

**Image Scraping and Classification Project**

Submitted by:

Poonam Singh

Internship 17

**ACKNOWLEDGMENT**

The internship opportunity I had with FlipRobo was a great chance for learning and professional development. Therefore, I consider myself as a very lucky individual as I was provided with an opportunity to be a part of it. I am also grateful for having a chance to meet so many wonderful people and professionals who led me though this project period.

I would like to thank our SME for suggesting this project and for his whole hearted cooperation and constant encouragement throughout the project.

**INTRODUCTION**

* Business Problem Framing

Images are one of the major sources of data in the field of data science and AI. This field is making appropriate use of information that can be gathered through images by examining its features and details. We are trying to give you an exposure of how an end to end project is developed in this field. The idea behind this project is to build a deep learning-based Image Classification model on images that will be scraped from e-commerce portal. This is done to make the model more and more robust.

* Conceptual Background of the Domain Problem

Nowadays internet is filled with an abundance of images and videos, which is encouraging the development of search applications and algorithms that can examine the semantic analysis of image and videos for presenting the user with better search content and their summarization. There have been major breakthroughs in image labelling, object detection, scene classification, areas reported by different researchers across the world. This leads to making it possible to formulate approaches concerning object detection and scene classification problems. Since artificial neural networks have shown a performance breakthrough in the area of object detection and scene classification, especially convolutional neural networks (CNN), this work focuses on identifying the best network for this purpose. Feature extraction is a key step of such algorithms. Feature extraction from images involves extracting a minimal set of features containing a high amount of object or scene information from low -level image pixel values, therefore, capturing the difference among the object categories involved.

* Review of Literature

Image classification is a task of great importance and a lot of real-world application. One of the best examples of this is in self-driving cars. When we drive a car we need to follow some traffic rules. But how can a self-driving car know what sign it is? This is where image classification comes into play. We can train a deep learning based model to predict from the image of sign what type of sign it is and can take actions accordingly. So in the similar manner we train our model to classify the image to be saree, jeans or trouser.

* Project Objectives:
* Collect a dataset of images of clothing of saree for women, jeans for men, trouser for men from the online shopping website amazon.com
* Produce a convolutional neural network which is capable correctly classifying images of clothes with an average confidence level of 80% or more.

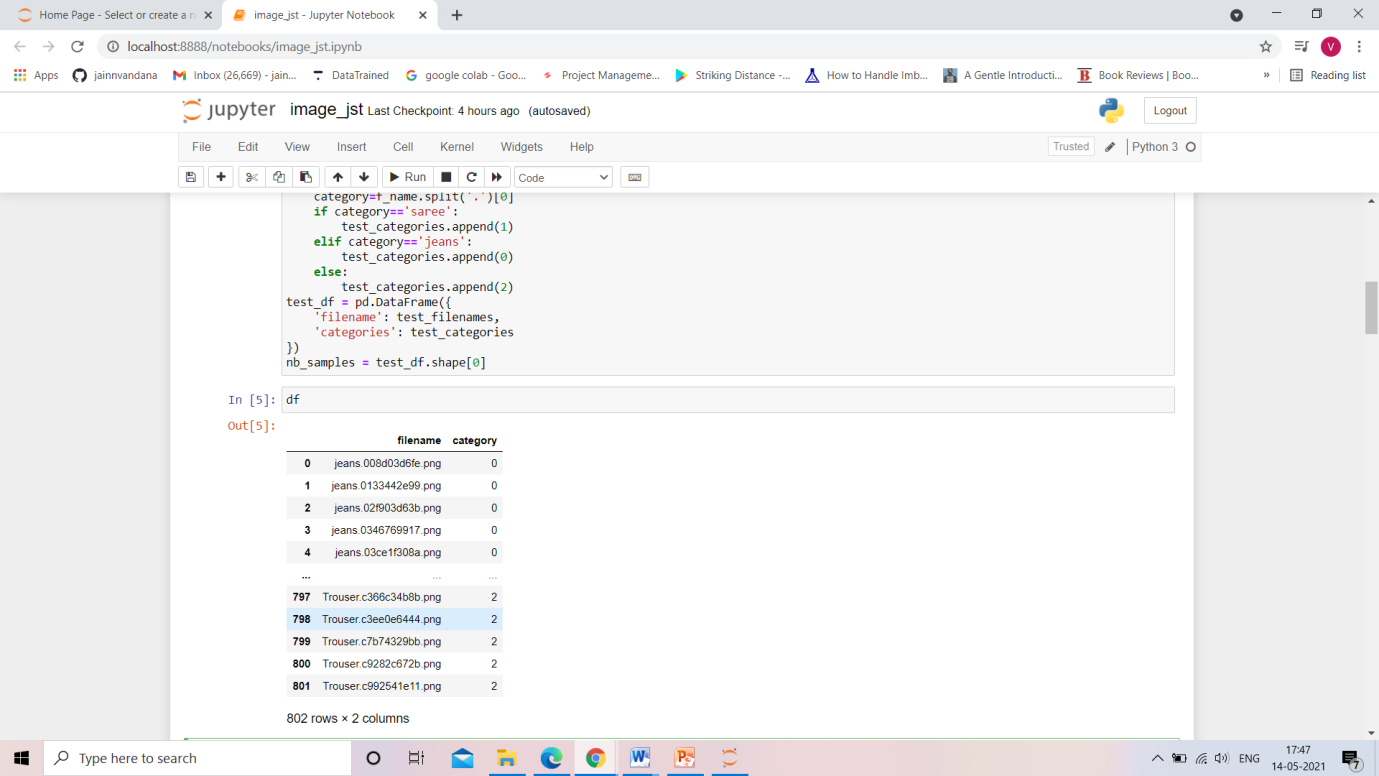
**Analytical Problem Framing**

* Mathematical/ Analytical Modeling of the Problem

The main aim of our work is to understand the performance of the networks for static as well as live video feeds. The first step for the following is to perform transfer learning on the networks with image datasets. This is followed by checking the prediction rate of the same object on static images .The different accuracy rates are observed and noted and presented in the tables given in further sections. Third important criteria for evaluating the performance were to check whether prediction accuracy varies across all CNNs chosen for the study. We are looking for best image classifier where the object is the main attribute for classification of dress category. Different layers of the convolutional neural network used are:

