition, while other products from the company are Anaconda Team Edition and Anaconda Enterprise Edition, both of which are not free.

Package versions in Anaconda are managed by the package management system conda. This package manager was spun out as a separate open-source package as it ended up being useful on its own and for other things than Python. There is also a small, bootstrap version of Anaconda called Miniconda, which includes only conda, Python, the packages they depend on, and a small number of other packages.

Smart Python

With PyCharm, developers are assisted by an intelligent platform that aids them when it comes to code completions, inspections, error pinpointing, quick fixes, and more. This enables them to bolster their productivity as their codes can be automatically completed. Plus, they do not need to waste time scouring their codes for errors to rectify.

Spyder IDE

Spyder is an open source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder integrates with a number of prominent packages in the scientific Python stack, including NumPy, SciPy, Matplotlib, pandas, IPython, SymPy and Cython, as well as other open source software. It is released under the MIT license

Initially created and developed by Pierre Raybaut in 2009, since 2012 Spyder has been maintained and continuously improved by a team of scientific Python developers and the community.

Spyder is extensible with first- and third-party plugins, includes support for interactive tools for data inspection and embeds Python-specific code quality assurance and introspection instruments, such as Pyflakes, Pylint and Rope. It is available cross-platform through Anaconda, on Windows, on macOS through MacPorts, and on major Linux distributions such as Arch Linux, Debian, Fedora, Gentoo Linux, openSUSE and Ubuntu.

4.3.2 Installation of Software

Installation Procedure of Anaconda and Spyder IDE on Windows

Installing Python

Step 1) to download and install Python visit the official website of Python <http://www.python.org/downloads/> and choose your version. Person have chosen Python version 3.6.3

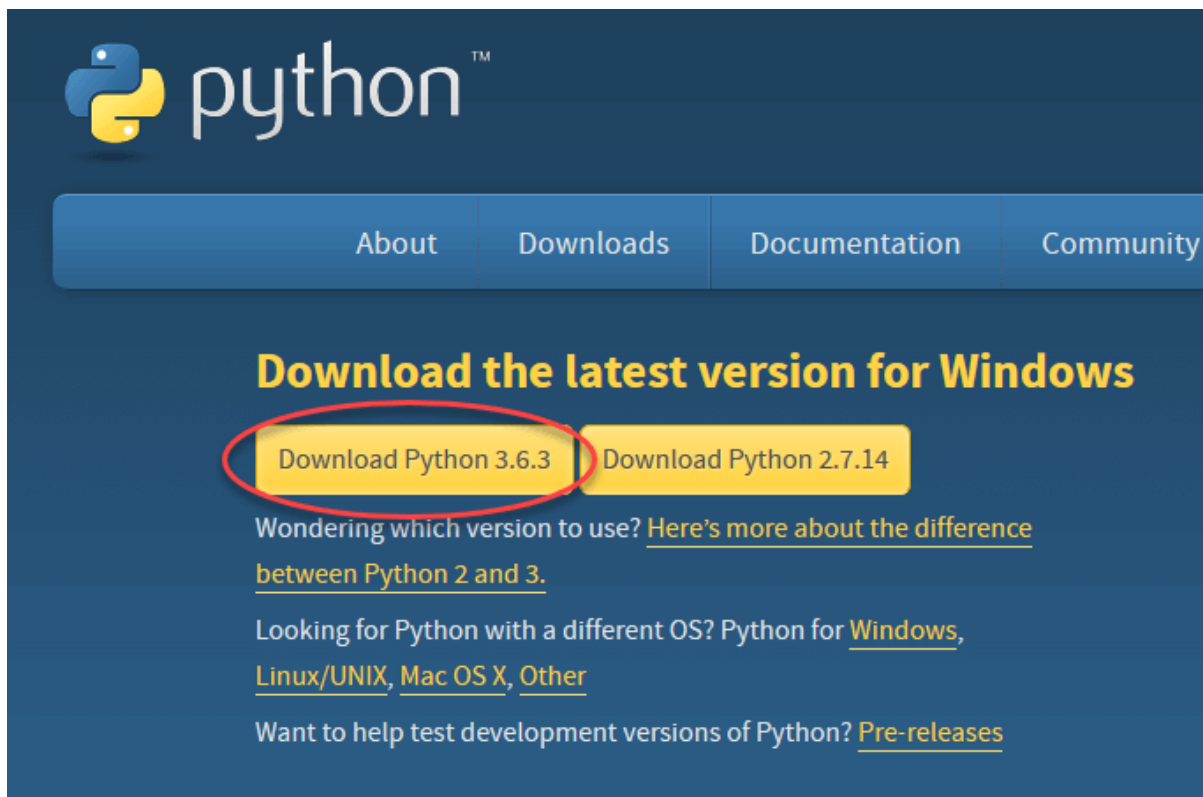
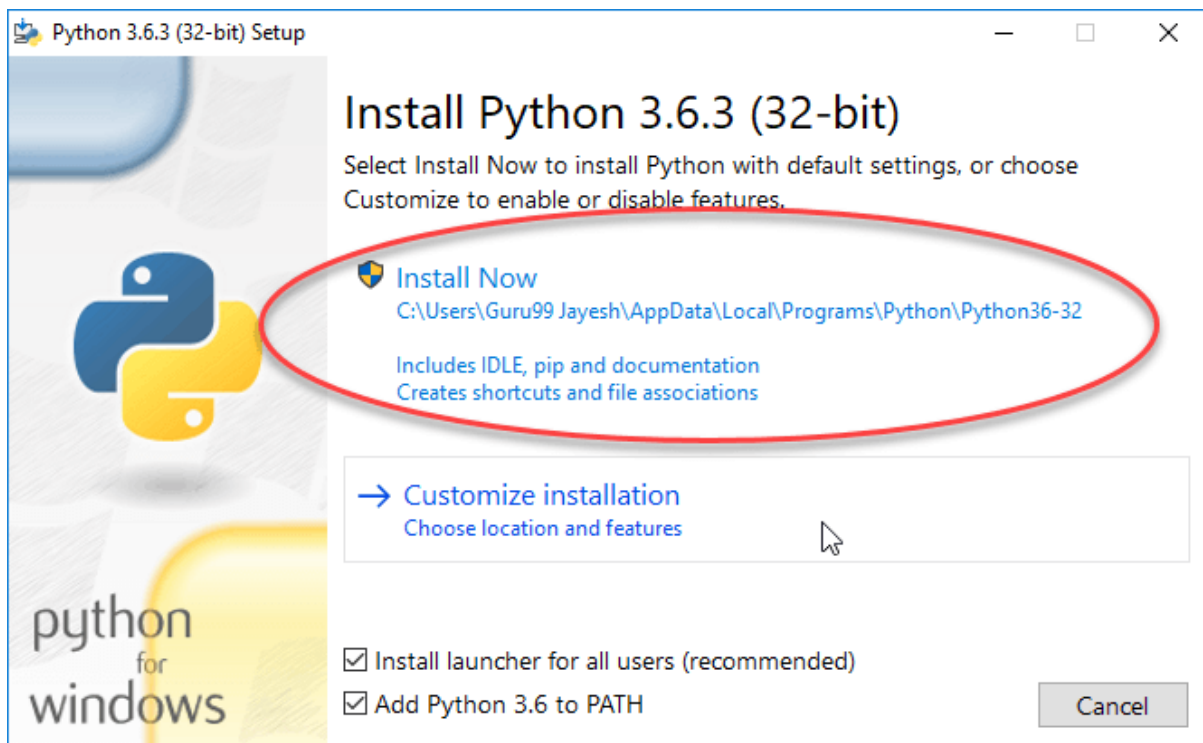


Figure 4.1

Step 2) once the download is complete, run the exe for install Python. Now click on Install Now.



Step 3) you can see Python installing at this point.

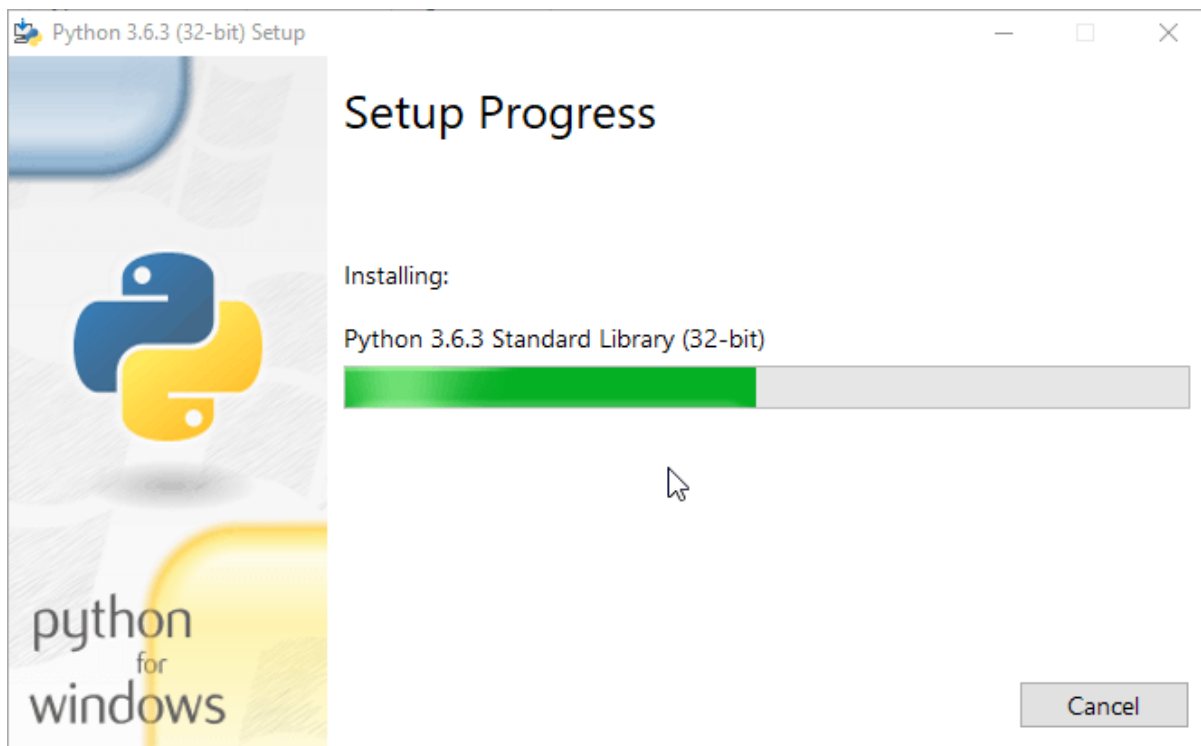


Figure 4.3

Step 4) When it finishes, you can see a screen that says the Setup was successful. Now click on "Close".

