# IMAGE SCRAPING AND CLASSIFICATION PROJECT

FLIP ROBO

Submitted by:

Charitha Lanka

# TABLE OF CONTENTS

# ACKNOWLEDGEMENT

I would like to express my gratitude to Data Trained for providing me with several learning opportunities. I would like to thank Flip Robo for giving me this platform to intern and learn in the vast field of Data Science. I have been learning through various online modes during this project.

# INTRODUCTION

## What is Image Classification?

When we human look at a picture we can easily identify what it depicts without any problem. Humans learn this skill of identification at early stages of life. This image identification problem though is not an easy task for a machine or a computer. They don't see the world the way humans do. Image classification thus becomes a challenge for computers or machines to learn, which is where the role of deep learning comes in.

Image classification is the ability of a computer to analyze an image and identify the class the image falls under. The class can be anything from automobiles to animals etc.

For instance, we input an image of a bus into the computer then the ability of computer to tell that the loaded image is a bus or the probability that the image is of a bus is what image classification is.

Image classification is a supervised learning problem.

## Background of the domain problem

Earlier image classification used raw pixel data which means that computer would break down the image into individual pixels. Problem with this approach was that two similar product images could look way distinct because of the different background, colors anglers, poses and many other factors. This made it a task for computers to identify the images.

## Review of literature

As earlier method of classification of image was not reliable this is where deep learning entered. Deep learning is a subset of machine learning that allows machine to learn from data. Deep learning is known to use neural networks for this work.

In Neural Networks, the input gets filtered through hidden layers of nodes. Each node processes the input and passes their individual result to the next layer of node. This process repeats until it reaches an output layer and machine is able to answer.

There are different types of neural networks based on how hidden layers work. Image classification with deep learning is mostly done using CNN which is convolutional neural network. In CNN the nodes in the hidden layers don't always share their output with every node in the next layer known as convolutional layer.

## Motivation of the problem undertaken

Deep learning allows machine to identify and extract features from images. This means they can learn the features to look for in images by analyzing a lot of pictures. User does not need to enter the filters manually. Deep learning has made this problem of image classification quite easy.

# ANALYTICAL PROBLEM FACING

## Data sources and their formats

Data used in the project is images of saree, jeans(men) and trouser(men) which is scraped from e-commerce amazon. Scraping of data is done using selenium. After scraping the images in three different folders. The images were separated into training and testing folders each consisting three sub folders belonging to one of the three- Saree, Jeans and Trouser. These subfolders would serve as three categories in the project.

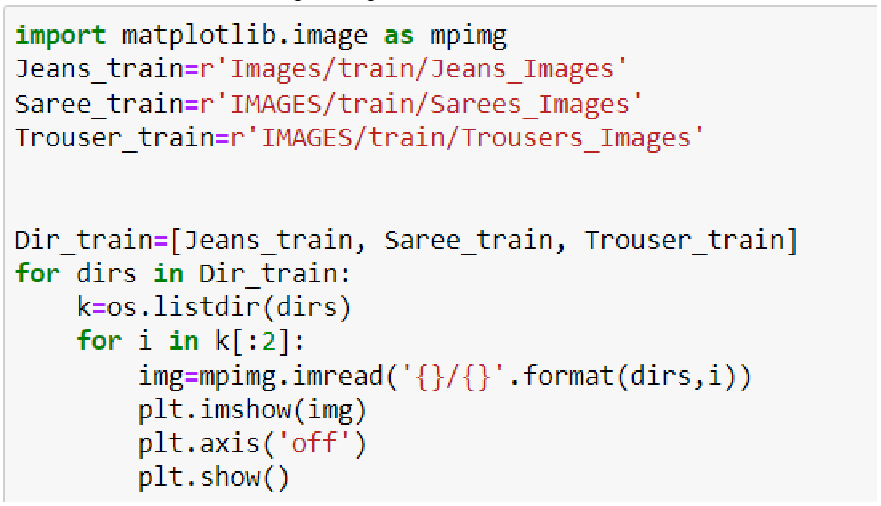
We will build a model that would take images as input and will classify them in one of the three categories- Saree, Jeans or Trouser.

## Visualization

To check whether the three folders of saree, jeans and trouser have the right images we visualize some images randomly using 'random'.

