A Project Report

on

A Deep Learning based System for Landmark and tourist place recognition

Submitted in partial fulfillment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

19WH1A05G1	Ms. G.VARSHA PRIYA
19WH1A05F0	Ms. S MEGHANA
20WH5A0517	Ms. S MALAVIKA

Under the esteemed guidance of

Ms. S Vidyullatha
Assistant Professor

Department of Computer Science and Engineering
BVRIT HYDERABAD

College of Engineering for Women

(NBA Accredited – EEE, ECE, CSE and IT)

(Approved by AICTE, New Delhi and Affiliated to JNTU, Hyderabad)
Bachupally, Hyderabad – 500090

June 2023

DECLARATION

We hereby declare that the work presented in this project entitled **“A Deep Learning Based system for Land mark and tourist place Recognization**”submitted towards completion of Project Work in IV year of B.Tech, CSE at ‘BVRIT HYDERABAD College of Engineering For Women**’**, Hyderabad is an authentic record of our original work carried out under the guidance of Ms. S Vidyullatha, Assistant Professor, Department of CSE.

Sign with date:

Ms. G VARSHA PRIYA

(19WH1A05G1)

Sign with date:

Ms. S MEGHANA

(19WH1A05F0)

Sign with date:

Ms. S MALAVIKA

(20WH5A0517)

BVRIT HYDERABAD College of Engineering for Women

(NBA Accredited – EEE, ECE, CSE and IT)

(Approved by AICTE, New Delhi and Affiliated to JNTU, Hyderabad)
Bachupally, Hyderabad – 500090

Department of Computer Science and Engineering

Certificate

This is to certify that the Project Work report on "**A Deep Learning Based system for Land mark and tourist place Recognization**" is a bonafide work carried out byMs G VARSHA PRIYA(19WH1A05G1); Ms.MEGHANA(19WH1A05F0); MS .S MALAVIKA(20WH5A0517) in the partial fulfillment for the award of B.Tech. degree in **Computer Science and Engineering, BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Head of the Department	**Guide**
Dr.E. Venkateswara Reddy	**Ms. S Vidyullatha**
Professor and HoD ,	**Assistant Professor**
Department of CSE	

External Examiner

Acknowledgements

We would like to express our sincere thanks to **Dr. K V N Sunitha, Principal**, **BVRIT HYDERABAD College of Engineering for Women**, for providing the working facilities in the college.

Our sincere thanks and gratitude to **Dr.E. Venkateswara Reddy, Professor**, Department of CSE,**BVRIT HYDERABAD College of Engineering for Women** for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Ms. S Vidyullatha Assistant Professor**, Department of CSE**, BVRIT HYDERABAD College of Engineering for Women** for her constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank our Project Coordinator, all the faculty and staff of **CSE** Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

Ms. G VARSHA PRIYA
(19WH1A05G1)

Ms. S MEGHANA
(19WH1A05F0)

Ms. S MALAVIKA
(20WH5A0517)

ABSTRACT

More and more information about tourist attractions is being portrayed visually rather than textually. As a result, tourists who are interested in a specific attraction represented in photographs may not know how to conduct a text search to learn more about the intriguing tourist places. In light of this issue, and in order to improve the tourism industry's competitiveness, this study proposes an innovative tourist spot identification mechanism based on deep learning-based object detection technologyfor real-time detection and identification of tourist spots by taking pictures on location or retrieving images from the Internet. This project creates a tourist spot recognition system, which is a Deep Learning AI framework that is used to identify tourist destinations by providing photographs and Images.

Contents

LIST OF FIGURES

1. INTRODUCTION

Motivation

Computer vision constitutes a multidisciplinary scientific field that deals with the way of how computers comprehend or perceive the visual world. In other words, how computers can identify and classify an object, based on a compilation of images or videos used for input . As humans, we have the ability to understand the three-dimensional structure of our surrounding environment. In the same way, we try to imitate this procedure in computers, using mathematical techniques to create a partial 3D model of an object . Nowadays, computer vision is utilized in a great variety of different sectors like Optical Character Recognition (OCR), Machine Formal Review, Retail, Motor Vehicles Safety, Surveillance, 3Dmodeling, Face Recognition, and many more. Of course, nothing of these would have been achieved without the recent advancements in the field of machine learning and more specifically in deep learning. With the advent of deep learning and the recent developments in the last few years, we now have the potential to build complex Machine Learning models for image or object detection without considering the characteristics of an object.

Problem Definition

More and more information about tourist attractions is being portrayed visually rather than textually. As a result, tourists who are interested in a specific attraction represented in photographs may not know how to conduct a text search to learn more about the intriguing tourist places. In light of this issue, and in order to improve the tourism industry's competitiveness, this study proposes an innovative tourist spot identification mechanism based on deep learning-based object detection technologyfor real-time detection and identification of tourist spots by taking pictures on location or retrieving images from the Internet.

Objective of Problem

This project creates a tourist spot recognition system, which is a Deep Learning AI framework that is used to identify tourist destinations by providing photographs and Images**.**

Limitations of Project

- Currently most of the searches are about tourist are done using Text. Although there are several instances where text information is not available and or is not enough.
- There are several Image based system which can identify tourist spots using Images which uses algorithms like SVM which are not accurate and are not well suited.

Organization of Documentation

Project Documentation is an important part of project management. It is substantiated by the essential two functions of documentation: to make sure thatproject requirements are fulfilled and to establish traceability with regard to whathas been done, who has done it, and when it has been done. Thus, documentationmust lay the foundation for quality, traceability, and history for both the individual document and for the entire project documentation. It is thus extremely important that the documentation is well arranged, easy to read, and adequate.

Experienced project managers excel at making and following standard templates for their project documents. They reuse successful project plans, business cases, requirement sheets, and project status reports. This helps them focus on their core competency of managing the project rather than balancing the unmanageablepaperwork.

Existing System

- Currently most of the searches are about tourist are done using Text. Although there are several instances where text information is not available and or is not enough.

- There are several Image based system which can identify tourist spots using Images which uses algorithms like SVM which are not accurate and are not well suited.

Disadvantages of Existing System

- Support vector machine algorithm is not acceptable for large data sets.

- It does not execute very well when the data set has more sound i.e. target classes are overlapping.

- In cases where the number of properties for each data point outstrips the number of training data specimens, the support vector machine will underperform.

- As the support vector classifier works by placing data points, above and below the classifying hyperplane there is no probabilistic clarification for the classification.

Proposed System

- We propose a Advance Deep Learning System which will be able to detect the Name and Details about a Tourist Spot just by using the Image provided.

- Currently the System will be trained to recognise some major Tourist Spots with a Web App based intuitive Interface.

We suggest an advanced deep learning system that can identify a tourist attraction's name and other information from the provided image alone. With a Web App based intuitive Interface, the System is currently being trained to recognise some major tourist attractions. A user can also access the system on their smartphone to upload an image and get information about a particular tourist attraction. No matter the image's colour, resolution, or rotation, it can accept it as input. To train our model, we used Resnet 50 V2, Resnet 101 V2, and transfer learning.

Why is ResNet so popular?

This model was immensely successful, as can be ascertained from the fact that its ensemble won the top position at the ILSVRC 2015 classification competition with an error of only 3.57%. Additionally, it also came first in the ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation in the ILSVRC & COCO competitions of 2015.

The Importance Of ResNet in Computer Vision

Deep Neural Networks in Computer Vision

When working with deep convolutional neural networks to solve a problem related to computer vision, Machine Learning experts engage in stacking more layers. These additional layers help solve complex problems more efficiently as the different layers could be trained for varying tasks to get highly accurate results.

While the number of stacked layers can enrich the features of the model, a deeper network can show the issue of degradation. In other words, as the number of layers of the Neural Network increases, the accuracy levels may get saturated and slowly degrade after a point. As a result, the performance of the model deteriorates both on the training and testing data.

This degradation is not a result of overfitting. Instead, it may result from the initialization of the network, optimization function, or, more importantly, the problem of vanishing or exploding gradients.

Usage of Deep Residual Learning

ResNet was created with the aim of tackling this exact problem. Deep residual nets make use of residual blocks to improve the accuracy of the models. The concept of “skip connections,” which lies at the core of the residual blocks, is the strength of this type of neural network.

Skip Connections in ResNet

These skip connections work in two ways. Firstly, they alleviate the issue of vanishing gradient by setting up an alternate shortcut for the gradient to pass through. In addition, they enable the model to learn an identity function. This ensures that the higher layers of the model do not perform any worse than the lower layers.

In short, the residual blocks make it considerably easier for the layers to learn identity functions. As a result, ResNet improves the efficiency of deep neural networks with more neural layers while minimizing the percentage of errors. In other words, the skip connections add the outputs from previous layers to the outputs of stacked layers, making it possible to train much deeper networks than previously possible.

RESNET 50 V2

Deep residual networks are convolutional neural networks (CNNs) with more than 50 layers, like the well-known ResNet-50 model. A Residual Neural Network (ResNet) is a type of ANN that creates a network by stacking residual blocks on top of one another.
ResNet stands for Residual Network and is a specific type of convolutional neural network (CNN) introduced in the 2015 paper “Deep Residual Learning for Image Recognition” by He Kaiming, Zhang Xiangyu, Ren Shaoqing, and Sun Jian. CNNs are commonly used to power computer vision applications.

Special characteristics of ResNet-50

ResNet-50 has an architecture based on the model depicted above, but with one important difference. The 50-layer ResNet uses a bottleneck design for the building block. A bottleneck residual block uses 1×1 convolutions, known as a “bottleneck”, which reduces the number of parameters and matrix multiplications. This enables much faster training of each layer. It uses a stack of three layers rather than two layers.

The 50-layer ResNet architecture includes the following elements, as shown in the table below:

- **A 7×7 kernel convolution** alongside 64 other kernels with a 2-sized stride.
- **A max pooling layer** with a 2-sized stride.
- **9 more layers**—3×3,64 kernel convolution, another with 1×1,64 kernels, and a third with 1×1,256 kernels. These 3 layers are repeated 3 times.
- **12 more layers** with 1×1,128 kernels, 3×3,128 kernels, and 1×1,512 kernels, iterated 4 times.
- **18 more layers** with 1×1,256 cores, and 2 cores 3×3,256 and 1×1,1024, iterated 6 times.
- **9 more layers** with 1×1,512 cores, 3×3,512 cores, and 1×1,2048 cores iterated 3 times.

(up to this point the network has 50 layers)

- **Average pooling**, followed by a fully connected layer with 1000 nodes, using the softmax activation

ResNet-50v2 Architecture

The original ResNet architecture was ResNet-34, which comprised 34 weighted layers. It provided a novel way to add more convolutional layers to a CNN, without running into the vanishing gradient problem, using the concept of shortcut connections. A shortcut connection “skips over” some layers, converting a regular network to a residual network.

The regular network was based on the VGG neural networks (VGG-16 and VGG-19)—each convolutional network had a 3×3 filter. However, a ResNet has fewer filters and is less complex than a VGGNet. A 34-layer ResNet can achieve a performance of 3.6 billion FLOPs, and a smaller 18-layer ResNet can achieve 1.8 billion FLOPs, which is significantly faster than a VGG-19 Network with 19.6 billion FLOPs (read more in the ResNet paper, He et, al, 2015).

The ResNet architecture follows two basic design rules. First, the number of filters in each layer is the same depending on the size of the output feature map. Second, if the feature map's size is halved, it has double the number of filters to maintain the time complexity of each layer.

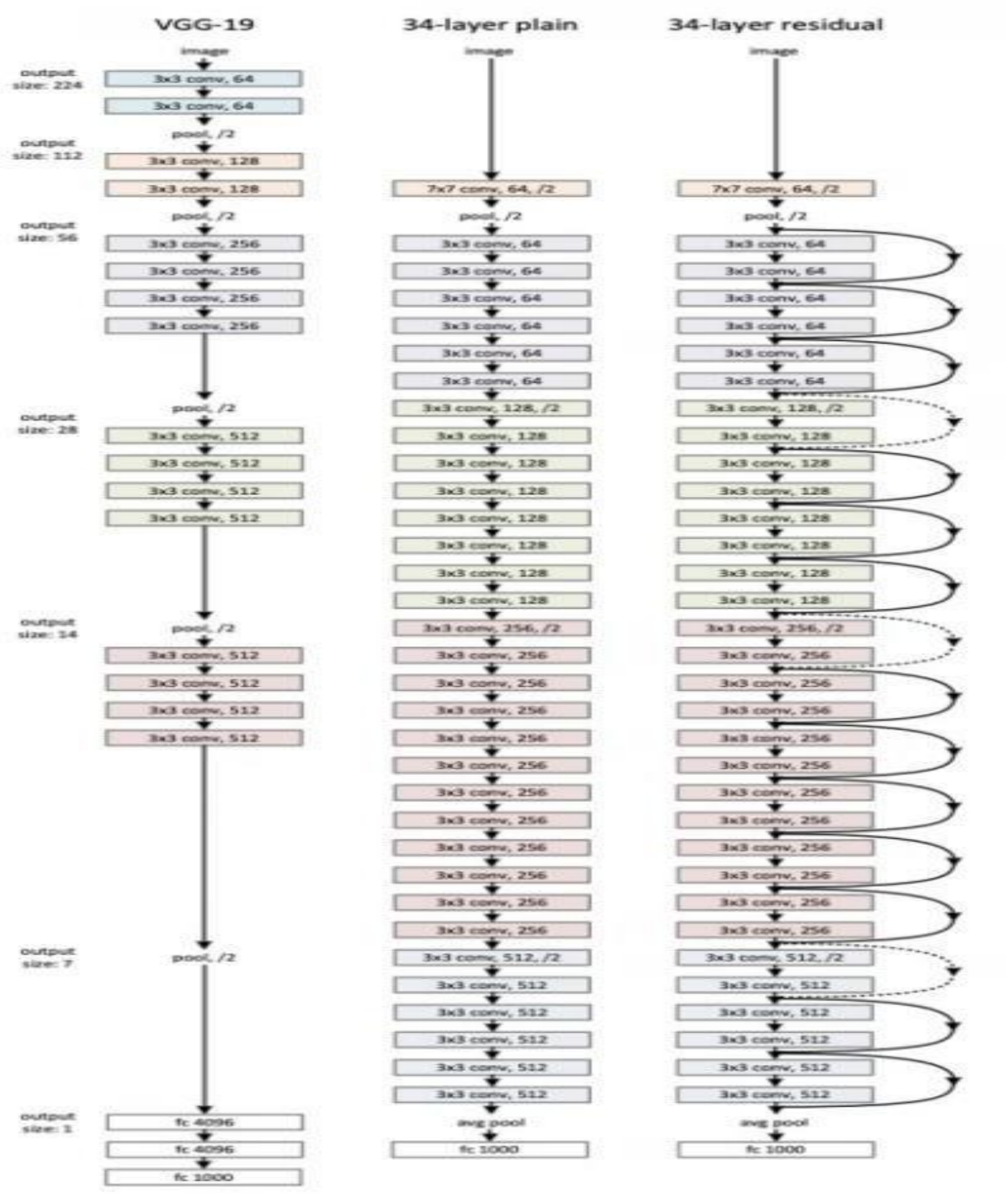
ResNet50 With Keras

KERAS is a deep learning API that is popular due to the simplicity of building models using it. Keras comes with several pre-trained models, including Resnet50, that anyone can use for their experiments.

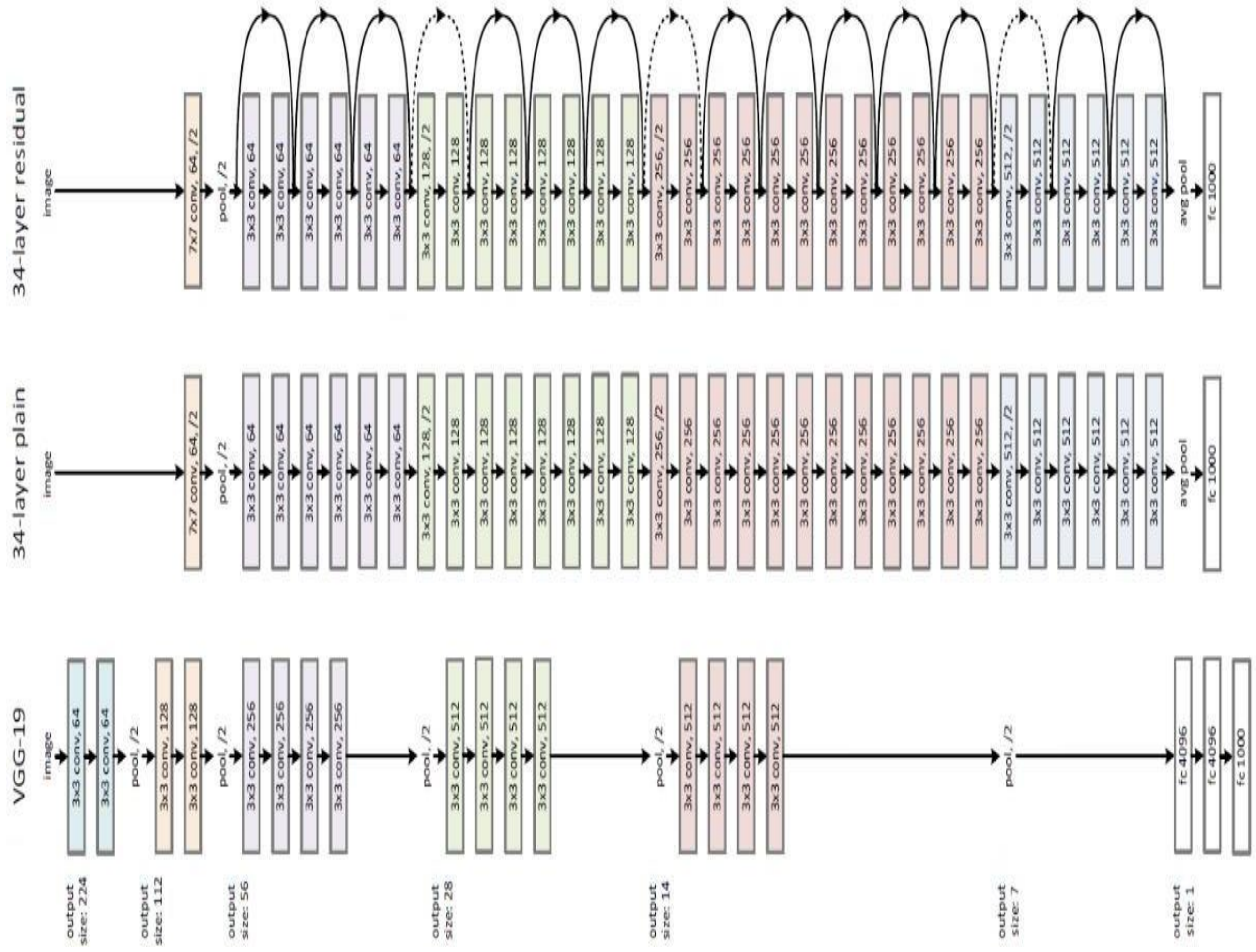
Therefore, building a residual network in Keras for computer vision tasks like image classification is relatively simple. You only need to follow a few simple steps.