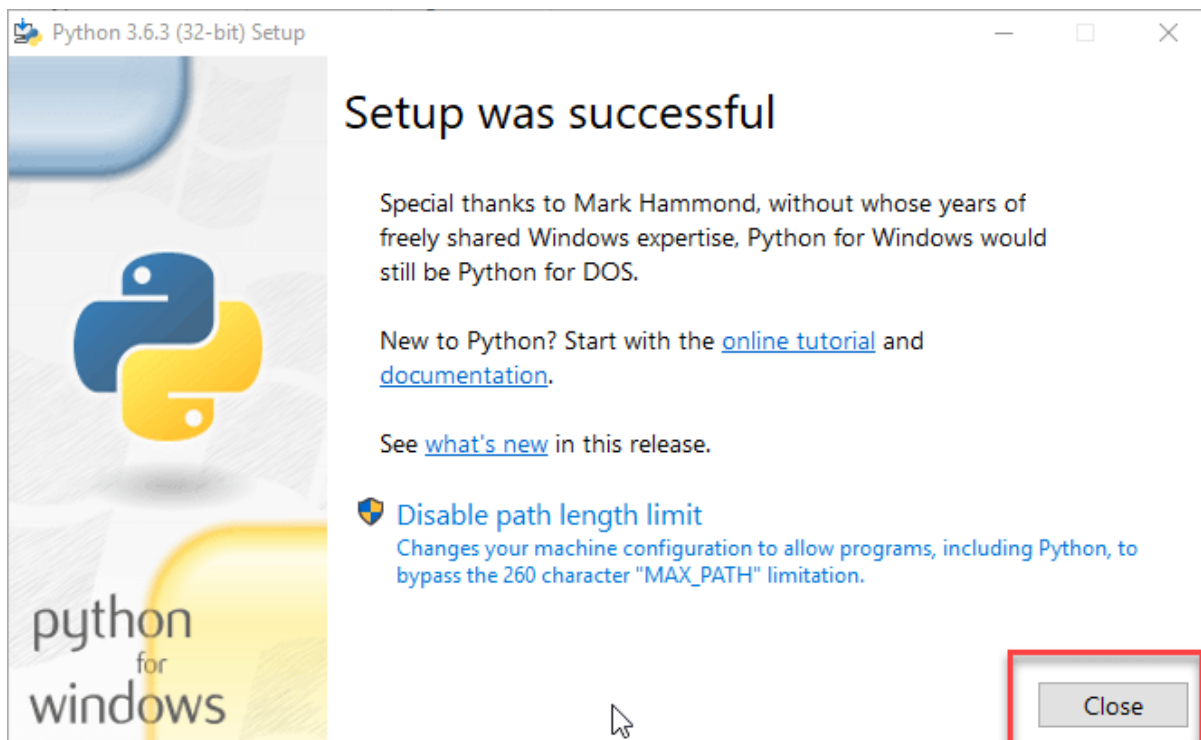


Figure 4.4

Download and Install Anaconda

1. Go to the Anaconda Website and choose a Python 3.x graphical installer (A) or a Python 2.x graphical installer (B). If you aren't sure which Python version you want to install, choose Python 3. Do not choose both.

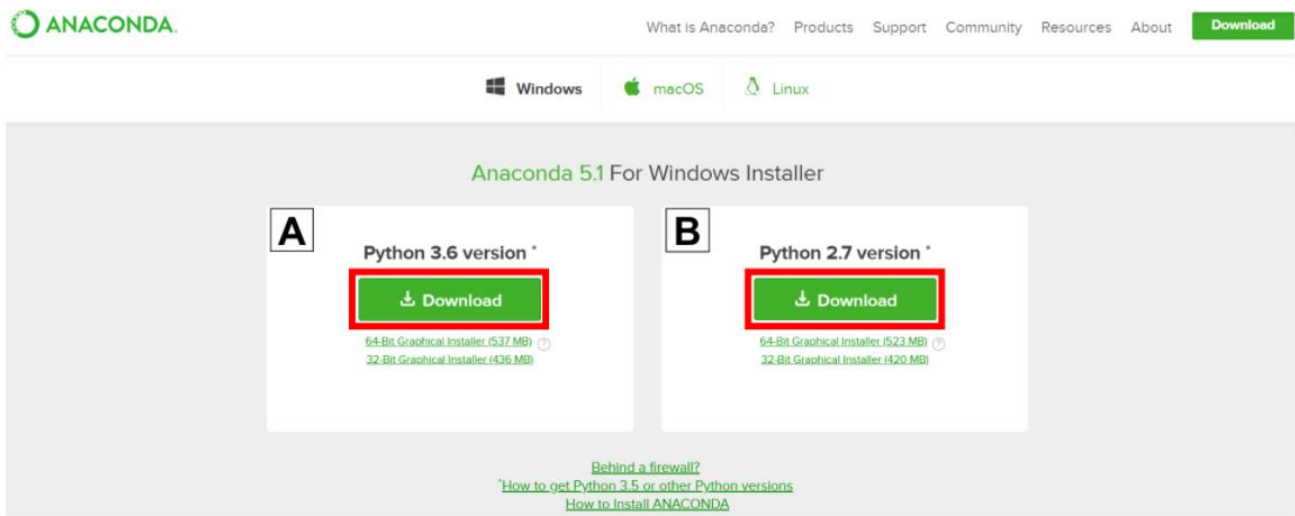


Figure 4.5

- 2) Locate your download and double click it.

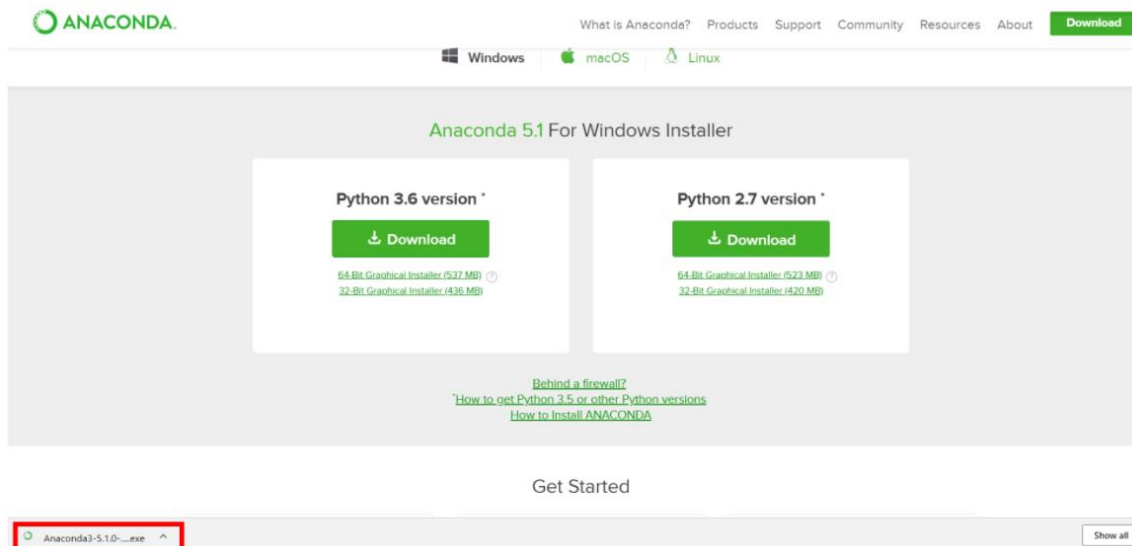


Figure 4.6

When the screen below appears, click on Next.



Figure 4.7

Read the license agreement and click on I Agree

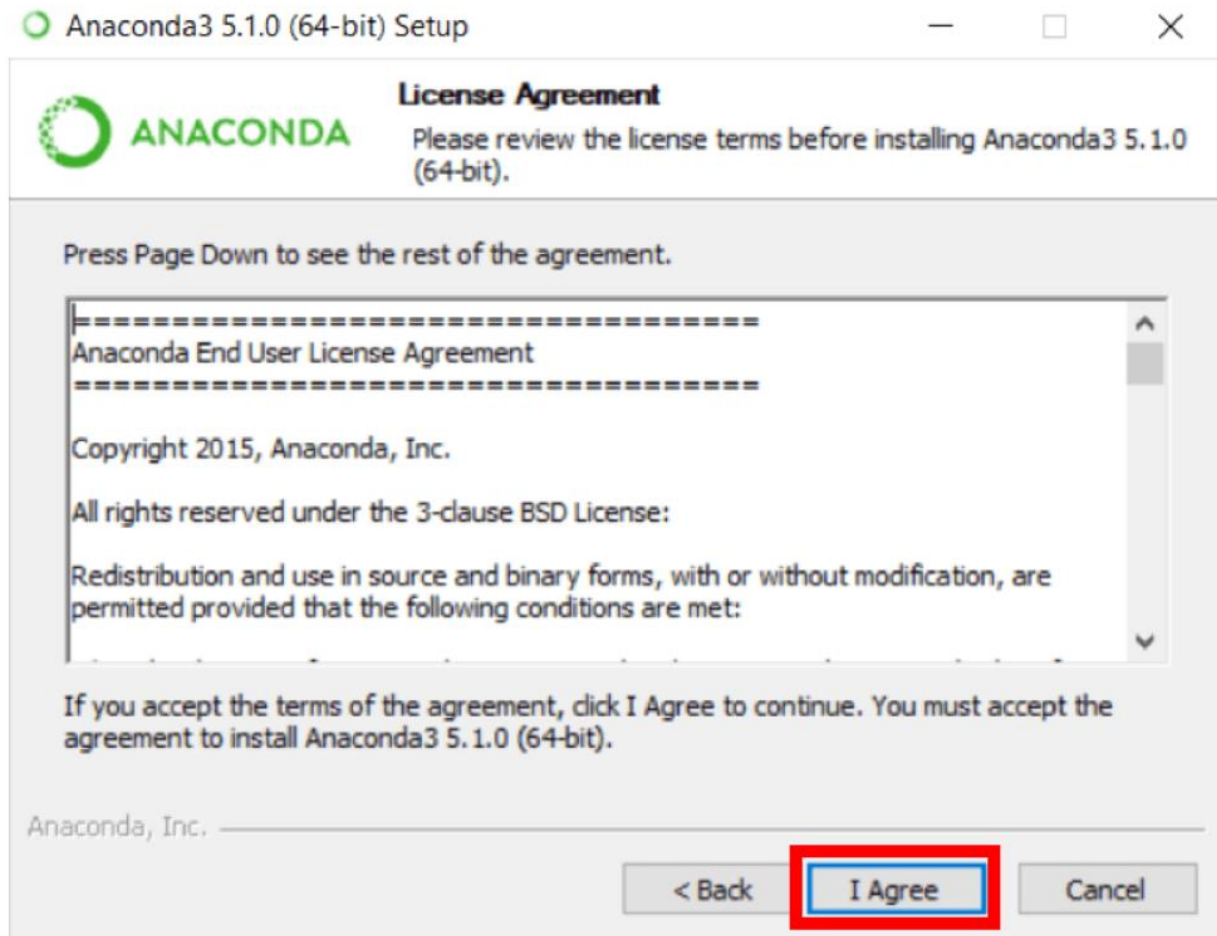


Figure 4.8

Click on Next.

