# MACHINE LEARNING – CS6301

## LAB PROJECT

## BACHELOR OF ENGINEERING

***in***


## COMPUTER SCIENCE AND ENGINEERING

## COLLEGE OF ENGINEERING GUINDY,

## ANNA UNIVERSITY, CHENNAI 600 025

## JUNE 2022

Submitted by

| | |
|---|---|
| **Barathraj T** | **2019103511** |
| **Pranav Chandar KR** | **2019103046** |
| **Shafeeq Ur Rahman PA** | **2019103058** |

# FESTIVAL IMAGE CLASSIFICATION USING DEEP LEARNING


## ABSTRACT:

Deep learning models have demonstrated improved efficacy in image classification. In this paper, we implement three different image classification models on our festival dataset containing augmented images. “Residual Attention Network”, a convolutional neural network that adopts a mixed attention mechanism into a very deep structure for image classification tasks. This particular model can generate attention-aware features and is robust to noisy images. We construct 56-layer Residual Attention Networks to classify small size images and solve low accuracy problems by tuning parameters.

Second model implemented is Transfer learning-based CNN. Classification of images has further augmented in the field of computer vision with the dawn of transfer learning. To train a model on a huge dataset demands huge computational resources and adds a lot of cost to learning. Transfer learning allows to reduce the cost of learning and also help avoid reinventing the wheel. This paper also demonstrates image classification using pretrained deep neural network model Resnet. After obtaining the convolutional base model, a new deep neural network model is built on top of it for image classification based on a fully connected network. This classifier will use features extracted from the convolutional base model.

This paper also aims to introduce a deep learning technique based on the combination of a convolutional neural network (CNN) and long short-term memory (LSTM). LSTMs have the capacity to selectively remember patterns for a long duration of time and CNNs are able to extract the important features out of it. This LSTM-CNN layered structure, when used for image classification, has an edge over conventional CNN classifiers.

To evaluate the effectiveness of models, performance analyses are conducted.

## ALGORITHMS:

- *Transfer Learning Using Resnet*
- *Long Short Term Memory*
- *Attention Learning Using Resnet*

## Dataset:

The dataset contains augmented images of various festivals prevalent among different parts of the world. We have collected images of 21 festivals which are celebrated throughout the year around the world.

Google drive link to dataset:
https://drive.google.com/drive/folders/1yiIkUHhQm6vShC2FXhlA-uGlzjRYgelh?usp=sharing

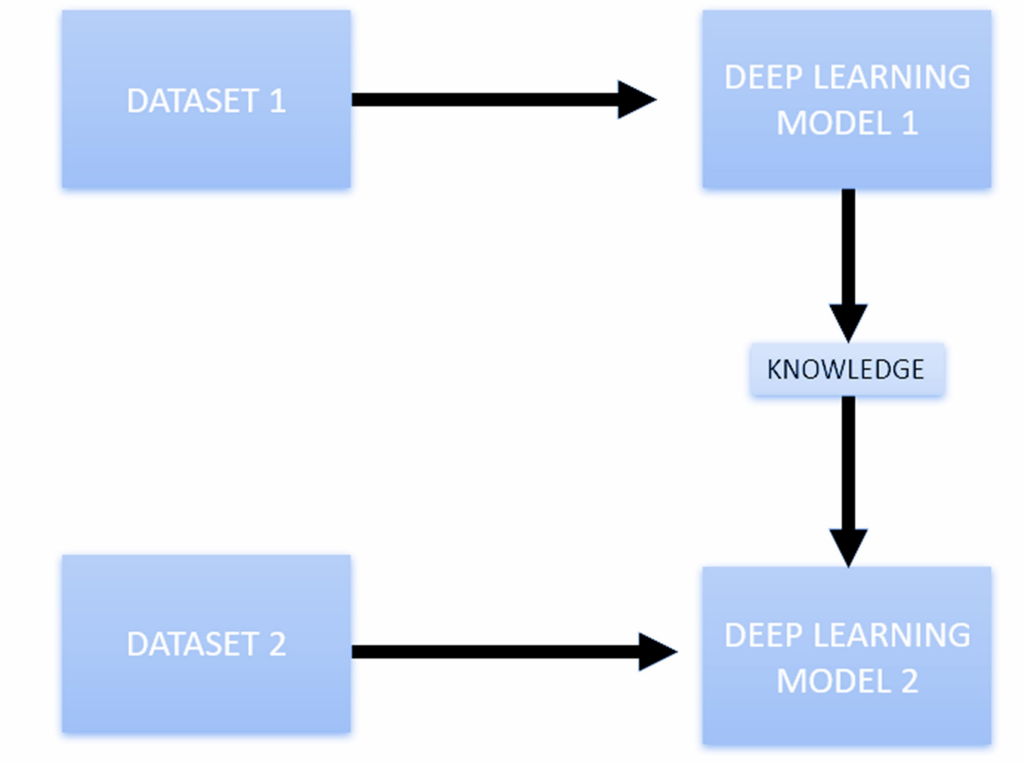
## TRANSFER LEARNING USING RESNET:

- **Methodology:**

  The Transfer Learning Algorithm is used for saving lots of time and in saving the usage of resources.

  This algorithm works by using the model obtained by training on a different dataset and using it to obtain the predicted and expected results from a different dataset which is in hand.

  In this scenario, we use the Resnet50 model which is trained with the ImageNet dataset. This model is then trained with the added output layers where the actual learning for our dataset (Festival Image Classification) takes place. In this way the Pre-trained Resnet50 model along with the newly added output Dense layers serves as Transfer Learning and allows us to reduce massive time and space complexity by using what other state-of-the-art models have learnt.