How to use ResNet 50 with Keras

- **Step #1:** Firstly, you need to run a code to define the identity blocks to transform the CNN into a residual network and build the convolution block.
- **Step #2:** The next step is building the 50-layer Resnet model by combining both blocks.
- **Step #3:** Finally, you need to train the model for the required task. Keras allows you to easily generate a detailed summary of the network architecture you built. This can be saved or printed for future use.


VGG-19
34-layer plain
34-layer residual
image
output size: 224
output size: 112
output size: 56
output size: 28
output size: 14
output size: 7
output size: 1
3x3 conv, 64
pool, /2
3x3 conv, 128
3x3 conv, 256
3x3 conv, 512
fc 4096
fc 1000
7x7 conv, 64, /2
3x3 conv, 128, /2
3x3 conv, 256, /2
3x3 conv, 512, /2
avg pool

RESNET101 V2



ResNet-101 is a convolutional neural network that is 101 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database [1]. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224.

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers---8x deeper than VGG nets but still having lower complexity.

An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10 with 100 and 1000 layers.The depth of representations is of central importance for many visual recognition tasks. Solely due to our extremely deep representations, we obtain a 28% relative improvement on the COCO object detection dataset. Deep residual nets are foundations of our submissions to ILSVRC & COCO 2015 competitions, where we also won the 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.

Transfer learning

A model created for one task is used as the basis for another using the machine learning technique known as transfer learning. Pre-trained models are frequently used as the foundation for deep learning tasks in computer vision and natural language processing because they save both time and money compared to developing neural network models from scratch and because they perform vastly better on related tasks.

Reusing a model that has already been trained on a different problem is known as transfer learning in machine learning. With transfer learning, a computer can use its understanding of one activity to better generalize about another. For instance, you may leverage the skills a classifier learned to identify drinks while training it to predict whether an image contains food.
Although transfer learning has other advantages as well, its key advantages are reducing training time, improving neural network performance (in most circumstances), and using less data. Transfer learning is useful in situations when access to large amounts of data is not always possible yet is required to train a neural network from start. Because the model has previously been trained,

transfer learning allows for the construction of reliable machine learning models with relatively minimal training data. This is particularly useful in natural language processing because producing sizable labelled data sets typically necessitates expert expertise. Because it can occasionally take days or even weeks to train a deep neural network from scratch on a tough problem, training time is also shortened. The isolated training approach used in traditional ML allows each model to be trained independently for a given task without relying on prior knowledge. Transfer learning, on the other hand, uses the information learned from the previously trained model to carry out the task. Because ImageNet does not contain images from the biomedical field, one cannot use the pre-trained model of ImageNet with biomedical images. Transfer learning models outperform conventional ML models in terms of performance. The reason for this is that the models that use information (features, weights, etc.) from previously trained models already have a thorough understanding of the features. It expedites the process compared to building neural networks from scratch.

2. DEEP LEARNING TECHNIQUES

Deep learning is a type of machine learning and artificial intelligence (AI) that imitates the way humans gain certain types of knowledge. Deep learning is an important element of data science, which includes statistics and predictive modeling. It is extremely beneficial to data scientists who are tasked with collecting, analyzing and interpreting large amounts of data; deep learning makes this process faster and easier. At its simplest, deep learning can be thought of as a way to automate predictive analytics. While traditional machine learning algorithms are linear, deep learning algorithms are stacked in a hierarchy of increasing complexity and abstraction.

- **How deep learning works**

Computer programs that use deep learning go through much the same process as the toddler learning to identify the dog. Each algorithm in the hierarchy applies a nonlinear transformation to its input and uses what it learns to create a statistical model as output. Iterations continue until the output has reached an acceptable level of accuracy. The number of processing layers through which data must pass is what inspired the label deep.

- **Deep learning examples**

Because deep learning models process information in ways similar to the human brain, they can be applied to many tasks people do. Deep learning is currently used in most common image recognition tools, natural language processing (NLP) andspeech recognition software. These tools are starting to appear in applications as diverse as self-driving cars and language translation services.

3. ARCHITECTURE

Web applications are by nature distributed applications, meaning that they are

programs that run on more than one computer and communicate through network or server. Specifically, web applications are accessed with a web browser and are popular because of the ease of using the browser as a user client. For the enterprise software on potentially thousands of client computers is a key reason for their popularity. Web applications are used for web mail, online retail sales, discussion boards, weblogs, online banking, and more. One web application can be accessed and used by millions of people.

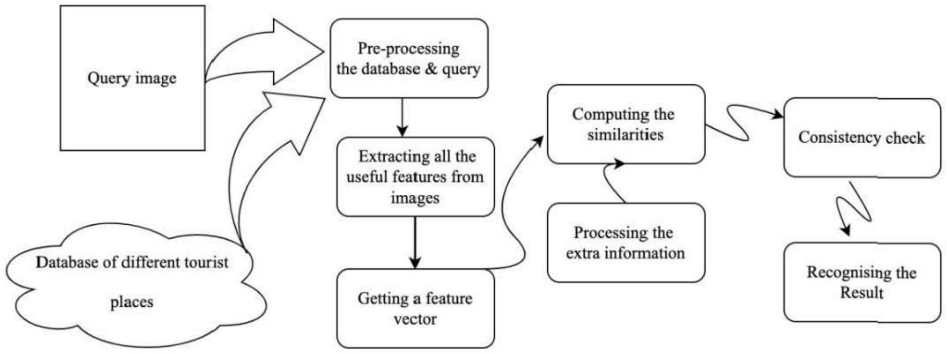

Query image
Pre-processing the database & query
Extracting all the useful features from images
Getting a feature vector
Computing the similarities
Processing the extra information
Consistency check
Recognising the Result
Database of different tourist places

Figure 1: Architecture

Uml diagrams

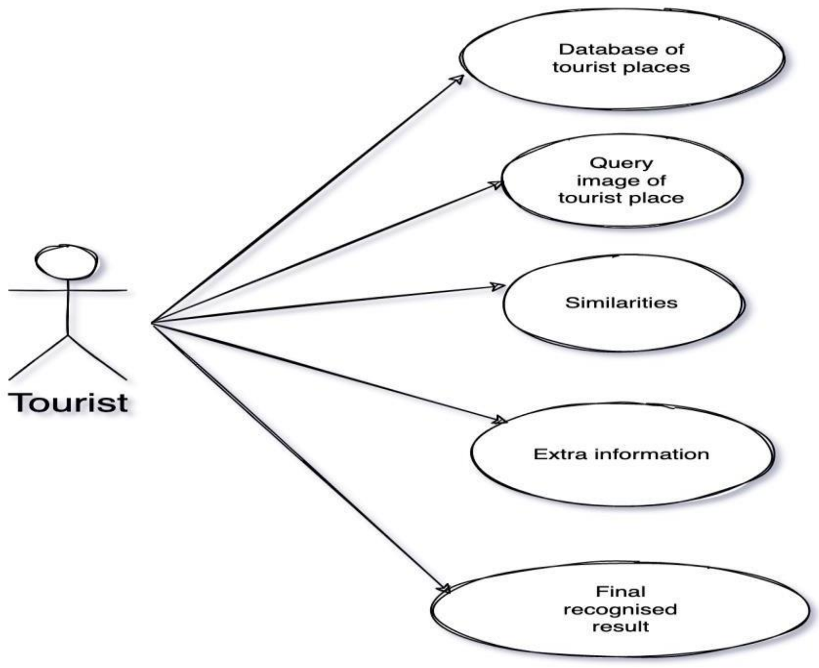
1. Use Case Diagram

To model a system, the most important aspect is to capture the dynamic behavior. Dynamic behavior means the behavior of the system when it is running/operating.

Only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML, there are five diagrams available to model the dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature, there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. Use case diagrams consist of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

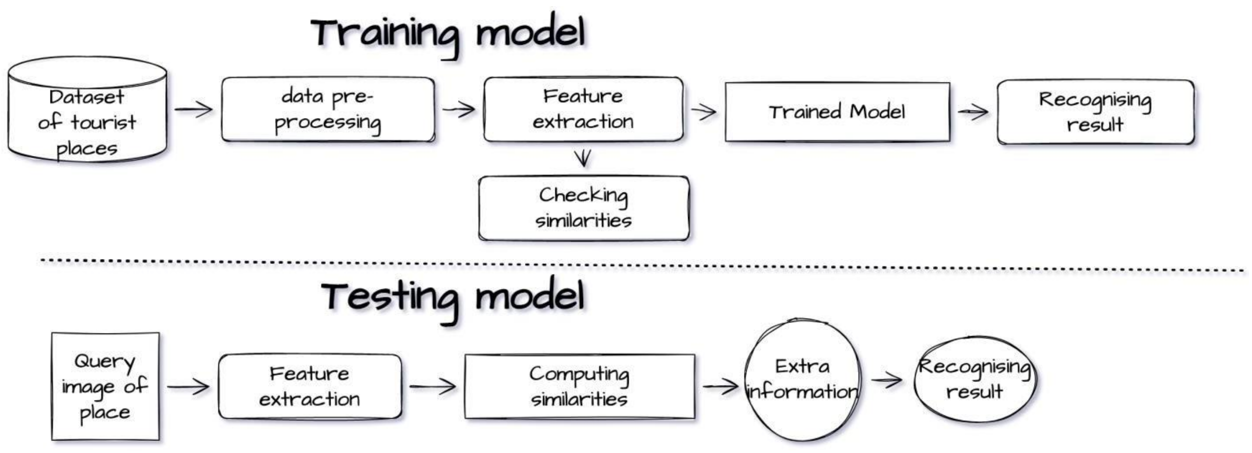
Hence to model the entire system, a number of use case diagrams are used.



1. Usecase Diagram

2. Activity Diagram

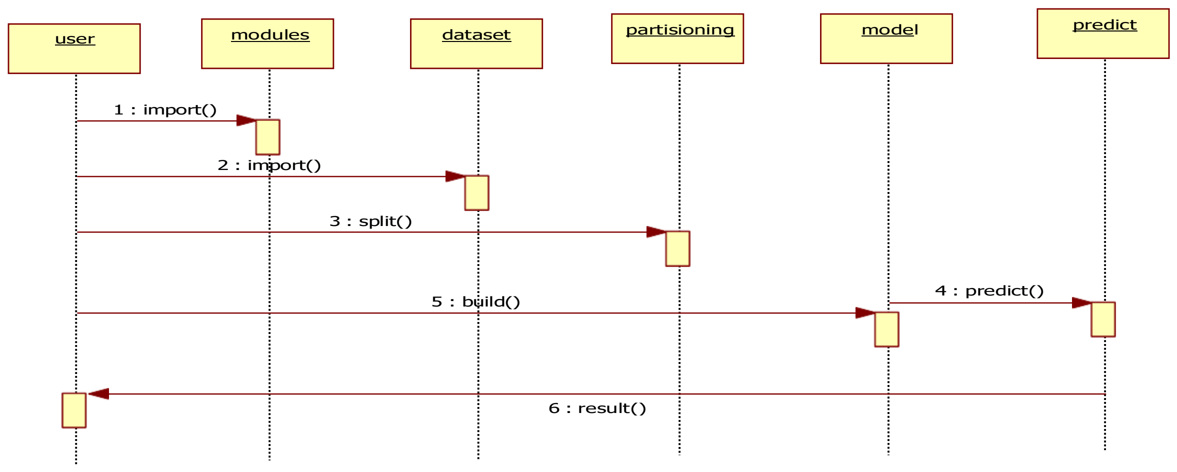
Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



2 Activity Diagram

3. Sequence Diagram

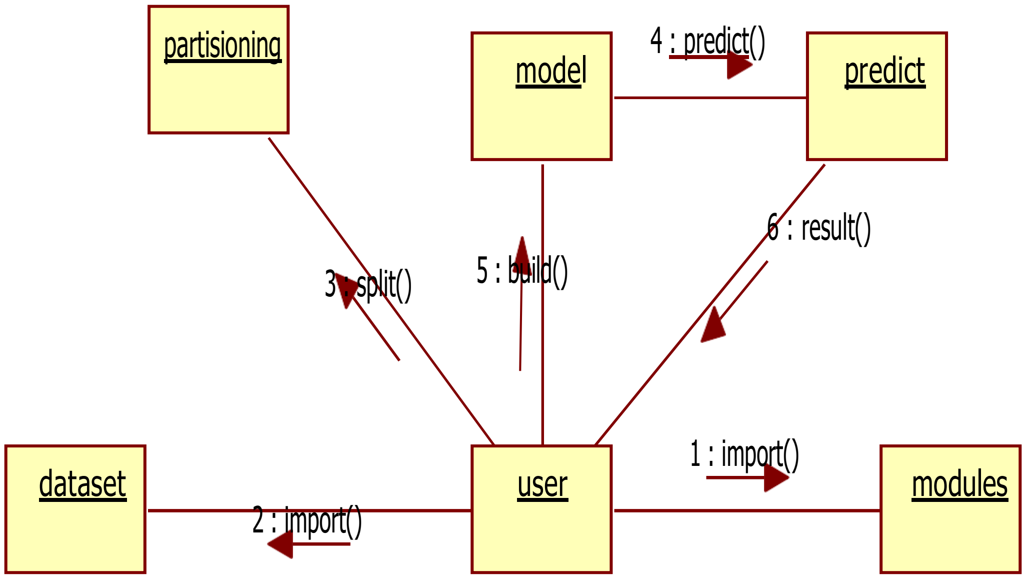
Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.



3. Sequence Diagram

4. Collaboration Diagram

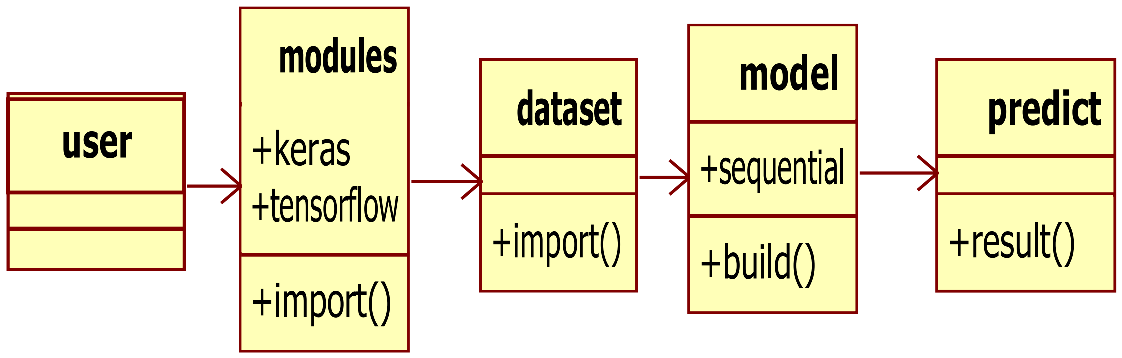
A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. The messages between objects are shown as arrows connecting the relevant rectangles along with labels that define the message sequencing.



4 Collabration Diagram

5. Class Diagram

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.



5 Class Diagram

4. THEORETICAL ANALYSIS OF THE PROPOSED PROJECT

Requirements Gathering

Software Requirements

- Programming Language: Python 3.6
- Graphical User Interface: HTML5, CSS3 with Bootstrap, JavaScript.Packages: Tensorflow, keras,numpy,pandas,matplotlib,Scikit-learn.
- Tool: Jupyter Notebook.

Hardware Requirements

- Processor: i3 7th Gen

  RAM: 8GB or better
- Storage: 120GB or more

5. TECHNOLOGIES DESCRIPTION

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

HTML

HTML stands for Hypertext Markup Language, and it is the most widely used language to write Web Pages.

- Hypertext refers to the way in which Web pages (HTML documents) are linked together. Thus, the link available on a webpage is called Hypertext.
- As its name suggests, HTML is a Markup Language which means you use HTML to simply "mark-up" a text document with tags that tell a Web browser how to structure it to display.

Originally, HTML was developed with the intent of defining the structure of documents like headings, paragraphs, lists, and so forth to facilitate the sharing of scientific information between researchers.

Now, HTML is being widely used to format web pages with the help of different tags available in HTML language.

CSS

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, and variations in display for different devices and screen sizes as well as a variety of other effects.

CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

JavaScript

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

JavaScript was first known as LiveScript, but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java. JavaScript made its first appearance in Netscape 2.0 in 1995 with the name LiveScript. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

The ECMA-262 Specification defined a standard version of the core JavaScript language.

- JavaScript is a lightweight, interpreted programming language.
- Designed for creating network-centric applications.
- Complementary to and integrated with Java.
- Complementary to and integrated with HTML.
- Open and cross-platform.

KERAS

- Keras is an open-source high-level Neural Network library, which is written in Python is capable enough to run on Theano, TensorFlow, or CNTK. It was developed by one of the

Google engineers, Francois Chollet. It is made user-friendly, extensible, and modular for facilitating faster experimentation with deep neural networks. It not only supports Convolutional Networks and Recurrent Networks individually but also their combination.

- It cannot handle low-level computations, so it makes use of the **Backend** library to resolve it. The backend library act as a high-level API wrapper for the low-level API, which lets it run on TensorFlow, CNTK, or Theano.
- Initially, it had over 4800 contributors during its launch, which now has gone up to 250,000 developers. It has a 2X growth ever since every year it has grown. Big companies like Microsoft, Google, NVIDIA, and Amazon have actively contributed to the development of Keras. It has an amazing industry interaction, and it is used in the development of popular firms likes Netflix, Uber, Google, Expedia, etc.

Importance of keras

- Focus on user experience has always been a major part of Keras.
- Large adoption in the industry.
- It is a multi backend and supports multi-platform, which helps all the encoders come together for coding.
- Research community present for Keras works amazingly with the production community.
- Easy to grasp all concepts.
- It supports fast prototyping.
- It seamlessly runs on CPU as well as GPU.
- It provides the freedom to design any architecture, which then later is utilized as an API for the project.
- It is really very simple to get started with.
- Easy production of models actually makes Keras special.

TENSORFLOW

- TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.
- TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

- **NumPy**: Base n-dimensional array package
- **SciPy**: Fundamental library for scientific computing
- **Matplotlib**: Comprehensive 2D/3D plotting
- **IPython**: Enhanced interactive console
- **Sympy**: Symbolic mathematics
- **Pandas**: Data structures and analysis
- Extensions or modules for SciPy care conventionally named SciKits. As such, the module provides learning algorithms and is named scikit-learn.

Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions

- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Jupyter Notebook

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

6. IMPLEMENTATION

This study is divided into five stages. The dataset that will be used for model training and testing must first be prepared. The training process is the second stage. The training stage is carried out using the structure of the created model. The model must then be tested, evaluated using web services, and put through a performance test. The dataset used for training and testing has the same number of batches and epochs. The complete stage process is as follows:

DataSet Overview

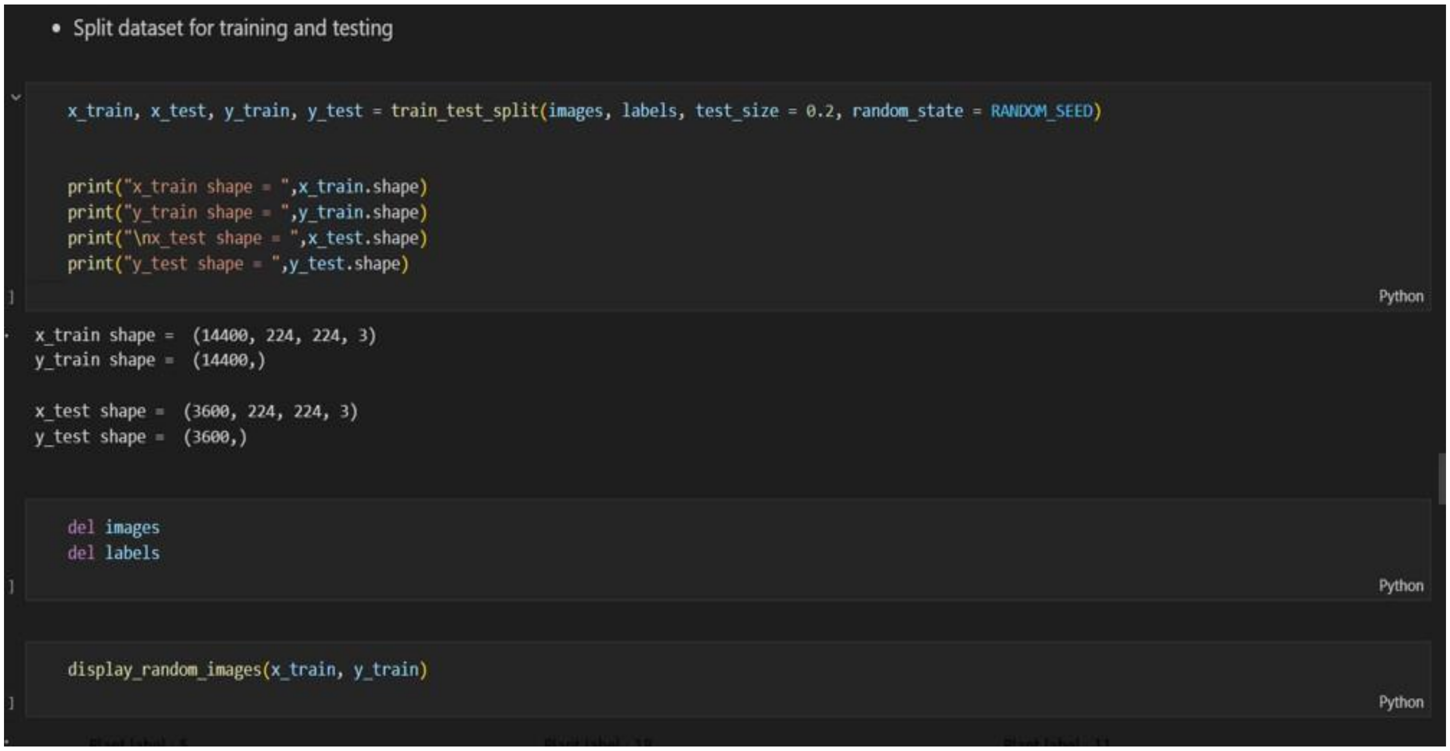
The following difficulties faced by commercial landmark recognition systems are what the Google Landmarks Dataset v2 aims to mimic: a sizeable scale A collection of millions of photographs is required to capture the entire planet. Within-class variation Pho Photos are taken in a variety of lighting situations and angles, including from inside and outside of buildings. Additionally, there will be images pertaining to the landmark but not featuring the landmark itself, such as floor plans and pictures of the architects, or views from the landmark.

Data Distribution

With landmarks from 246 of the 249 countries on the ISO 3166-1 country code list, the Google Landmarks Dataset v2 is a truly global dataset.The number of images in the top 20 nations reveals how many there are on each continent. We can see that despite the dataset being global, it is by no means comprehensive.Churches are by far the most popular category, followed by parks and museums. Only those categories with more than 18k images are included, and of those, about 28% are made by nature and 72% are artificial.

Dataset Preparation

The model that has been created will be evaluated using the Google Landmark Dataset, a public dataset, as part of the evaluation plan. The number of images in each class is uneven, so it needs to be filtered on a public dataset. There are classes with fewer than ten images and classes with more than one hundred images. In this study, 30 classes were used, with an average value of 50 taken from each class.A specific composition will be used to divide each class in the dataset. The dataset has three compositions: data train, data validation, and data testing. The composition of the comparison is 80% for data train, 20% for data validation, and 20% for data resulting.



- Split dataset for training and testing

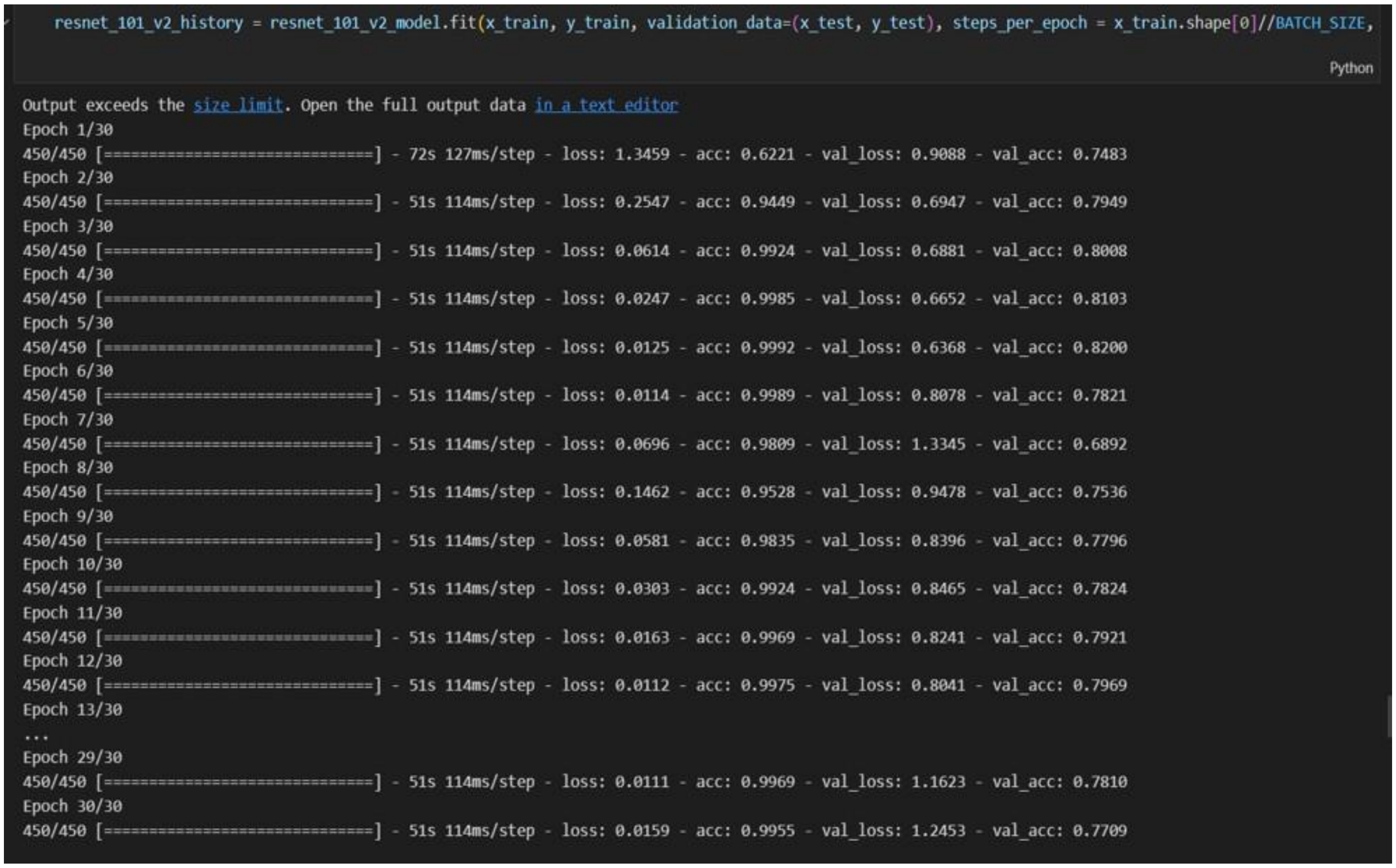
```python
x_train, x_test, y_train, y_test = train_test_split(images, labels, test_size = 0.2, random_state = RANDOM_SEED)


print("x_train shape = ",x_train.shape)
print("y_train shape = ",y_train.shape)
print("\nx_test shape = ",x_test.shape)
print("y_test shape = ",y_test.shape)
```

```
x_train shape =  (14400, 224, 224, 3)
y_train shape =  (14400,)

x_test shape =  (3600, 224, 224, 3)
y_test shape =  (3600,)
```

```python
del images
del labels
```

```python
display_random_images(x_train, y_train)
```

Training Model

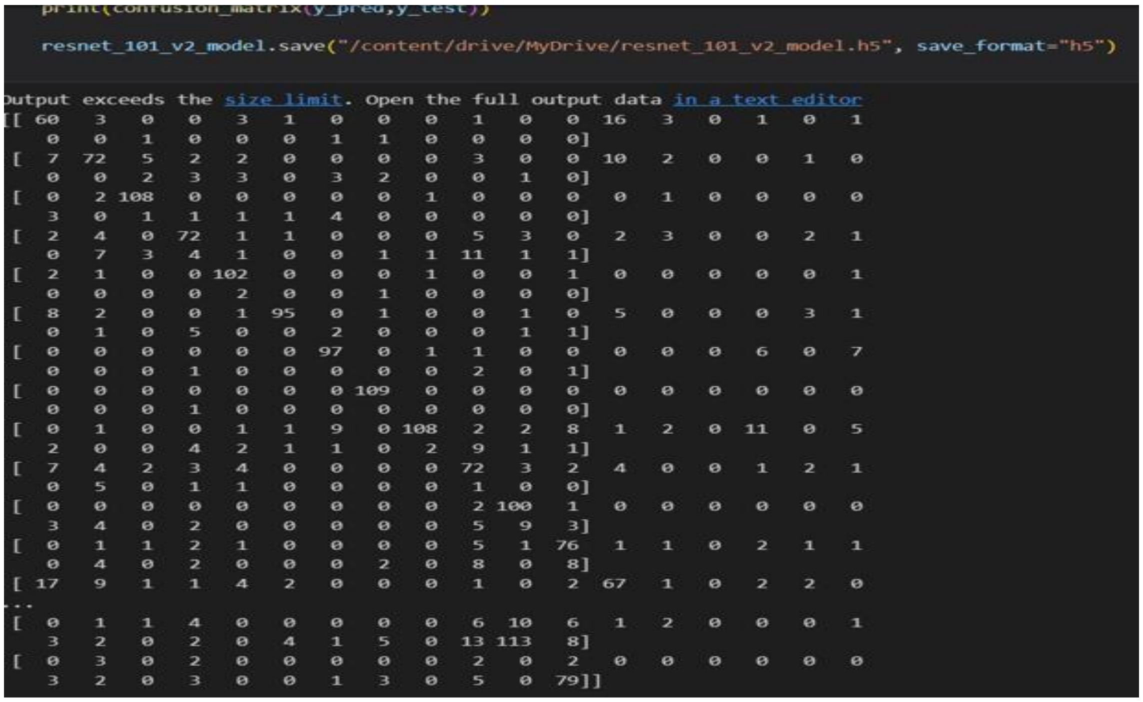


```
resnet_101_v2_history = resnet_101_v2_model.fit(x_train, y_train, validation_data=(x_test, y_test), steps_per_epoch = x_train.shape[0]//BATCH_SIZE,
                                                                                                                                        Python
Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/30
450/450 [==============================] - 72s 127ms/step - loss: 1.3459 - acc: 0.6221 - val_loss: 0.9088 - val_acc: 0.7483
Epoch 2/30
450/450 [==============================] - 51s 114ms/step - loss: 0.2547 - acc: 0.9449 - val_loss: 0.6947 - val_acc: 0.7949
Epoch 3/30
450/450 [==============================] - 51s 114ms/step - loss: 0.0614 - acc: 0.9924 - val_loss: 0.6881 - val_acc: 0.8008
Epoch 4/30
450/450 [==============================] - 51s 114ms/step - loss: 0.0247 - acc: 0.9985 - val_loss: 0.6652 - val_acc: 0.8103
Epoch 5/30
450/450 [==============================] - 51s 114ms/step - loss: 0.0125 - acc: 0.9992 - val_loss: 0.6368 - val_acc: 0.8200
Epoch 6/30
450/450 [==============================] - 51s 114ms/step - loss: 0.0114 - acc: 0.9989 - val_loss: 0.8078 - val_acc: 0.7821
Epoch 7/30
450/450 [==============================] - 51s 114ms/step - loss: 0.0696 - acc: 0.9809 - val_loss: 1.3345 - val_acc: 0.6892
Epoch 8/30
450/450 [==============================] - 51s 114ms/step - loss: 0.1462 - acc: 0.9528 - val_loss: 0.9478 - val_acc: 0.7536
Epoch 9/30
450/450 [==============================] - 51s 114ms/step - loss: 0.0581 - acc: 0.9835 - val_loss: 0.8396 - val_acc: 0.7796
Epoch 10/30
450/450 [==============================] - 51s 114ms/step - loss: 0.0303 - acc: 0.9924 - val_loss: 0.8465 - val_acc: 0.7824
Epoch 11/30
450/450 [==============================] - 51s 114ms/step - loss: 0.0163 - acc: 0.9969 - val_loss: 0.8241 - val_acc: 0.7921
Epoch 12/30
450/450 [==============================] - 51s 114ms/step - loss: 0.0112 - acc: 0.9975 - val_loss: 0.8041 - val_acc: 0.7969
Epoch 13/30
...
Epoch 29/30
450/450 [==============================] - 51s 114ms/step - loss: 0.0111 - acc: 0.9969 - val_loss: 1.1623 - val_acc: 0.7810
Epoch 30/30
450/450 [==============================] - 51s 114ms/step - loss: 0.0159 - acc: 0.9955 - val_loss: 1.2453 - val_acc: 0.7709
```

The training process is carried out on the data train and data validation. The training results are in the form of a model that has a weight, which is then stored for evaluation and image prediction. The test conducted is a test of the model of training results. The model is tested on test data in which the output is a percentage of overall accuracy. The additional testing process for the landmark dataset used is to make predictions on the inserted image, which provides output in the form of the landmark name of the inserted image Before the model can be used for evaluation, the model needs to be trained using CNN method with several different model architecture such as ResNet50V2, and Resnet101V2.

Model Testing

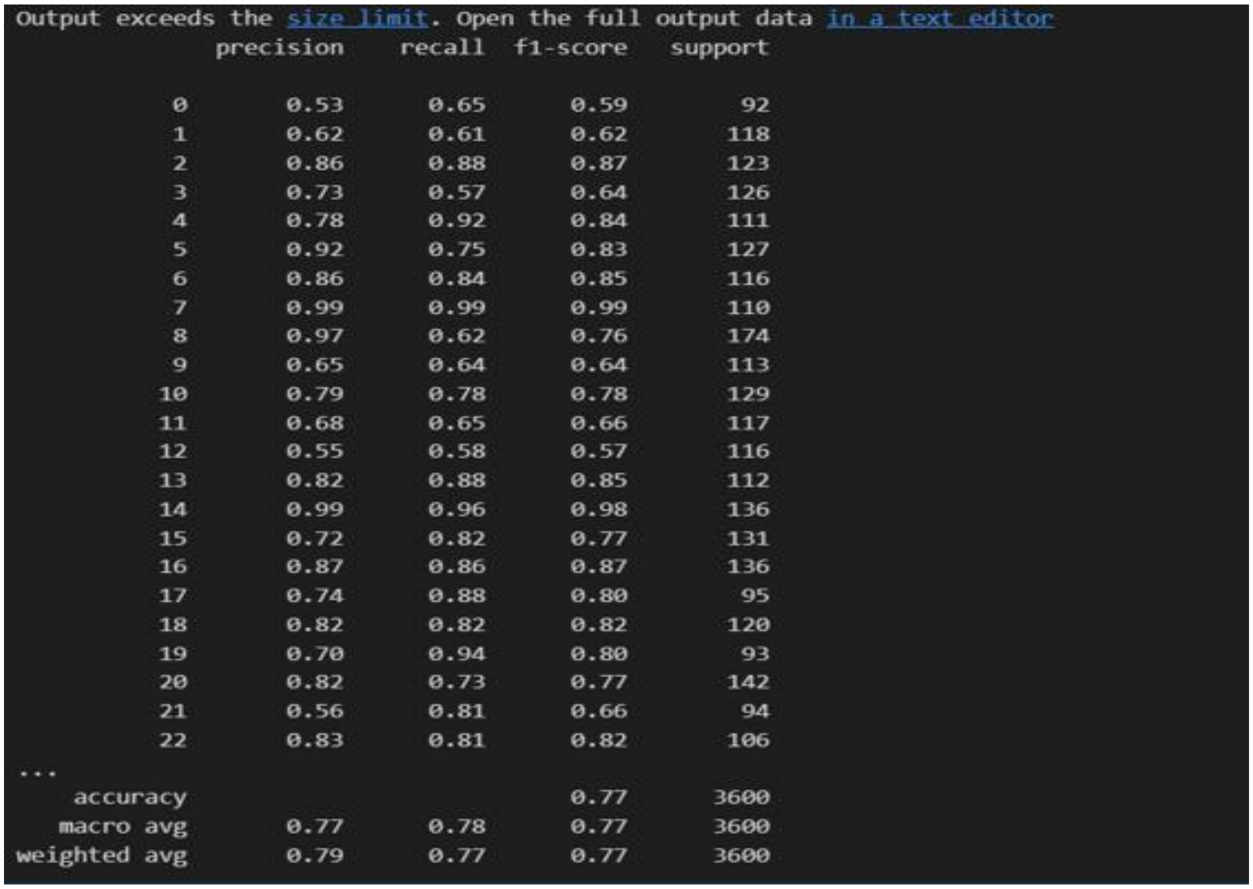
Confusion Matrix is a particular table used in machine learning fields, specifically the problem of statistical classification, that enables visualisation of an algorithm's performance. Classification can be accomplished by comparing the created models. The test data from each class is matched. The obtained results and predictions can be categorised into two categories: true positive and actual positive, indicating that the data is in fact consistent with the classification.



```
print(confusion_matrix(y_pred,y_test))

resnet_101_v2_model.save("/content/drive/MyDrive/resnet_101_v2_model.h5", save_format="h5")

Output exceeds the size limit. Open the full output data in a text editor
[[ 60   3   0   0   3   1   0   0   0   1   0   0  16   3   0   1   0   1
    0   0   1   0   0   0   1   1   0   0   0   0]
 [  7  72   5   2   2   0   0   0   0   3   0   0  10   2   0   0   1   0
    0   0   2   3   3   0   3   2   0   0   1   0]
 [  0   2 108   0   0   0   0   0   1   0   0   0   0   1   0   0   0   0
    3   0   1   1   1   1   4   0   0   0   0   0]
 [  2   4   0  72   1   1   0   0   0   5   3   0   2   3   0   0   2   1
    0   7   3   4   1   0   0   1   1  11   1   1]
 [  2   1   0   0 102   0   0   0   1   0   0   1   0   0   0   0   0   1
    0   0   0   0   2   0   0   1   0   0   0   0]
 [  8   2   0   0   1  95   0   1   0   0   1   0   5   0   0   0   3   1
    0   1   0   5   0   0   2   0   0   0   1   1]
 [  0   0   0   0   0   0  97   0   1   1   0   0   0   0   0   6   0   7
    0   0   0   1   0   0   0   0   0   2   0   1]
 [  0   0   0   0   0   0   0 109   0   0   0   0   0   0   0   0   0   0
    0   0   0   1   0   0   0   0   0   0   0   0]
 [  0   1   0   0   1   1   9   0 108   2   2   8   1   2   0  11   0   5
    2   0   0   4   2   1   1   0   2   9   1   1]
 [  7   4   2   3   4   0   0   0   0  72   3   2   4   0   0   1   2   1
    0   5   0   1   1   0   0   0   0   1   0   0]
 [  0   0   0   0   0   0   0   0   0   2 100   1   0   0   0   0   0   0
    3   4   0   2   0   0   0   0   0   5   9   3]
 [  0   1   1   2   1   0   0   0   0   5   1  76   1   1   0   2   1   1
    0   4   0   2   0   0   0   2   0   8   0   8]
 [ 17   9   1   1   4   2   0   0   0   1   0   2  67   1   0   2   2   0
...
 [  0   1   1   4   0   0   0   0   0   6  10   6   1   2   0   0   0   1
    3   2   0   2   0   4   1   5   0  13 113   8]
 [  0   3   0   2   0   0   0   0   0   2   0   2   0   0   0   0   0   0
    3   2   0   3   0   0   1   3   0   5   0  79]]
```