* Input Layer: The first layer of each CNN used is ‘input layer’ which takes images, resize them for passing onto further layers for feature extraction.
* Convolution Layer: The next few layers are ‘Convolution layers’ which act as filters for images, hence finding out features from images and also used for calculating the match feature points during testing.
* Pooling Layer: The extracted feature sets are then passed ‘pooling layer’. This layer takes large images and shrink them down while preserving the most important information in them. It keeps the maximum value from each window, it preserves the best fits of each feature within the window.
* Rectified Linear Unit Layer:The next ‘Rectified Linear Unit’ or ReLU layer swaps every negative number of the pooling layer with 0. This helps the CNN stay mathematically stable by keeping learned values from getting stuck near 0 or blowing up toward infinity.
* Fully Connected Layer: The final layer is the fully connected layers which takes the high-level filtered image and translate them into categories with labels.
* Data Sources and their formats

The data is scrapped from the online shopping website amazon.com . The dataset consists of around 1012 images in one subfolder with filenames looking like ‘saree.69ebfef69b.png’ , ‘jeans.a4de31a840’ and ‘Trouser.8c97a17e3c’. Image data is very big and we can’t load the complete dataset into RAM. So we need to make a method to pick an image or a batch of images from the dataset and we will only load that batch into memory.

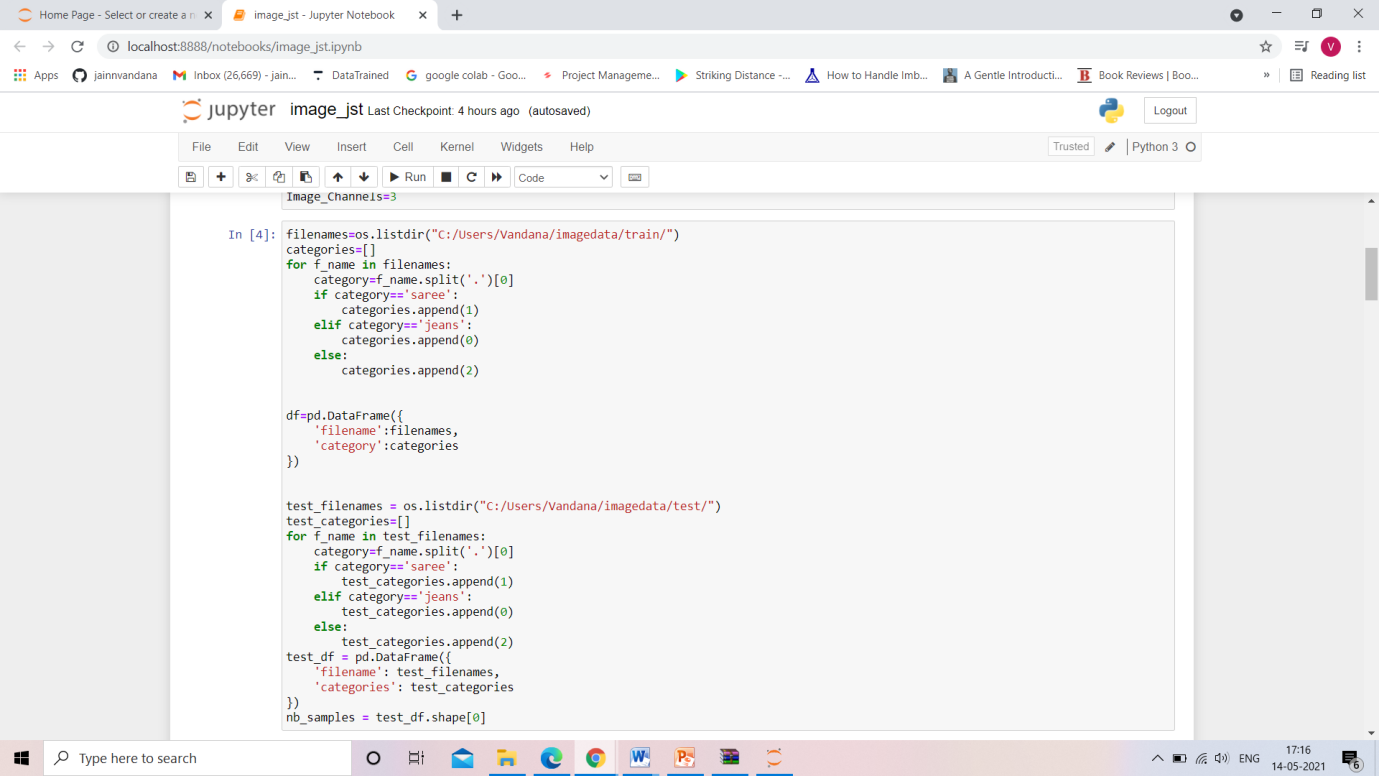


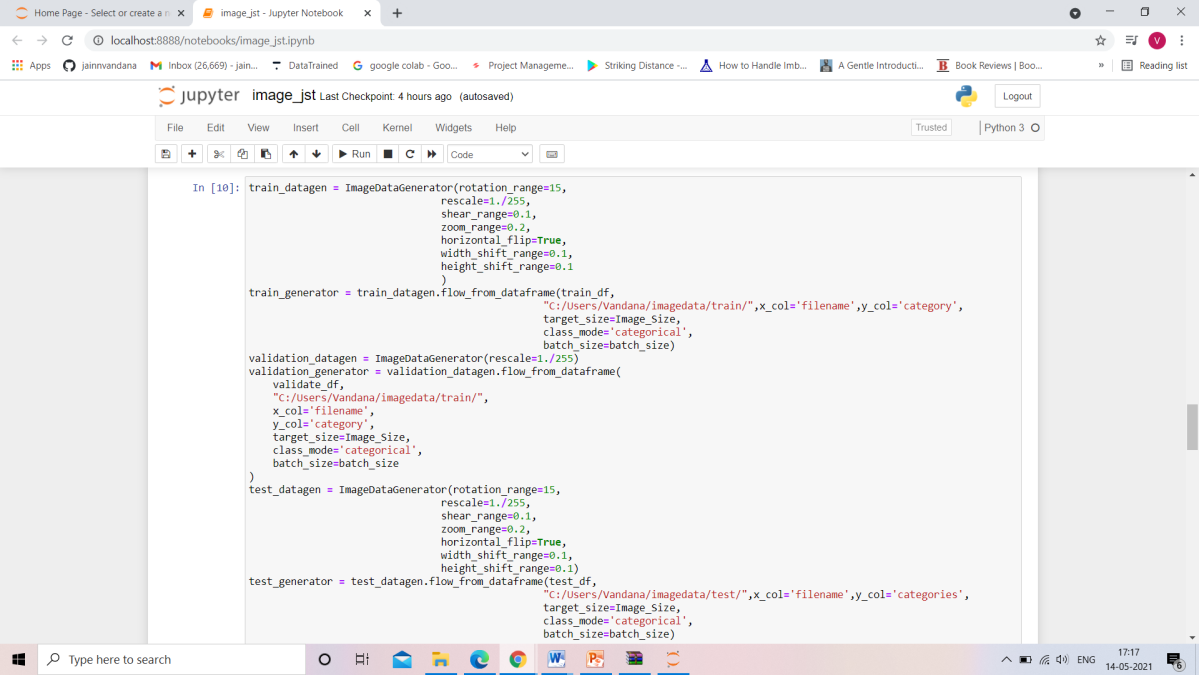
* Data Preprocessing Done