**Sarees:**

The code to load the images is given below:



```python
import matplotlib.image as mpimg
Jeans_train=r'Images/train/Jeans_Images'
Saree_train=r'IMAGES/train/Sarees_Images'
Trouser_train=r'IMAGES/train/Trousers_Images'


Dir_train=[Jeans_train, Saree_train, Trouser_train]
for dirs in Dir_train:
    k=os.listdir(dirs)
    for i in k[:2]:
        img=mpimg.imread('{}/{}'.format(dirs,i))
        plt.imshow(img)
        plt.axis('off')
        plt.show()
```

The output of the code printed five random images from the folder of saree.

Similarly, we randomly loaded images of trouser and jeans.

The loaded images are given below:

**Jeans Men:**

**Trouser Men:**

AESTHETIC

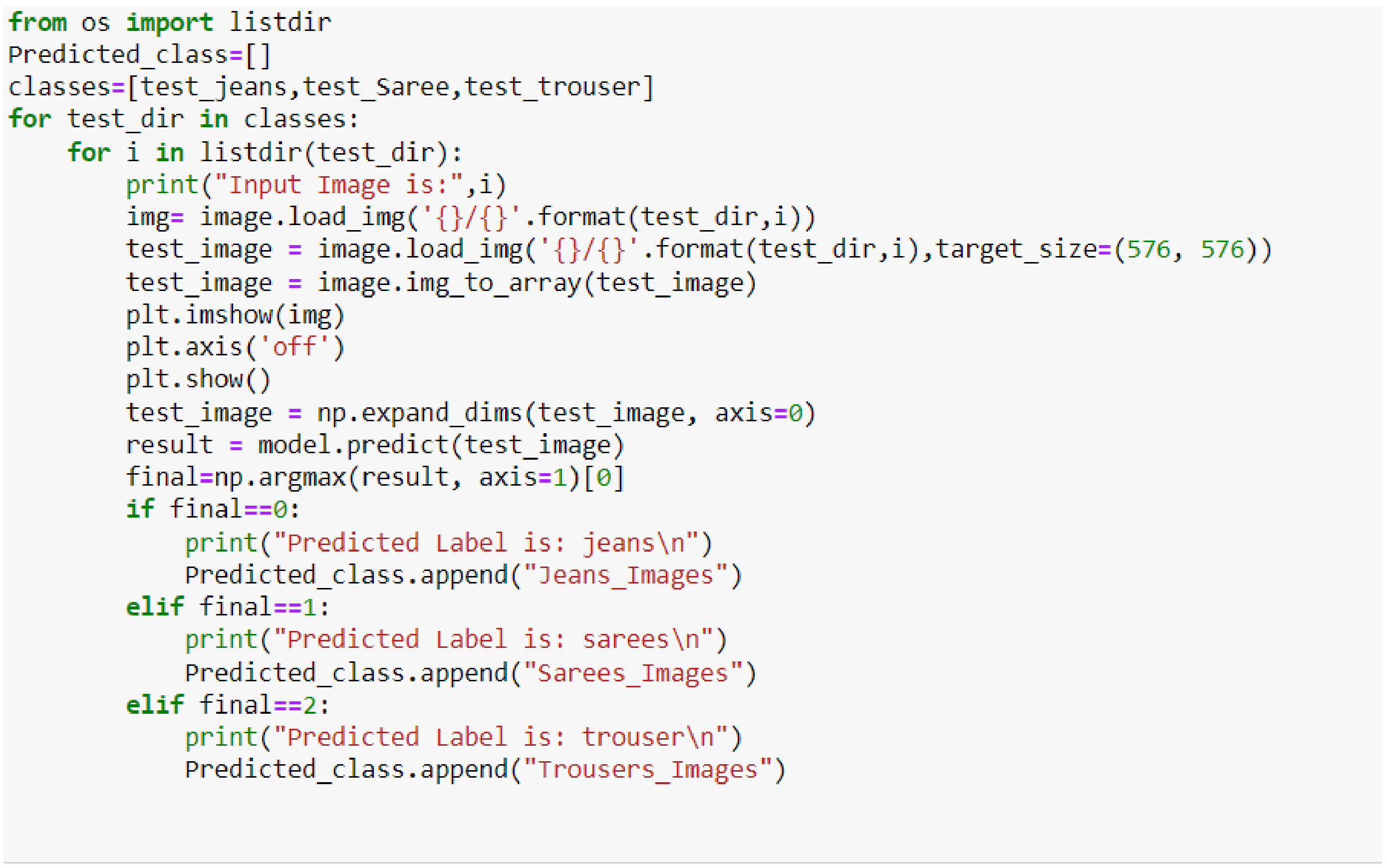
## Loading the Images in the python notebook

Train and test dataset was separately loaded using the open cv python library. There are many ways in which we can load the images, one can use PIL, open cv, IPython, Keras to load images in python. Open cv python library is the mostly used among all to load and resize the images. In the project we have also used open cv to load the train and test data.