Fig 6.Confusion matrix

Performance Measurement



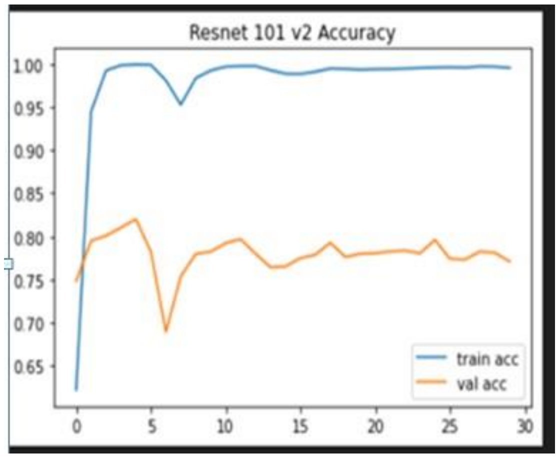
```
Output exceeds the size limit. Open the full output data in a text editor
              precision    recall  f1-score   support

           0       0.53      0.65      0.59        92
           1       0.62      0.61      0.62       118
           2       0.86      0.88      0.87       123
           3       0.73      0.57      0.64       126
           4       0.78      0.92      0.84       111
           5       0.92      0.75      0.83       127
           6       0.86      0.84      0.85       116
           7       0.99      0.99      0.99       110
           8       0.97      0.62      0.76       174
           9       0.65      0.64      0.64       113
          10       0.79      0.78      0.78       129
          11       0.68      0.65      0.66       117
          12       0.55      0.58      0.57       116
          13       0.82      0.88      0.85       112
          14       0.99      0.96      0.98       136
          15       0.72      0.82      0.77       131
          16       0.87      0.86      0.87       136
          17       0.74      0.88      0.80        95
          18       0.82      0.82      0.82       120
          19       0.70      0.94      0.80        93
          20       0.82      0.73      0.77       142
          21       0.56      0.81      0.66        94
          22       0.83      0.81      0.82       106
...
    accuracy                           0.77      3600
   macro avg       0.77      0.78      0.77      3600
weighted avg       0.79      0.77      0.77      3600
```

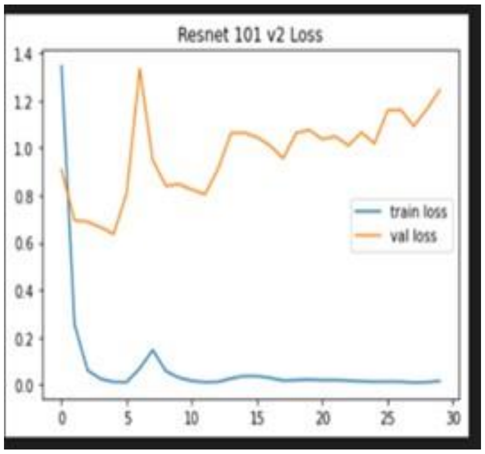
Fig: 7.Evlauation metrics

The performance test will test each model through the gateway API.Performance measurements are also carried out on the model through the API that has been made. The API was deployed to Kubernetes system on Google Cloud Platform, the server specification is 22 GB RAM and 6 CPU cores using Intel Skylake technology. Measurements were made using the JMeter application, each model will be distinguished through labels to be tested one by one. Each label has a different API url and from each label will produce a report with data including average, min, max, std. dev, error, throughput and Kb / Sec. Average is the average time taken by all samples to run a certain label. Min is the shortest time taken by a sample for a particular label.

Max is the longest time taken by a sample for a particular label. Std. Dev shows a series of cases that deviate from the average value of sample response time. The lower this value the more consistent the data flow from network. Std. A good dev should be less than equal to half the average time for a label. Error (%) is the percentage of failed requests per label. Throughput is the number of requests processed per unit of time by the server. This time is calculated from the beginning of the first sample to the end of the last sample. Bigger throughput is better. KB / Sec shows the amount of data downloaded from the server during the performance test.

RESULT AND ANALYSIS





As part of the evaluation strategy, the developed model will be assessed using the Google Landmark Dataset, a public dataset. It must be filtered on a public dataset because there are uneven numbers of images in each class. Less than ten images are used in some classes, and more than one hundred images are used in other classes. 30 classes were used in this study, with an average value of 50 being taken from each class. Each class in the dataset will be divided according to a particular composition. Data train, data validation, and data testing are the three components that make up the dataset. The comparison is made up of 80% train data, 20% validation data and data resulting. Resnet 50 v2 model is used for evaluating results. It results in the accuracy of 71% . Further the data is trained with Resnet 101 v2 that results in 77% .The accuracy gets increased in Resnet 101 v2 ,because the Resnet 101v2 is more efficient than Resnet 50v2 as it easily deals with large amounts of data with an ease.

FUTURE WORK

Using GPS to aid Recognition Tracking and correctly classifying an object is often a challenging and a difficult task, especially considering the numerous factors that are involved in a such type of process. Even well trained models, in some cases fail to provide adequately accurate, valid and precise data. A technology that all the mobile devices incorporate and could substantially reduce the failures on recognition, is thatof the Global Positioning System commonly known as GPS. Receiving information through the 4 existing GPS satellites, this feature enables us to obtain valuable data about the geolocation and the time of a device, operating independently and without influencing any other operation. The concept behind this idea is to combine the models capabilities with the reliability of GPS. With this way, by providing access to GPS, the application will immediately restrict significantly the area, excluding those monuments that are located far from the given location and reducing dramatically the chance of a wrong detection. Typically, with the recent developments particularly inthe technological field, the detection of a position using this feature can be achieved even with accuracy within a few meters. Therefore, fusing GPS with an object detection model may lead to a near infallible model.

CONCLUSION

In conclusion, residual network, also known as ResNet, was a significant advancement that altered the training of deep convolutional neural networks for computer vision tasks.

GPS as a Recognition Aid Tracking and accurately classifying an object is frequently a difficult task, especially when you consider the many variables that go into such a process. Even well-trained models occasionally fall short of providing data that is sufficiently accurate, valid, and precise. The Global Positioning System, or GPS, is a technology that all mobile devices include and could

Significantly lower the failures on recognition. This feature allows us to obtain useful information about a device's geo location and time by receiving data from the four active GPS satellite.

REFERENCES

1. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classificationwith deep convolutional neural networks. In Proceedings of the 25th InternationalConference on Neural Information Processing Systems - Volume 1, NIPS 2012, pages1097–1105, USA, 2012. Curran Associates Inc.

2. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014.

3. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich- Going Deeper with Convolutions- 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 10.1109/CVPR.2015.7298594

4. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015.

5. Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. CoRR, abs/1709.01507, 2017.

6. Barret Zoph, Vijay Vasudevan, Jonathon Shlens, Quoc V. Le- Learning Transferable Architectures for Scalable Image Recognition- arXiv:1707.07012.

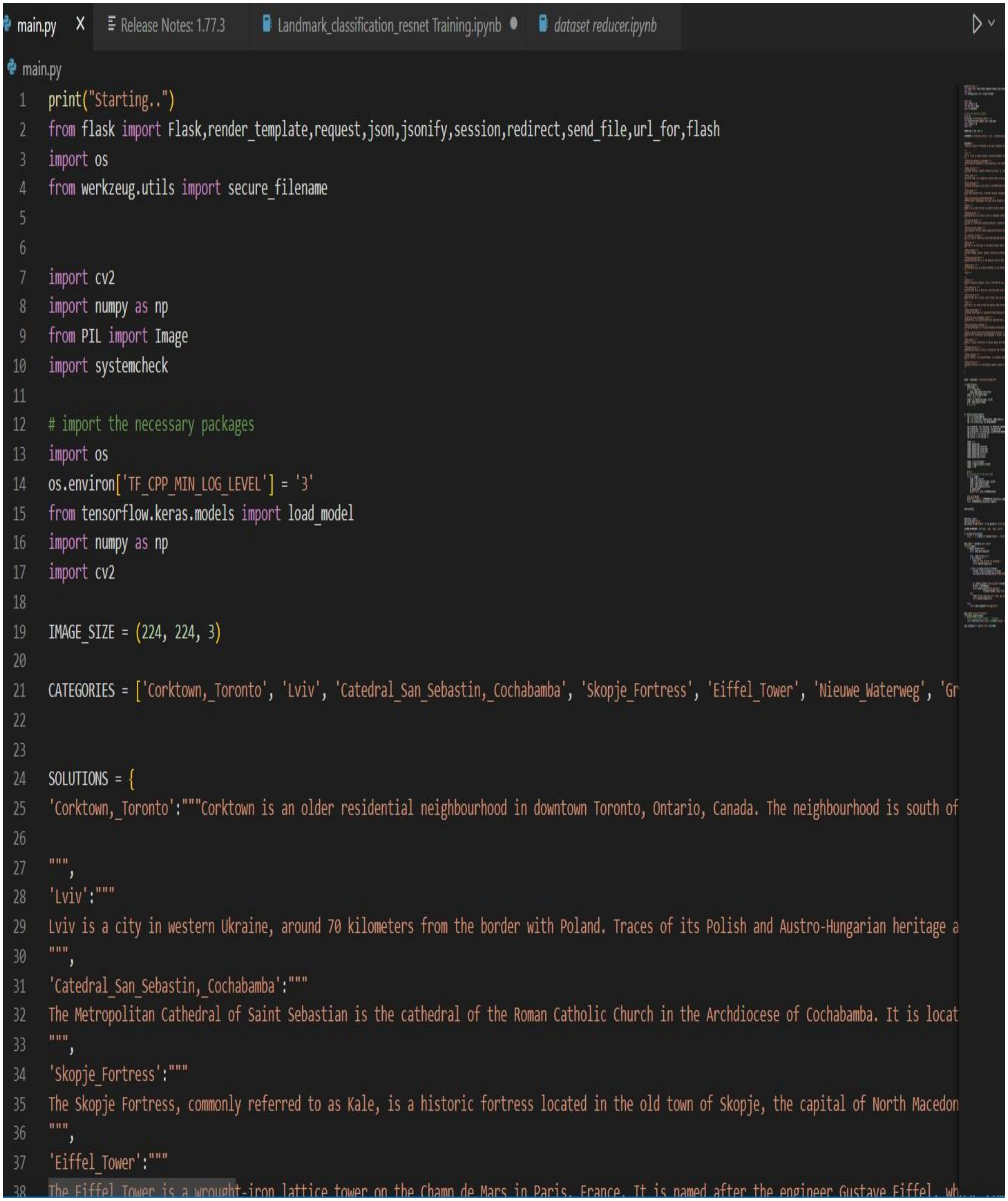
7. Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger- Densely Connected Convolutional Networks- arXiv:1608.06993.

8. Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On theimportance of initialization and momentum in deep learning. In Sanjoy Dasgupta andDavid McAllester, editors, Proceedings of the 30th International Conference onMachine Learning, volume 28 of Proceedings of Machine Learning Research, pages1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

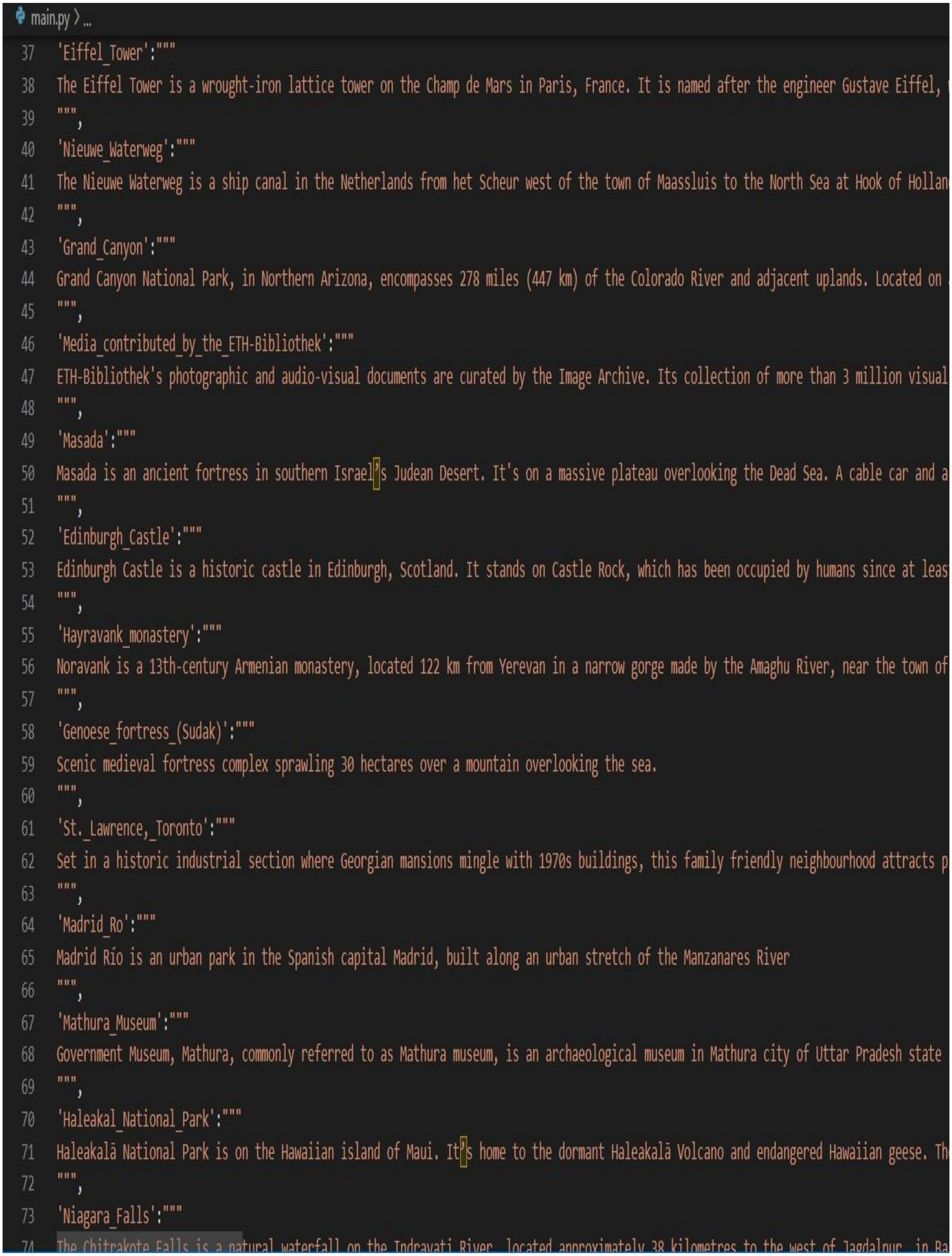
9. Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and ChunfangLiu. A survey on deep transfer learning. CoRR, abs/1808.01974, 2018.

10. Hyeonwoo Noh, Andre Araujo, Jack Sim, and Bohyung Han. Image retrieval withdeep local features and attention-based keypoints. CoRR, abs/1612.06321, 2016.