The steps involved in Transfer Learning are:

- Step 1: Import all the required libraries
- Step 2: Import your dataset
- Step 3: Split Your Data
- Step 4: Visualize your data
- Step 5: Import your Pre-trained Model
- Step 6: Model Evaluation
- Step 7: Model Inference

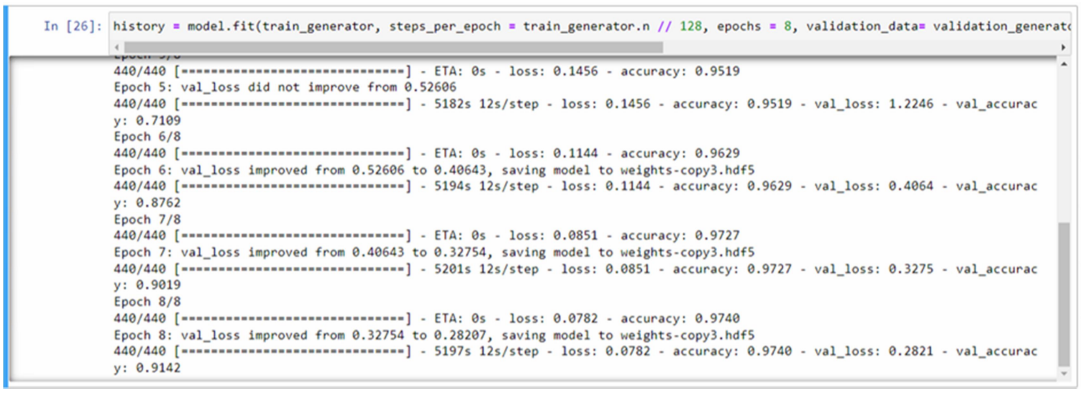
- **Result:**

  The execution of the Transfer Learning algorithm with the Resnet pre-trained model on the above-mentioned Festival Image dataset provides the following outputs.

  The Variations tested in the TL algorithm are:

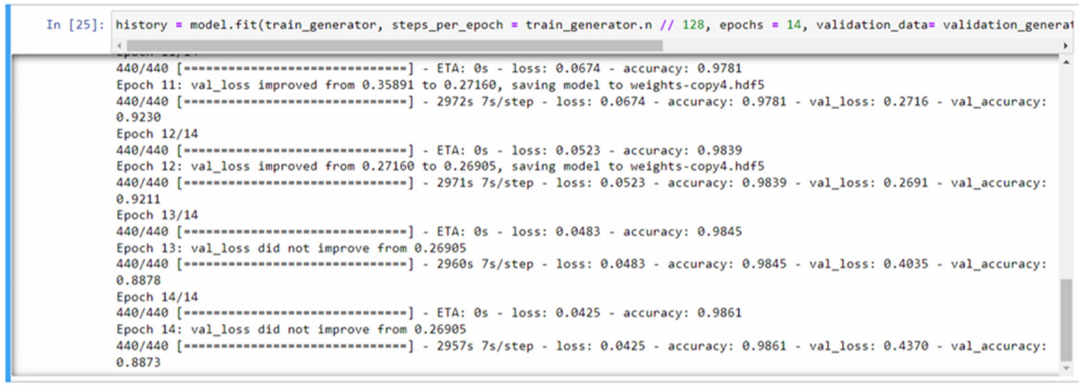
  - Number of Epochs
  - Batch size
  - Number of Blocks

- **Epoch – 8**



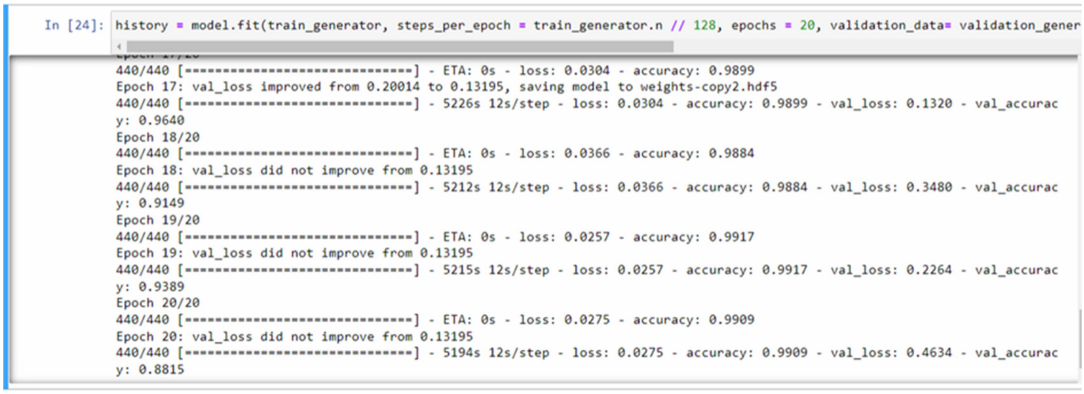
```
In [26]: history = model.fit(train_generator, steps_per_epoch = train_generator.n // 128, epochs = 8, validation_data= validation_generat
440/440 [==============================] - ETA: 0s - loss: 0.1456 - accuracy: 0.9519
Epoch 5: val_loss did not improve from 0.52606
440/440 [==============================] - 5182s 12s/step - loss: 0.1456 - accuracy: 0.9519 - val_loss: 1.2246 - val_accurac
y: 0.7109
Epoch 6/8
440/440 [==============================] - ETA: 0s - loss: 0.1144 - accuracy: 0.9629
Epoch 6: val_loss improved from 0.52606 to 0.40643, saving model to weights-copy3.hdf5
440/440 [==============================] - 5194s 12s/step - loss: 0.1144 - accuracy: 0.9629 - val_loss: 0.4064 - val_accurac
y: 0.8762
Epoch 7/8
440/440 [==============================] - ETA: 0s - loss: 0.0851 - accuracy: 0.9727
Epoch 7: val_loss improved from 0.40643 to 0.32754, saving model to weights-copy3.hdf5
440/440 [==============================] - 5201s 12s/step - loss: 0.0851 - accuracy: 0.9727 - val_loss: 0.3275 - val_accurac
y: 0.9019
Epoch 8/8
440/440 [==============================] - ETA: 0s - loss: 0.0782 - accuracy: 0.9740
Epoch 8: val_loss improved from 0.32754 to 0.28207, saving model to weights-copy3.hdf5
440/440 [==============================] - 5197s 12s/step - loss: 0.0782 - accuracy: 0.9740 - val_loss: 0.2821 - val_accurac
y: 0.9142
```