In keras, we get two predefined methods for data loading. One is flow from directory and other is flow from dataframe(pandas dataframe). So if we want to use predefined methods we need to either convert our dataset in subfolders of classes or we need to create a dataframe consisting of the filename and the classes. The second being easier we will use it.

import pandas as pd  
filenames = os.listdir('./train')  
categories = []  
for filename in filenames:  
category = filename.split('.')[0]  
if category == 'dog':  
categories.append('dog')  
else:  
categories.append('cat')  
  
  
df = pd.DataFrame({  
'filename': filenames,  
'category': categories  
})

This code will create a dataframe consisting of filenames and a corresponding label. Now we need to make data generator and we can flow through the data frame. In datagenerator, we will just normalize image between 0 and 1. We will also create a validation loader to validate the model. So we will divide the dataframe into two parts of train and validation. Train generator is the method which will generate one batch of data. The arguments of this method are the data frame we created, the path to images, name of columns, final image size, batch size. We also need to specify the categorical mode. Now we are done with the dataloader and we will create a model. We will also create a validation loader to validate the model.





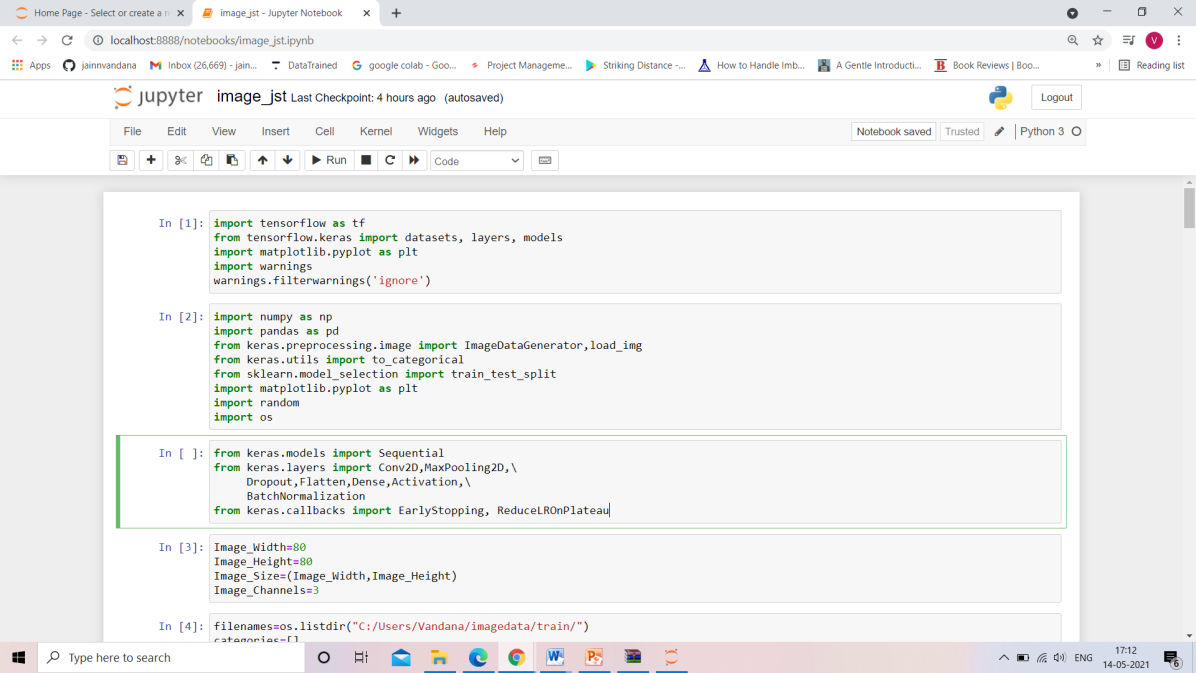
* Hardware and Software Requirements and Tools Used

Hardware : Since the computational aspect of the project is of

importance to PANDA, it is important to know the hardware that was used in the evaluation process. The training and evaluation of the neural network model has been done on a Windows 10 computer using a quad-core CPU at i3.