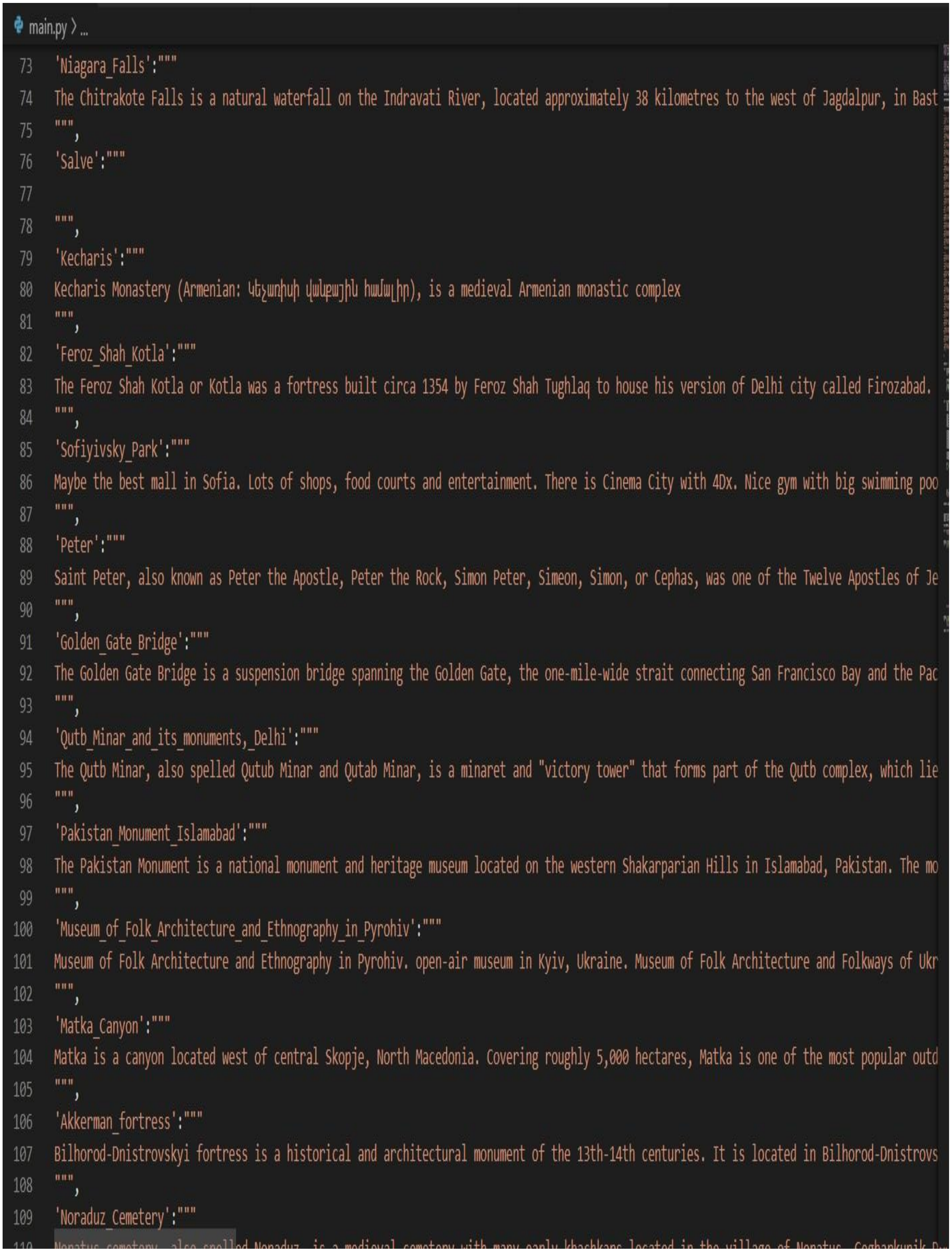
APPENDIX



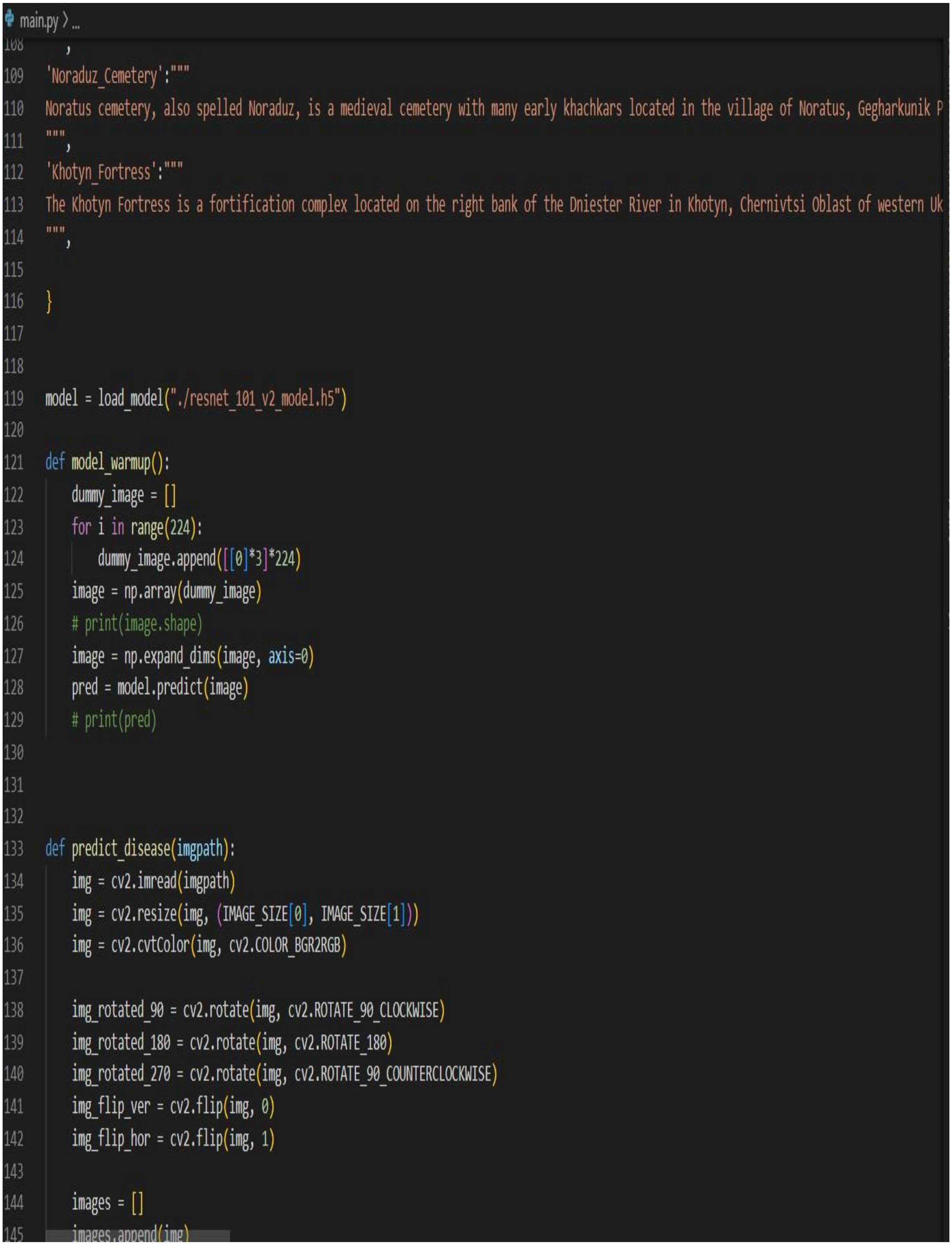
```python
print("Starting..")
from flask import Flask,render_template,request,json,jsonify,session,redirect,send_file,url_for,flash
import os
from werkzeug.utils import secure_filename


import cv2
import numpy as np
from PIL import Image
import systemcheck

# import the necessary packages
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
from tensorflow.keras.models import load_model
import numpy as np
import cv2

IMAGE_SIZE = (224, 224, 3)

CATEGORIES = ['Corktown,_Toronto', 'Lviv', 'Catedral_San_Sebastin,_Cochabamba', 'Skopje_Fortress', 'Eiffel_Tower', 'Nieuwe_Waterweg', 'Gr


SOLUTIONS = {
'Corktown,_Toronto':"""Corktown is an older residential neighbourhood in downtown Toronto, Ontario, Canada. The neighbourhood is south of

""",
'Lviv':"""
Lviv is a city in western Ukraine, around 70 kilometers from the border with Poland. Traces of its Polish and Austro-Hungarian heritage a
""",
'Catedral_San_Sebastin,_Cochabamba':"""
The Metropolitan Cathedral of Saint Sebastian is the cathedral of the Roman Catholic Church in the Archdiocese of Cochabamba. It is locat
""",
'Skopje_Fortress':"""
The Skopje Fortress, commonly referred to as Kale, is a historic fortress located in the old town of Skopje, the capital of North Macedon
""",
'Eiffel_Tower':"""
The Eiffel Tower is a wrought-iron lattice tower on the Champ de Mars in Paris, France. It is named after the engineer Gustave Eiffel, wh
```



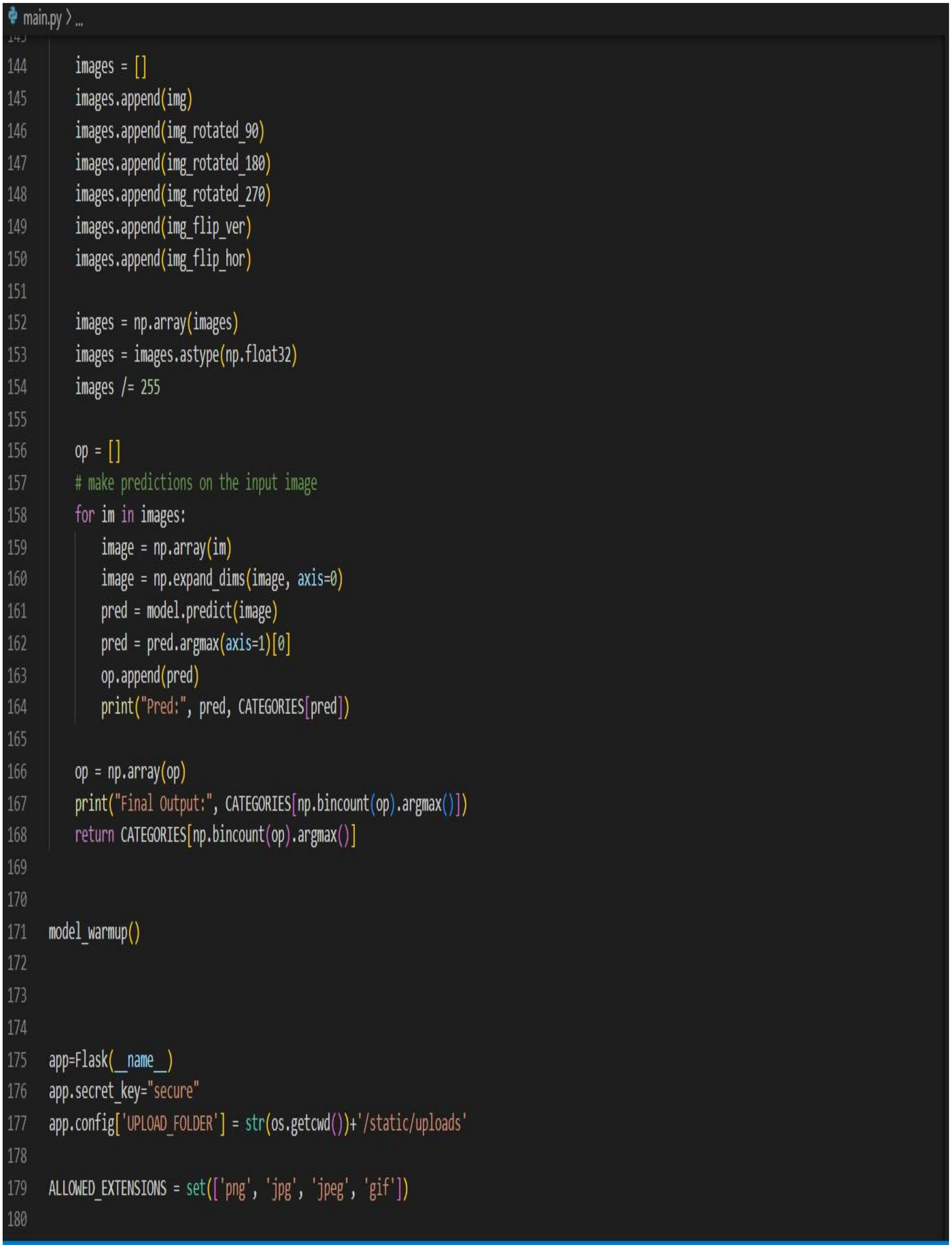
```
main.py > ...
'Eiffel_Tower':"""
The Eiffel Tower is a wrought-iron lattice tower on the Champ de Mars in Paris, France. It is named after the engineer Gustave Eiffel,
""",
'Nieuwe_Waterweg':"""
The Nieuwe Waterweg is a ship canal in the Netherlands from het Scheur west of the town of Maassluis to the North Sea at Hook of Hollan
""",
'Grand_Canyon':"""
Grand Canyon National Park, in Northern Arizona, encompasses 278 miles (447 km) of the Colorado River and adjacent uplands. Located on
""",
'Media_contributed_by_the_ETH-Bibliothek':"""
ETH-Bibliothek's photographic and audio-visual documents are curated by the Image Archive. Its collection of more than 3 million visual
""",
'Masada':"""
Masada is an ancient fortress in southern Israel's Judean Desert. It's on a massive plateau overlooking the Dead Sea. A cable car and a
""",
'Edinburgh_Castle':"""
Edinburgh Castle is a historic castle in Edinburgh, Scotland. It stands on Castle Rock, which has been occupied by humans since at leas
""",
'Hayravank_monastery':"""
Noravank is a 13th-century Armenian monastery, located 122 km from Yerevan in a narrow gorge made by the Amaghu River, near the town of
""",
'Genoese_fortress_(Sudak)':"""
Scenic medieval fortress complex sprawling 30 hectares over a mountain overlooking the sea.
""",
'St._Lawrence,_Toronto':"""
Set in a historic industrial section where Georgian mansions mingle with 1970s buildings, this family friendly neighbourhood attracts p
""",
'Madrid_Ro':"""
Madrid Río is an urban park in the Spanish capital Madrid, built along an urban stretch of the Manzanares River
""",
'Mathura_Museum':"""
Government Museum, Mathura, commonly referred to as Mathura museum, is an archaeological museum in Mathura city of Uttar Pradesh state
""",
'Haleakal_National_Park':"""
Haleakalā National Park is on the Hawaiian island of Maui. It's home to the dormant Haleakalā Volcano and endangered Hawaiian geese. Th
""",
'Niagara_Falls':"""
The Chitrakote Falls is a natural waterfall on the Indravati River, located approximately 38 kilometres to the west of Jagdalpur, in Ba
```



```python
'Niagara_Falls':"""
The Chitrakote Falls is a natural waterfall on the Indravati River, located approximately 38 kilometres to the west of Jagdalpur, in Bast
""",
'Salve':"""

""",
'Kecharis':"""
Kecharis Monastery (Armenian: Կեչառիսի վանքային համալիր), is a medieval Armenian monastic complex
""",
'Feroz_Shah_Kotla':"""
The Feroz Shah Kotla or Kotla was a fortress built circa 1354 by Feroz Shah Tughlaq to house his version of Delhi city called Firozabad.
""",
'Sofiyivsky_Park':"""
Maybe the best mall in Sofia. Lots of shops, food courts and entertainment. There is Cinema City with 4Dx. Nice gym with big swimming poo
""",
'Peter':"""
Saint Peter, also known as Peter the Apostle, Peter the Rock, Simon Peter, Simeon, Simon, or Cephas, was one of the Twelve Apostles of Je
""",
'Golden_Gate_Bridge':"""
The Golden Gate Bridge is a suspension bridge spanning the Golden Gate, the one-mile-wide strait connecting San Francisco Bay and the Pac
""",
'Qutb_Minar_and_its_monuments,_Delhi':"""
The Qutb Minar, also spelled Qutub Minar and Qutab Minar, is a minaret and "victory tower" that forms part of the Qutb complex, which lie
""",
'Pakistan_Monument_Islamabad':"""
The Pakistan Monument is a national monument and heritage museum located on the western Shakarparian Hills in Islamabad, Pakistan. The mo
""",
'Museum_of_Folk_Architecture_and_Ethnography_in_Pyrohiv':"""
Museum of Folk Architecture and Ethnography in Pyrohiv. open-air museum in Kyiv, Ukraine. Museum of Folk Architecture and Folkways of Ukr
""",
'Matka_Canyon':"""
Matka is a canyon located west of central Skopje, North Macedonia. Covering roughly 5,000 hectares, Matka is one of the most popular outd
""",
'Akkerman_fortress':"""
Bilhorod-Dnistrovskyi fortress is a historical and architectural monument of the 13th-14th centuries. It is located in Bilhorod-Dnistrovs
""",
'Noraduz_Cemetery':"""
Noratus cemetery, also spelled Noraduz, is a medieval cemetery with many early khachkars located in the village of Noratus, Gegharkunik Pr
```

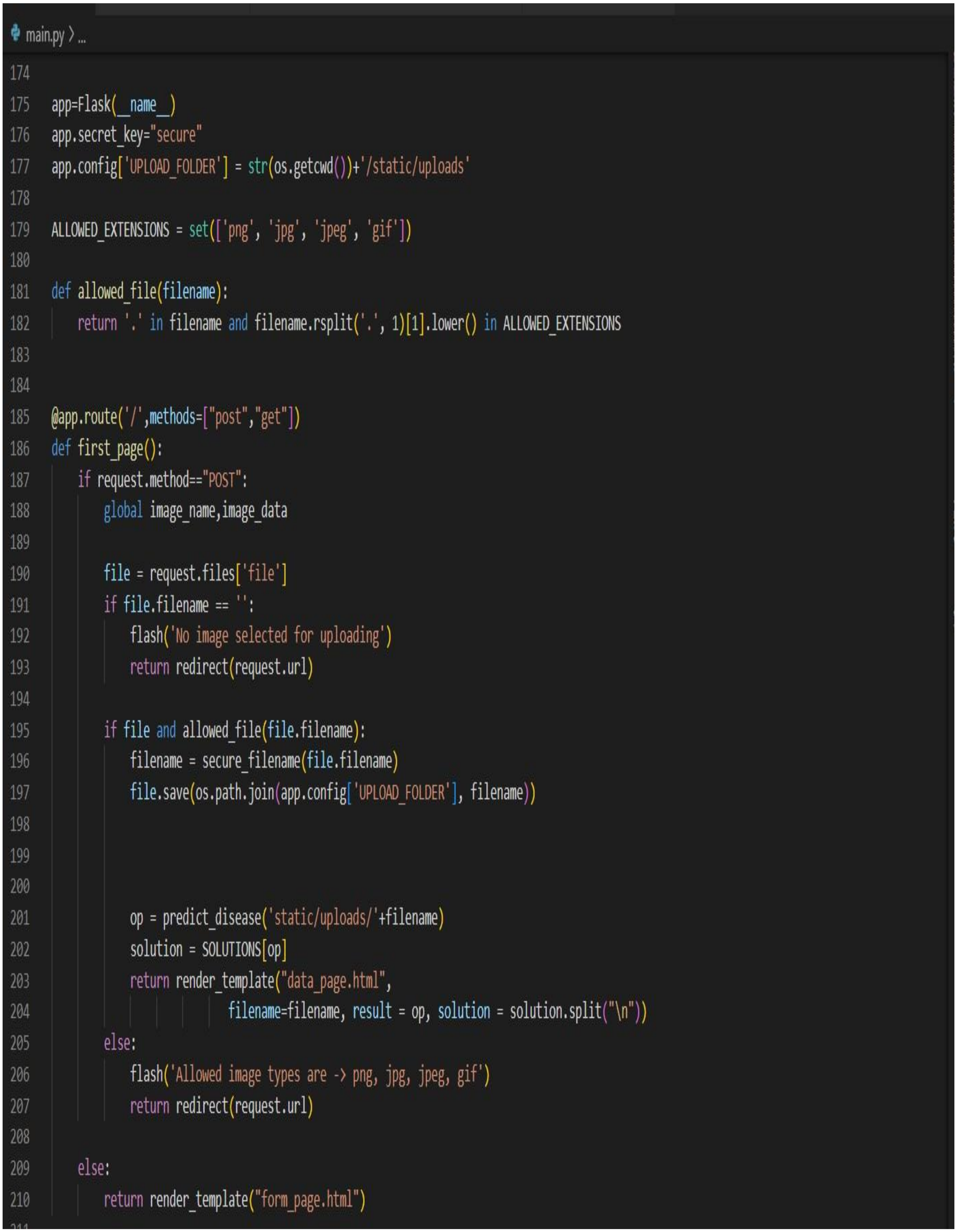


```
main.py > ...
    ,
'Noraduz_Cemetery':"""
Noratus cemetery, also spelled Noraduz, is a medieval cemetery with many early khachkars located in the village of Noratus, Gegharkunik P
""",
'Khotyn_Fortress':"""
The Khotyn Fortress is a fortification complex located on the right bank of the Dniester River in Khotyn, Chernivtsi Oblast of western Uk
""",


}



model = load_model("./resnet_101_v2_model.h5")

def model_warmup():
    dummy_image = []
    for i in range(224):
        dummy_image.append([[0]*3]*224)
    image = np.array(dummy_image)
    # print(image.shape)
    image = np.expand_dims(image, axis=0)
    pred = model.predict(image)
    # print(pred)



def predict_disease(imgpath):
    img = cv2.imread(imgpath)
    img = cv2.resize(img, (IMAGE_SIZE[0], IMAGE_SIZE[1]))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    img_rotated_90 = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)
    img_rotated_180 = cv2.rotate(img, cv2.ROTATE_180)
    img_rotated_270 = cv2.rotate(img, cv2.ROTATE_90_COUNTERCLOCKWISE)
    img_flip_ver = cv2.flip(img, 0)
    img_flip_hor = cv2.flip(img, 1)

    images = []
    images.append(img)
```