We have defined a function to load the datasets. Let us have a look at the function:

A function is defined which will return us the class name of each image and each image in the form of an array.

1. We have used os. listdir method to list all the files in the train folder. So as output it will give us three files of Saree, Jeans and Trousers folders.



```python
from os import listdir
Predicted_class=[]
classes=[test_jeans,test_Saree,test_trouser]
for test_dir in classes:
    for i in listdir(test_dir):
        print("Input Image is:",i)
        img= image.load_img('{}/{}'.format(test_dir,i))
        test_image = image.load_img('{}/{}'.format(test_dir,i),target_size=(576, 576))
        test_image = image.img_to_array(test_image)
        plt.imshow(img)
        plt.axis('off')
        plt.show()
        test_image = np.expand_dims(test_image, axis=0)
        result = model.predict(test_image)
        final=np.argmax(result, axis=1)[0]
        if final==0:
            print("Predicted Label is: jeans\n")
            Predicted_class.append("Jeans_Images")
        elif final==1:
            print("Predicted Label is: sarees\n")
            Predicted_class.append("Sarees_Images")
        elif final==2:
            print("Predicted Label is: trouser\n")
            Predicted_class.append("Trousers_Images")
```

2. We have used os.path.join method to join the paths of the folders listed by os.listdir with the train folder to get into the folders of Saree, Jeans and trouser in order to download the images.

3. Further, we have downloaded the images using open cv python method.

4. Image augmentation is performed on downloaded image in which the images are re-sized, converted into an array and normalized in the range of 0-1.

5. Image array and class to which the image belongs are returned.

The class_name returned by the function is a list containing the word jeans, saree and trouser. We need to convert this into integer type classification 0,1, and 2. That way we will be able to use the classes as label output.

6. Following steps were performed to convert the keywords saree, jeans and trouser to integer.

```
predt={k: v for v, k in enumerate(np.unique(class_name))}
predt
```

```
{'Jeans': 0, 'Sarees': 1, 'Trousers': 2}
```

```
# Creating the label column
pred_value=[predt[class_name[i]] for i in range(len(class_name))]
```

Here, we have built a dictionary in which we have the name of classes along with an integer representation. Further this dictionary is used to put label corresponding each image as 0,1 or 2.

7. Creating training data

```
# creating x_train and y_train
x_train=np.array(img_data, np.float32)
y_train=np.array(list(map(int,target_val)), np.float32)
```

We have used the above downloaded image array and pred_value list and stored the data under the names x_train and y_train respectively.

Similarly, we have loaded the test data and split the data into x_test and y_test.

## Hardware and Software Requirements and Tools Used

Hardware specifications-

Processor- Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11

GHzRAM-8.00 GB

Edition-Windows10

This project was completed using Python 3.0 which is an interpreted high-level and general-purpose programming language. Jupyter notebook which is an open-source web application provided GUI environment for the python notebook.

Important libraries that were used in the completion of this project are:

1. Numpy- Also known as Numerical python is a python library used to perform algebraic and statical analysis.

2. Matplotlib.pyplot- Matplotlib is a plotting library in python used for creating visualizations in python. Pyplot is the sub-module of matplot library which is the collection of functions that makes matplotlib works like pyplot. Pyplot in simple language helps in making the figure decorative and informative. We can put labels with the help of pyplot, we can insert some lines and points in the figure using pyplot etc.

3. Tensor flow- It is a free and open-source library for machine learning. It is used in deep learning. It allows programmers to create large scale neural networks with dense layers.

4. Keras - Keras is a free open-source python library that works on top of tensor flow. It allows user to develop and evaluate deep learning models.

5. OS – The main purpose of the OS module is to interact with your operating system. We can use OS to create folders, remove folders, move folders, and sometimes change the working directory. We have used it to access the names of files within a file path by doing listdir() and also we have used os.path.join to join different path easily.

6. Random – Random is a python module which is used to generate pseudo-random numbers. It is used to perform some action randomly. We have used random to randomly pick image paths to load random images of our data.

7. OpenCV-Python – We have used this module to load images and resizing the image.

## Model Training and Testing

We have a built a convolutional neural network.

```
model=Sequential()

# First convolution layer
model.add(Conv2D(32,(3,3),input_shape=input_shape))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

# Second convolution layer
model.add(Conv2D(32,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

# Third convolution layer
model.add(Conv2D(64,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

# Fourth convolution layer
model.add(Conv2D(64,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))


model.add(Flatten())
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(3))
model.add(Activation('softmax'))

print(model.summary())
```

The model type that we will be using is Sequential. Sequential is the easiest way to build a model in Keras. It allows you to build a model layer by layer.