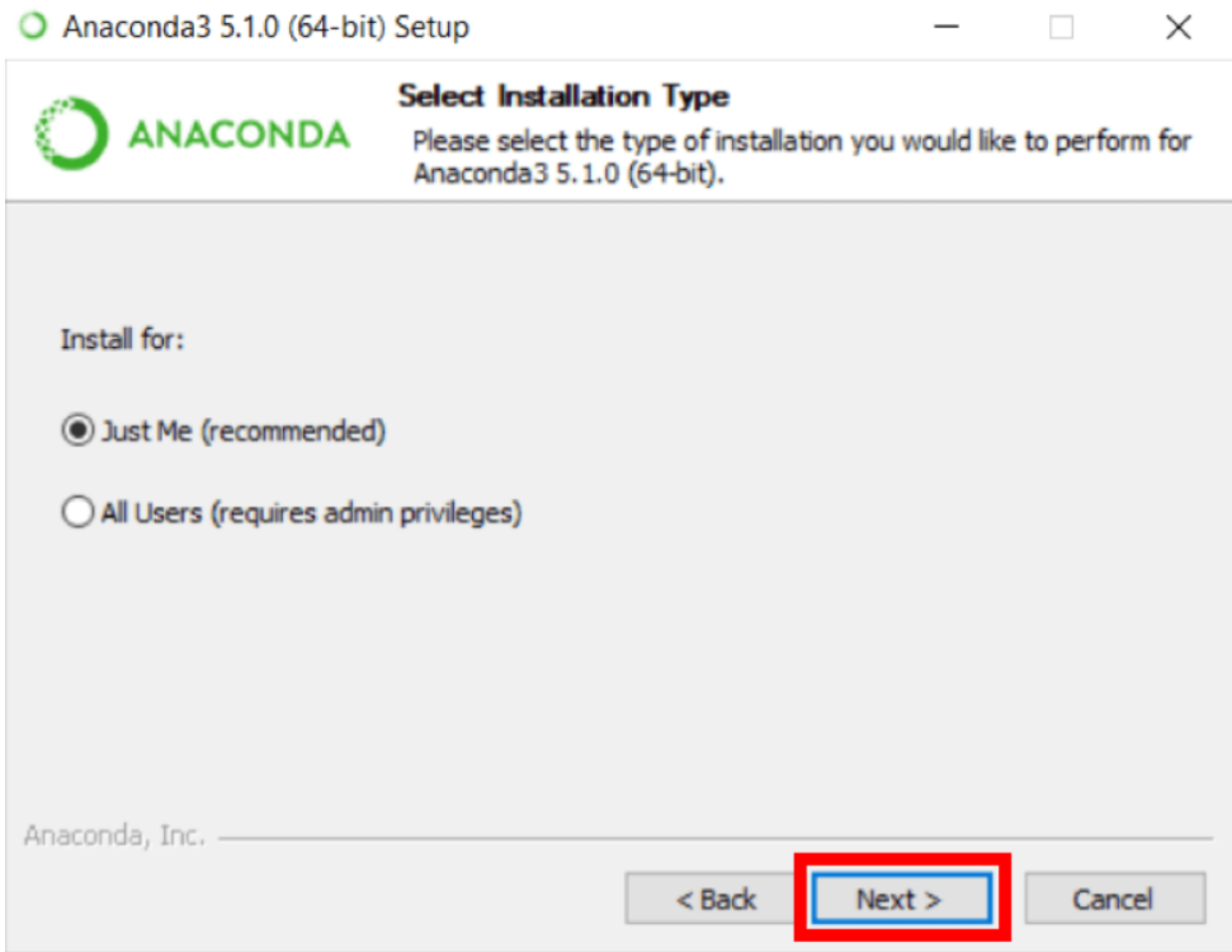


Figure 4.9

Note your installation location and then click Next.

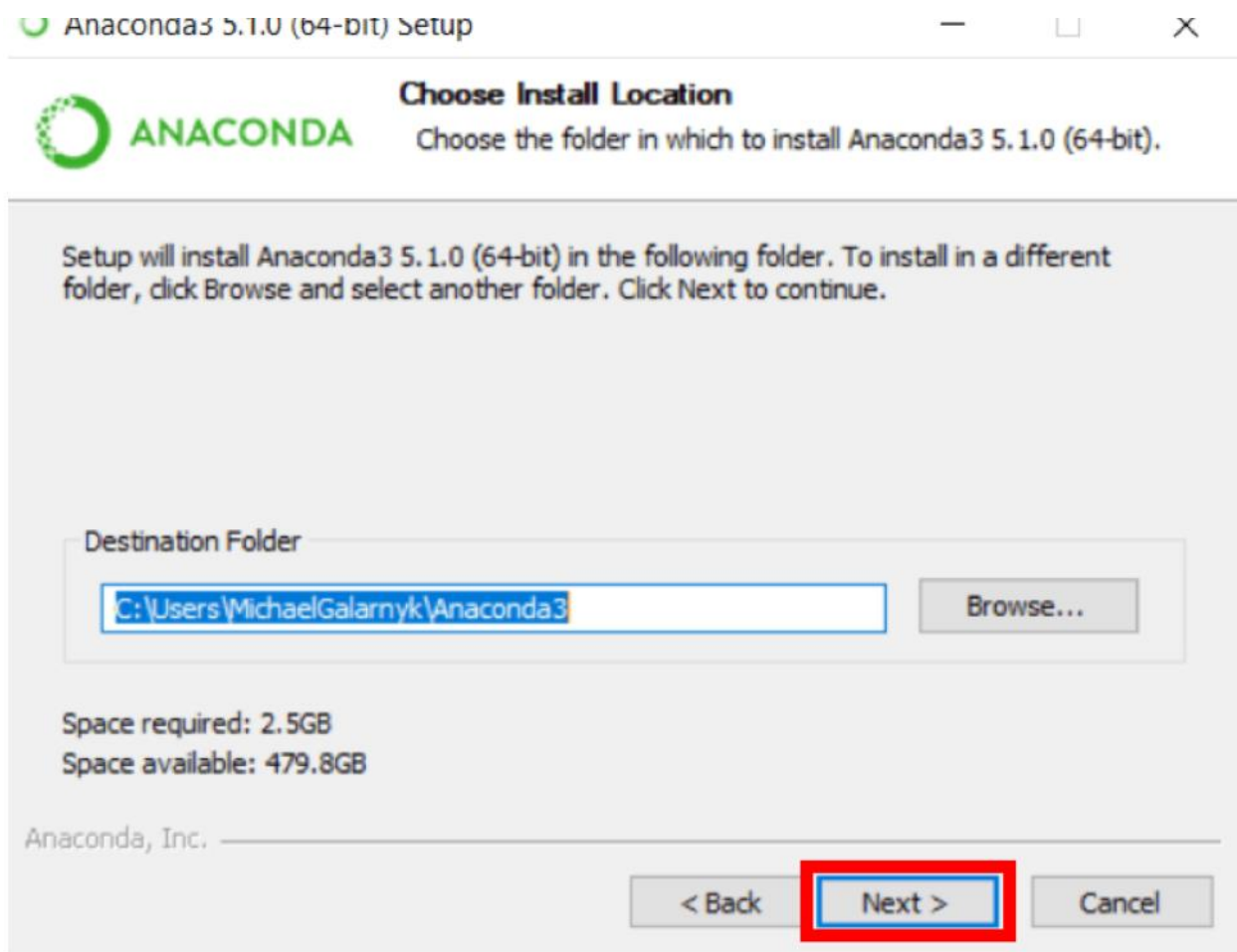


Figure 4.9

This is an important part of the installation process. The recommended approach is to not check the box to add Anaconda to your path. This means you will have to use Anaconda Navigator or the Anaconda Command Prompt (located in the Start Menu under "Anaconda") when you wish to use Anaconda (you can always add Anaconda to your PATH later if you don't check the box). If you want to be able to use Anaconda in your command prompt (or git bash, cmd, powershell etc), please use the alternative approach and check the box.