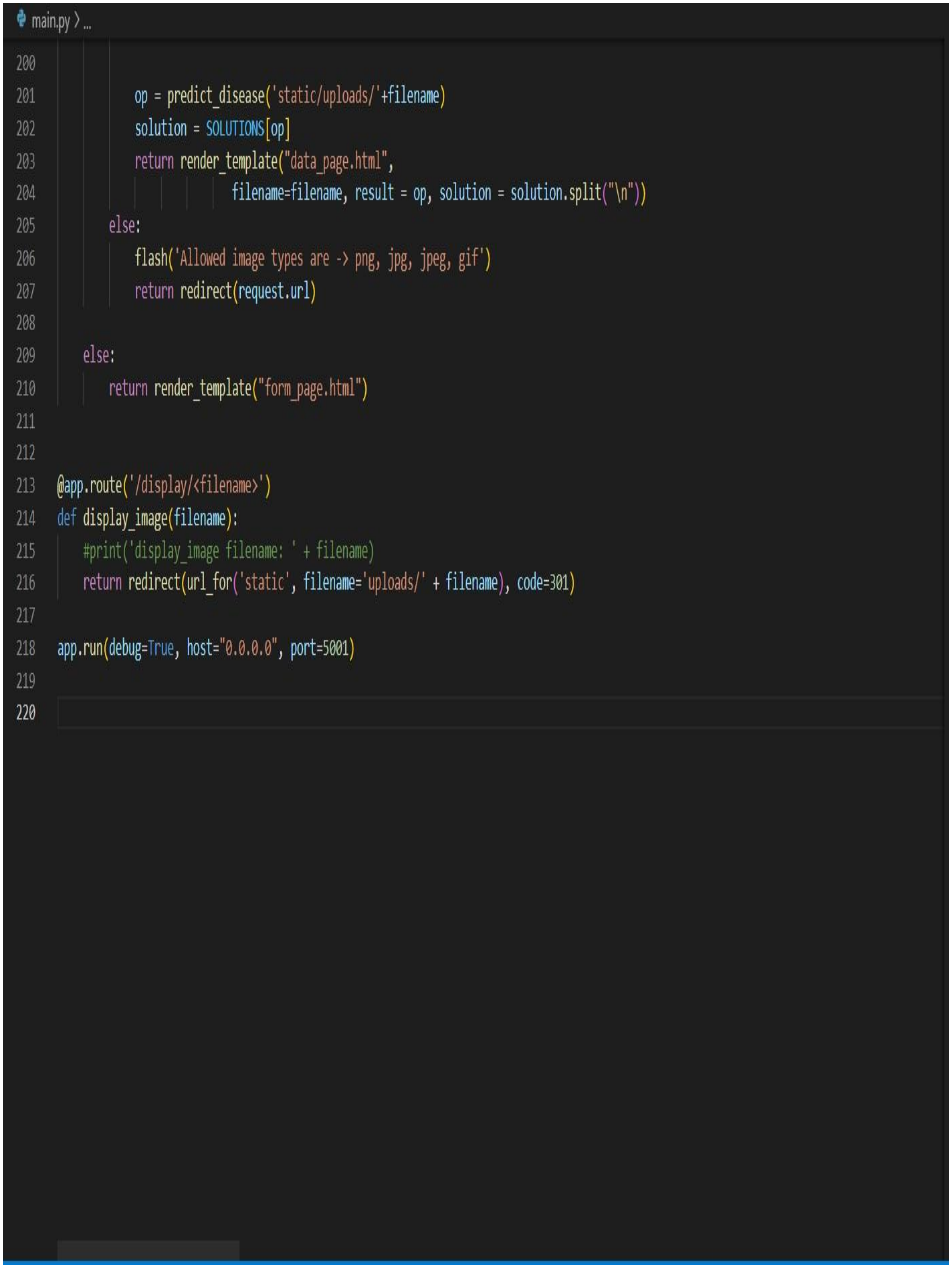
main.py > ...

```
    images = []
    images.append(img)
    images.append(img_rotated_90)
    images.append(img_rotated_180)
    images.append(img_rotated_270)
    images.append(img_flip_ver)
    images.append(img_flip_hor)

    images = np.array(images)
    images = images.astype(np.float32)
    images /= 255

    op = []
    # make predictions on the input image
    for im in images:
        image = np.array(im)
        image = np.expand_dims(image, axis=0)
        pred = model.predict(image)
        pred = pred.argmax(axis=1)[0]
        op.append(pred)
        print("Pred:", pred, CATEGORIES[pred])

    op = np.array(op)
    print("Final Output:", CATEGORIES[np.bincount(op).argmax()])
    return CATEGORIES[np.bincount(op).argmax()]


model_warmup()



app=Flask(__name__)
app.secret_key="secure"
app.config['UPLOAD_FOLDER'] = str(os.getcwd())+'/static/uploads'

ALLOWED_EXTENSIONS = set(['png', 'jpg', 'jpeg', 'gif'])

```



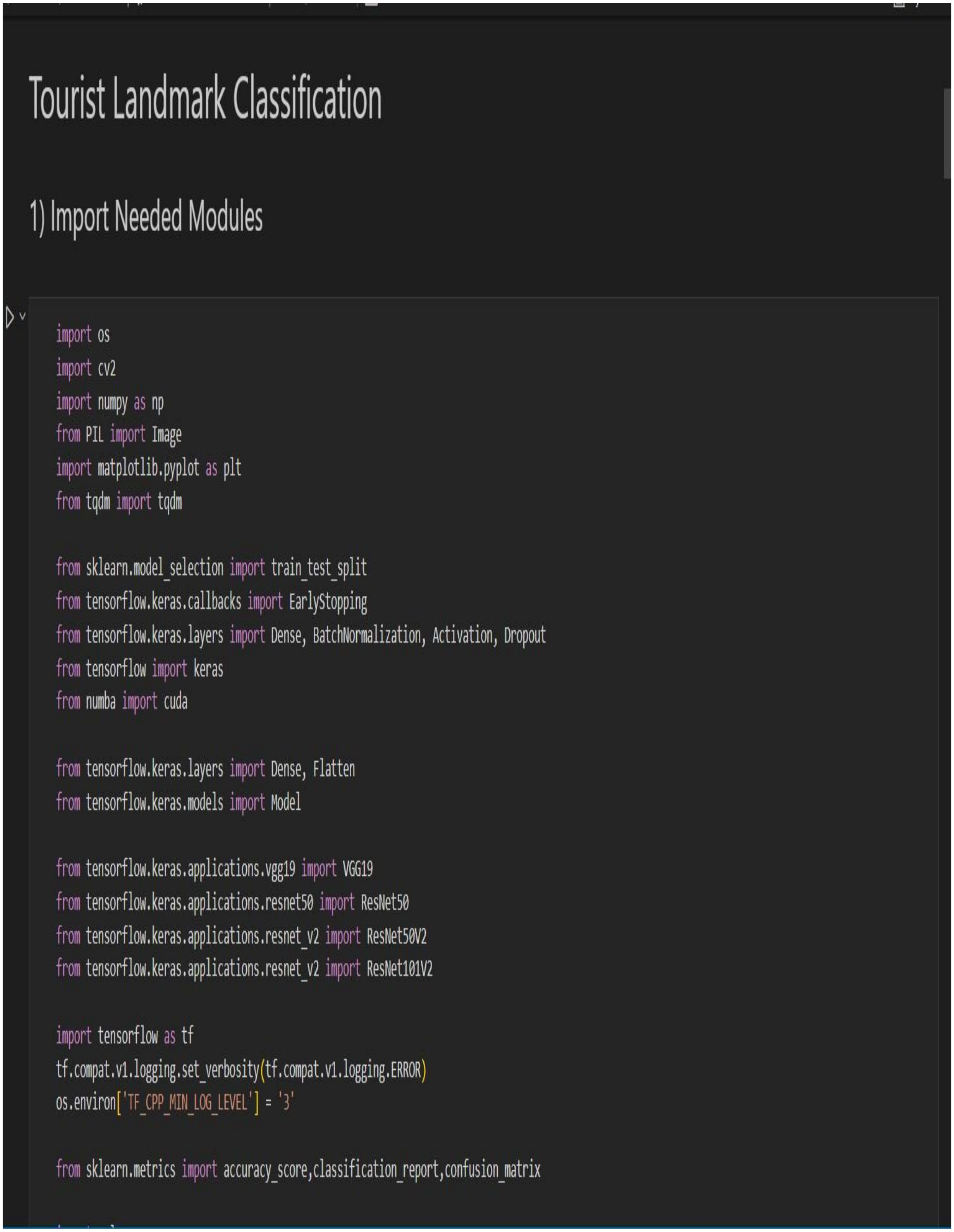
main.py > ...

```python

app=Flask(__name__)
app.secret_key="secure"
app.config['UPLOAD_FOLDER'] = str(os.getcwd())+'/static/uploads'

ALLOWED_EXTENSIONS = set(['png', 'jpg', 'jpeg', 'gif'])

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS


@app.route('/',methods=["post","get"])
def first_page():
    if request.method=="POST":
        global image_name,image_data

        file = request.files['file']
        if file.filename == '':
            flash('No image selected for uploading')
            return redirect(request.url)

        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))



            op = predict_disease('static/uploads/'+filename)
            solution = SOLUTIONS[op]
            return render_template("data_page.html",
                                    filename=filename, result = op, solution = solution.split("\n"))
        else:
            flash('Allowed image types are -> png, jpg, jpeg, gif')
            return redirect(request.url)

    else:
        return render_template("form_page.html")
```



main.py > ...

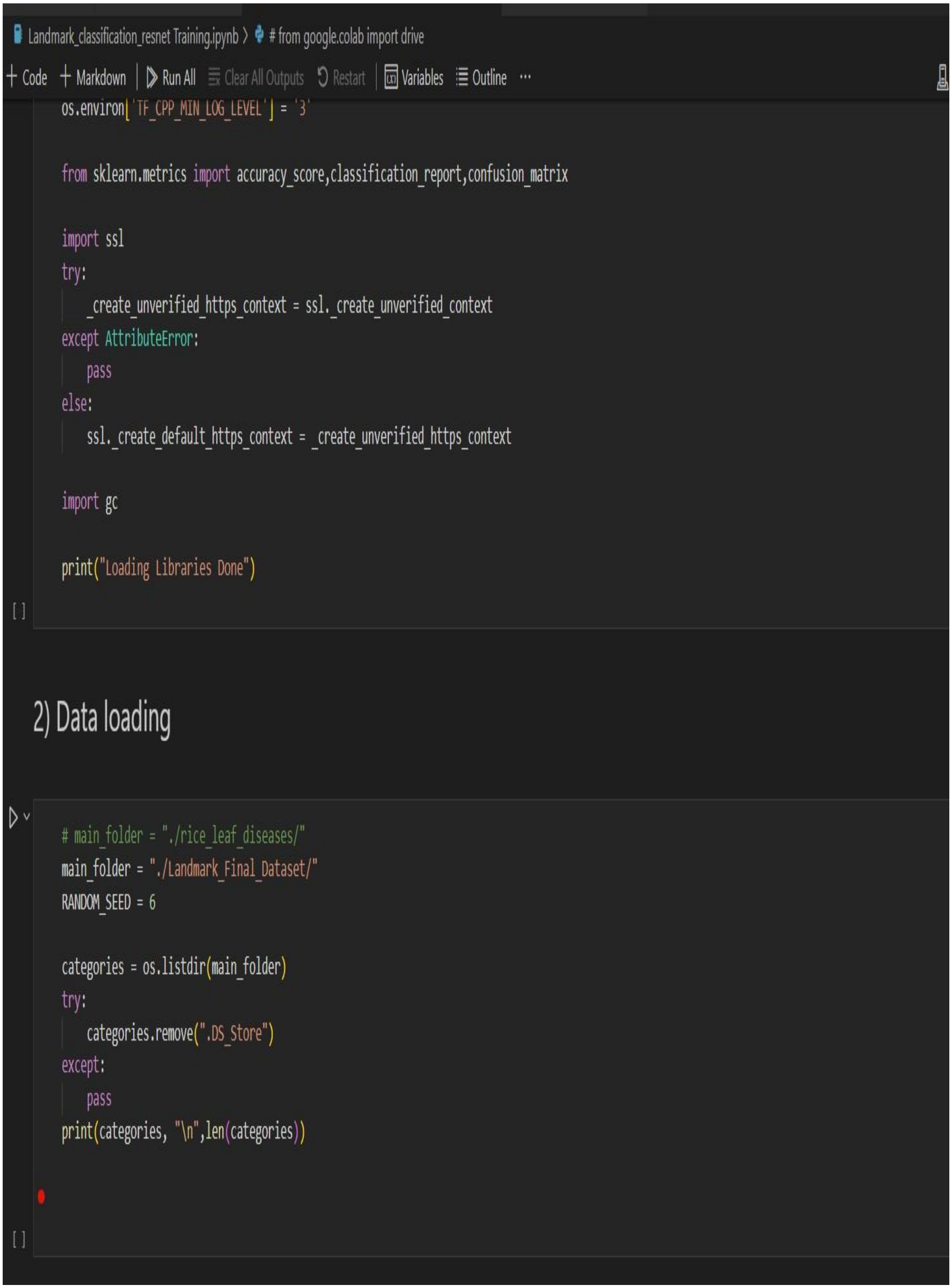
```python

            op = predict_disease('static/uploads/'+filename)
            solution = SOLUTIONS[op]
            return render_template("data_page.html",
                                   filename=filename, result = op, solution = solution.split("\n"))
        else:
            flash('Allowed image types are -> png, jpg, jpeg, gif')
            return redirect(request.url)

    else:
        return render_template("form_page.html")


@app.route('/display/<filename>')
def display_image(filename):
    #print('display_image filename: ' + filename)
    return redirect(url_for('static', filename='uploads/' + filename), code=301)

app.run(debug=True, host="0.0.0.0", port=5001)


```



Tourist Landmark Classification

1) Import Needed Modules

```
import os
import cv2
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
from tqdm import tqdm

from sklearn.model_selection import train_test_split
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.layers import Dense, BatchNormalization, Activation, Dropout
from tensorflow import keras
from numba import cuda

from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model

from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications.resnet_v2 import ResNet50V2
from tensorflow.keras.applications.resnet_v2 import ResNet101V2

import tensorflow as tf
tf.compat.v1.logging.set_verbosity(tf.compat.v1.logging.ERROR)
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
```


Landmark_classification_resnet Training.ipynb > # from google.colab import drive
+ Code + Markdown | Run All Clear All Outputs Restart | Variables Outline ···

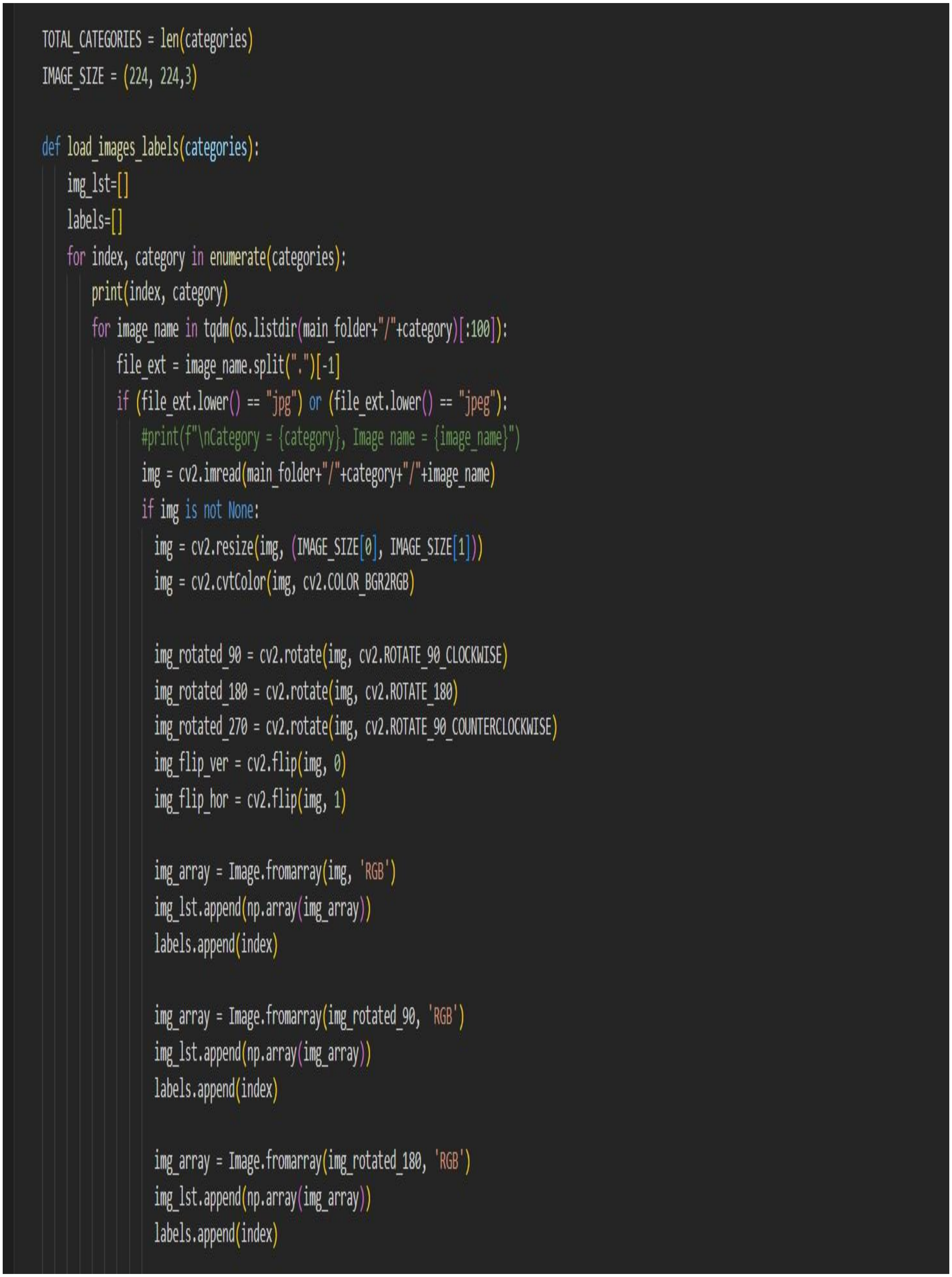
```python
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

from sklearn.metrics import accuracy_score,classification_report,confusion_matrix

import ssl
try:
    _create_unverified_https_context = ssl._create_unverified_context
except AttributeError:
    pass
else:
    ssl._create_default_https_context = _create_unverified_https_context

import gc

print("Loading Libraries Done")
```

[ ]