We have 2 Conv2D layers. Convolution layer will deal with our input images, which are seen as

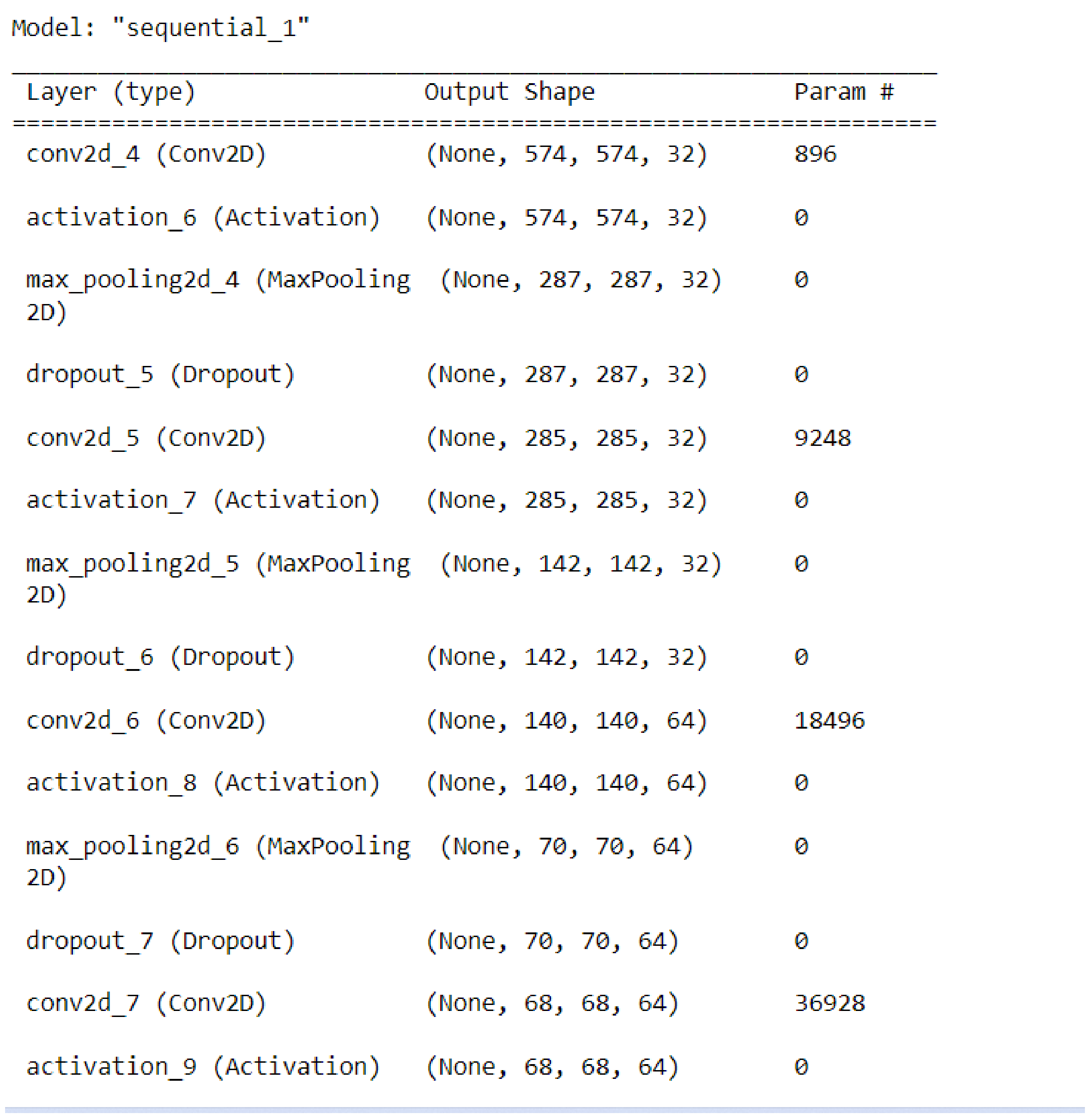
2- dimensional matrices.

32 in the first layer and 64 in the second layer are the number of nodes in each layer. This number can be adjusted to be higher or lower, depending on the size of the dataset. In our case, 32 and 64 works well, so we will stick with this for now.

