MULTILABEL IMAGE CLASSIFIER

A

Report submitted

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

Bachelor of Technology

in

Computer Science and Engineering

BY:

KASHIKA ADARSH (1805210024)
VARUN SAINI (1805210062)
GAURISH GUPTA (1805210021)
NIKHIL JAMWAL (1805210030)

Under the guidance of

DR. MANIK CHANDRA

MS. MUDITA SHARAN

Department of Computer Science and Engineering
Institute of Engineering and Technology, Lucknow
Dr. A.P.J. Abdul Kalam Technical University, Lucknow, Uttar Pradesh

TABLE OF CONTENTS

DECLARATION

We hereby declare that this submission is our own work and that it contains no material previously published or written by another person, or material that has been accepted for the award of any degree or diploma by a university or other institute of higher learning due to a substantial error, except where the acknowledgement has been made in the text. We have not submitted the project to any other institute for any other degree requirement.

Date: 27th May 2022

Submitted by: -

(1) Name: Kashika Adarsh
Roll No.: 1805210024
Branch: CSE
Signature:

(2) Name: Varun Saini
Roll No.: 1805210062
Branch: CSE
Signature:

(3) Name: Gaurish Gupta
Roll No.: 1805210021
Branch: CSE
Signature:

(4) Name: Nikhil Jamwal
Roll No.: 1805210030
Branch: CSE
Signature:

CERTIFICATE

This is to certify that the project report entitled “Multilabel Image Classifier” presented by Kashika Adarsh, Varun Saini, Gaurish Gupta and Nikhil Jamwal in the partial fulfillment for the granting of Bachelor of Technology in Computer Science and Engineering, is a record of work completed by them at the Department of Computer Science and Engineering at the Institute of Engineering and Technology, Lucknow, under my supervision and assistance.

It is also certified that, to the best of my knowledge, this project has not been submitted to any other Institute for the award of any other degrees.

(Dr. Manik Chandra)
Department of Computer Science and Engineering
Institute of Engineering and Technology, Lucknow

(Ms. Mudita Sharan)
Department of Computer Science and Engineering
Institute of Engineering and Technology, Lucknow

ACKNOWLEDGEMENT

Our team is highly indebted to **Dr. Manik Chandra** and **Ms Mudita Sharan** of Computer Science & Engineering Department, IET Lucknow, for their continued support and suggestions so far They've been extremely helpful, supporting, and nice, making it much easier to write up this project's work. Their unwavering support inspired us to work on this article and assisted us in gaining knowledge in a variety of sectors. We'd also like to express our gratitude for their feedback and critiques on the early project effort. They helped us identify areas for improvement, which helped us produce this interim report. Our mentor answered all of our inquiries with a smile, thoughtfulness, and a great deal of patience. By continually listening to our perspectives, appreciating and developing them, and giving us a free hand in our project, they never make us feel like we're on our backsides. The current job has progressed to this point due to their overwhelming interest and accommodating demeanor. Finally, I want to express my gratitude to our Institution and colleagues, whose persistent encouragement helped us to revive our spirits, refocus our attention and energy, and finish this project.

We express our thanks to **Dr. Pawan Kumar Tiwari** for their unwavering cooperation in completing this project on time.

Kashika Adarsh
Varun Saini
Gaurish Gupta
Nikhil Jamwal

ABSTRACT

If we show you a visual of a ball, you'll immediately recognize it as such. We'll show you a terrace in the next shot. The image is divided into two categories: ball and no-ball. A binary image classification problem is a classification problem in which the images can be classified into two classes. The photographs can be categorized into more than two different groups. A single image can't be assigned to more than one category. Multi-label classification is a classification method in which an object can be classified into multiple categories. As an example, We will categorize a movie poster's genre in the dataset based just on seeing the movie poster.

Image classification is one of the most popular machine learning applications. It has a wide range of applications, including facial recognition algorithms, classification of complicated patterns in photos for crime detection, and a variety of other social, medical, and technical uses.

Image classification entails more than simply classifying photos into categories; it also entails providing machines the ability to visualize the environment.

Another example of image categorization in action is the prediction of movie genres using posters. This involves sorting movie posters into predetermined categories like drama, romance, action, tragedy and so on.

LIST OF FIGURES

INTRODUCTION

We are already moved to the process of automating everything whether it is prediction or suggestion and any other process. We are living in the era governed by data. With Artificial Intelligence (AI) and Machine Learning becoming ubiquitous technologies, we now have a very large amount of data being generated. The main purpose of this classification process is to divide the pixels of data into any of the predefined classes. In most cases, the data utilized to do the classification is multi-spectral, and the numerical basis for categorizing is the spectral pattern present within the data for each pixel.

1.1 BACKGROUND INFORMATION

The demand for picture classification with various labels has resulted in a means to filter them in our local storage, which will simplify the process of automatically filtering photos from a big data set with many labels. Image classification has a wide range of applications. It can distinguish between several types of land use. Natural disasters such as floods, volcanoes, and severe droughts also use high-resolution imagery to assess the effects and damage they create. According to a recent survey, more than 70% of the population now considers filtering them to be a monumental undertaking. For such people, this concept is advantageous.

The convolutional neural network (CNN) is a type of deep neural network that has excelled in computer vision applications, particularly image categorization.. Convolutional Neural Networks are a type of multi-layer neurological network inspired by human visual and neural systems' operations (CNN). To filter out our platform's essential photographs, we need to create a resourceful and well-trained environment.

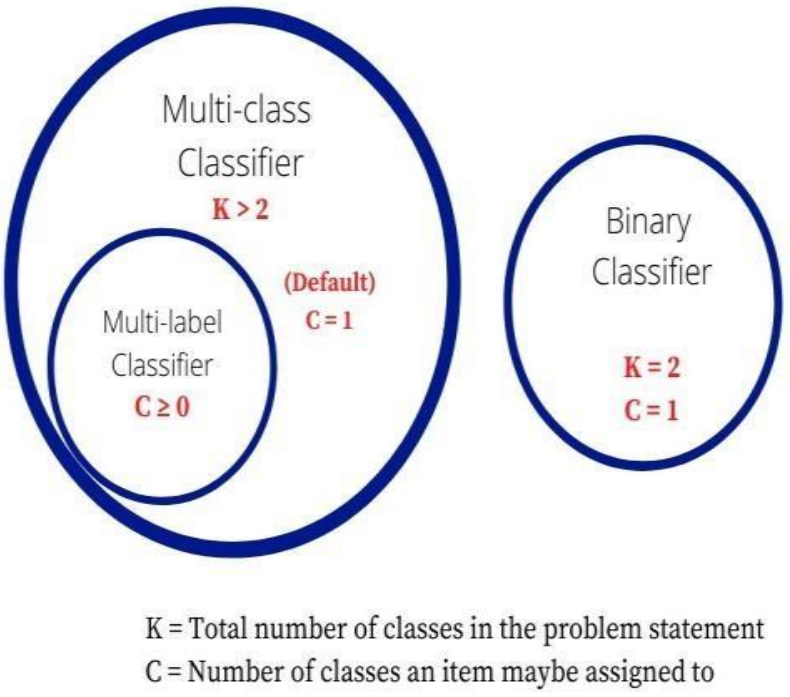
In the year 2012, AlexNet, a deep convolutional neural network, performed exceptionally well in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). This signalled the beginning of the widespread use and development of convolutional neural network models (CNN) such as VGGNet and GoogleNet.

Multi-label classification is a type of classification problem in which each image can have many labels, with some photos having all of them at the same time. In some aspects, the issue statement resembles single-label categorization, but the problem statement is more sophisticated.

Labeling photos can also be thought of as a multi-label classification problem, with labels indicating the existence of objects in the images.

Posters represent how humans utilize media to influence human behavior. The film industry is heavily reliant on the effectiveness of posters. Good posters are important for movies since they increase interest in watching the film. Good posters connect key parts of a film, such as cast, topic, and story elements. As a result, designers are motivated to add notable characteristics in their posters in order to make their films more appealing. For a variety of reasons, the image recognition topic is crucial when it comes to movie posters. To begin with, the movie poster would have practical consequences for the movie's viewers. Second, it would be interesting to learn what features of posters people find appealing, and this information would be beneficial for media providers in general. Finally, the subject of picture recognition allows us to experiment with awide range of methods and network designs in order to develop a successful model.

Current project is aimed at using different machine learning models, to analyze different data points to help get reasonable labels to an image on different machine learning models. These results can be further used for image classification, filtering, and automated clustering. It also works closely with genre prediction for movie posters.


Multi-class Classifier
K > 2
(Default)
C = 1
Multi-label Classifier
C ≥ 0
Binary Classifier
K = 2
C = 1
K = Total number of classes in the problem statement
C = Number of classes an item maybe assigned to

1.2 MOTIVATION

We live in a data-driven era. As a result of the massive amount of data produced today, the Internet of Things (IoT) and Artificial Intelligence (AI) are becoming universal technologies. The information could be in any format, including speech, text, image, or a combination of these. It could be in the form of photographs or videos.

Simply put, if a user has tens of thousands of photographs and wants to filter a document that was created three years ago, it will be a monumental feat for him to do so with the efficiency that our system gives. It is based on the notion of managing the user's images in the most compact and accessible manner possible in order to reduce the user's time and effort.

1.3 PROJECT OBJECTIVE

To create a Multilabel image classifier that will classify a given set of images into predefined labels. Evaluating the datasets by checking the loss and accuracy of our model.

1.4 REPORT LAYOUT

- Literature Review describes all the previous works done in this field.
- Methodology describes the architecture of the project.
- Implementation Details describes the tools that are used and the process that needs to be followed.
- Results show the final outputs that are done in the course of this project.
- Conclusion specifies the limitations and future scope of this project.

LITERATURE REVIEW

In the literature, there have been attempts to develop models that predict a film's genre based on non-visual advertising materials. Hoang used plot summaries to predict a movie's genre using several machine learning algorithms such as Naive Bayes and RNN. In 80.5 percent of trials, a Gated Recurrent Units neural network was able to determine genre. Makita and Lenskiy employed a multivariate Bernoulli event model to determine the likelihood of genre based on a movie's ratings in another study. It achieved a 50 percent success rate, which is a really substantial result. There have also been attempts at genre prediction using photographs, in addition to the above studies.