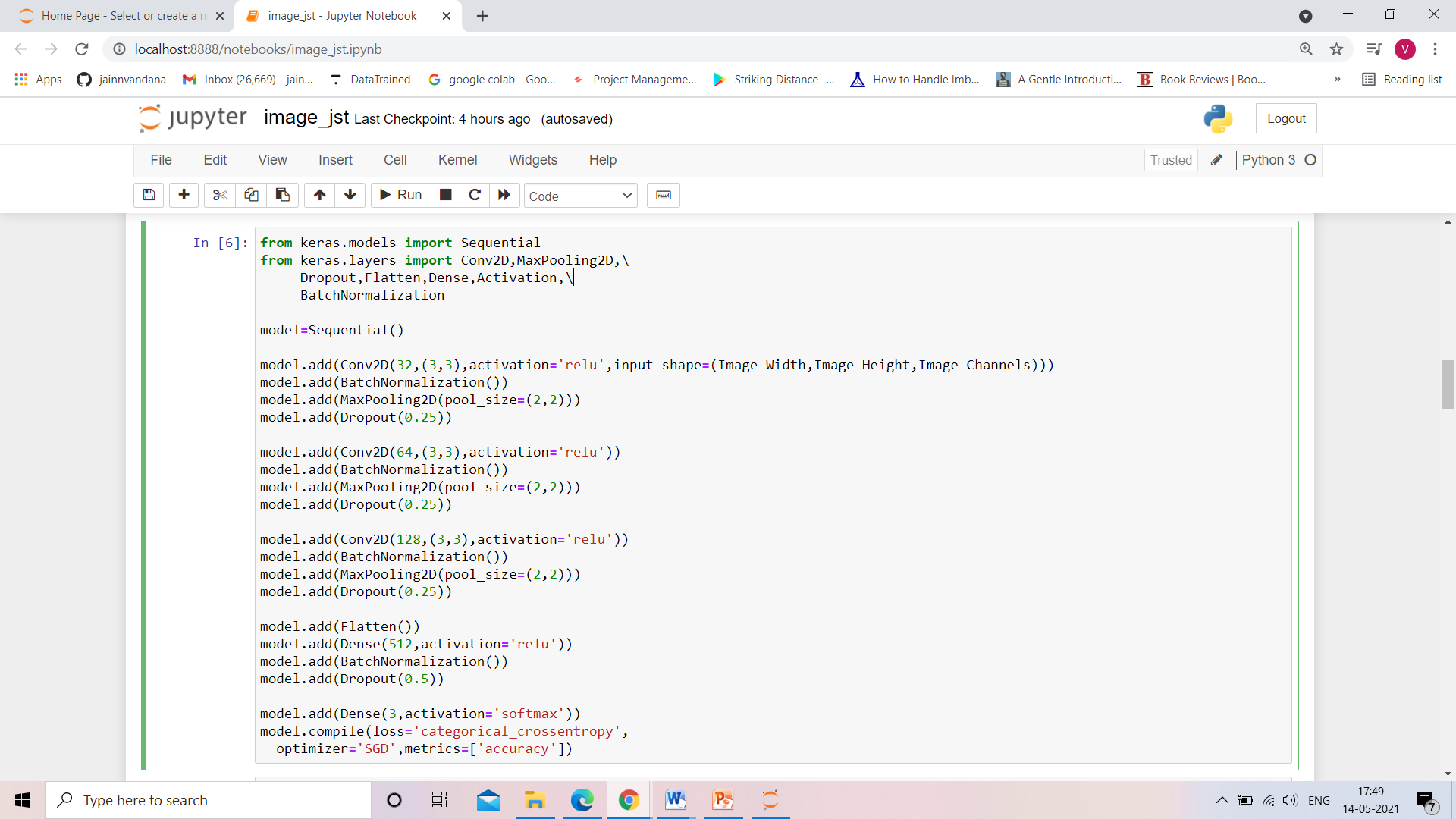
Software : Anaconda 3 , Windows 10 , Microsoft office.