Kernel size is the size of the filter matrix for our convolution. So, a kernel size of 3 means we will have a3x3 filter matrix.

Activation is the activation function for the layer. The activation function we will be using for convolutional layers is ReLU, or Rectified Linear Activation. This activation function has been proven to work well in neural networks.

In between the Conv2D layers and the dense layer, there is a ‘Flatten’ layer. Flatten serves as a connection between the convolution and dense layers.



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 574, 574, 32)      896

 activation_6 (Activation)   (None, 574, 574, 32)      0

 max_pooling2d_4 (MaxPooling  (None, 287, 287, 32)     0
 2D)

 dropout_5 (Dropout)         (None, 287, 287, 32)      0

 conv2d_5 (Conv2D)           (None, 285, 285, 32)      9248

 activation_7 (Activation)   (None, 285, 285, 32)      0

 max_pooling2d_5 (MaxPooling  (None, 142, 142, 32)     0
 2D)

 dropout_6 (Dropout)         (None, 142, 142, 32)      0

 conv2d_6 (Conv2D)           (None, 140, 140, 64)      18496

 activation_8 (Activation)   (None, 140, 140, 64)      0

 max_pooling2d_6 (MaxPooling  (None, 70, 70, 64)       0
 2D)

 dropout_7 (Dropout)         (None, 70, 70, 64)        0

 conv2d_7 (Conv2D)           (None, 68, 68, 64)        36928

 activation_9 (Activation)   (None, 68, 68, 64)        0
```

‘Dense’ is the layer type we will use in for our output layer. Dense is a standard layer type that is used in many cases for neural networks.

We will have 3 nodes in our output layer, one for each possible outcome (0-2).

The activation is ‘softmax’. Softmax makes the output sum up to 1 so the output can be interpreted as probabilities. The model will then make its prediction based on which option has the highest probability.