To conduct multi-label genre categorization from movie trailers, Wehrmann and Barros used Convolutional Neural Networks. This demonstrates attempts to determine a film's genre based on the aesthetic characteristics of its marketing. CNNs outperformed other genre-classification methods such as feature limiting, according to the study. We intend to change the network architecture and use CNNs for the sake of our project. One of the earliest attempts to divide movies into genres based on their posters was by Ivasic-Kos, Pobar, and Mikec. They were able to predict a film's whole genre classification 14 percent of the time using a dataset of 1500 posters with six genre classifications . Our approach builds on this foundation by employing a larger dataset, more genre labels, and more current methodologies. Chu and Guo have recently attempted to apply more modern frameworks to the problem of poster categorization by using the pre-trained YOLO version 2 network for object detection, which has yielded encouraging results.

METHODOLOGY

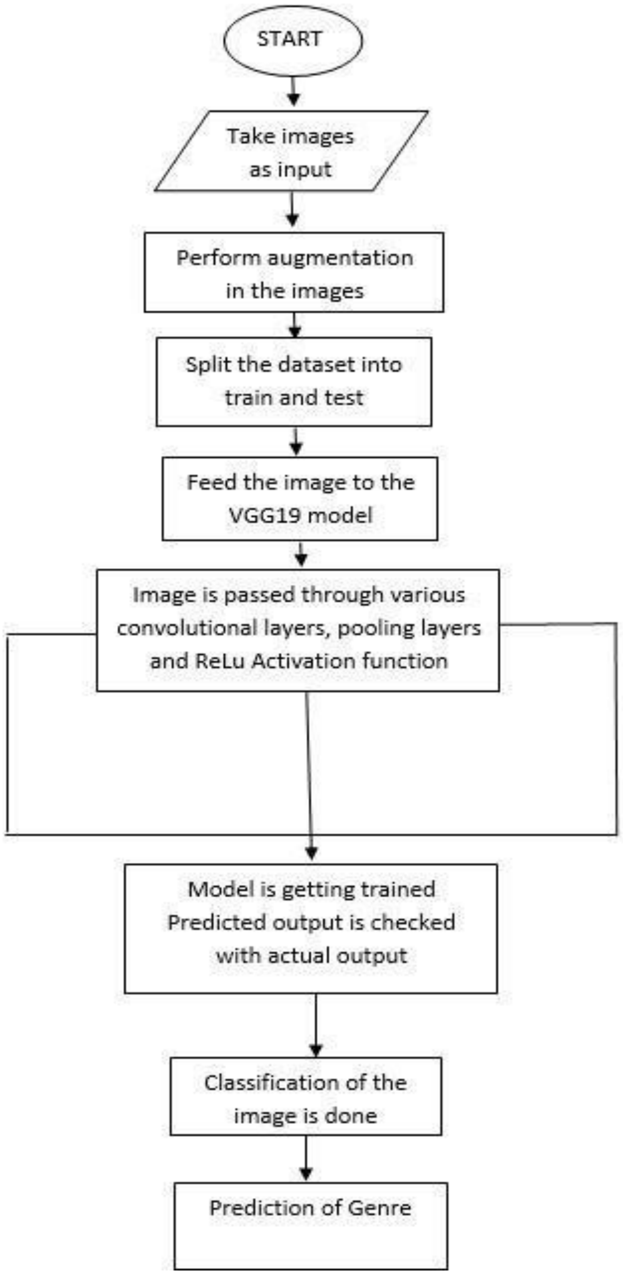
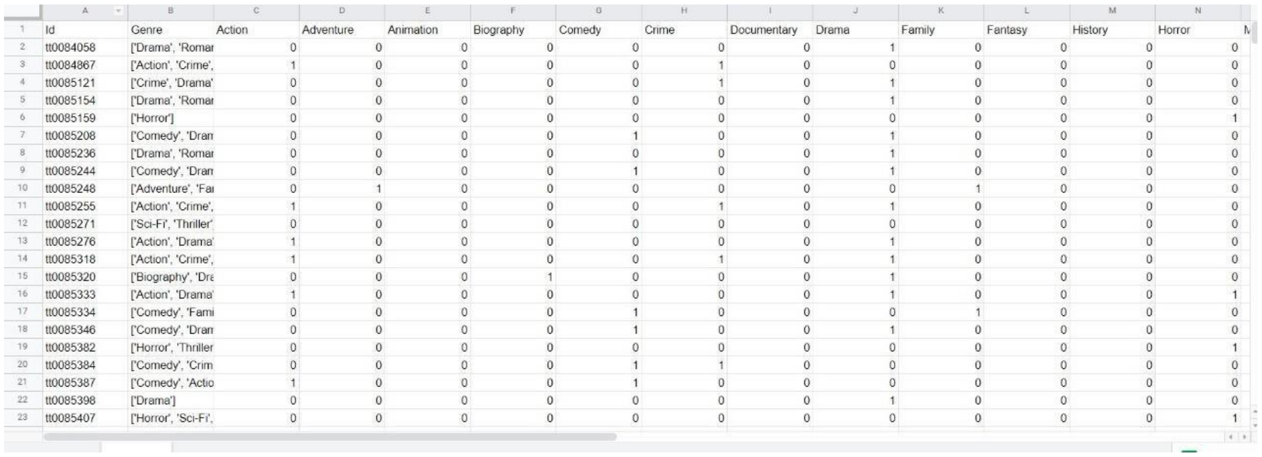

START
Take images as input
Perform augmentation in the images
Split the dataset into train and test
Feed the image to the VGG19 model
Image is passed through various convolutional layers, pooling layers and ReLu Activation function
Model is getting trained Predicted output is checked with actual output
Classification of the image is done
Prediction of Genre

Fig:-Dataflow Diagram

DATASET

A Kaggle dataset is made up of a variety of movie posters from different genres. Well-annotated media of movie posters must be trained, tested, and validated in order to create a Movie genre prediction system.. The IDs in the collection correspond to distinct movie poster genres, with columns having a binary value of 0 or 1 indicating whether the genre is present or not.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Id	Genre	Action	Adventure	Animation	Biography	Comedy	Crime	Documentary	Drama	Family	Fantasy	History	Horror
2	tt0084058	['Drama', 'Romar	0	0	0	0	0	0	0	1	0	0	0	0
3	tt0084867	['Action', 'Crime',	1	0	0	0	0	1	0	0	0	0	0	0
4	tt0085121	['Crime', 'Drama'	0	0	0	0	0	1	0	1	0	0	0	0
5	tt0085154	['Drama', 'Romar	0	0	0	0	0	0	0	1	0	0	0	0
6	tt0085159	['Horror']	0	0	0	0	0	0	0	0	0	0	0	1
7	tt0085208	['Comedy', 'Dran	0	0	0	0	1	0	0	1	0	0	0	0
8	tt0085236	['Drama', 'Romar	0	0	0	0	0	0	0	1	0	0	0	0
9	tt0085244	['Comedy', 'Dran	0	0	0	0	1	0	0	1	0	0	0	0
10	tt0085248	['Adventure', 'Fai	0	1	0	0	0	0	0	0	1	0	0	0
11	tt0085255	['Action', 'Crime',	1	0	0	0	0	1	0	1	0	0	0	0
12	tt0085271	['Sci-Fi', 'Thriller'	0	0	0	0	0	0	0	0	0	0	0	0
13	tt0085276	['Action', 'Drama'	1	0	0	0	0	0	0	1	0	0	0	0
14	tt0085318	['Action', 'Crime',	1	0	0	0	0	1	0	1	0	0	0	0
15	tt0085320	['Biography', 'Dre	0	0	0	1	0	0	0	1	0	0	0	0
16	tt0085333	['Action', 'Drama'	1	0	0	0	0	0	0	1	0	0	0	1
17	tt0085334	['Comedy', 'Fami	0	0	0	0	1	0	0	0	1	0	0	0
18	tt0085346	['Comedy', 'Dran	0	0	0	0	1	0	0	1	0	0	0	0
19	tt0085382	['Horror', 'Thriller	0	0	0	0	0	0	0	0	0	0	0	1
20	tt0085384	['Comedy', 'Crim	0	0	0	0	1	1	0	0	0	0	0	0
21	tt0085387	['Comedy', 'Actio	1	0	0	0	1	0	0	0	0	0	0	0
22	tt0085398	['Drama']	0	0	0	0	0	0	0	1	0	0	0	0
23	tt0085407	['Horror', 'Sci-Fi',	0	0	0	0	0	0	0	0	0	0	0	1

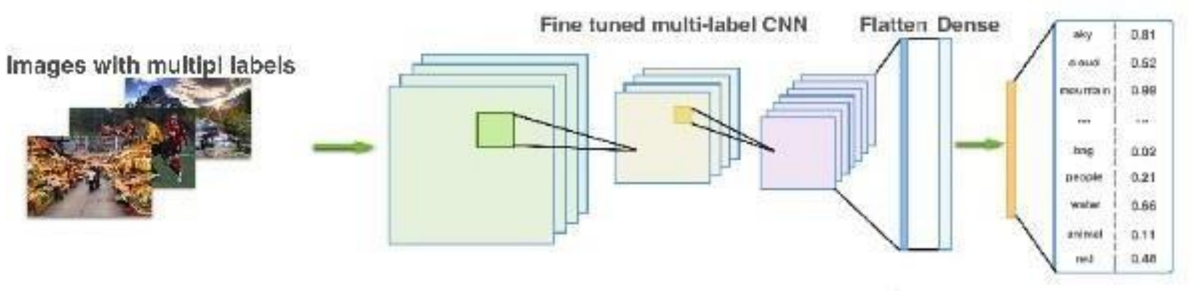
Fig: Snippet of training dataset table

In the training set, there are 28,000 labelled photos, whereas in the test set, there are 5,500 identified images. We also developed our own dataset for testing purposes, which will be used to implement our model.