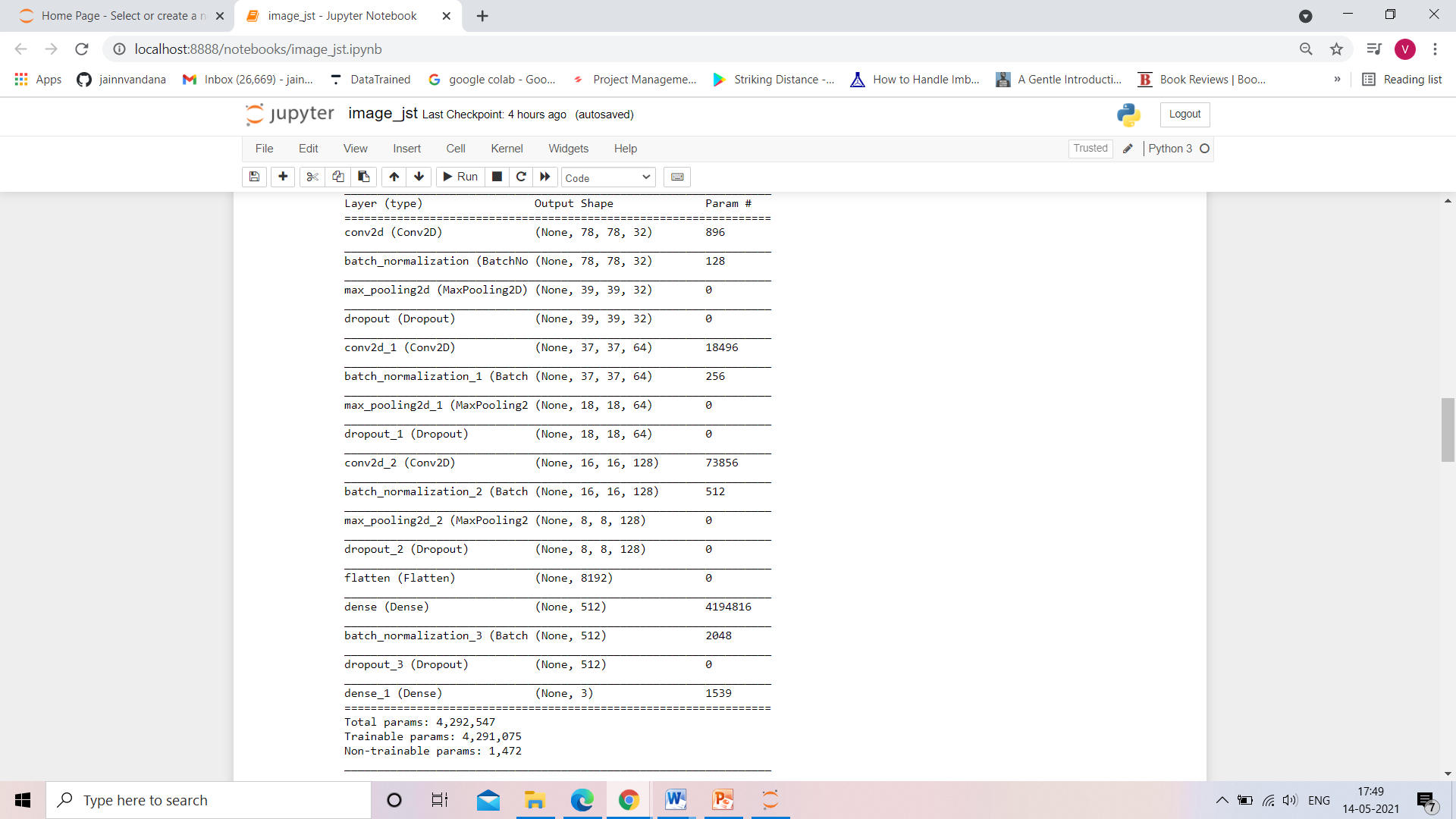
Tools used : Python , Machine learning libraries, Tensorflow a machine with Keras, SciPy, PIL installed