## Compiling the model

Next, we need to compile our model. Compiling the model takes three parameters: optimizer, loss and metrics.

The optimizer controls the learning rate. We have used ‘adam’ as our optimizer. Adam is generally a good optimizer to use for many cases. The Adam optimizer adjusts the learning rate throughout training.

The learning rate determines how fast the optimal weights for the model are calculated. A smaller learning rate may lead to more accurate weights (up to a certain point), but the time it takes to compute the weights will be longer.

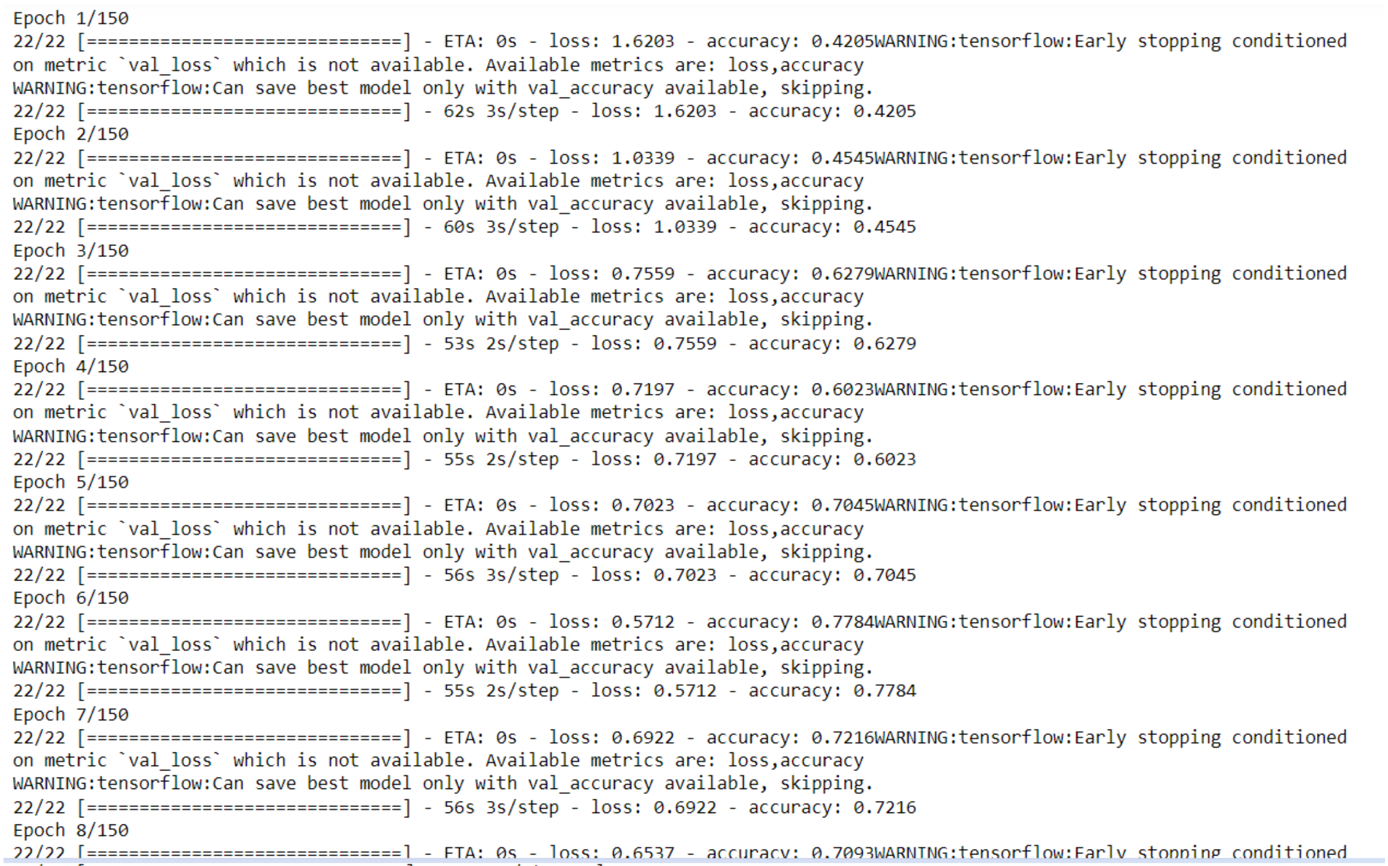
We will use ‘sparse_categorical_crossentropy’ for our loss function. This is the most common choice for integer type classification. One advantage of using sparse categorical cross entropy is it saves time in memory as well as computation because it simply uses a single integer for a class, rather than a whole vector. A lower score indicates that the model is performing better.

To make things even easier to interpret, we will use the ‘accuracy’ metric to see the accuracy score when we train the model.

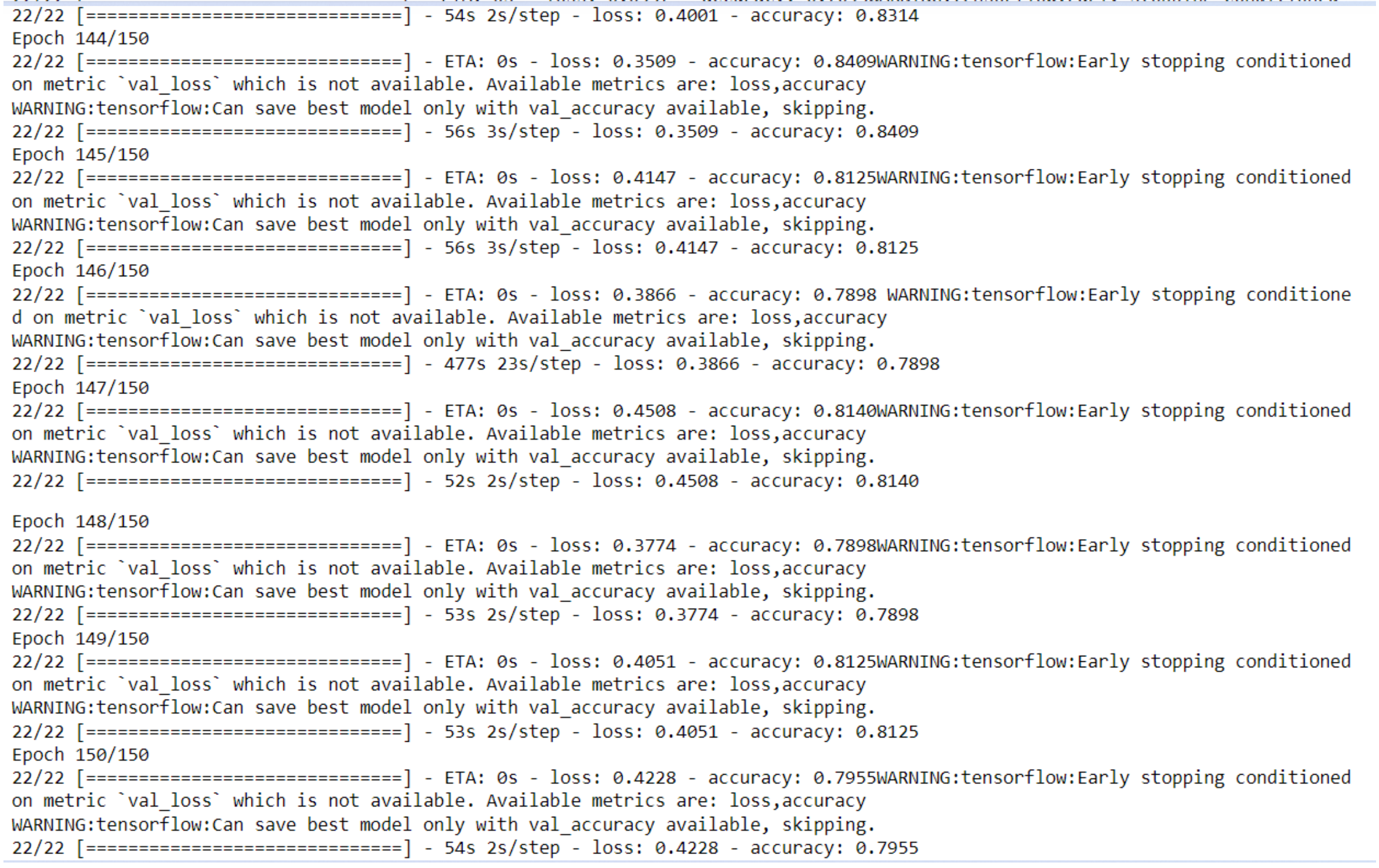
## Training the model

Now we will train our model. To train, we will use the ‘fit()’ function on our model with the following parameters: training data (x_train), target data (y_train) and the number of epochs.

The number of epochs is the number of times the model will cycle through the data. The more epochs we run, the more the model will improve, up to a certain point. After that point, the model will stop improving during each epoch. For our model, we will set the number of epochs to 25.