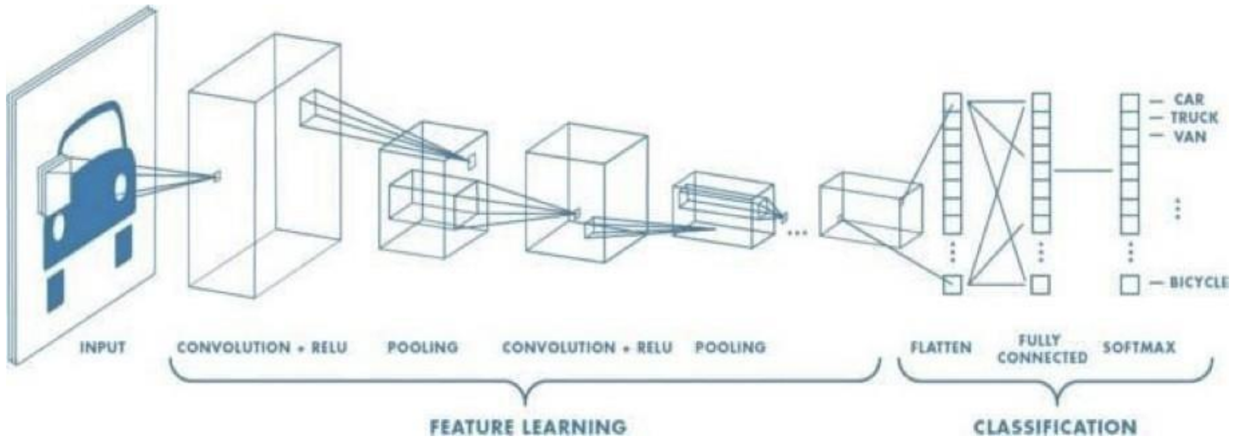
CONVOLUTIONAL NEURAL NETWORK

Convolutional neural networks are the most frequent deep neural network type (CNN or ConvNet). A Convolution Neural Network combines learnt options with input files using second convolution layers, and this design is used to process second knowledge such as photographs.

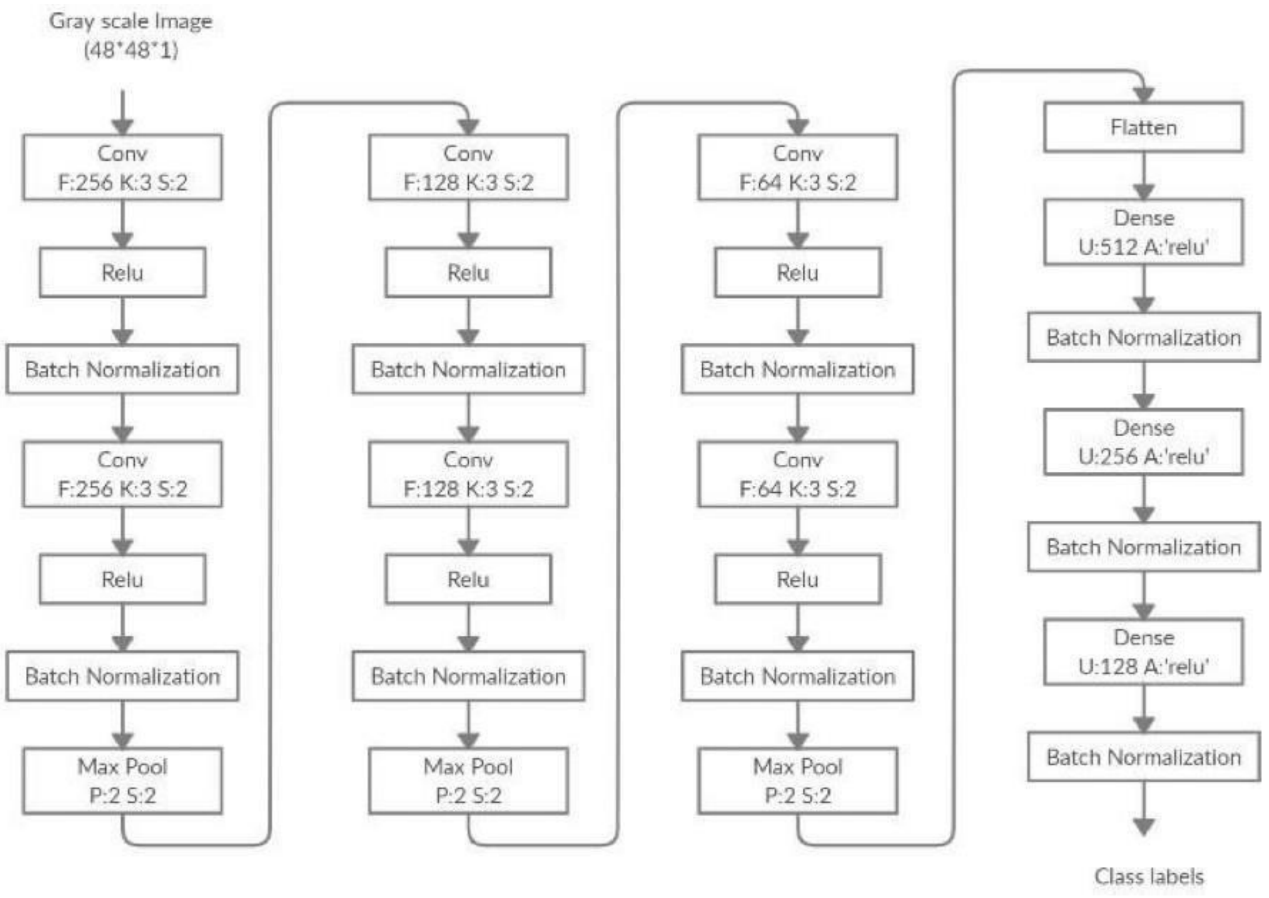
The need for manual feature extraction is no longer necessary. As a result, we have a tendency to set up the alternatives for categorising photos. Using CNN, the alternatives are extracted straight from the images. There is no prior training for the connected options. They can only be taught, whereas the network is trained using a collection of images. Using this automatic feature extraction for applications like object categorization, deep learning models become incredibly accurate.



.



CNN ARCHITECTURE



<u>**Fig: - CNN ARCHITECTURE 2**</u>

Keras Layers Using in This Model -

1. Flatten
2. Dense
3. Dropout
4. Batch Normalization
5. Maxpool2D
6. conv2D

Preprocessing

The noise reduction property adjustment part (angular rotation, scaling, and exact position detection) as well as image augmentation are very significant parts of the dataset pre-processing.

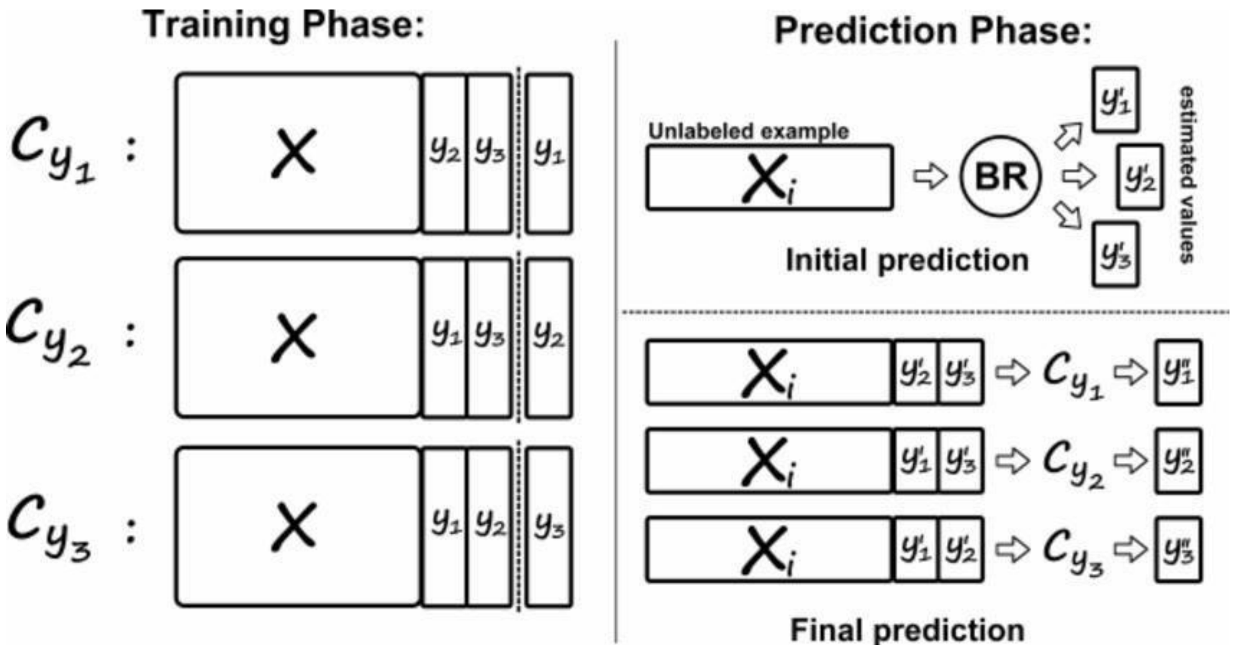
Normalization - An image is normalized to reduce illumination variations and improve the appearance of the face.

Gray scaling is the conversion of a colored image into an image whose pixel value is determined by the intensity of light on the image. Because colored images are harder for algorithms to process, grey scaling is used.

Image resizing - The image is scaled to remove any superfluous elements. This minimizes the amount of memory used and speeds up calculation.

Dilation is a morphological procedure that enlarges the region and emphasizes features.