**Model/s Development and Evaluation**

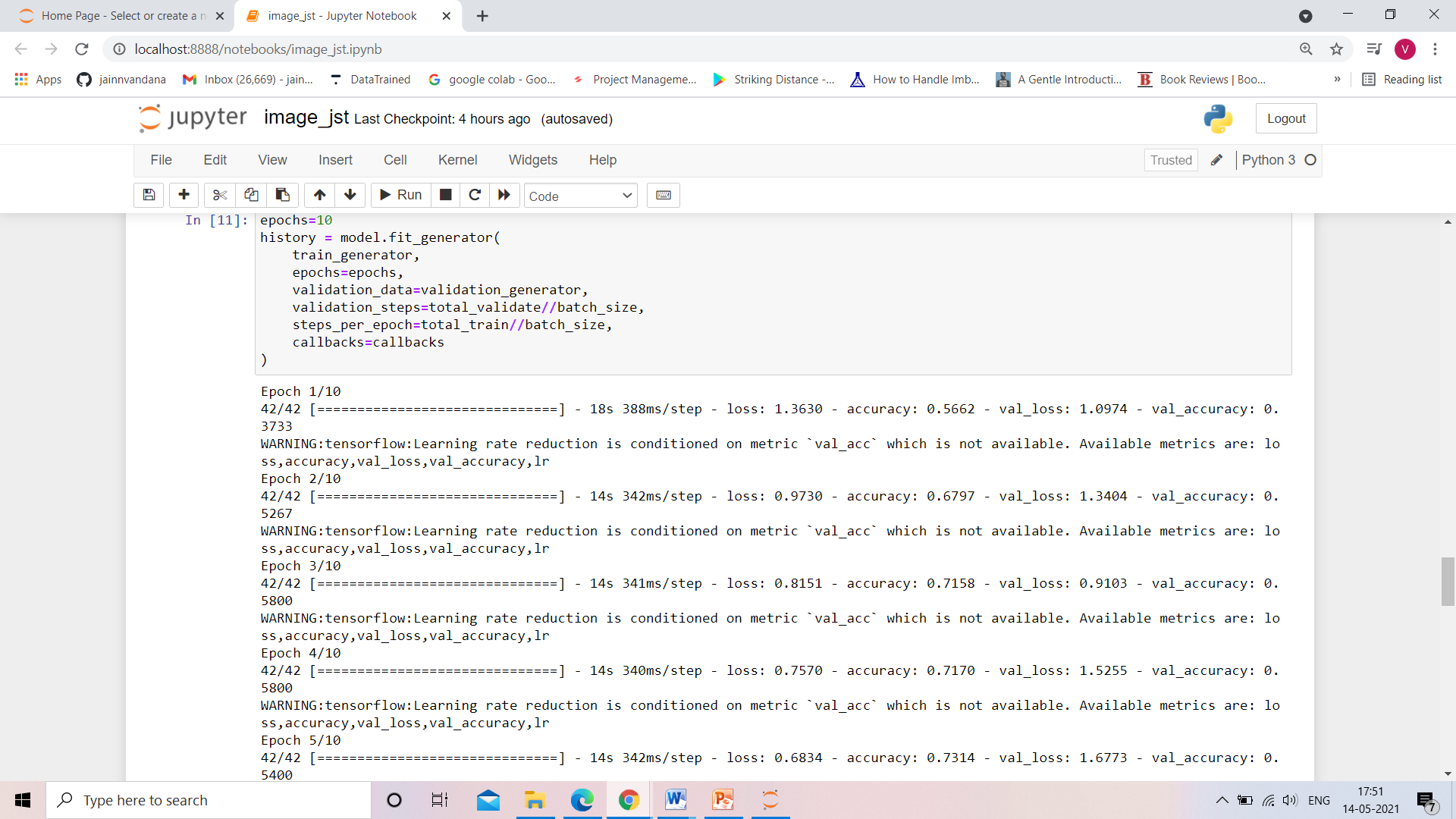
* Identification of possible problem-solving approaches (methods)
* Sequential: Sequential is the easiest way to build a model in Keras. It allows you to build a model layer by layer. Each layer has weights that correspond to the layer the follows it. We use the 'add()' function to add layers to our model. We will add two layers and an output layer.
* Convo2D: Keras Conv2D is a 2D Convolution Layer, this layer creates a convolution kernel that is wind with layers input which helps produce a tensor of outputs. In image processing kernel is a convolution matrix or masks which can be used for blurring, sharpening, embossing, edge detection, and more by doing a convolution between a kernel and an image.
* MaxPooling2D: Maximum pooling, or max pooling, is a pooling operation that calculates the maximum, or largest, value in each patch of each feature map. The results are down sampled or pooled feature maps that highlight the most present feature in the patch, not the average presence of the feature in the case of average pooling. This has been found to work better in practice than average pooling for computer vision tasks like image classification.
* BatchNormalization: The goal of Batch Normalization is to achieve a stable distribution of activation values throughout training, and in our experiments we apply it before the nonlinearity since that is where matching the first and second moments is more likely to result in a stable distribution
* Dropout: Dropout is an approach to regularization in neural networks which helps reducing interdependent learning amongst the neurons. At each training stage, individual nodes are either dropped out of the net with probability 1-p or kept with probability p, so that a reduced network is left; incoming and outgoing edges to a dropped-out node are also removed.
* Flatten: Flattening is converting the data into a 1-dimensional array for inputting it to the next layer. We flatten the output of the convolutional layers to create a single long feature vector. And it is connected to the final classification model, which is called a fully-connected layer.
* Optimizer: Optimizers are algorithms or methods used to change the attributes of your neural network such as weights and learning rate in order to reduce the losses. Optimization algorithms or strategies are responsible for reducing the losses and to provide the most accurate results possible.
* Stochastic Gradient Descent: It’s a variant of Gradient Descent. It tries to update the model’s parameters more frequently. In this, the model parameters are altered after computation of loss on each training example. So, if the dataset contains 1000 rows SGD will update the model parameters 1000 times in one cycle of dataset instead of one time as in Gradient Descent. Frequent updates of model parameters hence, converges in less time. Requires less memory as no need to store values of loss functions. May get new minima’s.