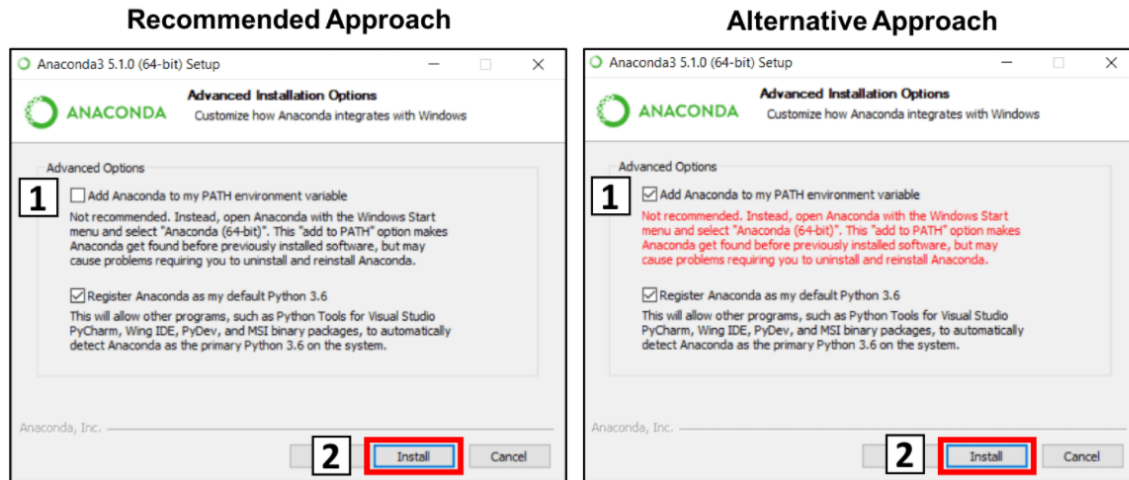


Figure 4.10

7. Click on Next.

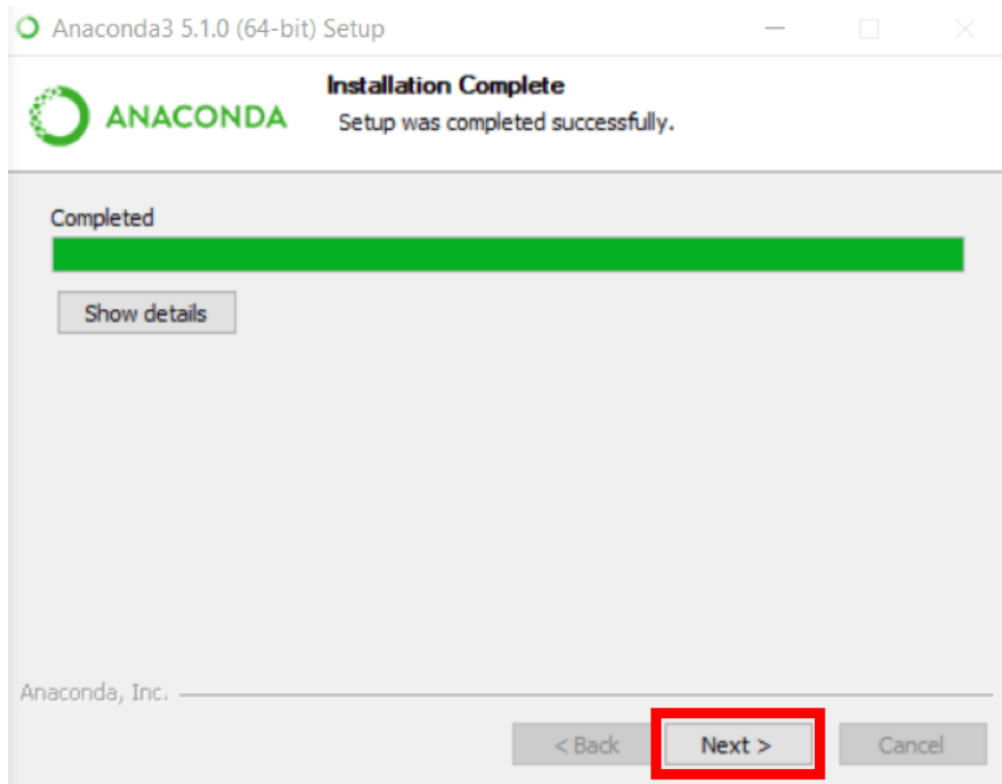


Figure 4.11

You can install Microsoft VSCode if you wish, but it is optional

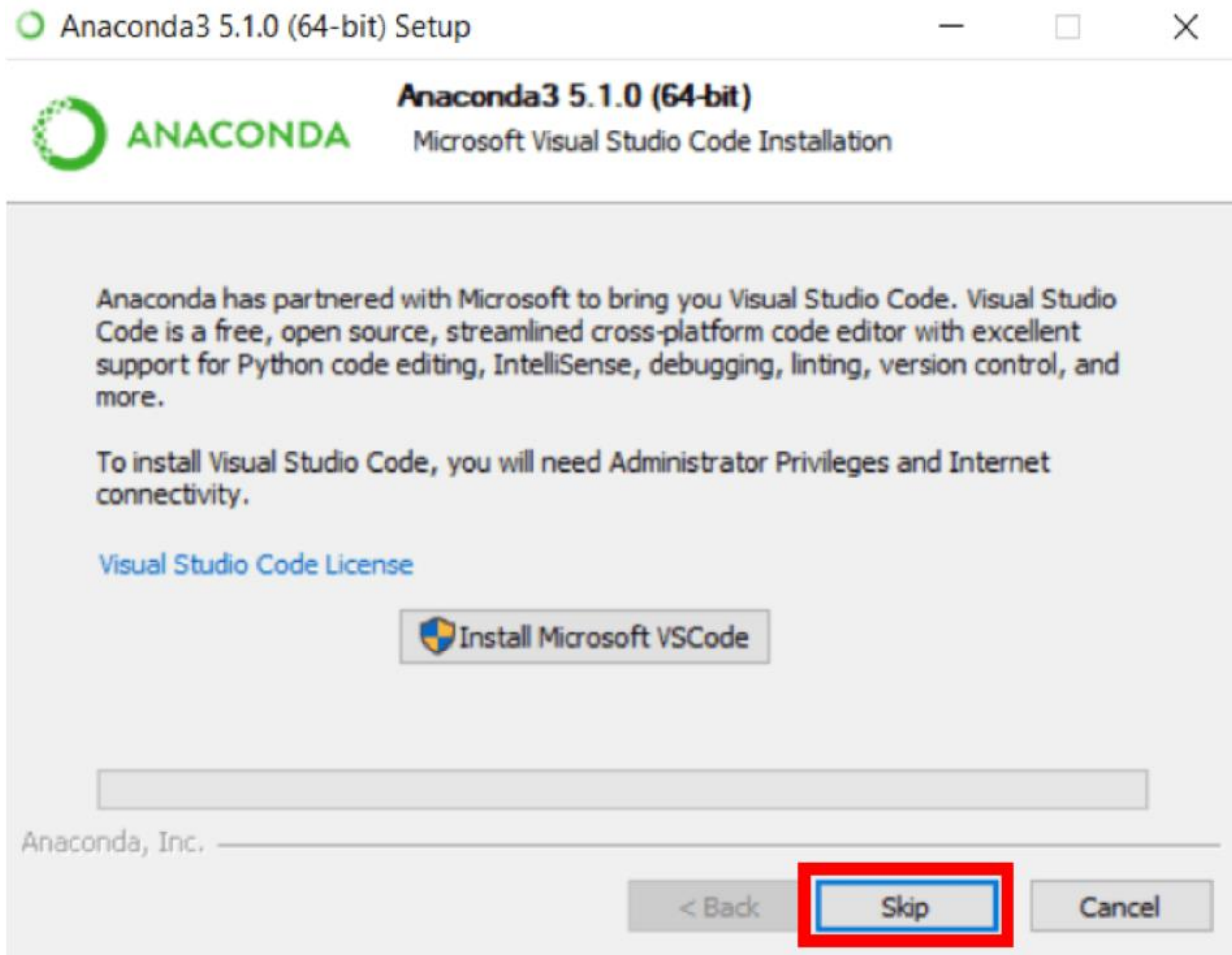


Figure 4.12

Click on Finish

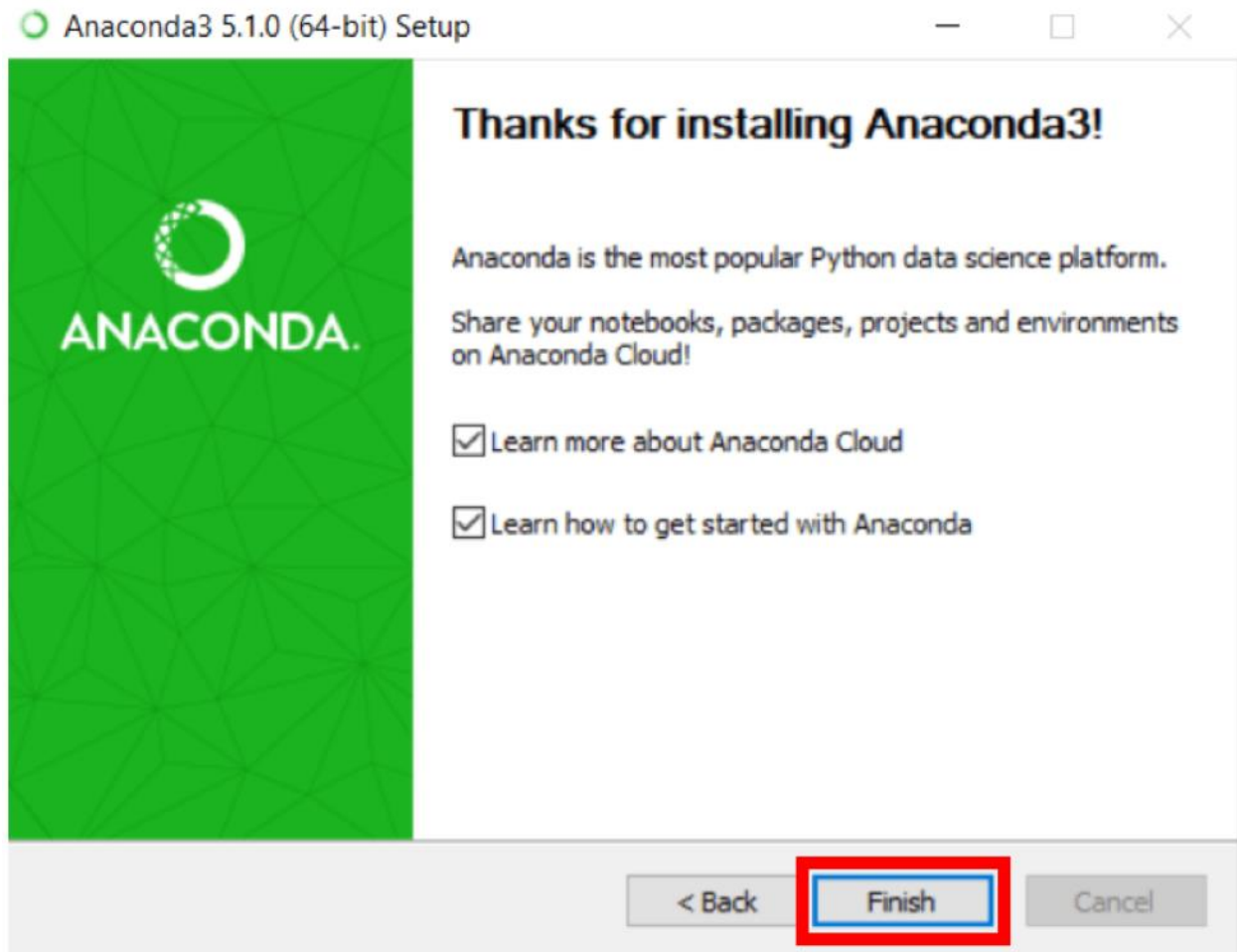


Figure 4.13

Once you will installed Anaconda the Sypder IDE is Available inside the Anaconda IDE