TRAINING OF MODEL



DEFINITION OF CNN MODEL

1. **Sequential Layer**
2. **Convolutional Layer (16) kernel = 5*5**
3. **Batch Normalization Layer**
4. **Maxpool Layer**
5. **Dropout Layer**

6. **Convolutional Layer (16) kernel = 5*5**
7. **Batch Normalization Layer**
8. **Maxpool Layer**
9. **Dropout Layer**

10. **Convolutional Layer (16) kernel = 5*5**
11. **Batch Normalization Layer**
12. **Maxpool Layer**
13. **Dropout Layer**

14. **Convolutional Layer (16) kernel = 5*5**
15. **Batch Normalization Layer**
16. **Maxpool Layer**
17. **Dropout Layer**

18. **Flatten Layer**

19. **Dense Layer**
20. **Batch Normalization**
21. **Dropout Layer**

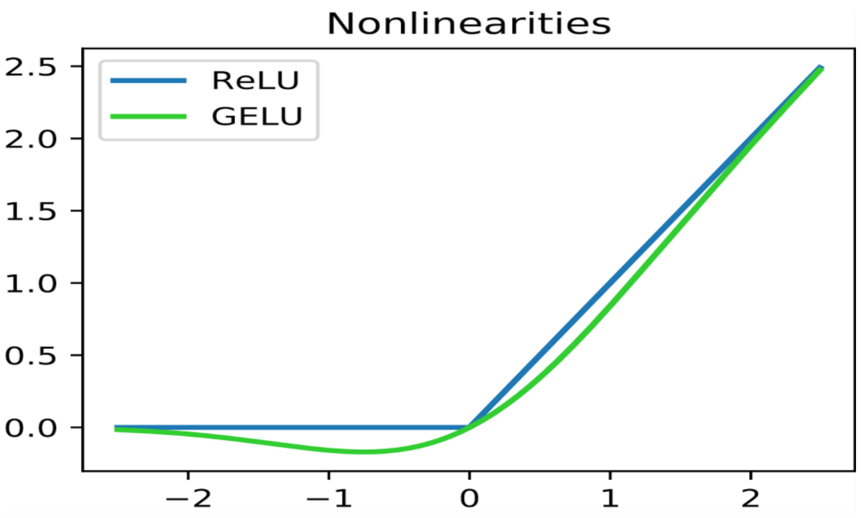
22. **Dense Layer**
23. **Batch Normalization**
24. **Dropout Layer**

25. **Dense Layer (25) activation = Sigmoid**

ACTIVATION FUNCTIONS

1. RELU ACTIVATION FUNCTION -

The activation function of the rectifier, or ReLU (Rectified Linear Unit), is defined by the positive component of its argument: x represents the neuron's input. This is known as a ramp function in electrical engineering and is equivalent to half-wave rectification. This activation function has been employed in the context of visual feature extraction in hierarchical neural networks since the late 1960s. It was later suggested that it has substantial biological and mathematical underpinnings. When compared to the extensively used activation functions previous to 2011, such as the logistic sigmoid (influenced by probability theory; see logistic regression) and its more practical equivalent, the hyperbolic tangent, it was discovered in 2011 that it enabled better training of deeper networks. The rectifier is the most used deep neural network activation function as of 2017.



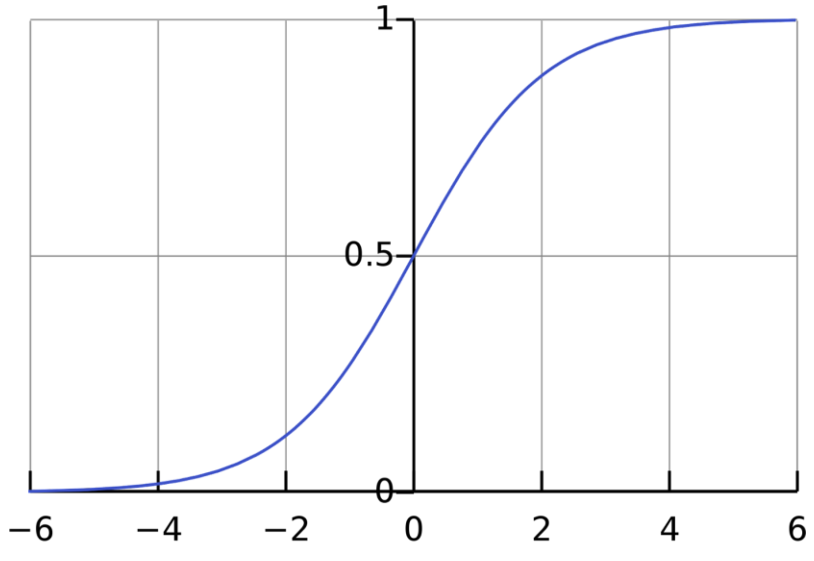
ReLU ACTIVATION FUNCTIION

2. The Sigmoid Function

A sigmoid function is a mathematical function that has a distinctive "S"-shaped curve, also called a sigmoid curve.

A common example of a sigmoid function is the logistic function (shown in Figure 1 and represented by the formula:).
More standard sigmoid functions can be found in the Examples section. In numerous disciplines, including artificial neural networks, the term "sigmoid function" is used as an alias for the logistic function.

The sigmoid function is represented by the Gompertz curve (used in modelling systems that saturate at big x values) and the ogee curve (used in the spillway of some dams). All real numbers are in the domain of sigmoid functions, and the return (response) value is usually monotonically increasing but can alternatively be decreasing. Sigmoid functions normally have a return value (y axis) in the range of 0 to 1. The 1 to 1 range is another often used range.



SIGMOID FUNCTION

LOSS FUNCTION -

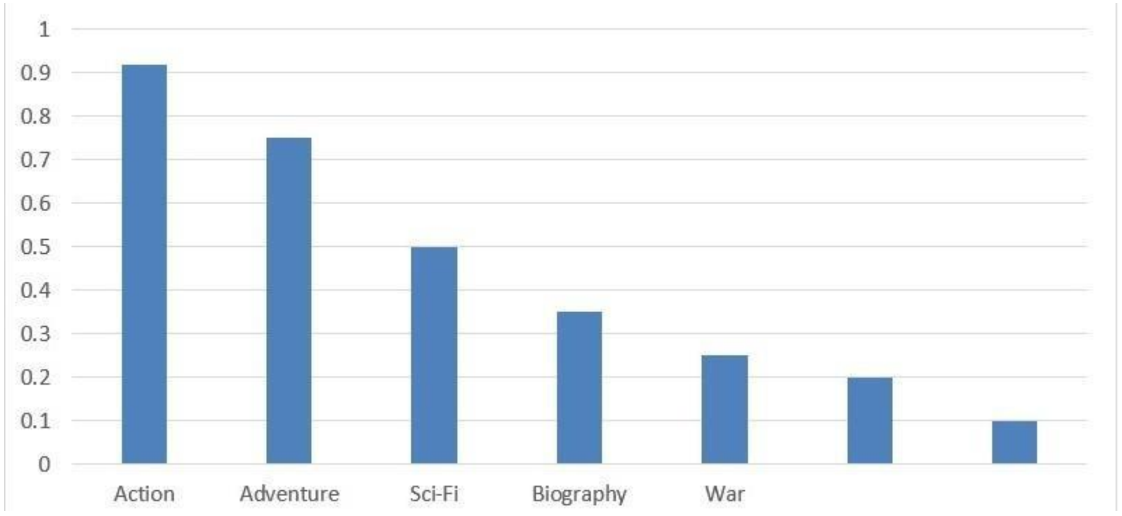
$$logloss = - \frac{1}{N}\sum_{i}^{N}\sum_{j}^{M} y_{ij}\log(p_{ij})$$

- N is the number of rows
- M is the number of classes

The actual class output, which might be either 0 or 1, is compared to each of the projected probabilities. The score is then computed, with the probabilities being penalized based on their divergence from the expected value. This is the distance between the value and the actual value.

TESTING OF MODEL

MARVEL STUDIOS
AVENGERS
ENDGAME
APRIL 26



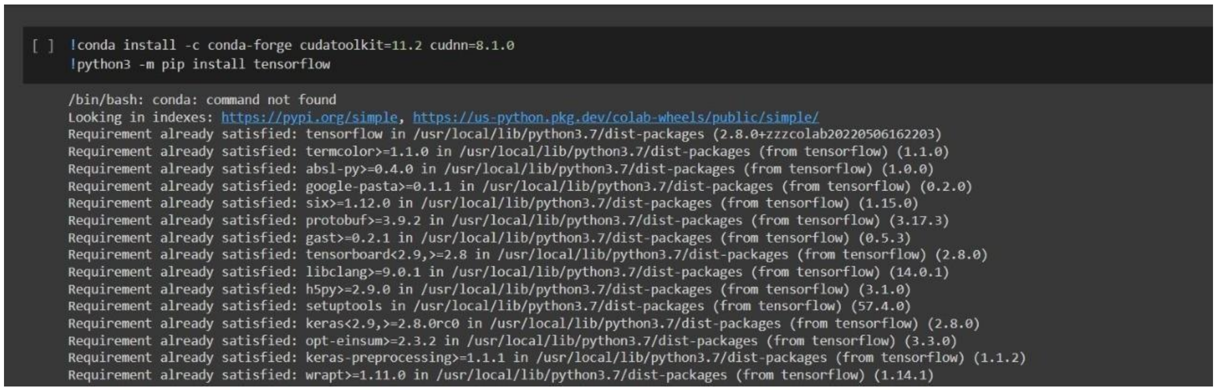
The network has now been trained using the picture pairings and labels that we collected earlier. 'Binary-cross entropy' is the loss function, while 'Adagrad' is the best optimizer.

During training, the model was penalised for making classification errors on the negative class. This was done with the knowledge that the bank would lose more money if an incorrect signature was classified as correct.

EXPERIMENTAL RESULTS

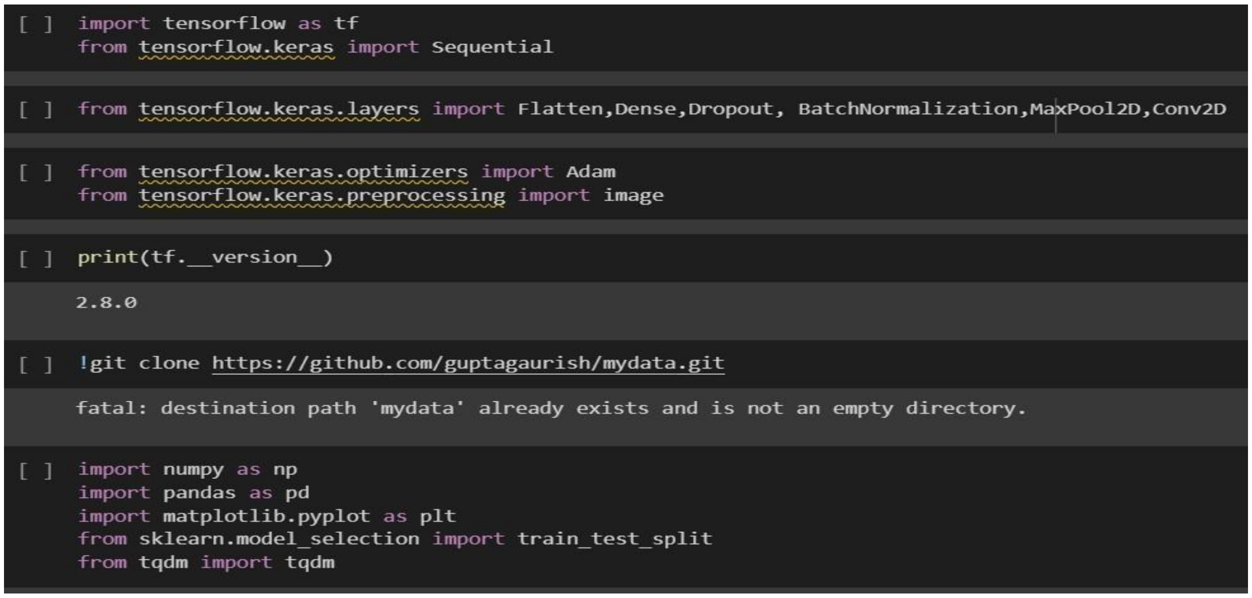
4.1 Installing Tensorflow and Other Libraries -