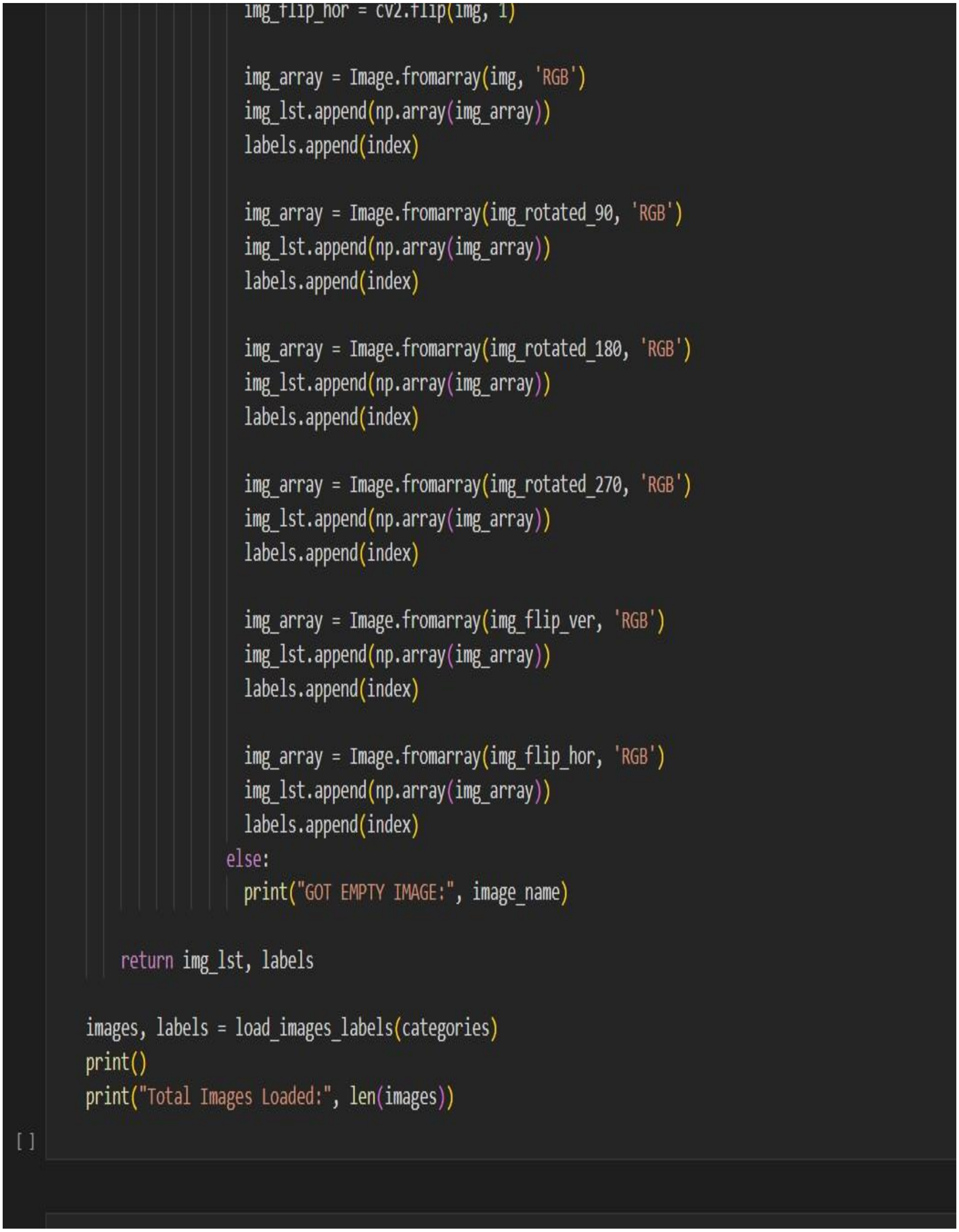
2) Data loading

```python
# main_folder = "./rice_leaf_diseases/"
main_folder = "./Landmark_Final_Dataset/"
RANDOM_SEED = 6

categories = os.listdir(main_folder)
try:
    categories.remove(".DS_Store")
except:
    pass
print(categories, "\n",len(categories))
```

[ ]

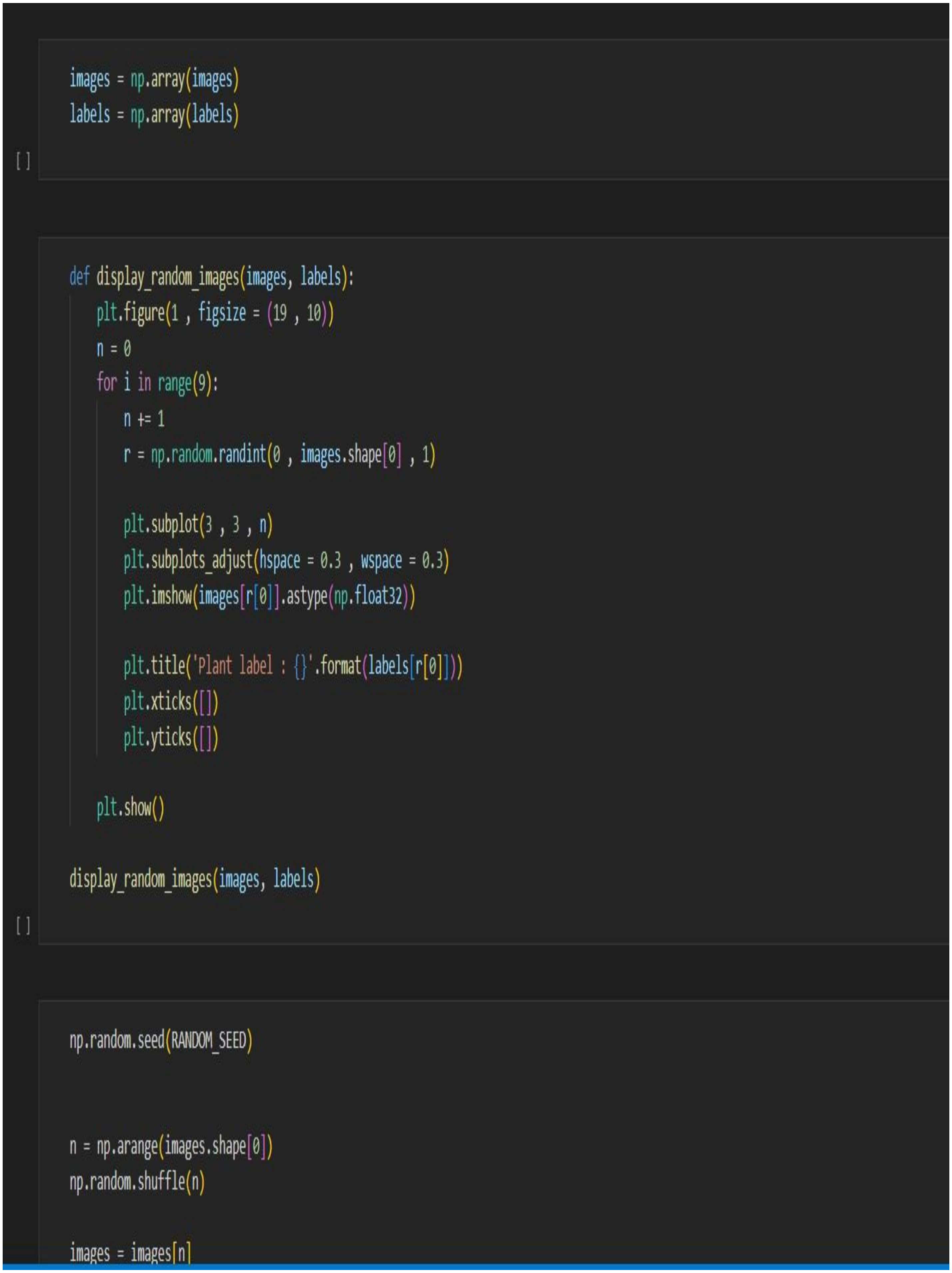


```python
TOTAL_CATEGORIES = len(categories)
IMAGE_SIZE = (224, 224,3)

def load_images_labels(categories):
    img_lst=[]
    labels=[]
    for index, category in enumerate(categories):
        print(index, category)
        for image_name in tqdm(os.listdir(main_folder+"/"+category)[:100]):
            file_ext = image_name.split(".")[-1]
            if (file_ext.lower() == "jpg") or (file_ext.lower() == "jpeg"):
                #print(f"\nCategory = {category}, Image name = {image_name}")
                img = cv2.imread(main_folder+"/"+category+"/"+image_name)
                if img is not None:
                  img = cv2.resize(img, (IMAGE_SIZE[0], IMAGE_SIZE[1]))
                  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

                  img_rotated_90 = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)
                  img_rotated_180 = cv2.rotate(img, cv2.ROTATE_180)
                  img_rotated_270 = cv2.rotate(img, cv2.ROTATE_90_COUNTERCLOCKWISE)
                  img_flip_ver = cv2.flip(img, 0)
                  img_flip_hor = cv2.flip(img, 1)

                  img_array = Image.fromarray(img, 'RGB')
                  img_lst.append(np.array(img_array))
                  labels.append(index)

                  img_array = Image.fromarray(img_rotated_90, 'RGB')
                  img_lst.append(np.array(img_array))
                  labels.append(index)

                  img_array = Image.fromarray(img_rotated_180, 'RGB')
                  img_lst.append(np.array(img_array))
                  labels.append(index)
```



```
                  img_flip_hor = cv2.flip(img, 1)

                  img_array = Image.fromarray(img, 'RGB')
                  img_lst.append(np.array(img_array))
                  labels.append(index)

                  img_array = Image.fromarray(img_rotated_90, 'RGB')
                  img_lst.append(np.array(img_array))
                  labels.append(index)

                  img_array = Image.fromarray(img_rotated_180, 'RGB')
                  img_lst.append(np.array(img_array))
                  labels.append(index)

                  img_array = Image.fromarray(img_rotated_270, 'RGB')
                  img_lst.append(np.array(img_array))
                  labels.append(index)

                  img_array = Image.fromarray(img_flip_ver, 'RGB')
                  img_lst.append(np.array(img_array))
                  labels.append(index)

                  img_array = Image.fromarray(img_flip_hor, 'RGB')
                  img_lst.append(np.array(img_array))
                  labels.append(index)
                else:
                  print("GOT EMPTY IMAGE:", image_name)

    return img_lst, labels

images, labels = load_images_labels(categories)
print()
print("Total Images Loaded:", len(images))
```

[ ]



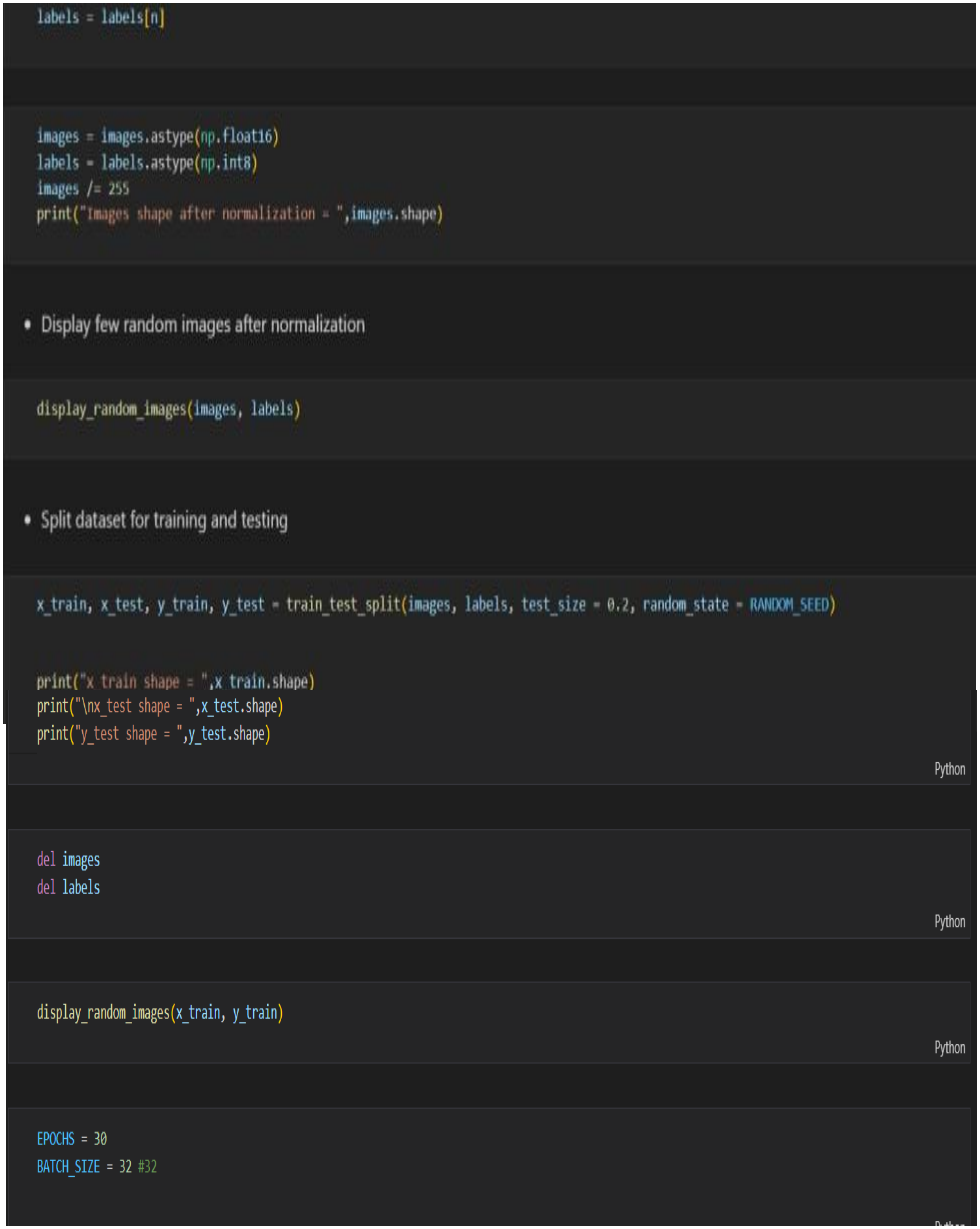
```
images = np.array(images)
labels = np.array(labels)
```

[ ]

```
def display_random_images(images, labels):
    plt.figure(1 , figsize = (19 , 10))
    n = 0
    for i in range(9):
        n += 1
        r = np.random.randint(0 , images.shape[0] , 1)

        plt.subplot(3 , 3 , n)
        plt.subplots_adjust(hspace = 0.3 , wspace = 0.3)
        plt.imshow(images[r[0]].astype(np.float32))

        plt.title('Plant label : {}'.format(labels[r[0]]))
        plt.xticks([])
        plt.yticks([])

    plt.show()

display_random_images(images, labels)
```

[ ]

```
np.random.seed(RANDOM_SEED)



n = np.arange(images.shape[0])
np.random.shuffle(n)


images = images[n]
```



```
labels = labels[n]
```

```
images = images.astype(np.float16)
labels = labels.astype(np.int8)
images /= 255
print("Images shape after normalization = ",images.shape)
```

- Display few random images after normalization

```
display_random_images(images, labels)
```

- Split dataset for training and testing

```
x_train, x_test, y_train, y_test = train_test_split(images, labels, test_size = 0.2, random_state = RANDOM_SEED)


print("x_train shape = ",x_train.shape)
print("\nx_test shape = ",x_test.shape)
print("y_test shape = ",y_test.shape)
```