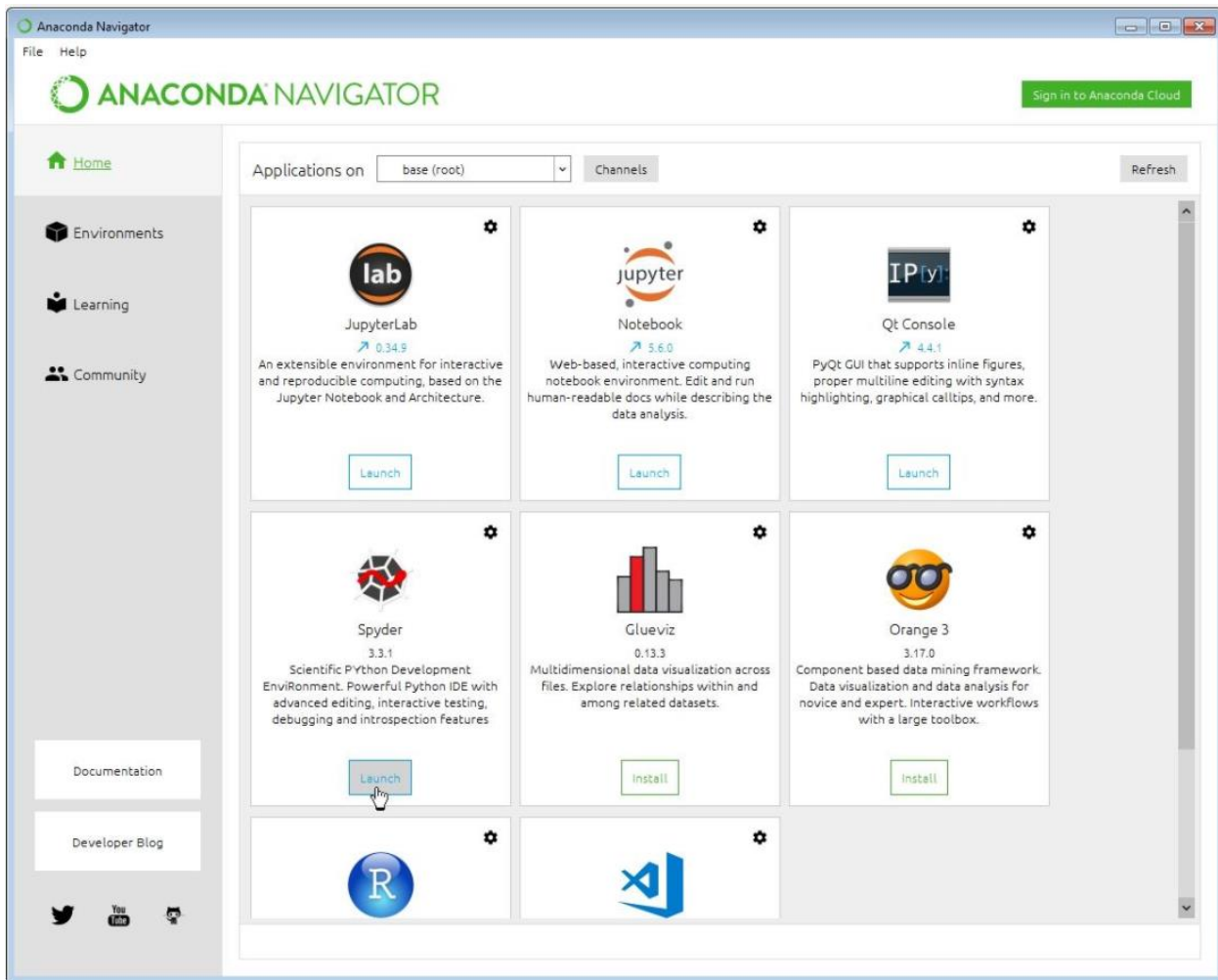


Figure 4.14

If you'd like run **Spyder**, just click on its **Launch** button. The IDE will open.

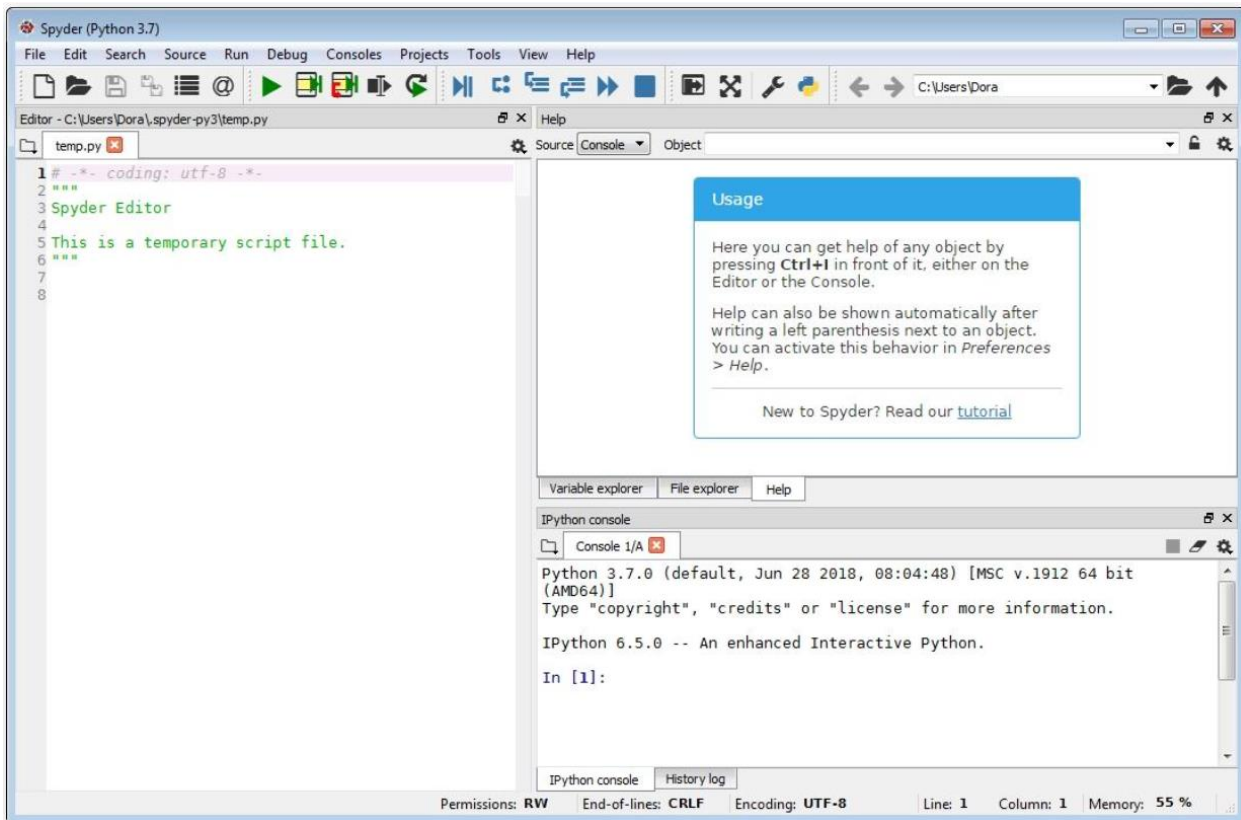


Figure 4.15

4.4 User Manual Documentation

Car Brand Detection Module

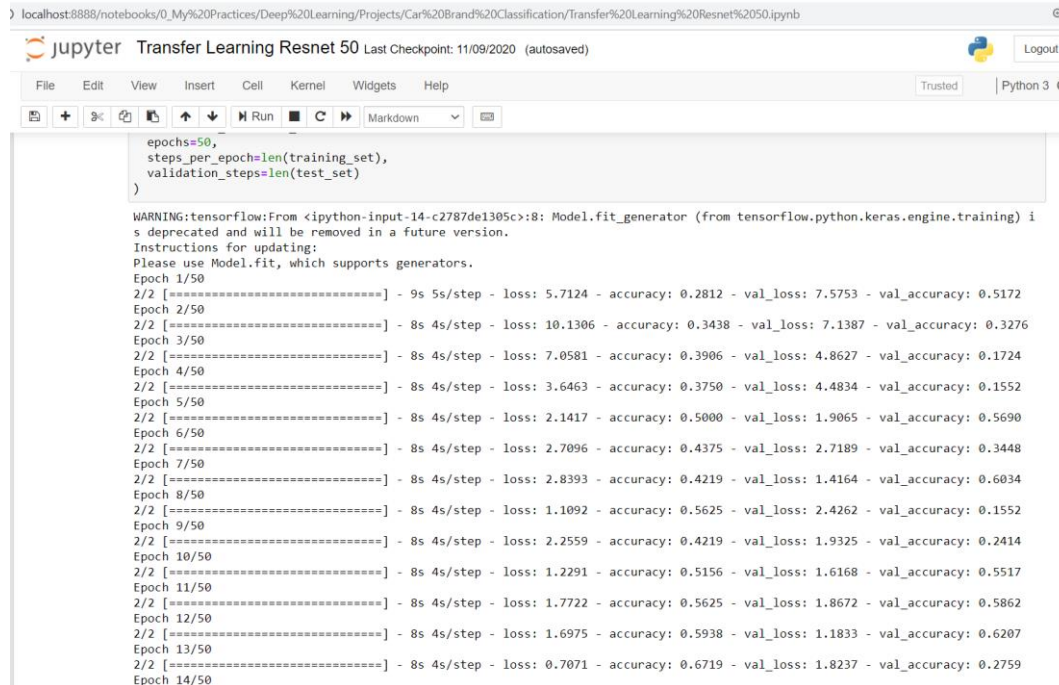
4.4.1 Introduction

The User Manual contains all essential information for the user to make full use of the information system. This manual includes a description of the system functions and capabilities, contingencies and alternate modes of operation, and step-by-step procedures for system access and use. Use graphics where possible in this manual. The manual format may be altered if another format is more suitable for the particular project.

4.4.2 Purpose and Scope

This section provides a description of the purpose and scope of the User Manual. The main purpose of the project detecting various kinds of car brands just capturing images.