- **Epoch - 14**



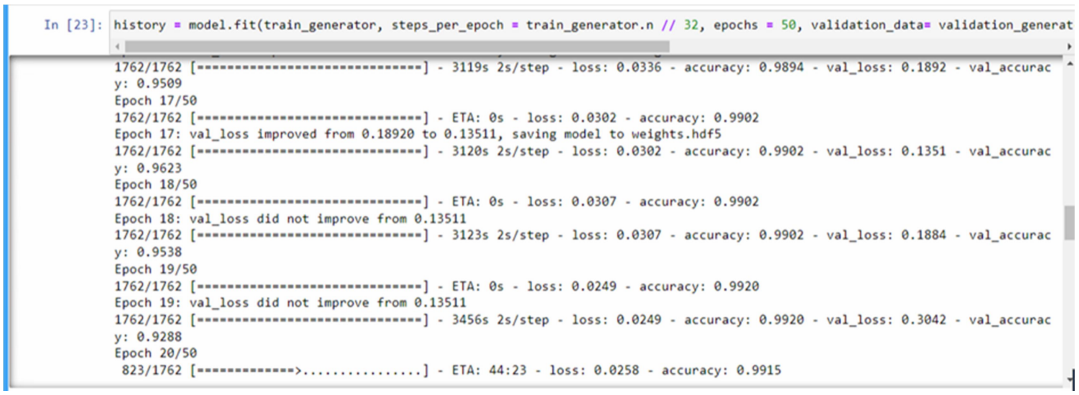
```
In [25]: history = model.fit(train_generator, steps_per_epoch = train_generator.n // 128, epochs = 14, validation_data= validation_generat
440/440 [==============================] - ETA: 0s - loss: 0.0674 - accuracy: 0.9781
Epoch 11: val_loss improved from 0.35891 to 0.27160, saving model to weights-copy4.hdf5
440/440 [==============================] - 2972s 7s/step - loss: 0.0674 - accuracy: 0.9781 - val_loss: 0.2716 - val_accuracy:
0.9230
Epoch 12/14
440/440 [==============================] - ETA: 0s - loss: 0.0523 - accuracy: 0.9839
Epoch 12: val_loss improved from 0.27160 to 0.26905, saving model to weights-copy4.hdf5
440/440 [==============================] - 2971s 7s/step - loss: 0.0523 - accuracy: 0.9839 - val_loss: 0.2691 - val_accuracy:
0.9211
Epoch 13/14
440/440 [==============================] - ETA: 0s - loss: 0.0483 - accuracy: 0.9845
Epoch 13: val_loss did not improve from 0.26905
440/440 [==============================] - 2960s 7s/step - loss: 0.0483 - accuracy: 0.9845 - val_loss: 0.4035 - val_accuracy:
0.8878
Epoch 14/14
440/440 [==============================] - ETA: 0s - loss: 0.0425 - accuracy: 0.9861
Epoch 14: val_loss did not improve from 0.26905
440/440 [==============================] - 2957s 7s/step - loss: 0.0425 - accuracy: 0.9861 - val_loss: 0.4370 - val_accuracy:
0.8873
```