```
Epoch 1/150
22/22 [==============================] - ETA: 0s - loss: 1.6203 - accuracy: 0.4205WARNING:tensorflow:Early stopping conditioned
on metric `val_loss` which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
22/22 [==============================] - 62s 3s/step - loss: 1.6203 - accuracy: 0.4205
Epoch 2/150
22/22 [==============================] - ETA: 0s - loss: 1.0339 - accuracy: 0.4545WARNING:tensorflow:Early stopping conditioned
on metric `val_loss` which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
22/22 [==============================] - 60s 3s/step - loss: 1.0339 - accuracy: 0.4545
Epoch 3/150
22/22 [==============================] - ETA: 0s - loss: 0.7559 - accuracy: 0.6279WARNING:tensorflow:Early stopping conditioned
on metric `val_loss` which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
22/22 [==============================] - 53s 2s/step - loss: 0.7559 - accuracy: 0.6279
Epoch 4/150
22/22 [==============================] - ETA: 0s - loss: 0.7197 - accuracy: 0.6023WARNING:tensorflow:Early stopping conditioned
on metric `val_loss` which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
22/22 [==============================] - 55s 2s/step - loss: 0.7197 - accuracy: 0.6023
Epoch 5/150
22/22 [==============================] - ETA: 0s - loss: 0.7023 - accuracy: 0.7045WARNING:tensorflow:Early stopping conditioned
on metric `val_loss` which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
22/22 [==============================] - 56s 3s/step - loss: 0.7023 - accuracy: 0.7045
Epoch 6/150
22/22 [==============================] - ETA: 0s - loss: 0.5712 - accuracy: 0.7784WARNING:tensorflow:Early stopping conditioned
on metric `val_loss` which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
22/22 [==============================] - 55s 2s/step - loss: 0.5712 - accuracy: 0.7784
Epoch 7/150
22/22 [==============================] - ETA: 0s - loss: 0.6922 - accuracy: 0.7216WARNING:tensorflow:Early stopping conditioned
on metric `val_loss` which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
22/22 [==============================] - 56s 3s/step - loss: 0.6922 - accuracy: 0.7216
Epoch 8/150
22/22 [==============================] - ETA: 0s - loss: 0.6537 - accuracy: 0.7093WARNING:tensorflow:Early stopping conditioned
```



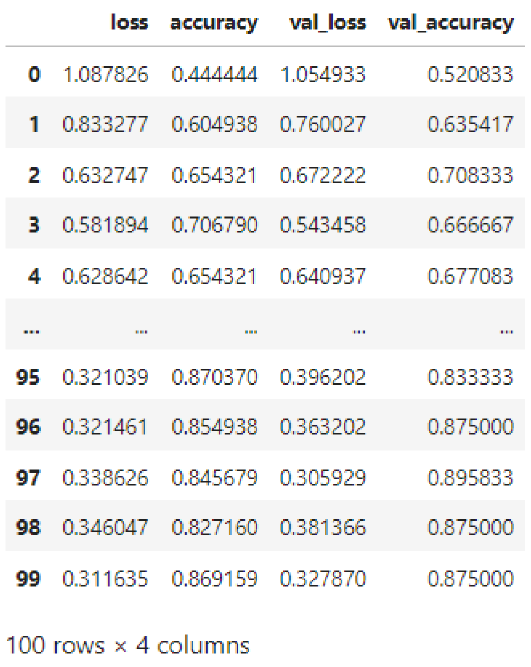
```
22/22 [==============================] - 54s 2s/step - loss: 0.4001 - accuracy: 0.8314
Epoch 144/150
22/22 [==============================] - ETA: 0s - loss: 0.3509 - accuracy: 0.8409WARNING:tensorflow:Early stopping conditioned
on metric `val_loss` which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
22/22 [==============================] - 56s 3s/step - loss: 0.3509 - accuracy: 0.8409
Epoch 145/150
22/22 [==============================] - ETA: 0s - loss: 0.4147 - accuracy: 0.8125WARNING:tensorflow:Early stopping conditioned
on metric `val_loss` which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
22/22 [==============================] - 56s 3s/step - loss: 0.4147 - accuracy: 0.8125
Epoch 146/150
22/22 [==============================] - ETA: 0s - loss: 0.3866 - accuracy: 0.7898 WARNING:tensorflow:Early stopping conditione
d on metric `val_loss` which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
22/22 [==============================] - 477s 23s/step - loss: 0.3866 - accuracy: 0.7898
Epoch 147/150
22/22 [==============================] - ETA: 0s - loss: 0.4508 - accuracy: 0.8140WARNING:tensorflow:Early stopping conditioned
on metric `val_loss` which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
22/22 [==============================] - 52s 2s/step - loss: 0.4508 - accuracy: 0.8140