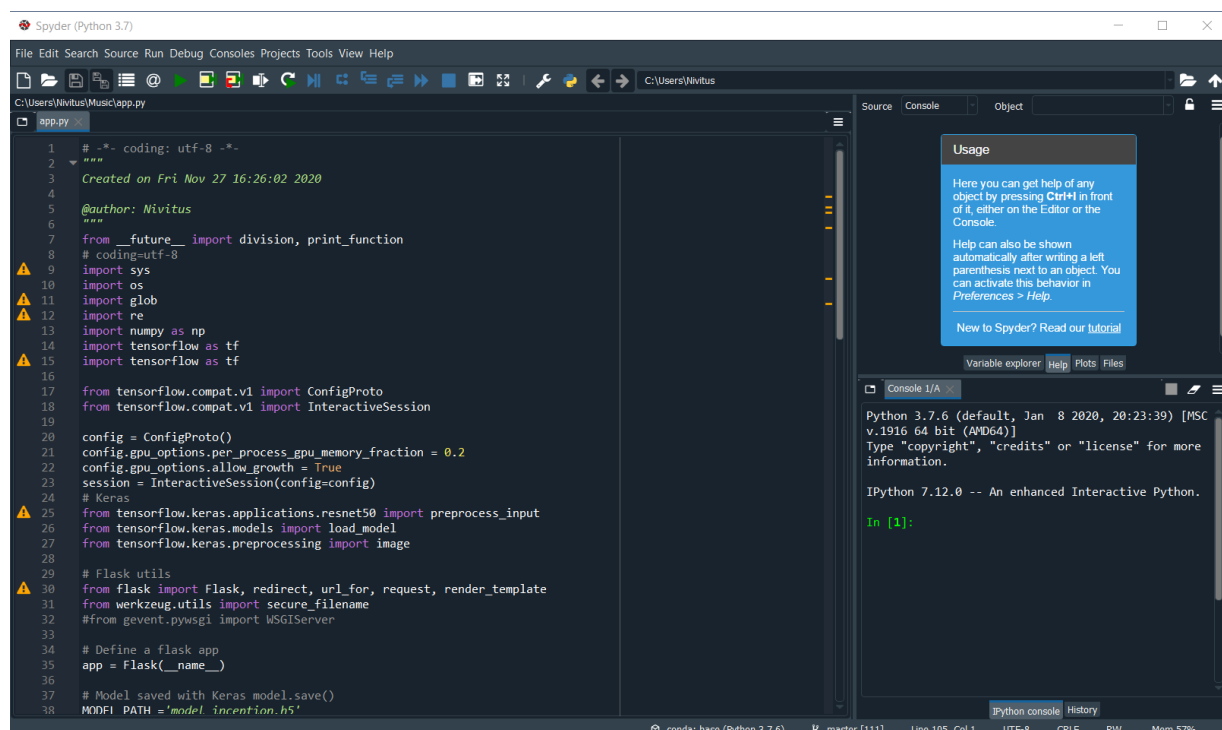
4.4.3 Car Brand Detection Model in CNN



```
epochs=50,
steps_per_epoch=len(training_set),
validation_steps=len(test_set)
)

WARNING:tensorflow:From <ipython-input-14-c2787de1305c>:8: Model.fit_generator (from tensorflow.python.keras.engine.training) is
deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/50
2/2 [=====] - 9s 5s/step - loss: 5.7124 - accuracy: 0.2812 - val_loss: 7.5753 - val_accuracy: 0.5172
Epoch 2/50
2/2 [=====] - 8s 4s/step - loss: 10.1306 - accuracy: 0.3438 - val_loss: 7.1387 - val_accuracy: 0.3276
Epoch 3/50
2/2 [=====] - 8s 4s/step - loss: 7.0581 - accuracy: 0.3906 - val_loss: 4.8627 - val_accuracy: 0.1724
Epoch 4/50
2/2 [=====] - 8s 4s/step - loss: 3.6463 - accuracy: 0.3750 - val_loss: 4.4834 - val_accuracy: 0.1552
Epoch 5/50
2/2 [=====] - 8s 4s/step - loss: 2.1417 - accuracy: 0.5000 - val_loss: 1.9065 - val_accuracy: 0.5690
Epoch 6/50
2/2 [=====] - 8s 4s/step - loss: 2.7096 - accuracy: 0.4375 - val_loss: 2.7189 - val_accuracy: 0.3448
Epoch 7/50
2/2 [=====] - 8s 4s/step - loss: 2.8393 - accuracy: 0.4219 - val_loss: 1.4164 - val_accuracy: 0.6034
Epoch 8/50
2/2 [=====] - 8s 4s/step - loss: 1.1092 - accuracy: 0.5625 - val_loss: 2.4262 - val_accuracy: 0.1552
Epoch 9/50
2/2 [=====] - 8s 4s/step - loss: 2.2559 - accuracy: 0.4219 - val_loss: 1.9325 - val_accuracy: 0.2414
Epoch 10/50
2/2 [=====] - 8s 4s/step - loss: 1.2291 - accuracy: 0.5156 - val_loss: 1.6168 - val_accuracy: 0.5517
Epoch 11/50
2/2 [=====] - 8s 4s/step - loss: 1.7722 - accuracy: 0.5625 - val_loss: 1.8672 - val_accuracy: 0.5862
Epoch 12/50
2/2 [=====] - 8s 4s/step - loss: 1.6975 - accuracy: 0.5938 - val_loss: 1.1833 - val_accuracy: 0.6207
Epoch 13/50
2/2 [=====] - 8s 4s/step - loss: 0.7071 - accuracy: 0.6719 - val_loss: 1.8237 - val_accuracy: 0.2759
Epoch 14/50
```

Figure 4.16



```
1  #-*- coding: utf-8 -*-
2  """
3  Created on Fri Nov 27 16:26:02 2020
4
5  @author: Nivitus
6  """
7  from _future_ import division, print_function
8  # coding=utf-8
9  import sys
10 import os
11 import glob
12 import re
13 import numpy as np
14 import tensorflow as tf
15 import tensorflow as tf
16
17 from tensorflow.compat.v1 import ConfigProto
18 from tensorflow.compat.v1 import InteractiveSession
19
20 config = ConfigProto()
21 config.gpu_options.per_process_gpu_memory_fraction = 0.2
22 config.gpu_options.allow_growth = True
23 session = InteractiveSession(config=config)
24 # Keras
25 from tensorflow.keras.applications.resnet50 import preprocess_input
26 from tensorflow.keras.models import load_model
27 from tensorflow.keras.preprocessing import image
28
29 # Flask utils
30 from flask import Flask, redirect, url_for, request, render_template
31 from werkzeug.utils import secure_filename
32 #from gevent.pywsgi import WSGIServer
33
34 # Define a flask app
35 app = Flask(__name__)
36
37 # Model saved with Keras model.save()
38 MODEL_PATH = "model_inception.h5"
```

Flask file for detecting car brands - app.py Figure 4.17

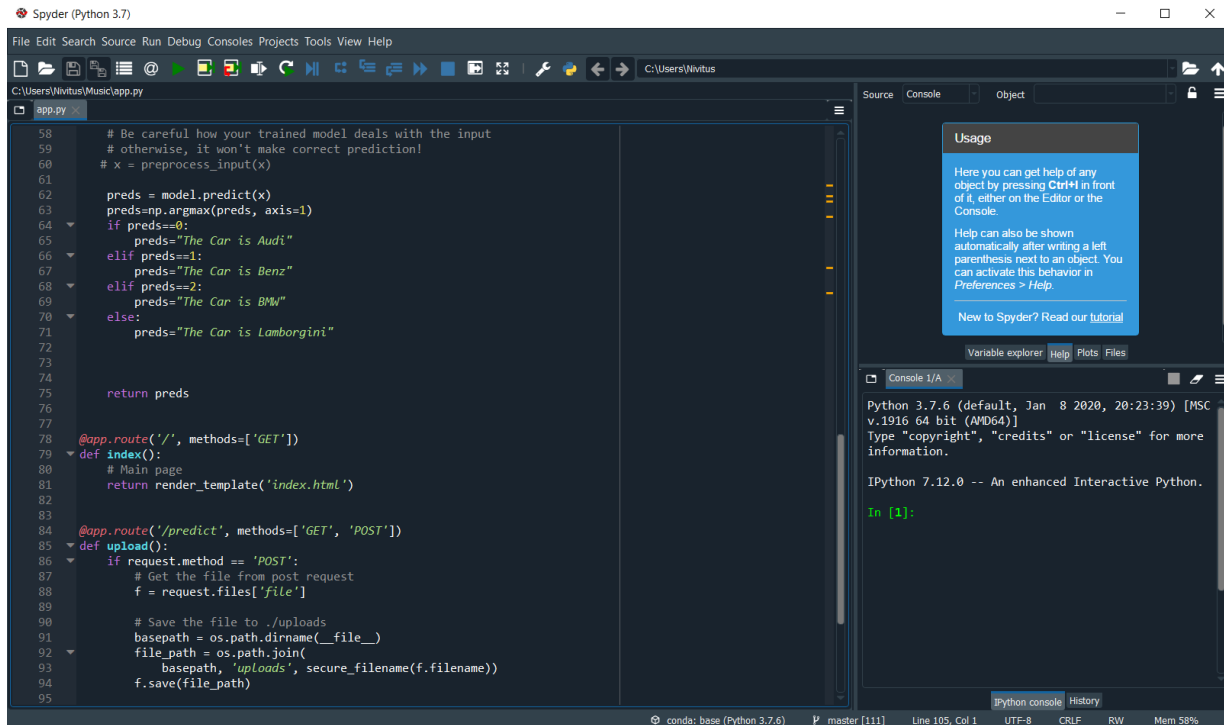


Figure 4.18

4.4.4 How to use this Application

- Step 1:** Select the car image from your pc or whatever
- Step2:** After once image will display the image container.
- Step 3:** Press the predict button.
- Step 4:** Result will show under the image container.