- **Epoch - 20**



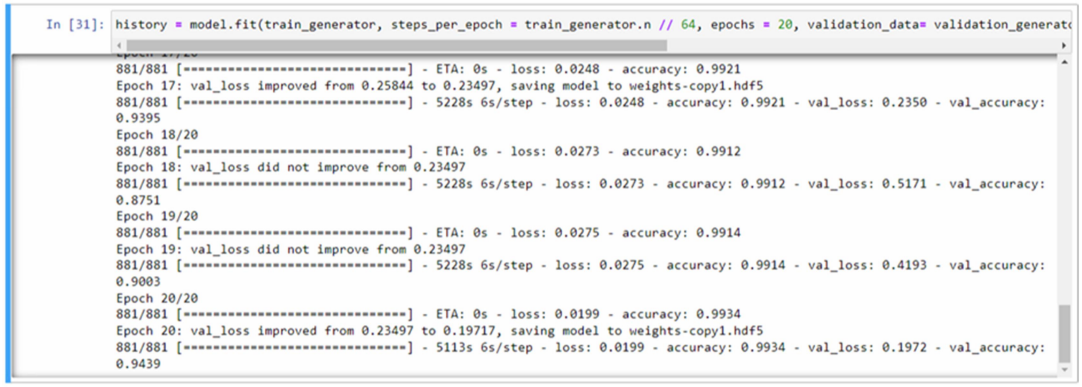
```
In [24]: history = model.fit(train_generator, steps_per_epoch = train_generator.n // 128, epochs = 20, validation_data= validation_gener
440/440 [==============================] - ETA: 0s - loss: 0.0304 - accuracy: 0.9899
Epoch 17: val_loss improved from 0.20014 to 0.13195, saving model to weights-copy2.hdf5
440/440 [==============================] - 5226s 12s/step - loss: 0.0304 - accuracy: 0.9899 - val_loss: 0.1320 - val_accurac
y: 0.9640
Epoch 18/20
440/440 [==============================] - ETA: 0s - loss: 0.0366 - accuracy: 0.9884
Epoch 18: val_loss did not improve from 0.13195
440/440 [==============================] - 5212s 12s/step - loss: 0.0366 - accuracy: 0.9884 - val_loss: 0.3480 - val_accurac
y: 0.9149
Epoch 19/20
440/440 [==============================] - ETA: 0s - loss: 0.0257 - accuracy: 0.9917
Epoch 19: val_loss did not improve from 0.13195
440/440 [==============================] - 5215s 12s/step - loss: 0.0257 - accuracy: 0.9917 - val_loss: 0.2264 - val_accurac
y: 0.9389
Epoch 20/20
440/440 [==============================] - ETA: 0s - loss: 0.0275 - accuracy: 0.9909
Epoch 20: val_loss did not improve from 0.13195
440/440 [==============================] - 5194s 12s/step - loss: 0.0275 - accuracy: 0.9909 - val_loss: 0.4634 - val_accurac
y: 0.8815
```

- **Batch size – 32**



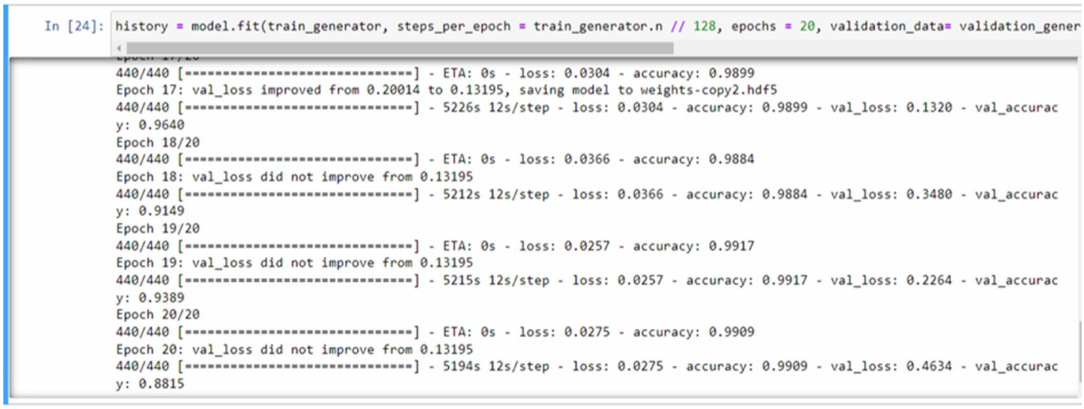
```
In [23]: history = model.fit(train_generator, steps_per_epoch = train_generator.n // 32, epochs = 50, validation_data= validation_generat
1762/1762 [==============================] - 3119s 2s/step - loss: 0.0336 - accuracy: 0.9894 - val_loss: 0.1892 - val_accurac
y: 0.9509
Epoch 17/50
1762/1762 [==============================] - ETA: 0s - loss: 0.0302 - accuracy: 0.9902
Epoch 17: val_loss improved from 0.18920 to 0.13511, saving model to weights.hdf5
1762/1762 [==============================] - 3120s 2s/step - loss: 0.0302 - accuracy: 0.9902 - val_loss: 0.1351 - val_accurac
y: 0.9623
Epoch 18/50
1762/1762 [==============================] - ETA: 0s - loss: 0.0307 - accuracy: 0.9902
Epoch 18: val_loss did not improve from 0.13511
1762/1762 [==============================] - 3123s 2s/step - loss: 0.0307 - accuracy: 0.9902 - val_loss: 0.1884 - val_accurac
y: 0.9538
Epoch 19/50
1762/1762 [==============================] - ETA: 0s - loss: 0.0249 - accuracy: 0.9920
Epoch 19: val_loss did not improve from 0.13511
1762/1762 [==============================] - 3456s 2s/step - loss: 0.0249 - accuracy: 0.9920 - val_loss: 0.3042 - val_accurac
y: 0.9288
Epoch 20/50
 823/1762 [============>.............] - ETA: 44:23 - loss: 0.0258 - accuracy: 0.9915
```