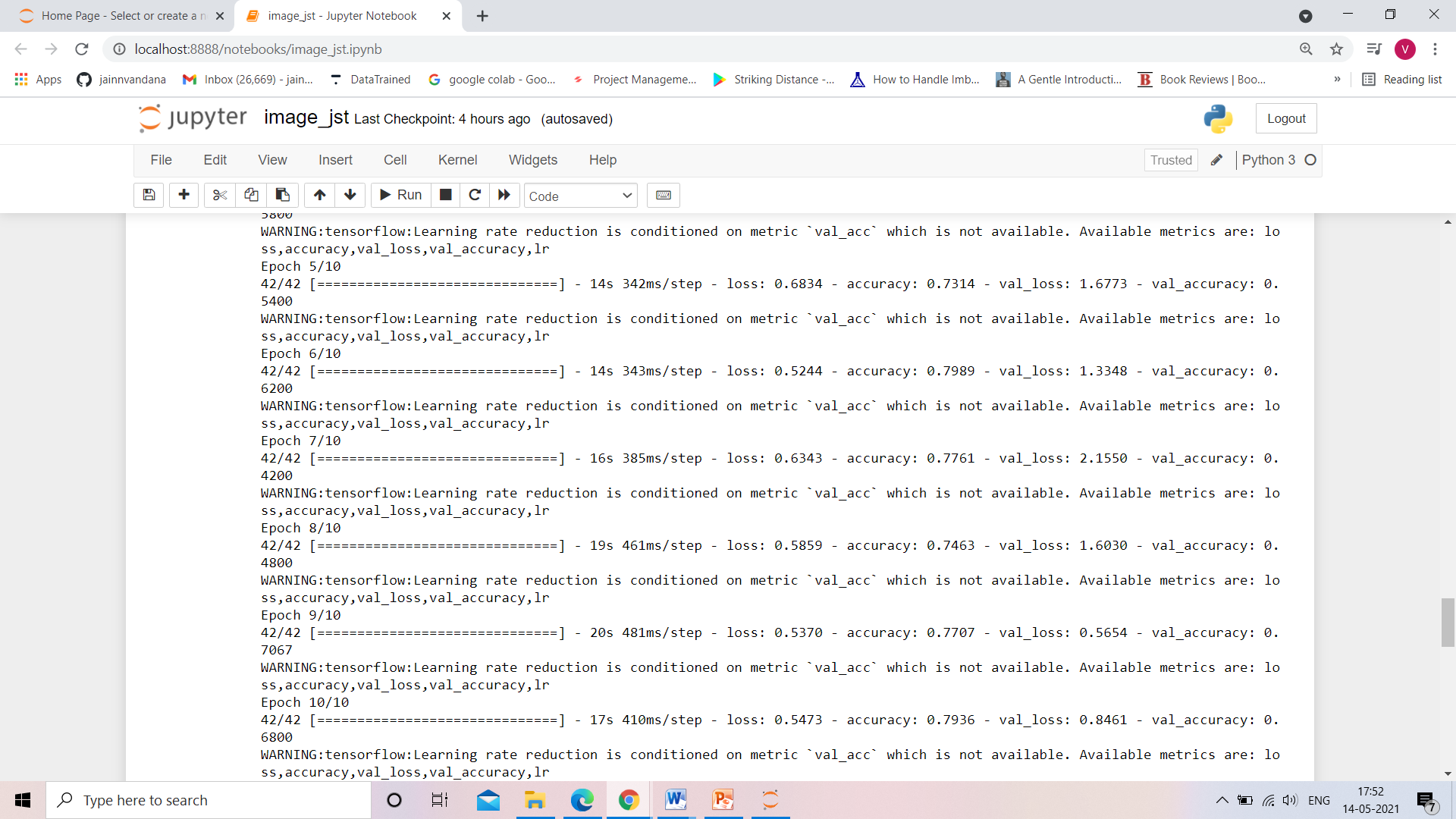


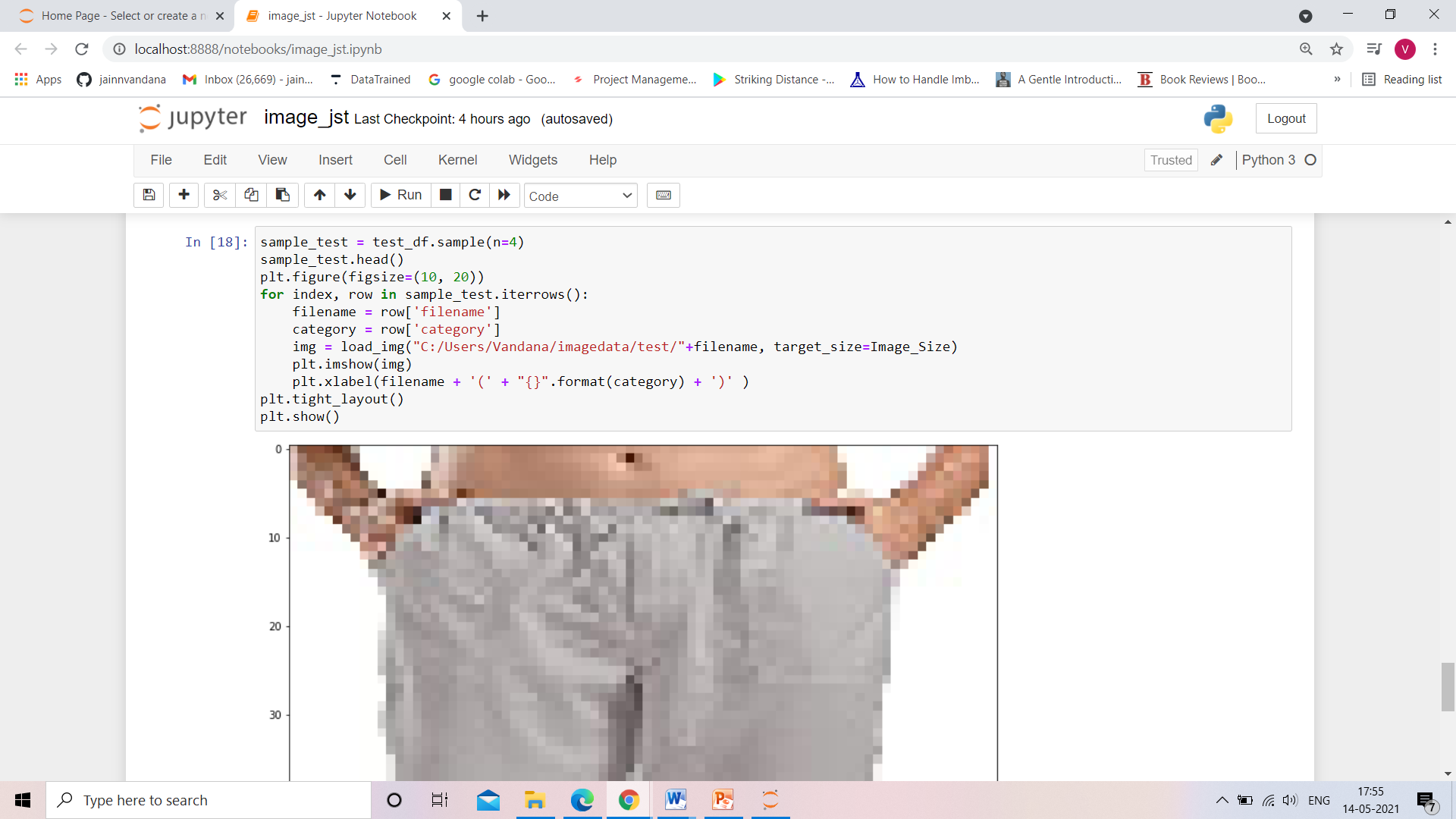


* Run and Evaluate selected models

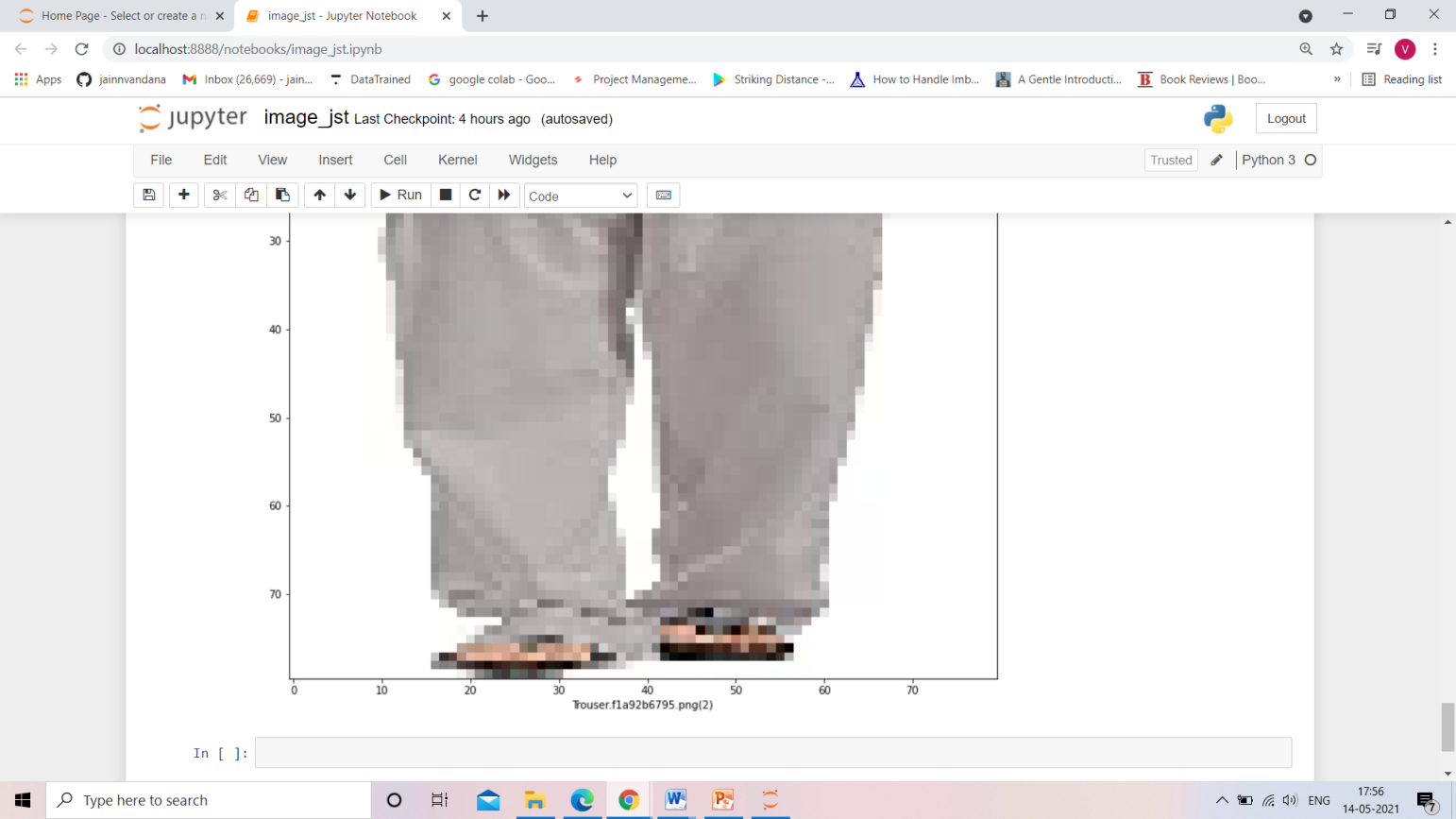
The right tool for an image classification job is a convnet, so let's try to train one on our data, as an initial baseline. Since we only have few examples, our number one concern should be overfitting. Overfitting happens when a model exposed to too few examples learns patterns that do not generalize to new data, i.e. when the model starts using irrelevant features for making predictions. For instance, if you, as a human, only see three images of people who are lumberjacks, and three, images of people who are sailors, and among them only one lumberjack wears a cap, you might start thinking that wearing a cap is a sign of being a lumberjack as opposed to a sailor. You would then make a pretty lousy lumberjack/sailor classifier.Data augmentation is one way to fight overfitting, but it isn't enough since our augmented samples are still highly correlated. Your main focus for fighting overfitting should be the entropic capacity of your model --how much information your model is allowed to store. A model that can store a lot of information has the potential to be more accurate by leveraging more features, but it is also more at risk to start storing irrelevant features. Meanwhile, a model that can only store a few features will have to focus on the most significant features found in the data, and these are more likely to be truly relevant and to generalize better.There are different ways to modulate entropic capacity. The main one is the choice of the number of parameters in your model, i.e. the number of layers and the size of each layer. Another way is the use of weight regularization, such as L1 or L2 regularization, which consists in forcing model weights to taker smaller values.In our case we will use a very small convnet with few layers and few filters per layer, alongside data augmentation and dropout. Dropout also helps reduce overfitting, by preventing a layer from seeing twice the exact same pattern, thus acting in a way analoguous to data augmentation (you could say that both dropout and data augmentation tend to disrupt random correlations occuring in your data).The code snippet below is our first model, a simple stack of 3 convolution layers with a ReLU activation and followed by max-pooling layers. This is very similar to the architectures that Yann LeCun advocated in the 1990s for image classification (with the exception of ReLU).







* Visualizations



**CONCLUSION**

The objective of the image classification project was to enable the model to understand about the image and its features and make model able to predict the given image’s label. To start working with Keras to solve real-time deep learning problems.

In this keras deep learning Project, we talked about the image classification paradigm for digital image analysis, such as human.

In the above model building we use different optimizer to train our model from which the best one is SGD but by improving the other aspects of our model we can see may be some other is performing better.