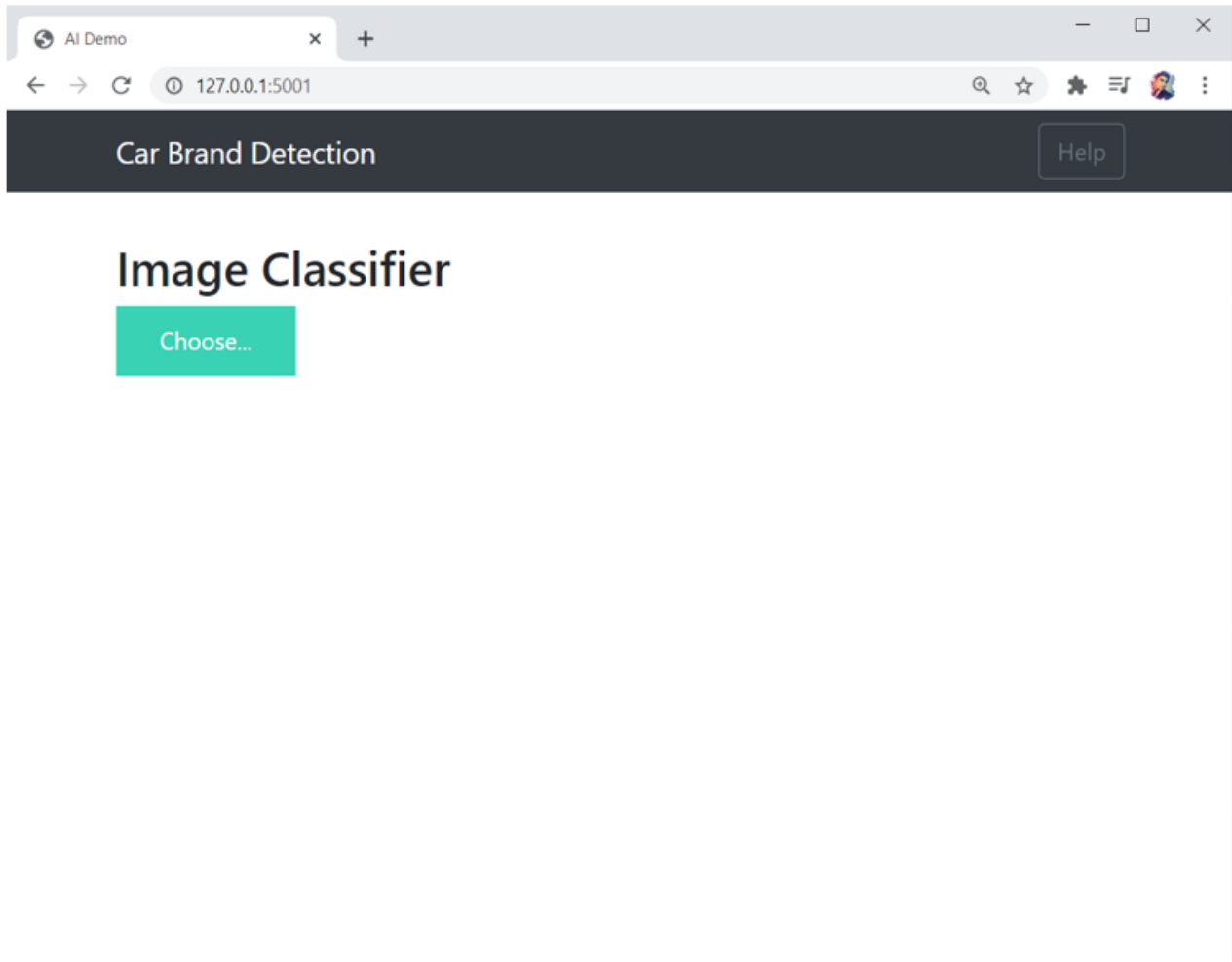


Figure 4.20

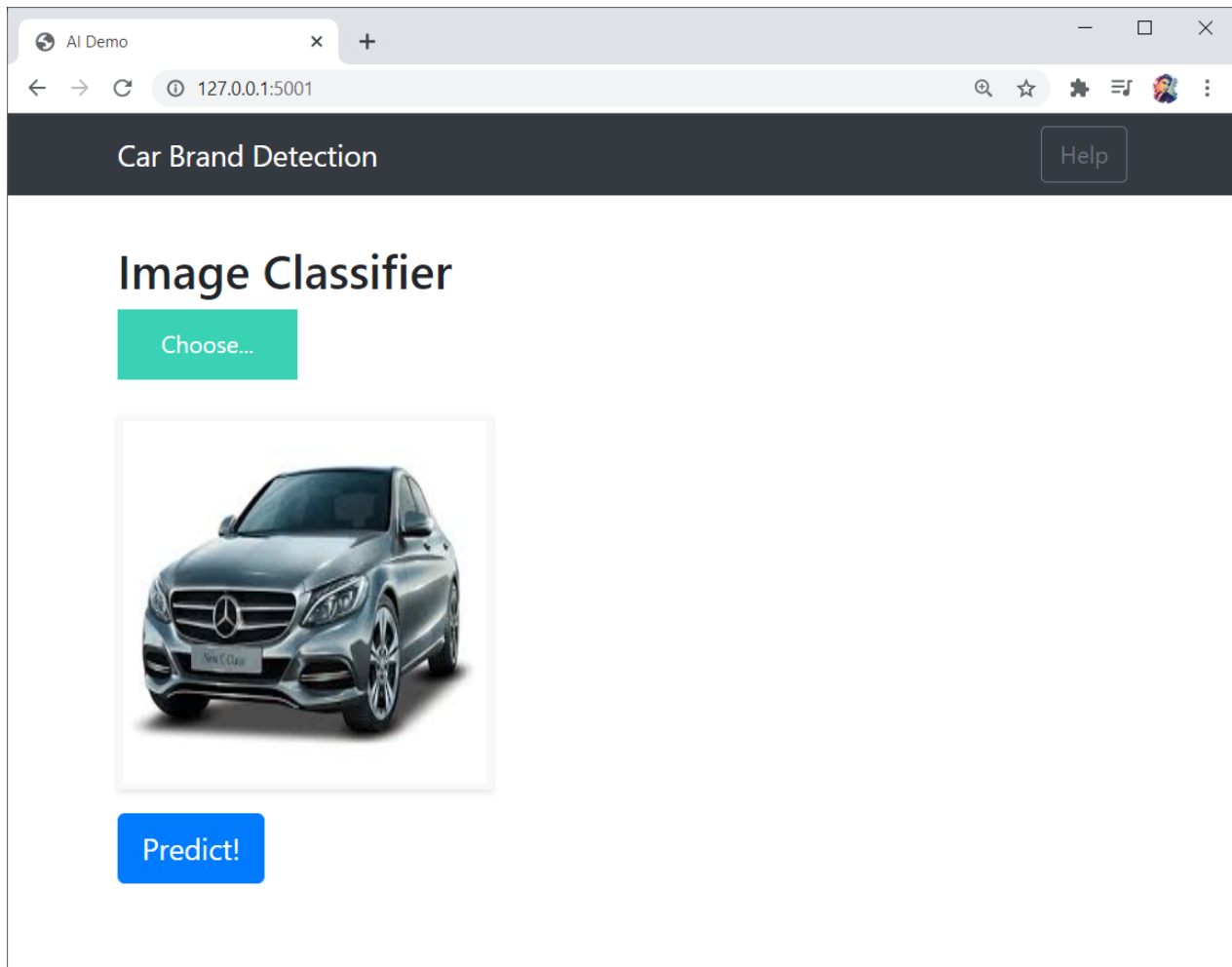
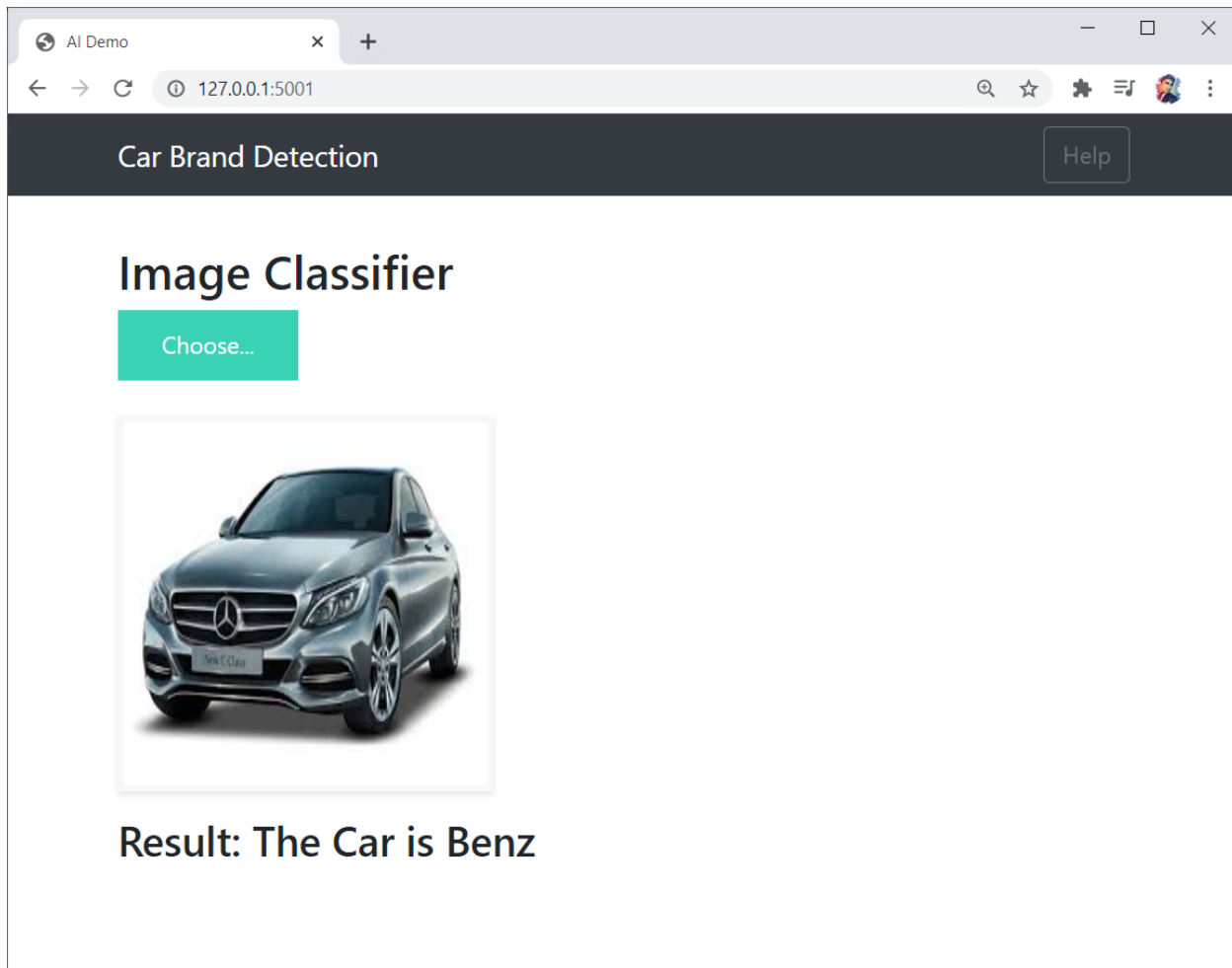


Image pre-processing Figure 4.21



Brand Detecting Figure 4.22

4.5 Conclusion

The application was tested under different testing strategies which was detecting various kind of images of cars. Various Test cases were considered on both the requirement and user expectation basis. Both Technical and User document were created with detailed pasteurization of how the application was developed / deployed and how to use the application from user's point of view.