Firstly we download all the required Libaries which will be used in the implementation of model and apply. Important libraries to install are Keras, Tensorflow, Matplotlib, sequential, Layers and many more.



```
[ ]  !conda install -c conda-forge cudatoolkit=11.2 cudnn=8.1.0
     !python3 -m pip install tensorflow

     /bin/bash: conda: command not found
     Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
     Requirement already satisfied: tensorflow in /usr/local/lib/python3.7/dist-packages (2.8.0+zzzcolab20220506162203)
     Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.1.0)
     Requirement already satisfied: absl-py>=0.4.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.0.0)
     Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (0.2.0)
     Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.15.0)
     Requirement already satisfied: protobuf>=3.9.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (3.17.3)
     Requirement already satisfied: gast>=0.2.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (0.5.3)
     Requirement already satisfied: tensorboard<2.9,>=2.8 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (2.8.0)
     Requirement already satisfied: libclang>=9.0.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (14.0.1)
     Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (3.1.0)
     Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (from tensorflow) (57.4.0)
     Requirement already satisfied: keras<2.9,>=2.8.0rc0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (2.8.0)
     Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (3.3.0)
     Requirement already satisfied: keras-preprocessing>=1.1.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.1.2)
     Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.14.1)
```

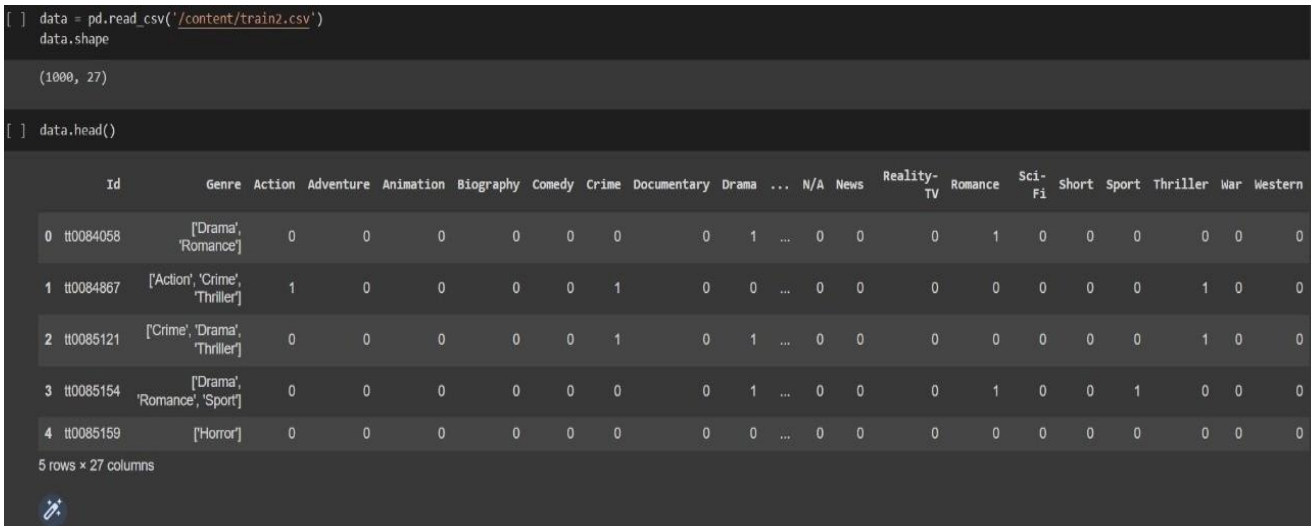
4.2 Initializing the Libraries to use them in code -



```
[ ]  import tensorflow as tf
     from tensorflow.keras import Sequential

[ ]  from tensorflow.keras.layers import Flatten,Dense,Dropout, BatchNormalization,MaxPool2D,Conv2D

[ ]  from tensorflow.keras.optimizers import Adam
     from tensorflow.keras.preprocessing import image

[ ]  print(tf.__version__)

     2.8.0

[ ]  !git clone https://github.com/guptagaurish/mydata.git

     fatal: destination path 'mydata' already exists and is not an empty directory.

[ ]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     from sklearn.model_selection import train_test_split
     from tqdm import tqdm
```

4.3 Data Reading and Visualization -

We use the csv file to have a dataset ready and and work on that file to implement and train the model. The model will then test on different kind of movie posters.



```
data = pd.read_csv('/content/train2.csv')
data.shape
```

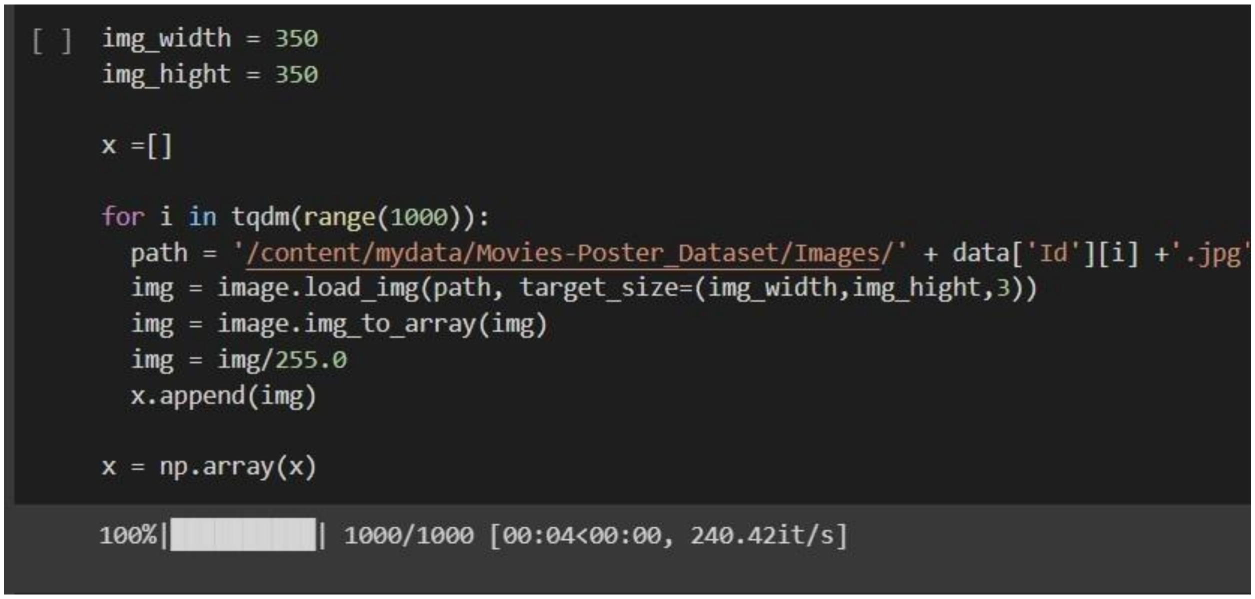
```
(1000, 27)
```

```
data.head()
```

	Id	Genre	Action	Adventure	Animation	Biography	Comedy	Crime	Documentary	Drama	...	N/A	News	Reality-TV	Romance	Sci-Fi	Short	Sport	Thriller	War	Western
0	tt0084058	['Drama', 'Romance']	0	0	0	0	0	0	0	1	...	0	0	0	1	0	0	0	0	0	0
1	tt0084867	['Action', 'Crime', 'Thriller']	1	0	0	0	0	1	0	0	...	0	0	0	0	0	0	0	1	0	0
2	tt0085121	['Crime', 'Drama', 'Thriller']	0	0	0	0	0	1	0	1	...	0	0	0	0	0	0	0	1	0	0
3	tt0085154	['Drama', 'Romance', 'Sport']	0	0	0	0	0	0	0	1	...	0	0	0	1	0	0	1	0	0	0
4	tt0085159	['Horror']	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 27 columns