Epoch 148/150
22/22 [==============================] - ETA: 0s - loss: 0.3774 - accuracy: 0.7898WARNING:tensorflow:Early stopping conditioned
on metric `val_loss` which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
22/22 [==============================] - 53s 2s/step - loss: 0.3774 - accuracy: 0.7898
Epoch 149/150
22/22 [==============================] - ETA: 0s - loss: 0.4051 - accuracy: 0.8125WARNING:tensorflow:Early stopping conditioned
on metric `val_loss` which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
22/22 [==============================] - 53s 2s/step - loss: 0.4051 - accuracy: 0.8125
Epoch 150/150
22/22 [==============================] - ETA: 0s - loss: 0.4228 - accuracy: 0.7955WARNING:tensorflow:Early stopping conditioned
on metric `val_loss` which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
22/22 [==============================] - 54s 2s/step - loss: 0.4228 - accuracy: 0.7955
```

## Using our model to make predictions

To see the actual predictions that our model has made for the test data, we are using the predict function. The predict function will give an array with 3 numbers. These numbers are the probabilities that the input image represents each class (0–2). The array index with the highest number represents the model prediction. Performance metrics used

We have a classification type data so performance metrics like accuracy score, confusion matrix, classification report, f1 score were used to study the performance of the models.

## Interpretation of the result



| | loss | accuracy | val_loss | val_accuracy |
|---|---|---|---|---|
| **0** | 1.087826 | 0.444444 | 1.054933 | 0.520833 |
| **1** | 0.833277 | 0.604938 | 0.760027 | 0.635417 |
| **2** | 0.632747 | 0.654321 | 0.672222 | 0.708333 |
| **3** | 0.581894 | 0.706790 | 0.543458 | 0.666667 |
| **4** | 0.628642 | 0.654321 | 0.640937 | 0.677083 |
| **...** | ... | ... | ... | ... |
| **95** | 0.321039 | 0.870370 | 0.396202 | 0.833333 |
| **96** | 0.321461 | 0.854938 | 0.363202 | 0.875000 |
| **97** | 0.338626 | 0.845679 | 0.305929 | 0.895833 |
| **98** | 0.346047 | 0.827160 | 0.381366 | 0.875000 |
| **99** | 0.311635 | 0.869159 | 0.327870 | 0.875000 |

100 rows × 4 columns

Our model was able to classify the images into the categories fairly well as we got the accuracy score and f1 score of 0.84 each.

# CONCLUSION

We were successful in building an end-to-end deep learning model to classify the three categories of images.