- **Batch size 64**



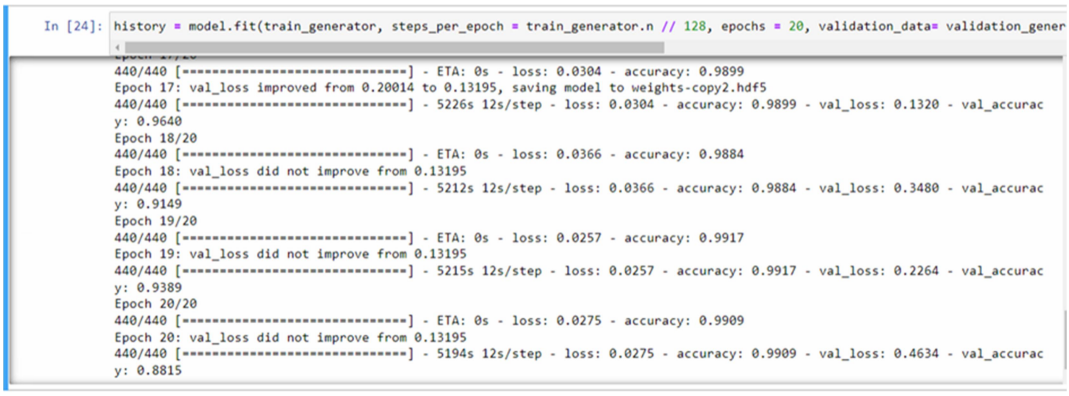
```
In [31]: history = model.fit(train_generator, steps_per_epoch = train_generator.n // 64, epochs = 20, validation_data= validation_generat
881/881 [==============================] - ETA: 0s - loss: 0.0248 - accuracy: 0.9921
Epoch 17: val_loss improved from 0.25844 to 0.23497, saving model to weights-copy1.hdf5
881/881 [==============================] - 5228s 6s/step - loss: 0.0248 - accuracy: 0.9921 - val_loss: 0.2350 - val_accuracy:
0.9395
Epoch 18/20
881/881 [==============================] - ETA: 0s - loss: 0.0273 - accuracy: 0.9912
Epoch 18: val_loss did not improve from 0.23497
881/881 [==============================] - 5228s 6s/step - loss: 0.0273 - accuracy: 0.9912 - val_loss: 0.5171 - val_accuracy:
0.8751
Epoch 19/20
881/881 [==============================] - ETA: 0s - loss: 0.0275 - accuracy: 0.9914
Epoch 19: val_loss did not improve from 0.23497
881/881 [==============================] - 5228s 6s/step - loss: 0.0275 - accuracy: 0.9914 - val_loss: 0.4193 - val_accuracy:
0.9003
Epoch 20/20
881/881 [==============================] - ETA: 0s - loss: 0.0199 - accuracy: 0.9934
Epoch 20: val_loss improved from 0.23497 to 0.19717, saving model to weights-copy1.hdf5
881/881 [==============================] - 5113s 6s/step - loss: 0.0199 - accuracy: 0.9934 - val_loss: 0.1972 - val_accuracy:
0.9439
```

- **Batch size 128**



```
In [24]: history = model.fit(train_generator, steps_per_epoch = train_generator.n // 128, epochs = 20, validation_data= validation_gener
440/440 [==============================] - ETA: 0s - loss: 0.0304 - accuracy: 0.9899
Epoch 17: val_loss improved from 0.20014 to 0.13195, saving model to weights-copy2.hdf5
440/440 [==============================] - 5226s 12s/step - loss: 0.0304 - accuracy: 0.9899 - val_loss: 0.1320 - val_accurac
y: 0.9640
Epoch 18/20
440/440 [==============================] - ETA: 0s - loss: 0.0366 - accuracy: 0.9884
Epoch 18: val_loss did not improve from 0.13195
440/440 [==============================] - 5212s 12s/step - loss: 0.0366 - accuracy: 0.9884 - val_loss: 0.3480 - val_accurac
y: 0.9149
Epoch 19/20
440/440 [==============================] - ETA: 0s - loss: 0.0257 - accuracy: 0.9917
Epoch 19: val_loss did not improve from 0.13195
440/440 [==============================] - 5215s 12s/step - loss: 0.0257 - accuracy: 0.9917 - val_loss: 0.2264 - val_accurac
y: 0.9389
Epoch 20/20
440/440 [==============================] - ETA: 0s - loss: 0.0275 - accuracy: 0.9909
Epoch 20: val_loss did not improve from 0.13195
440/440 [==============================] - 5194s 12s/step - loss: 0.0275 - accuracy: 0.9909 - val_loss: 0.4634 - val_accurac
y: 0.8815
```

- **Blocks 1 Conventional and 2 Identity Blocks**



```
In [24]: history = model.fit(train_generator, steps_per_epoch = train_generator.n // 128, epochs = 20, validation_data= validation_gener
440/440 [==============================] - ETA: 0s - loss: 0.0304 - accuracy: 0.9899
Epoch 17: val_loss improved from 0.20014 to 0.13195, saving model to weights-copy2.hdf5
440/440 [==============================] - 5226s 12s/step - loss: 0.0304 - accuracy: 0.9899 - val_loss: 0.1320 - val_accurac
y: 0.9640
Epoch 18/20
440/440 [==============================] - ETA: 0s - loss: 0.0366 - accuracy: 0.9884
Epoch 18: val_loss did not improve from 0.13195
440/440 [==============================] - 5212s 12s/step - loss: 0.0366 - accuracy: 0.9884 - val_loss: 0.3480 - val_accurac
y: 0.9149
Epoch 19/20
440/440 [==============================] - ETA: 0s - loss: 0.0257 - accuracy: 0.9917
Epoch 19: val_loss did not improve from 0.13195
440/440 [==============================] - 5215s 12s/step - loss: 0.0257 - accuracy: 0.9917 - val_loss: 0.2264 - val_accurac
y: 0.9389
Epoch 20/20
440/440 [==============================] - ETA: 0s - loss: 0.0275 - accuracy: 0.9909
Epoch 20: val_loss did not improve from 0.13195
440/440 [==============================] - 5194s 12s/step - loss: 0.0275 - accuracy: 0.9909 - val_loss: 0.4634 - val_accurac
y: 0.8815
```