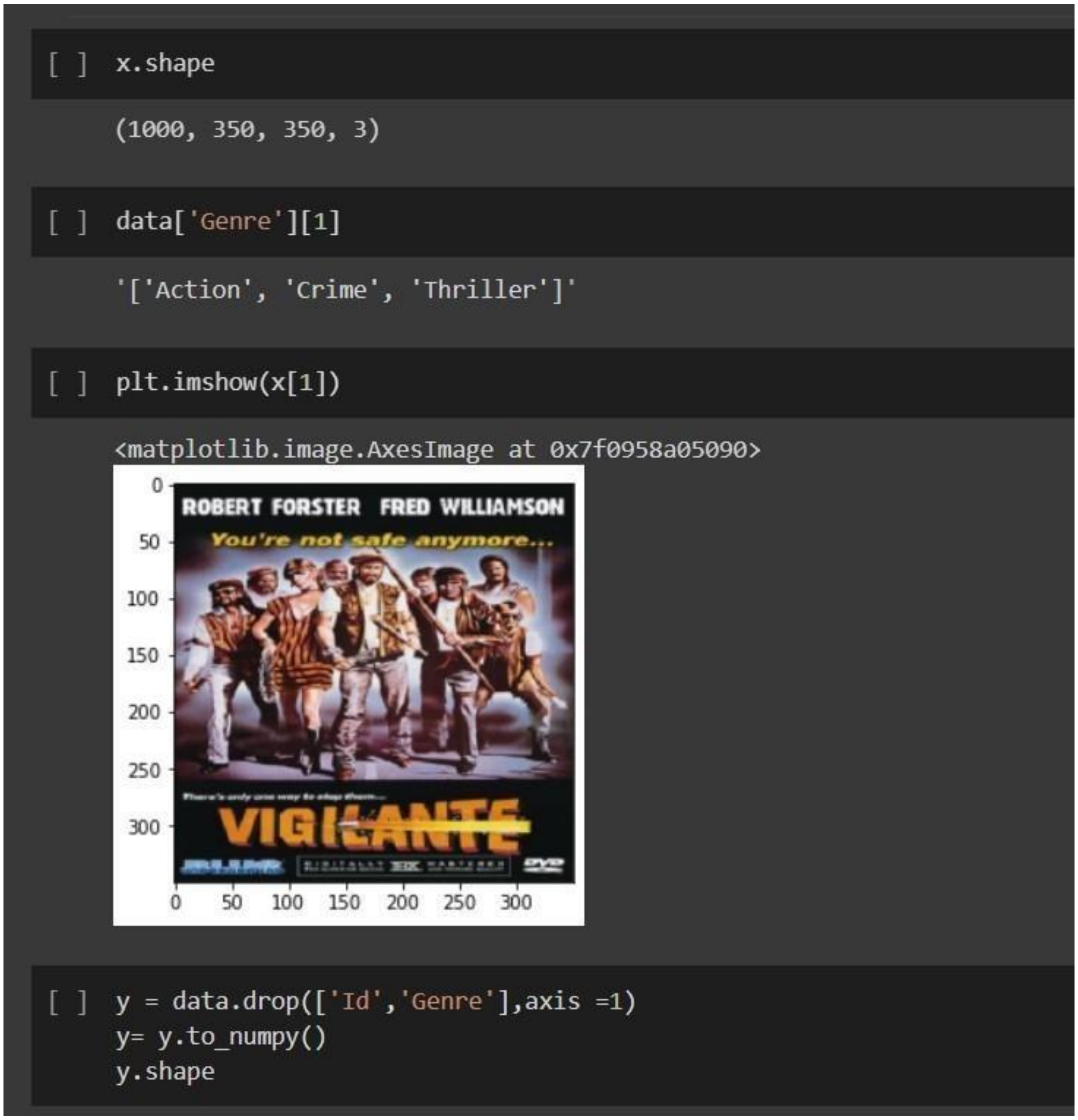
4.4 DataSet Loading and Preprocessing -



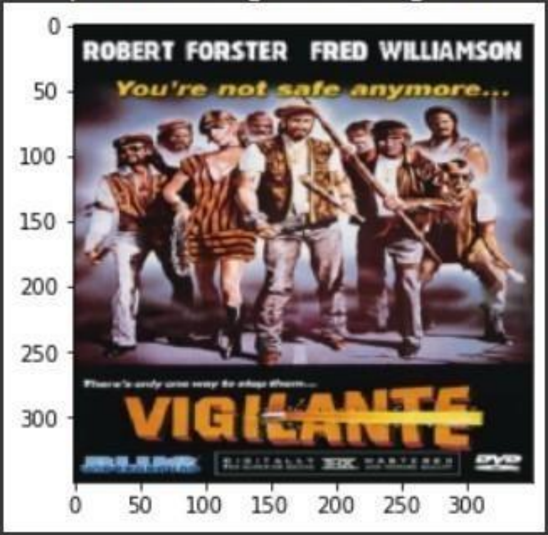
```
[ ]  img_width = 350
     img_hight = 350

     x =[]

     for i in tqdm(range(1000)):
       path = '/content/mydata/Movies-Poster_Dataset/Images/' + data['Id'][i] +'.jpg'
       img = image.load_img(path, target_size=(img_width,img_hight,3))
       img = image.img_to_array(img)
       img = img/255.0
       x.append(img)

     x = np.array(x)

     100%|██████████| 1000/1000 [00:04<00:00, 240.42it/s]
```

Now the data is loaded to the model to train. This method requires a good amount of RAM and Hardware, Software requirements as well. Otherwise the model will be crashed for sure.

4.5 Data Visualization -

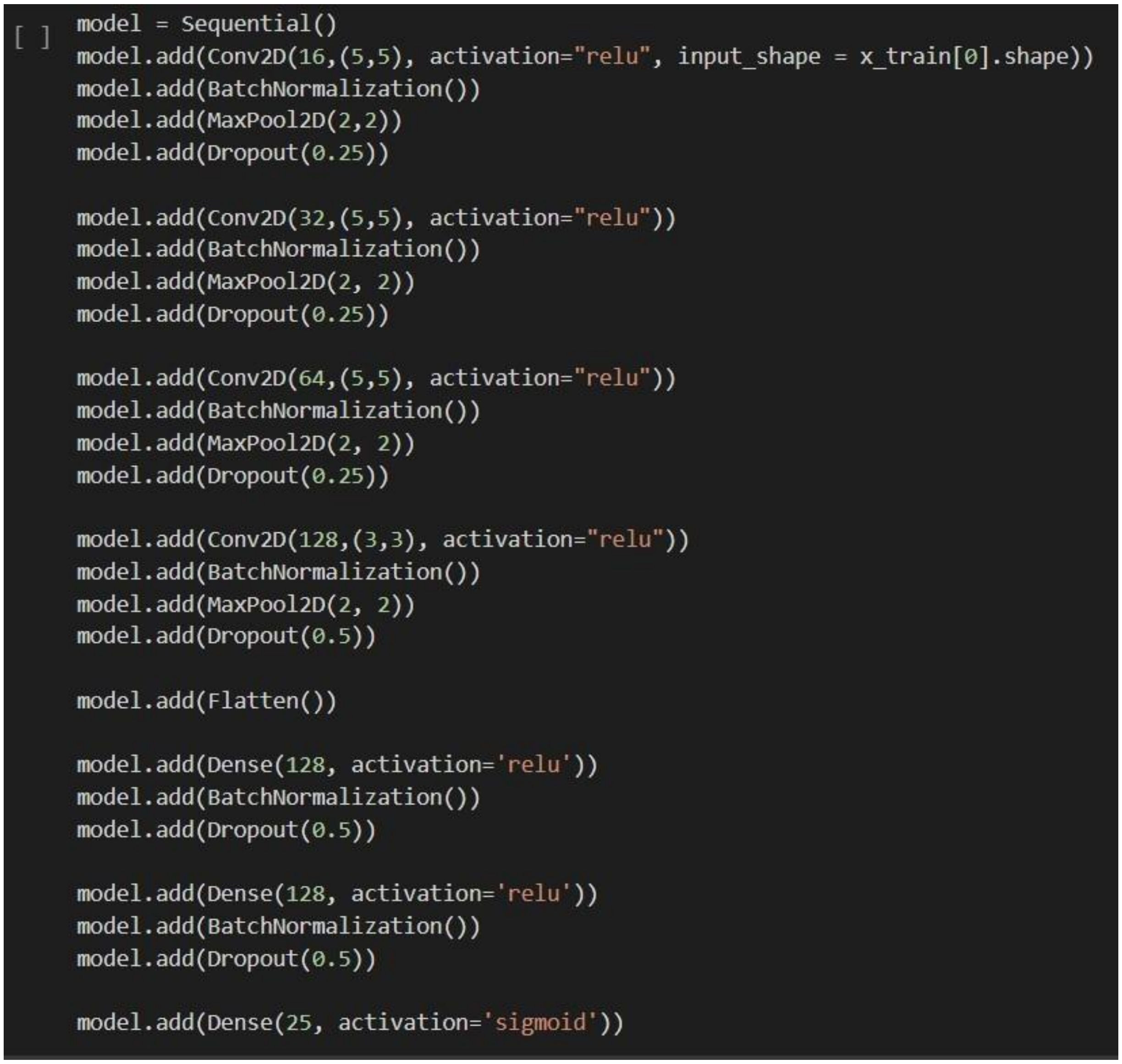


```
[ ]  x.shape

     (1000, 350, 350, 3)

[ ]  data['Genre'][1]

     '['Action', 'Crime', 'Thriller']'

[ ]  plt.imshow(x[1])

     <matplotlib.image.AxesImage at 0x7f0958a05090>
```



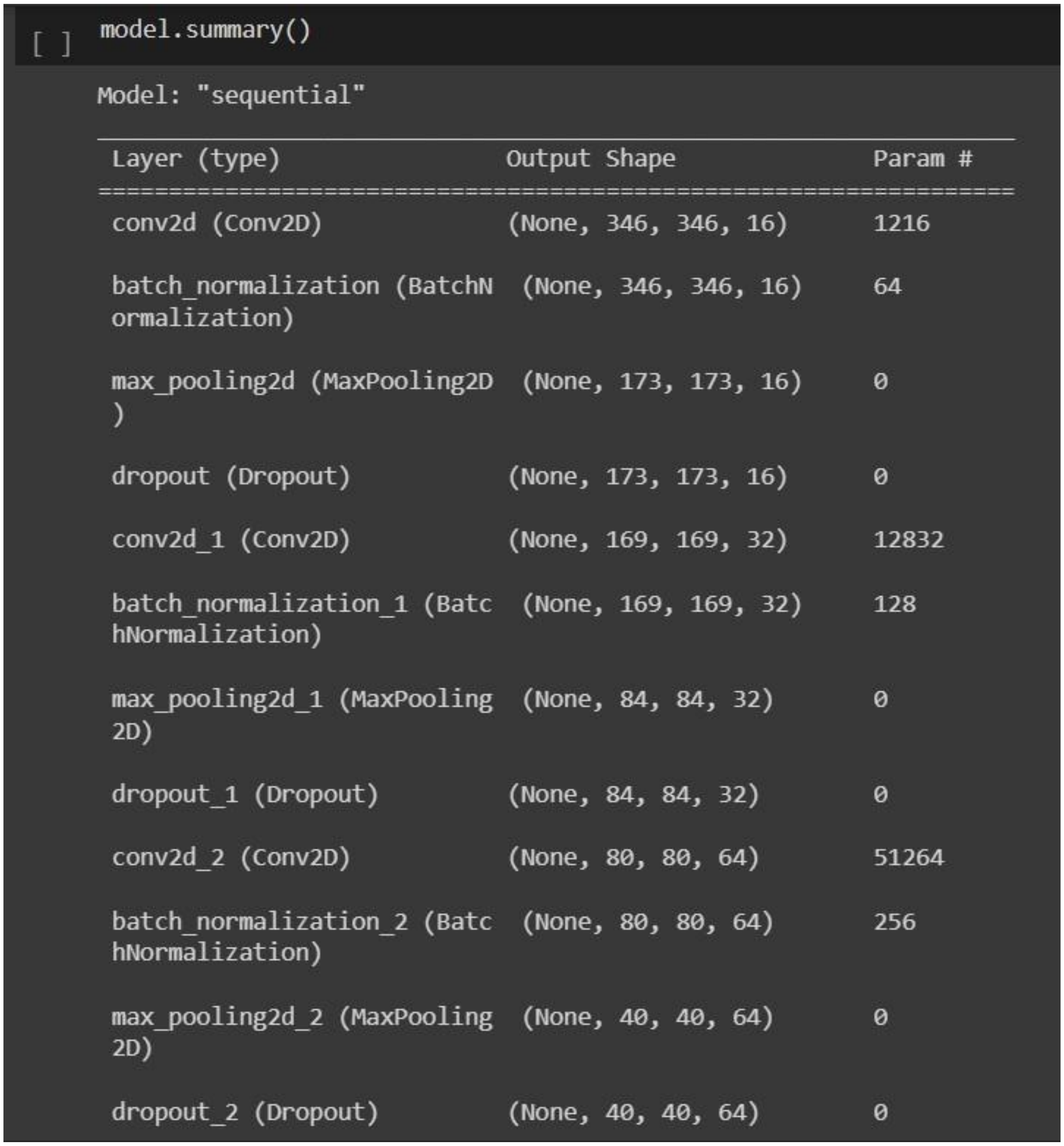
```
[ ]  y = data.drop(['Id','Genre'],axis =1)
     y= y.to_numpy()
     y.shape
```