Python

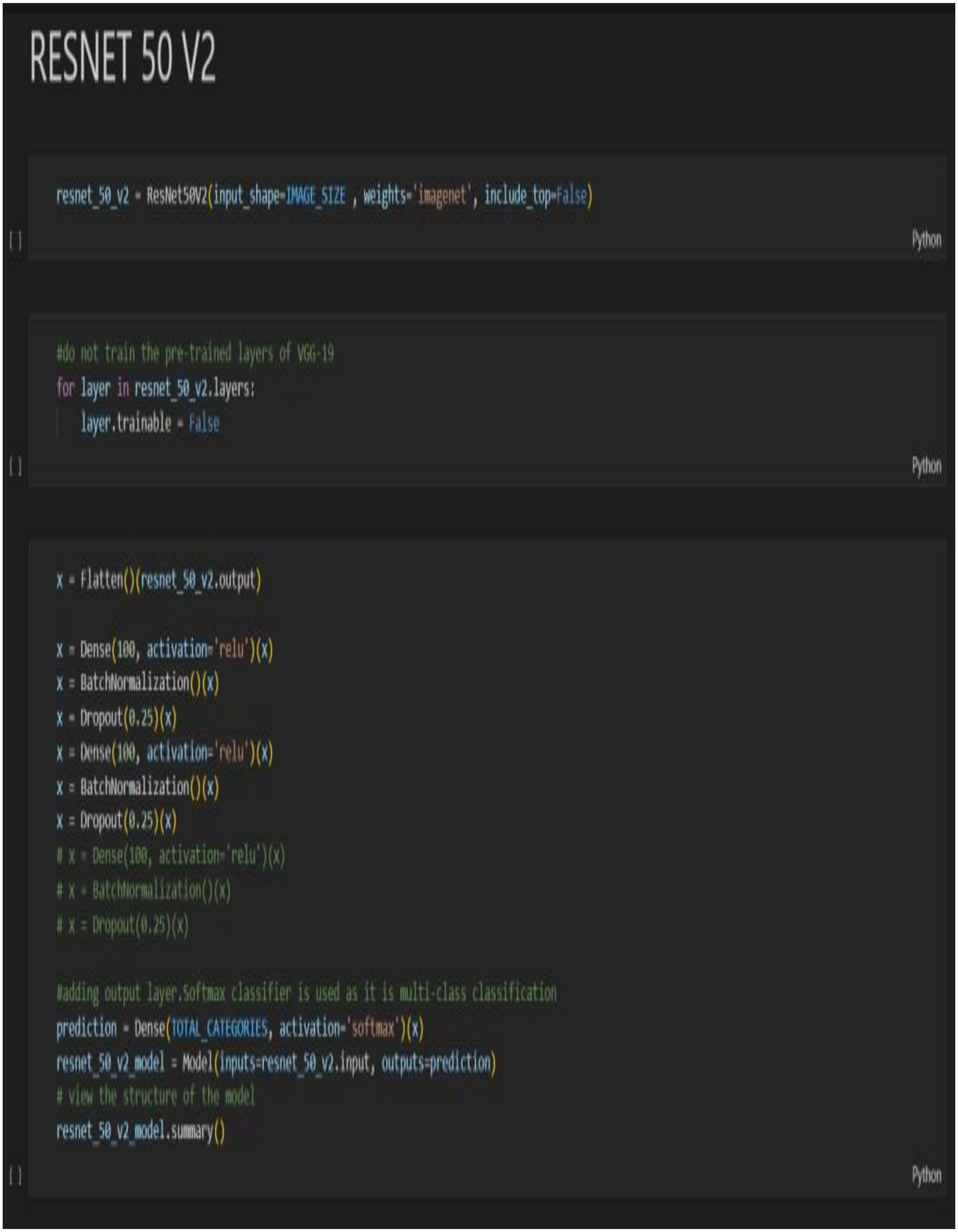
```
del images
del labels
```

Python

```
display_random_images(x_train, y_train)
```

Python

```
EPOCHS = 30
BATCH_SIZE = 32 #32
```

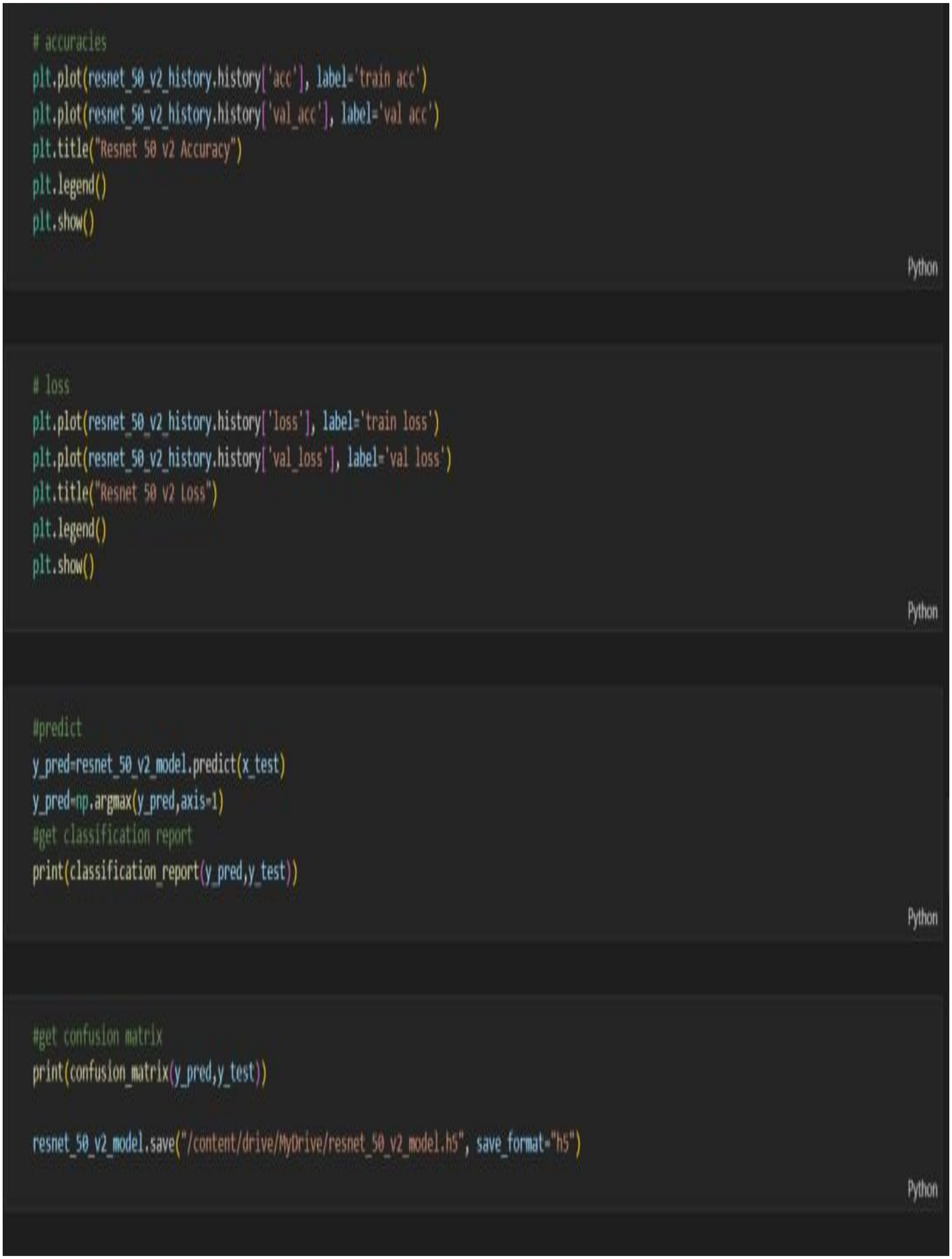


RESNET 50 V2

```python
resnet_50_v2 = ResNet50V2(input_shape=IMAGE_SIZE , weights='imagenet', include_top=False)
```

```python
#do not train the pre-trained layers of VGG-19
for layer in resnet_50_v2.layers:
    layer.trainable = False
```

```python
x = Flatten()(resnet_50_v2.output)

x = Dense(100, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.25)(x)
x = Dense(100, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.25)(x)
# x = Dense(100, activation='relu')(x)
# x = BatchNormalization()(x)
# x = Dropout(0.25)(x)

#adding output layer.Softmax classifier is used as it is multi-class classification
prediction = Dense(TOTAL_CATEGORIES, activation='softmax')(x)
resnet_50_v2_model = Model(inputs=resnet_50_v2.input, outputs=prediction)
# view the structure of the model
resnet_50_v2_model.summary()
```

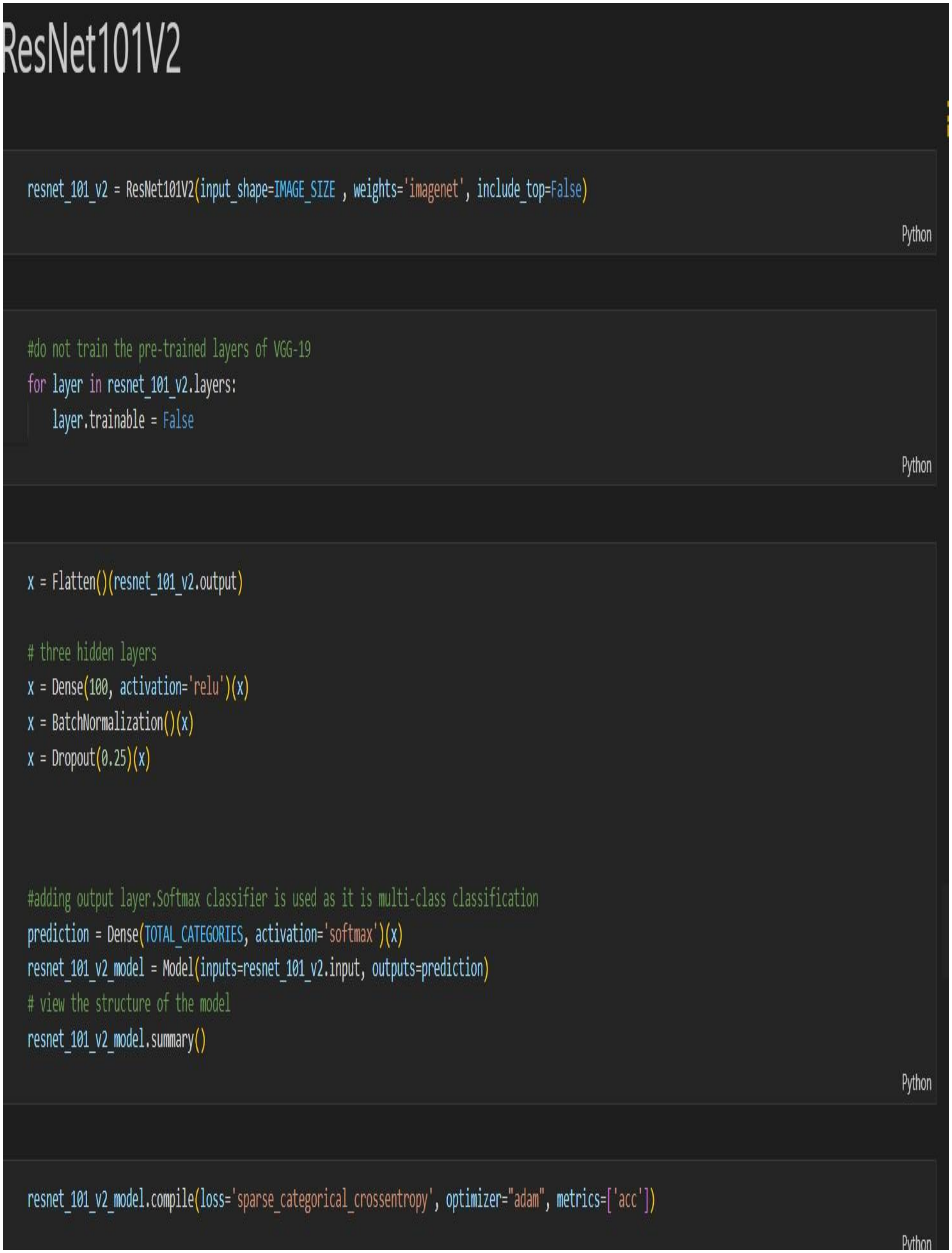


```python
# accuracies
plt.plot(resnet_50_v2_history.history['acc'], label='train acc')
plt.plot(resnet_50_v2_history.history['val_acc'], label='val acc')
plt.title("Resnet 50 v2 Accuracy")
plt.legend()
plt.show()
```

```python
# loss
plt.plot(resnet_50_v2_history.history['loss'], label='train loss')
plt.plot(resnet_50_v2_history.history['val_loss'], label='val loss')
plt.title("Resnet 50 v2 Loss")
plt.legend()
plt.show()
```

```python
#predict
y_pred=resnet_50_v2_model.predict(x_test)
y_pred=np.argmax(y_pred,axis=1)
#get classification report
print(classification_report(y_pred,y_test))
```

```python
#get confusion matrix
print(confusion_matrix(y_pred,y_test))

resnet_50_v2_model.save("/content/drive/MyDrive/resnet_50_v2_model.h5", save_format="h5")
```



ResNet101V2

```
resnet_101_v2 = ResNet101V2(input_shape=IMAGE_SIZE , weights='imagenet', include_top=False)
```

Python

```
#do not train the pre-trained layers of VGG-19
for layer in resnet_101_v2.layers:
    layer.trainable = False
```

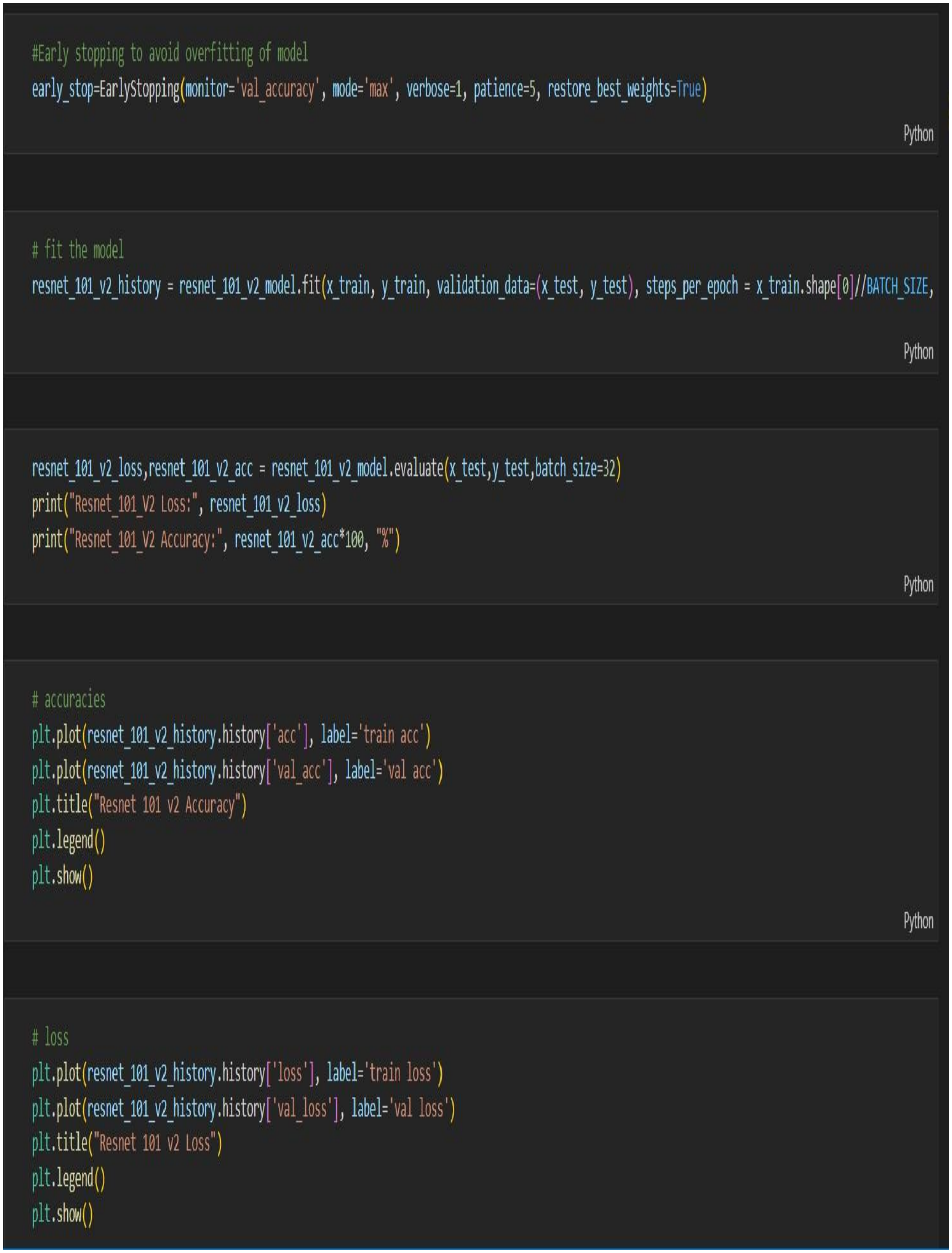
Python

```
x = Flatten()(resnet_101_v2.output)

# three hidden layers
x = Dense(100, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.25)(x)



#adding output layer.Softmax classifier is used as it is multi-class classification
prediction = Dense(TOTAL_CATEGORIES, activation='softmax')(x)
resnet_101_v2_model = Model(inputs=resnet_101_v2.input, outputs=prediction)
# view the structure of the model
resnet_101_v2_model.summary()
```

Python

```
resnet_101_v2_model.compile(loss='sparse_categorical_crossentropy', optimizer="adam", metrics=['acc'])
```

Python



```python
#Early stopping to avoid overfitting of model
early_stop=EarlyStopping(monitor='val_accuracy', mode='max', verbose=1, patience=5, restore_best_weights=True)
```

Python

```python
# fit the model
resnet_101_v2_history = resnet_101_v2_model.fit(x_train, y_train, validation_data=(x_test, y_test), steps_per_epoch = x_train.shape[0]//BATCH_SIZE,
```

Python

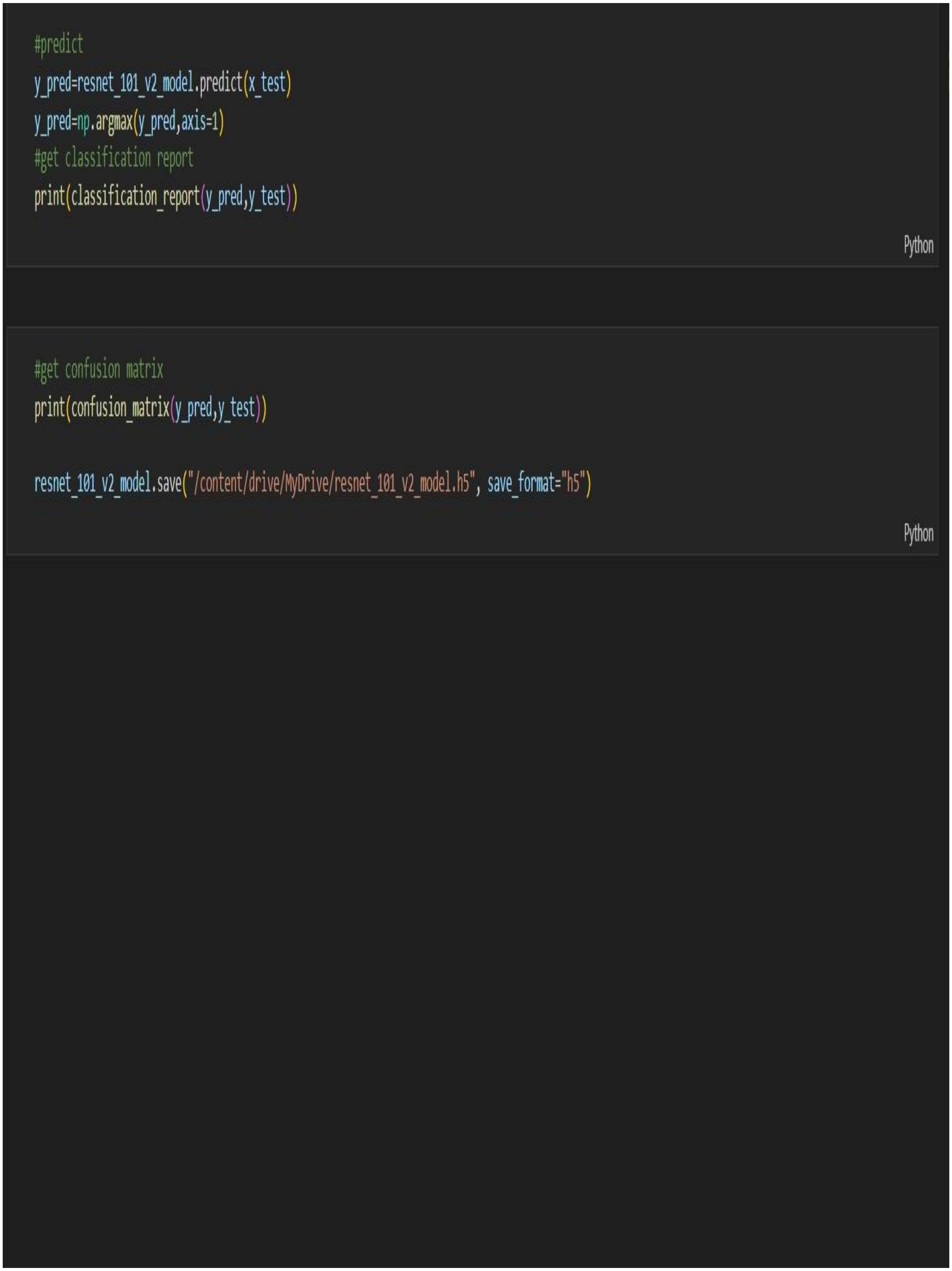
```python
resnet_101_v2_loss,resnet_101_v2_acc = resnet_101_v2_model.evaluate(x_test,y_test,batch_size=32)
print("Resnet_101_V2 Loss:", resnet_101_v2_loss)
print("Resnet_101_V2 Accuracy:", resnet_101_v2_acc*100, "%")
```

Python

```python
# accuracies
plt.plot(resnet_101_v2_history.history['acc'], label='train acc')
plt.plot(resnet_101_v2_history.history['val_acc'], label='val acc')
plt.title("Resnet 101 v2 Accuracy")
plt.legend()
plt.show()
```

Python

```python
# loss
plt.plot(resnet_101_v2_history.history['loss'], label='train loss')
plt.plot(resnet_101_v2_history.history['val_loss'], label='val loss')
plt.title("Resnet 101 v2 Loss")
plt.legend()
plt.show()
```



```python
#predict
y_pred=resnet_101_v2_model.predict(x_test)
y_pred=np.argmax(y_pred,axis=1)
#get classification report
print(classification_report(y_pred,y_test))
```

Python

```python
#get confusion matrix
print(confusion_matrix(y_pred,y_test))

resnet_101_v2_model.save("/content/drive/MyDrive/resnet_101_v2_model.h5", save_format="h5")
```

Python