- **Blocks 1 Conventional and 1 Identity Block**

```
history = model.fit(train_generator, steps_per_epoch = train_generator.n // 128, epochs = 8, validation_data= validation_ge
Epoch 1/8
440/440 [==============================] - ETA: 0s - loss: 1.5904 - accuracy: 0.5560
Epoch 1: val_loss improved from inf to 2.37366, saving model to weights.hdf5
440/440 [==============================] - 4640s 11s/step - loss: 1.5904 - accuracy: 0.5560 - val_loss: 2.3737 - val_accur
acy: 0.3210
Epoch 2/8
440/440 [==============================] - ETA: 0s - loss: 0.6839 - accuracy: 0.7865
Epoch 2: val_loss improved from 2.37366 to 1.18092, saving model to weights.hdf5
440/440 [==============================] - 4520s 10s/step - loss: 0.6839 - accuracy: 0.7865 - val_loss: 1.1809 - val_accur
acy: 0.6621
Epoch 3/8
440/440 [==============================] - ETA: 0s - loss: 0.3676 - accuracy: 0.8816
Epoch 3: val_loss improved from 1.18092 to 0.78560, saving model to weights.hdf5
440/440 [==============================] - 4451s 10s/step - loss: 0.3676 - accuracy: 0.8816 - val_loss: 0.7856 - val_accur
acy: 0.7722
Epoch 4/8
440/440 [==============================] - ETA: 0s - loss: 0.2140 - accuracy: 0.9307
Epoch 4: val_loss did not improve from 0.78560
440/440 [==============================] - 29090s 66s/step - loss: 0.2140 - accuracy: 0.9307 - val_loss: 1.1148 - val_accu
racy: 0.7221
Epoch 5/8
440/440 [==============================] - ETA: 0s - loss: 0.1303 - accuracy: 0.9569
Epoch 5: val_loss improved from 0.78560 to 0.72826, saving model to weights.hdf5
440/440 [==============================] - 4524s 10s/step - loss: 0.1303 - accuracy: 0.9569 - val_loss: 0.7283 - val_accur
acy: 0.8173
Epoch 6/8
440/440 [==============================] - ETA: 0s - loss: 0.0981 - accuracy: 0.9683
Epoch 6: val_loss improved from 0.72826 to 0.42053, saving model to weights.hdf5
440/440 [==============================] - 4541s 10s/step - loss: 0.0981 - accuracy: 0.9683 - val_loss: 0.4205 - val_accur
acy: 0.8912
Epoch 7/8
440/440 [==============================] - ETA: 0s - loss: 0.0853 - accuracy: 0.9727
Epoch 7: val_loss did not improve from 0.42053
440/440 [==============================] - 4538s 10s/step - loss: 0.0853 - accuracy: 0.9727 - val_loss: 0.8326 - val_accur
acy: 0.7878
Epoch 8/8
440/440 [==============================] - ETA: 0s - loss: 0.0639 - accuracy: 0.9799
Epoch 8: val_loss improved from 0.42053 to 0.36034, saving model to weights.hdf5
440/440 [==============================] - 4539s 10s/step - loss: 0.0639 - accuracy: 0.9799 - val_loss: 0.3603 - val_accur
acy: 0.8964
```

- **Blocks 1 Conventional Block**

```
history = model.fit(train_generator, steps_per_epoch = train_generator.n // 128, epochs = 8, validation_data= validation_ge
```