4.6 BUILDING CNN FOR MODEL -

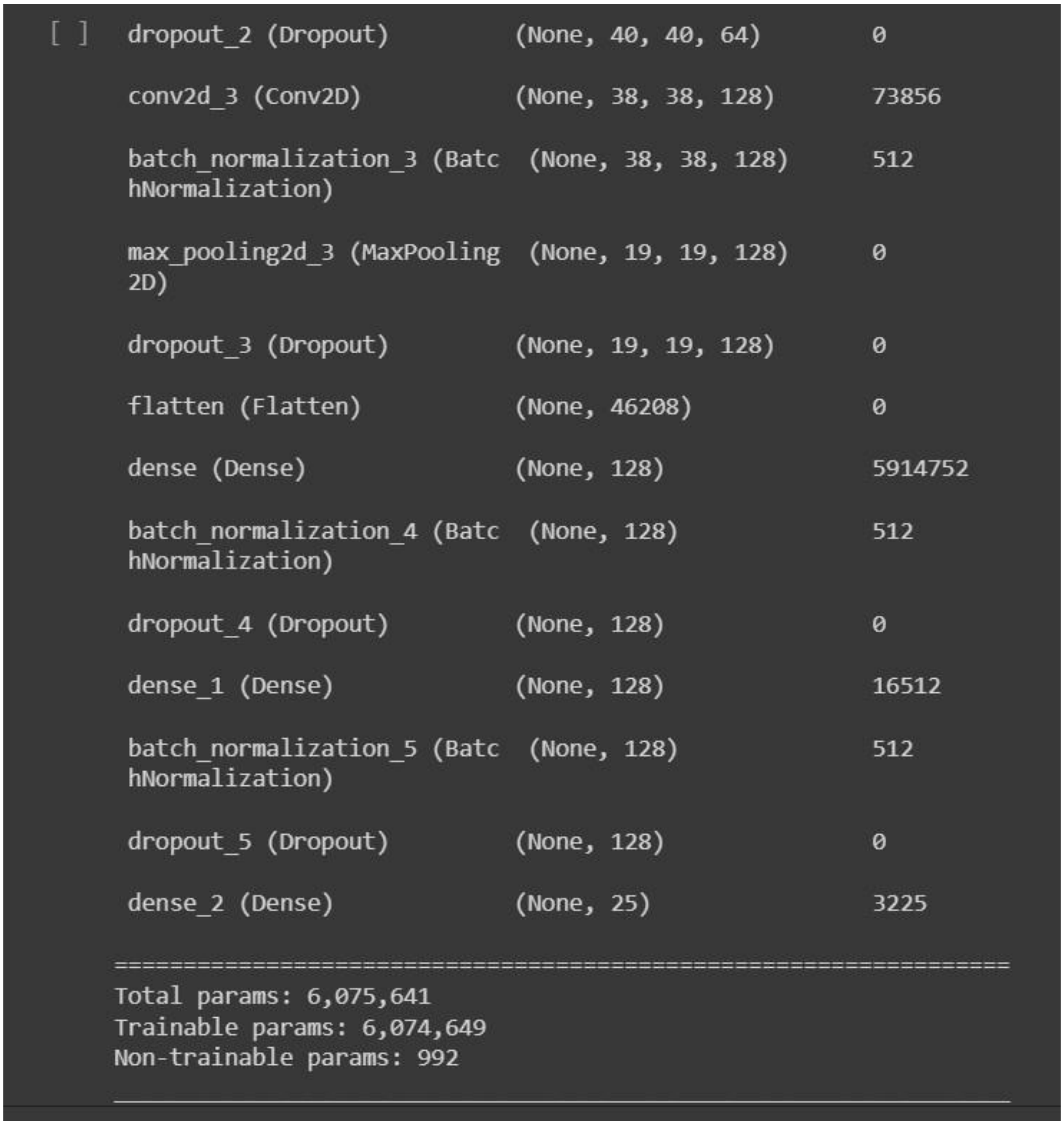


```
[ ]  model = Sequential()
     model.add(Conv2D(16,(5,5), activation="relu", input_shape = x_train[0].shape))
     model.add(BatchNormalization())
     model.add(MaxPool2D(2,2))
     model.add(Dropout(0.25))

     model.add(Conv2D(32,(5,5), activation="relu"))
     model.add(BatchNormalization())
     model.add(MaxPool2D(2, 2))
     model.add(Dropout(0.25))

     model.add(Conv2D(64,(5,5), activation="relu"))
     model.add(BatchNormalization())
     model.add(MaxPool2D(2, 2))
     model.add(Dropout(0.25))

     model.add(Conv2D(128,(3,3), activation="relu"))
     model.add(BatchNormalization())
     model.add(MaxPool2D(2, 2))
     model.add(Dropout(0.5))

     model.add(Flatten())

     model.add(Dense(128, activation='relu'))
     model.add(BatchNormalization())
     model.add(Dropout(0.5))

     model.add(Dense(128, activation='relu'))
     model.add(BatchNormalization())
     model.add(Dropout(0.5))

     model.add(Dense(25, activation='sigmoid'))
```

4.7 Model Summary -



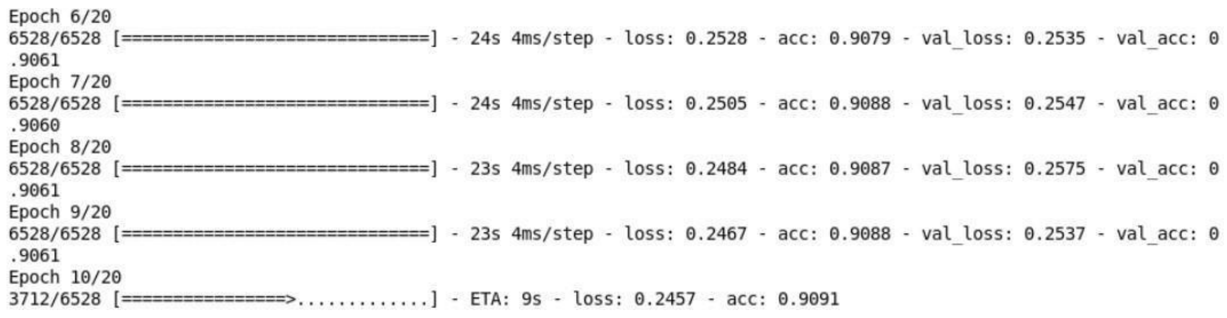
```
[ ]  model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 346, 346, 16)      1216

 batch_normalization (BatchN  (None, 346, 346, 16)     64
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 173, 173, 16)     0
 )

 dropout (Dropout)           (None, 173, 173, 16)      0

 conv2d_1 (Conv2D)           (None, 169, 169, 32)      12832

 batch_normalization_1 (Batc  (None, 169, 169, 32)     128
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 84, 84, 32)       0
 2D)

 dropout_1 (Dropout)         (None, 84, 84, 32)        0

 conv2d_2 (Conv2D)           (None, 80, 80, 64)        51264

 batch_normalization_2 (Batc  (None, 80, 80, 64)       256
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 40, 40, 64)       0
 2D)

 dropout_2 (Dropout)         (None, 40, 40, 64)        0
```



```
[ ]  dropout_2 (Dropout)         (None, 40, 40, 64)        0

     conv2d_3 (Conv2D)           (None, 38, 38, 128)       73856

     batch_normalization_3 (Batc  (None, 38, 38, 128)      512
     hNormalization)

     max_pooling2d_3 (MaxPooling  (None, 19, 19, 128)      0
     2D)

     dropout_3 (Dropout)         (None, 19, 19, 128)       0

     flatten (Flatten)           (None, 46208)             0

     dense (Dense)               (None, 128)               5914752

     batch_normalization_4 (Batc  (None, 128)              512
     hNormalization)

     dropout_4 (Dropout)         (None, 128)               0

     dense_1 (Dense)             (None, 128)               16512

     batch_normalization_5 (Batc  (None, 128)              512
     hNormalization)

     dropout_5 (Dropout)         (None, 128)               0

     dense_2 (Dense)             (None, 25)                3225

    =================================================================
    Total params: 6,075,641
    Trainable params: 6,074,649
    Non-trainable params: 992
    _________________________________________________________________
```

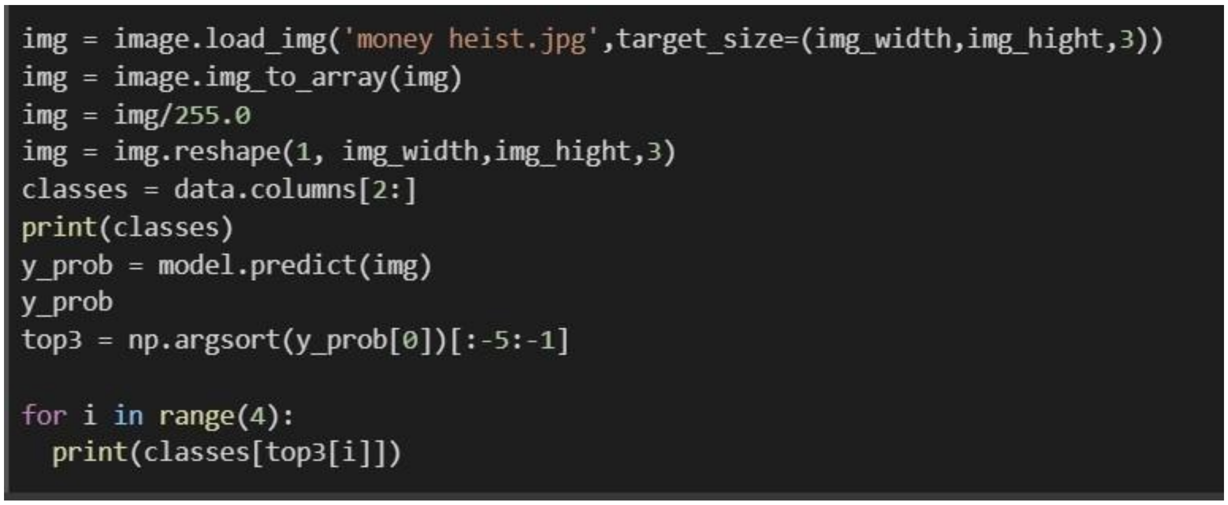
This is the way we designed our CNN model so that the accuracy will be more than 90%.