CHAPTER 5

CONCLUSION

5.0 Introduction

This chapter deals with the evaluation of the entire application in various views, future enhancements which could be incorporated and the technical skills which the developer has learnt in the process of developing this application. This chapter closes the entire report document of the mini project handled.

5.1 Evaluation

Evaluation plays an important role as this is where the fulfillment on completion of each module which in turn checks the objectives of each and every phase. Here User processed evaluation of 2 methods such as evaluate existing images and real-world images.

Evaluate on Existing Images

Based on Existing images result was pretty good. But while I train and test the existing images into the model it was gave good amount of accuracy. System has the power for detecting the cars brands which are really important weather a car is original or duplicate. But sometimes it was detecting another car brand which is not suitable for respective car brand.

Evaluate on real-world Images

Based on Real-world images this model work pretty good. When I am saying about the night time images it will also predict or detect the respective car brands accurately. People want to use this application weather car images are real world or google images for detecting the respective brands.

Below are 3 perspectives on the satisfactory regards along with a 3 grade scale.

5.1.1 UI/UX

Questions	Evaluation		
	Good	Moderate	Poor
Would you like this Application?	✓		
How was the design about the Application?		✓	
Would you like this Overall Design?	✓		
Does the design pattern resemble in all modules?	✓		
How stable is the font size of each label?	✓		
Response time for each click and other events?	✓		
Ease of using the application in design regards	✓		
How this Application is Responsive?	✓		
How appealing are the colors in the application?		✓	
Overall Look of the application?		✓	

Table 5.1

5.1.2 Functionality check

Question description	Evaluation		
	Good	Moderate	Poor
How this application was work?	✓		
Mentioned the speed of this application?	✓		
Is it work well for images?	✓		
Is it work well for real-world images?	✓		
Is it well for night time images?	✓		
Would you like the application functionality?		✓	
Is it working well in platform independent?		✓	
Overall check on user view portal	✓		
Displaying images is good quality		✓	
It is working various car brands?	✓		

Table 5.2

5.1.3 Data Source / Objectives

Question description	Evaluation		
	Good	Moderate	Poor
How this application work as web app?	✓		
Is it capturing images easy for you?	✓		
Is it working for all format of images?	✓		
Is it working as mobile application?	✓		
Is it working for videos?			✓
Would you like the overall objectives?	✓		
Is it find car brand logo?	✓		
Overall data quality of this project	✓		
Displaying brand name?	✓		
It is working various car logo at night time also?	✓		

Table 5.3

5.2 Future Enhancements

Convolutional neural networks perform well and are innovative in the field of image analysis, including object detection, instance segmentation, and object tracking. Combined with large datasets, CNN outperforms traditional machine learning techniques. In vehicle detection, person have proposed a good detection model for two-dimensional vehicle detection, which means a person only compute the two-dimensional bounding box of specific objects. People are getting more interested in three-dimensional vehicle detection, which is a great technique for autonomous

driving. A popular method in 3D vehicle detection is first computing the 2D bounding box by YOLO or Faster R-CNN, and then determining the dimension and orientation of the objects that are used for the determination of the location of 3D objects. In future days in this project should be integrated with any IOT devices for classifying and detecting real time cars such as help us to prevent threat in road traffic analysis. As well as in upcoming days it will deploy as mobile app using tensorflow.js for using this app anywhere anytime.

5.3 Technical skills learn

In this project I learn most important objective is that the growth of artificial intelligence (AI) has inspired more software engineers, data scientists, and other professionals to explore the possibility of a career in machine learning. Apart from that User learn most of the new technologies and its techniques which is really help me to complete this project. These are the skills which are User to learn while I am developing this project.

Skill Name	Area	Self-Evaluate
Machine Learning	Artificial Intelligence	Good
Deep Learning	Artificial Intelligence	Moderate
Transfer Learning	Artificial Intelligence	Moderate
Hyper-parameter Tuning	Artificial Intelligence	Moderate
CNN	Web Based	Moderate
Flask	Web Based	Moderate
Django	Web Based	Moderate
JavaScript	Web Based	Moderate
NodeJS	Web Based	Moderate
Git	Container Tool	Good
Docker	Container Tool	Moderate
Tensorflow	AI Libraries	Moderate
Keras	AI Libraries	Moderate
Python	Programming Language	Good

Table 5.1

5.4 Conclusion

This thesis introduces and improves basic deep learning components that are important in vehicle detection. The implementations and results demonstrate that deep learning techniques have great Performance in vehicle detection. The proposed model improves the performance and saves testing time. By incorporating extended RPN, the model is more robust to handle the large variation of vehicle scales. This application has satisfied all the aforesaid objectives and functionalities. Evaluation of the entire application is done in terms of various aspects and possible future enhancements are also attached, so that the same could be referred for better clarity.

BIBLIOGRAPHY

- [1] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards realtime object detection with region proposal networks." In *Advances in Neural Information Processing Systems*, pp. 91-99, 2015.
- [2] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. "Vision meets robotics: The KITTI dataset." *The International Journal of Robotics Research*, 32, no. 11, 2013: 1231-1237.
- [3] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick and Ali Farhadi. "You Only Look Once: Unified, Real-Time Object Detection." *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016: 779-788.
- [4] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, ChengYang Fu and Alexander C. Berg. "SSD: Single Shot MultiBox Detector." *European Conference on Computer Vision (ECCV)*, 2016.
- [5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. "ImageNet classification with deep convolutional neural networks." In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'12)*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.), Vol. 1. Curran Associates Inc., USA, 1097-1105.
- [6] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for LargeScale Image Recognition." *Computing Research Repository (CoRR)*, abs/1409.1556, 2014.
- [7] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. "Deep Residual Learning for Image Recognition." *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016: 770-778.
- [8] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke and Andrew Rabinovich. "Going deeper with convolutions." *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

APPENDIX

Data Source



Lamborghini



Audi

The Images are collected by from Google images as well as some other resources.

Screenshots

MY PROJECTS MACHINE LEARNING DEEP LEARNING NLP COMPUTER VISION

DEEP LEARNING PROJECTS

Project 1

Car Brand Detection using CNN

In this project we are going to do classifying or Detecting the car brands based on some of the car images using CNN algorithms and, used on some of the state-of-the-art- algorithms.

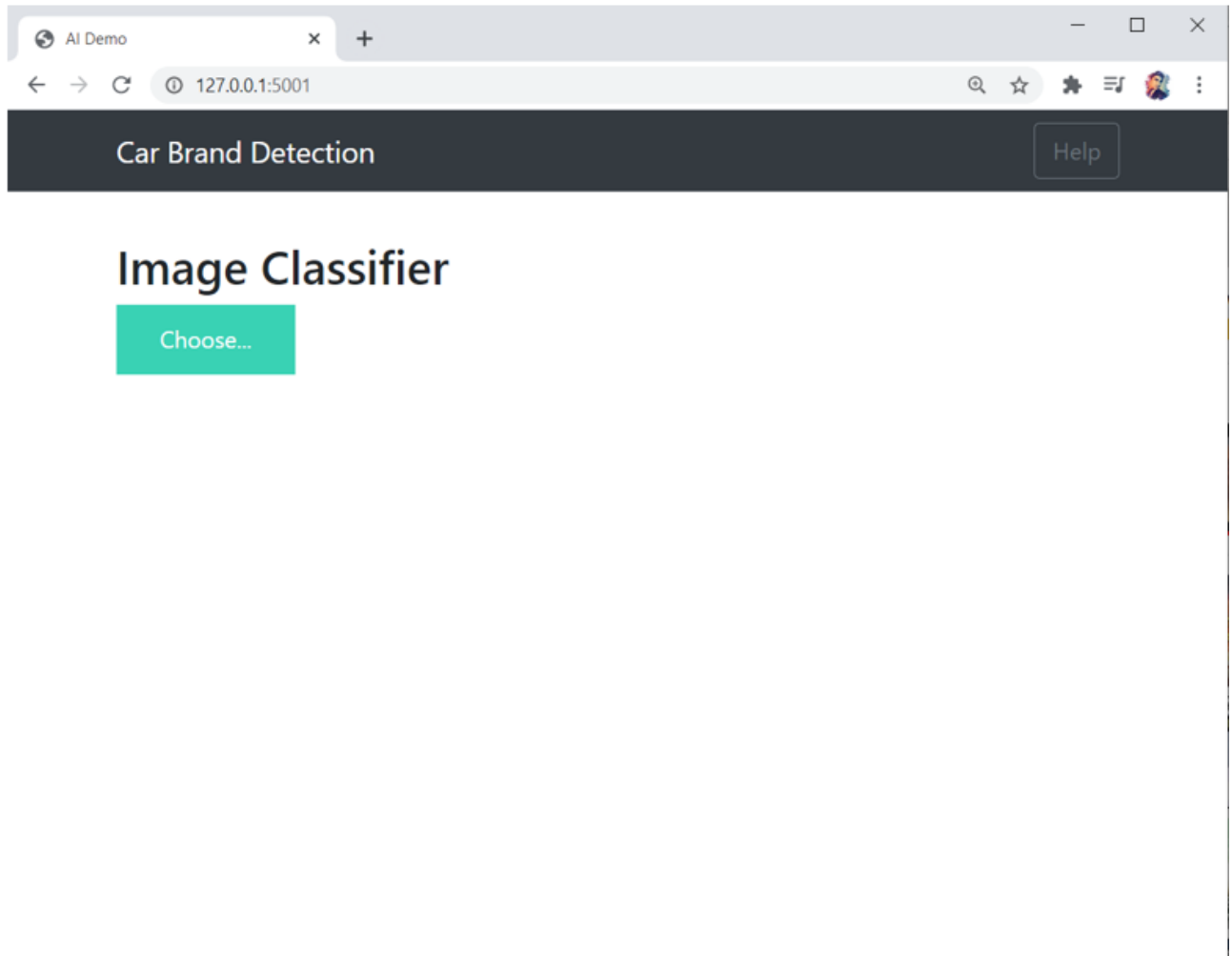
	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010
(2016)	0.32	0.24	0.36	0.28	0.30	0.21	0.32	0.27	0.29	0.30
(2015)	0.49	0.40	0.57	0.42	0.44	0.37	0.46	0.38	0.43	0.36



Project 2

MNIST Fashion Clothes Detection





Car Brand Detection

Image Classifier

Choose...



Predict!

Car Brand Detection

Image Classifier

Choose...



Result: The Car is Lamborghini