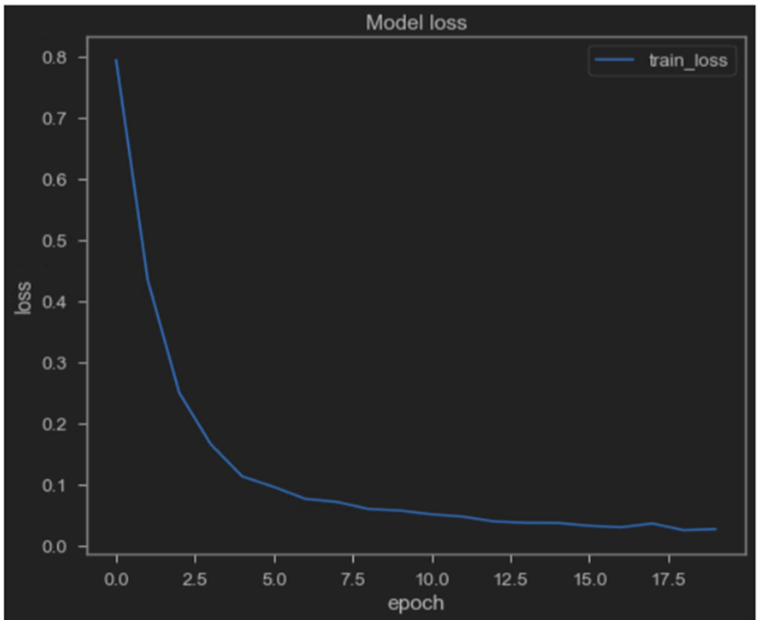
```
Epoch 1/8
440/440 [==============================] - ETA: 0s - loss: 1.3386 - accuracy: 0.6144
Epoch 1: val_loss improved from inf to 1.65990, saving model to weights-no-id.hdf5
440/440 [==============================] - 3700s 8s/step - loss: 1.3386 - accuracy: 0.6144 - val_loss: 1.6599 - val_accura
cy: 0.5133
Epoch 2/8
440/440 [==============================] - ETA: 0s - loss: 0.6015 - accuracy: 0.8106
Epoch 2: val_loss improved from 1.65990 to 1.01188, saving model to weights-no-id.hdf5
440/440 [==============================] - 3669s 8s/step - loss: 0.6015 - accuracy: 0.8106 - val_loss: 1.0119 - val_accura
cy: 0.7168
Epoch 3/8
440/440 [==============================] - ETA: 0s - loss: 0.3277 - accuracy: 0.8960
Epoch 3: val_loss did not improve from 1.01188
440/440 [==============================] - 3660s 8s/step - loss: 0.3277 - accuracy: 0.8960 - val_loss: 1.1941 - val_accura
cy: 0.6815
Epoch 4/8
440/440 [==============================] - ETA: 0s - loss: 0.1945 - accuracy: 0.9372
Epoch 4: val_loss improved from 1.01188 to 0.38125, saving model to weights-no-id.hdf5
440/440 [==============================] - 3648s 8s/step - loss: 0.1945 - accuracy: 0.9372 - val_loss: 0.3812 - val_accura
cy: 0.8818
Epoch 5/8
440/440 [==============================] - ETA: 0s - loss: 0.1269 - accuracy: 0.9593
Epoch 5: val_loss did not improve from 0.38125
440/440 [==============================] - 3654s 8s/step - loss: 0.1269 - accuracy: 0.9593 - val_loss: 0.6861 - val_accura
cy: 0.8101
Epoch 6/8
440/440 [==============================] - ETA: 0s - loss: 0.0977 - accuracy: 0.9677
Epoch 6: val_loss did not improve from 0.38125
440/440 [==============================] - 3637s 8s/step - loss: 0.0977 - accuracy: 0.9677 - val_loss: 0.5089 - val_accura
cy: 0.8553
Epoch 7/8
440/440 [==============================] - ETA: 0s - loss: 0.0765 - accuracy: 0.9757
Epoch 7: val_loss did not improve from 0.38125
440/440 [==============================] - 3670s 8s/step - loss: 0.0765 - accuracy: 0.9757 - val_loss: 0.4166 - val_accura
cy: 0.8890
Epoch 8/8
440/440 [==============================] - ETA: 0s - loss: 0.0734 - accuracy: 0.9763
Epoch 8: val_loss did not improve from 0.38125
440/440 [==============================] - 3644s 8s/step - loss: 0.0734 - accuracy: 0.9763 - val_loss: 0.4793 - val_accura
cy: 0.8740
```