4.8 Testing the Model -



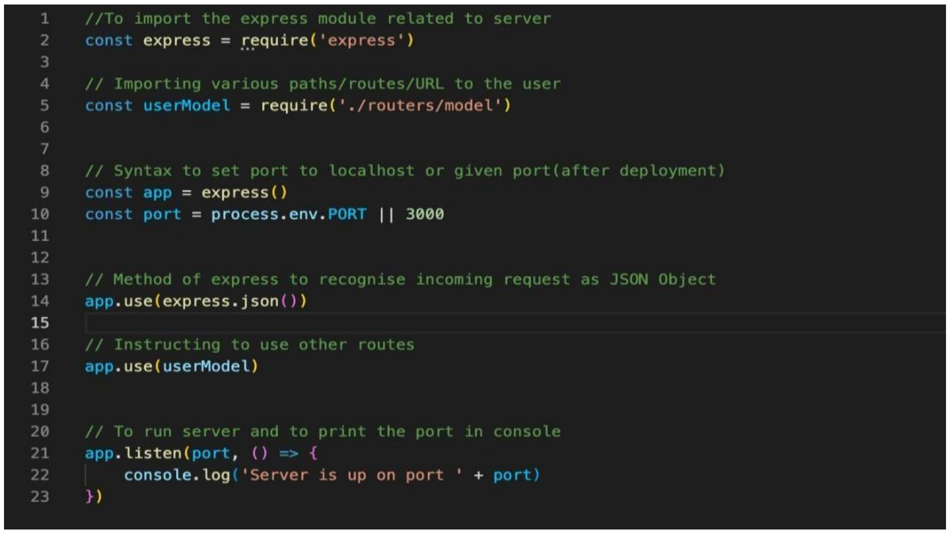
```
Epoch 6/20
6528/6528 [==============================] - 24s 4ms/step - loss: 0.2528 - acc: 0.9079 - val_loss: 0.2535 - val_acc: 0
.9061
Epoch 7/20
6528/6528 [==============================] - 24s 4ms/step - loss: 0.2505 - acc: 0.9088 - val_loss: 0.2547 - val_acc: 0
.9060
Epoch 8/20
6528/6528 [==============================] - 23s 4ms/step - loss: 0.2484 - acc: 0.9087 - val_loss: 0.2575 - val_acc: 0
.9061
Epoch 9/20
6528/6528 [==============================] - 23s 4ms/step - loss: 0.2467 - acc: 0.9088 - val_loss: 0.2537 - val_acc: 0
.9061
Epoch 10/20
3712/6528 [================>.............] - ETA: 9s - loss: 0.2457 - acc: 0.9091
```

4.9 Apply the model to test with the random movie posters -

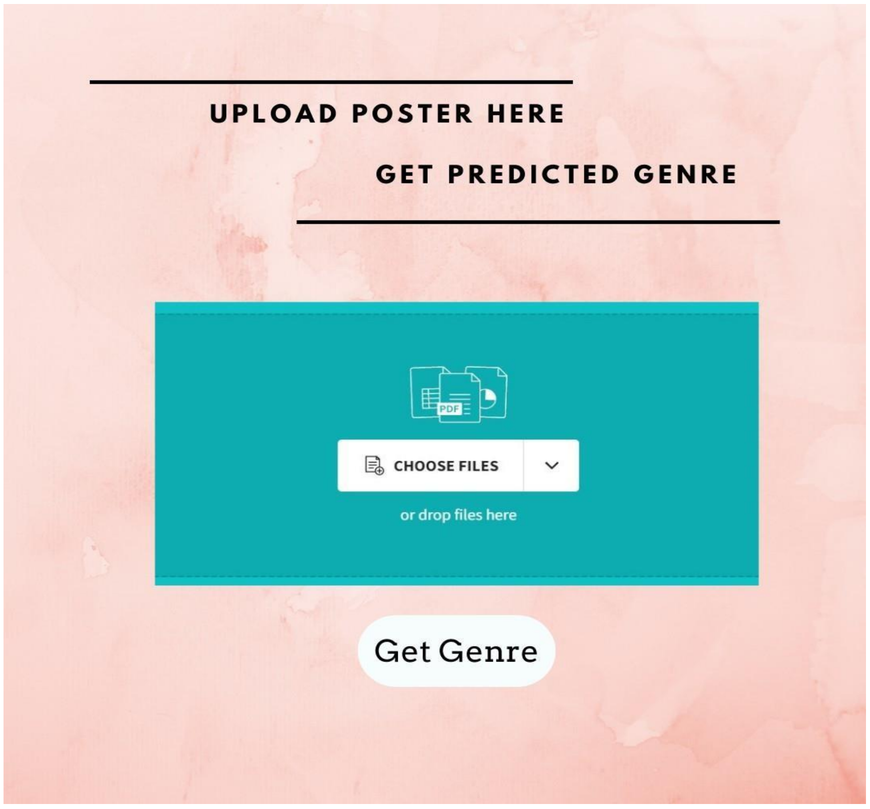


```
img = image.load_img('money heist.jpg',target_size=(img_width,img_hight,3))
img = image.img_to_array(img)
img = img/255.0
img = img.reshape(1, img_width,img_hight,3)
classes = data.columns[2:]
print(classes)
y_prob = model.predict(img)
y_prob
top3 = np.argsort(y_prob[0])[:-5:-1]

for i in range(4):
  print(classes[top3[i]])
```

4.10 Backend Integration -



```
//To import the express module related to server
const express = require('express')

// Importing various paths/routes/URL to the user
const userModel = require('./routers/model')


// Syntax to set port to localhost or given port(after deployment)
const app = express()
const port = process.env.PORT || 3000


// Method of express to recognise incoming request as JSON Object
app.use(express.json())

// Instructing to use other routes
app.use(userModel)


// To run server and to print the port in console
app.listen(port, () => {
    console.log('Server is up on port ' + port)
})
```



CONCLUSION

- After implementing the model and deploying it, the user can upload all their images to the website i.e., thousands of images and then can search for the different labels to find the correct image the user is looking for.
- It will automate the whole process to find an image which can take less than a few seconds so that it saves a lot of time for the user and clustering helps to improve the user experience.
- It will be useful to all kinds of user images.
- We will be evaluating the datasets by checking the loss and accuracy of the implemented model.
- Frontend will contain sample label images so that it will help users to give an idea of how they can search them.
- A major part of our project works on movie genre prediction which takes posters of a movie and assigns a set of labels to each image and then predicting the genres which have the mostfrequency.

Limitations of Model

1. The model is developed by using limited training data set hence it can be not that accurate to specify the Genre.

2. The model is very light weight and cannot handle multiple images in one time.

3. Model has to use the internet for predicting the genre. It can be made offline usage for better experience.

4. Model can not predict same Genre everytime when it sees the different poster as it differ from poster to poster.

5. The model can be implemented to read the words present in poster picture for better understanding of movie.

FUTURE WORK

Here, we have used VGG19 as the CNN model for our project. In future, we will be implementing our project using other CNN models such as Alexnet, VGG16, Resnet etc to see how the model will react to the data sets and also how better would be the accuracy of the result.

One way to improve our model is to train it with more balanced data. We could supplement our dataset to expose our model to more cases, making the model more resilient, because the dataset that was used is imbalanced. We could add to the dataset by oversampling minority classes, undersampling majority classes, or a combination of the two to make it more balanced. We also had to limit the number of datapoints we could evaluate for KNN as well as the number of probable neighbors to analyse owing to memory constraints. As a result, with increased computational resources, we will be able to investigate the performance of KNN in greater depth, as well as train much larger designs. Finally, we used elements on posters to determine the genre of a film. Movie posters, on the other hand, can reveal a lot more about a film than just its genre. If we had more time, we'd try to predict other movie characteristics from posters, such as ratings. In an ideal world, we'd train a single model that could predict all of the essential features we're looking for.

REFERENCES

[1] General Multi-Label Image Classification with Transformers.

[2] Multi-label Image Classification with A Probabilistic Label Enhancement Model **Xin Li and Feipeng Zhao and Yuhong Guo**;Department of Computer and Information Sciences Temple University Philadelphia, PA 19122, USA; 2017.

[3] Improving Pairwise Ranking for Multi-Label Image Classification; **Yuncheng Li, Yale Song, Jiebo Luo**; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 3617-3625.

[4] **Aaliyah Alshehri, Yakoub Bazi, Nassim Ammour, Haidar Almubarak, and Naif Alajlan. 2019.** Deep Attention Neural Network for Multi-Label Classification in Unmanned Aerial VehicleImagery. IEEE Access, Vol. 7 (2019), 119873--119880.

[5] Multi-Label Image Classification by Feature Attention Network **Zheng Yan; Weiwei Liu;**

[6] **Shiping Wen; Yin Yang;**Published in the IEEE access, 18 July 2019.

[7] ***Jack Lanchantin, Tianlu Wang, Vicente Ordonez, Yanjun Qi***; Proceedings of the IEEE/CVFConference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 16478-16488.

[8] Multilabel Image Classification via Feature/Label Co-Projection.

[9] IEEE Transactions on Systems, Man, and Cybernetics: Systems **(Volume: 51, Nov. 2021).**