- **Final Result**

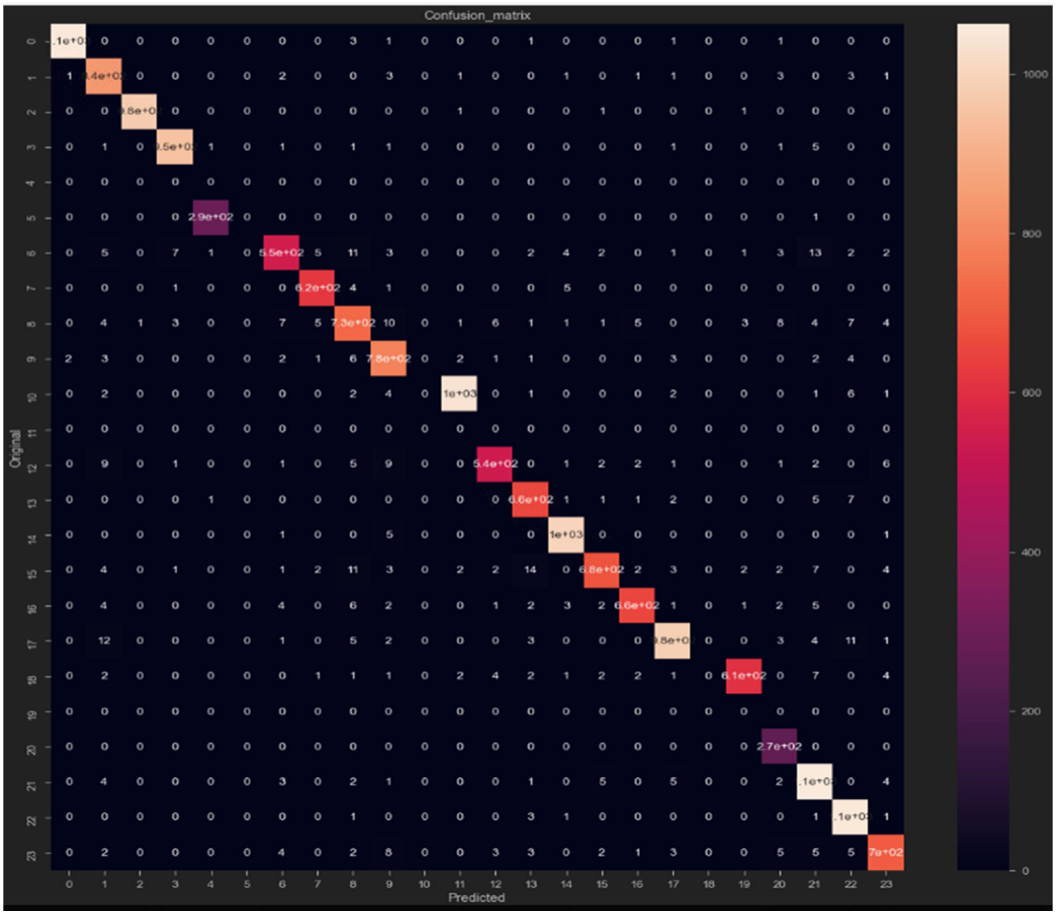
```
print(classification_report(np.asarray(original), np.asarray(prediction)))
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Agitagueda art festival | 1.00 | 0.99 | 1.00 | 1062 |
| Baisakhi | 0.94 | 0.98 | 0.96 | 859 |
| Bhandara | 1.00 | 1.00 | 1.00 | 982 |
| Bihu | 0.99 | 0.99 | 0.99 | 961 |
| Chapchar Kut | 0.00 | 0.00 | 0.00 | 0 |
| Chapchar kut | 0.00 | 0.00 | 0.00 | 294 |
| Christmas | 0.95 | 0.90 | 0.93 | 612 |
| Diwali | 0.98 | 0.98 | 0.98 | 633 |
| Ganesh Chathurthi | 0.92 | 0.91 | 0.92 | 798 |
| Holi | 0.94 | 0.97 | 0.95 | 812 |
| Jaipur Elephant Festival | 0.00 | 0.00 | 0.00 | 1059 |
| Jaipur Elephant festival | 0.00 | 0.00 | 0.00 | 0 |
| Krishna Jayanti | 0.97 | 0.93 | 0.95 | 583 |
| Kumbhmela | 0.95 | 0.97 | 0.96 | 678 |
| Lantern festival | 0.98 | 0.99 | 0.99 | 1005 |
| Maha Shivratri | 0.97 | 0.92 | 0.95 | 742 |
| Onam | 0.98 | 0.95 | 0.97 | 689 |
| Pongal | 0.98 | 0.96 | 0.97 | 1027 |
| Puli Kali | 0.00 | 0.00 | 0.00 | 637 |
| Puli kali | 0.00 | 0.00 | 0.00 | 0 |
| Raksha Bandhan | 0.90 | 1.00 | 0.95 | 268 |
| Ramzan | 0.94 | 0.98 | 0.96 | 1090 |
| Rath Yatri | 0.96 | 0.99 | 0.98 | 1059 |
| Thaipusam | 0.96 | 0.94 | 0.95 | 743 |
| accuracy | | | 0.85 | 16593 |
| macro avg | 0.72 | 0.72 | 0.72 | 16593 |
| weighted avg | 0.85 | 0.85 | 0.85 | 16593 |

- **Graph and Confusion matrix**



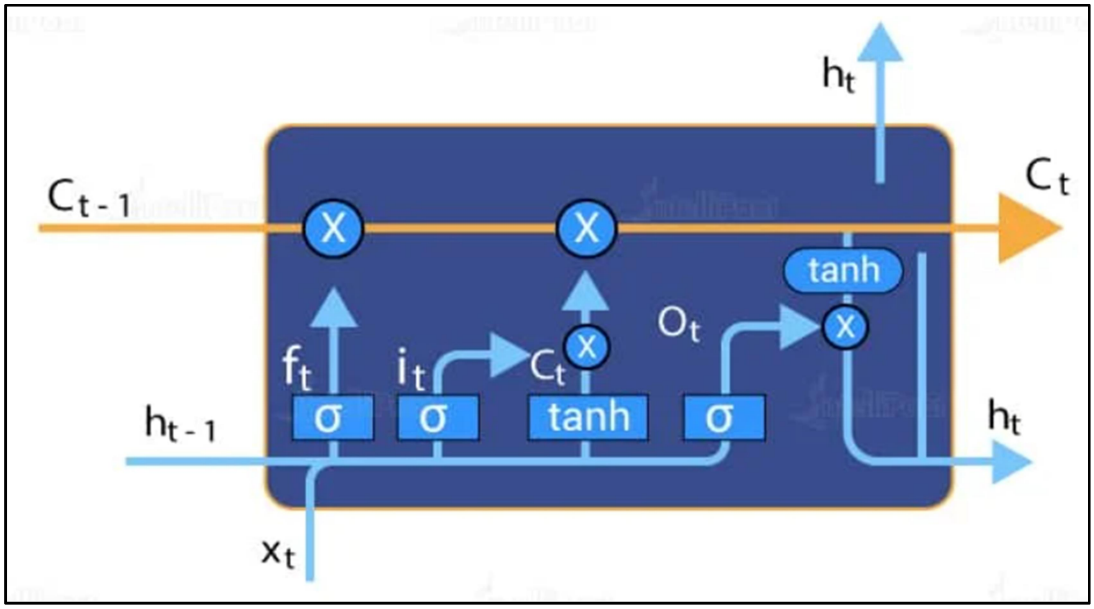

- **Confusion Matrix:**



## LONG SHORT TERM MEMORY:

- **Methodology:**

LSTM stands for long short-term memory networks, used in the field of Deep Learning. It is a variety of recurrent neural networks (RNNs) that are capable of learning long-term dependencies, especially in sequence prediction problems. LSTM has feedback connections, i.e., it is capable of processing the entire sequence of data, apart from single data points such as images.




The central role of an LSTM model is held by a memory cell known as a 'cell state' that maintains its state over time. The cell state is the horizontal line that runs through the top of the below diagram. It can be visualized as a conveyor belt through which information just flows, unchanged.

Information can be added to or removed from the cell state in LSTM and is regulated by gates. These gates optionally let the information flow in and out of the cell. It contains a pointwise multiplication operation and a sigmoid neural net layer that assist the mechanism. The sigmoid layer gives out numbers between zero and one, where zero means 'nothing should be let through' and one means 'everything should be let through'.

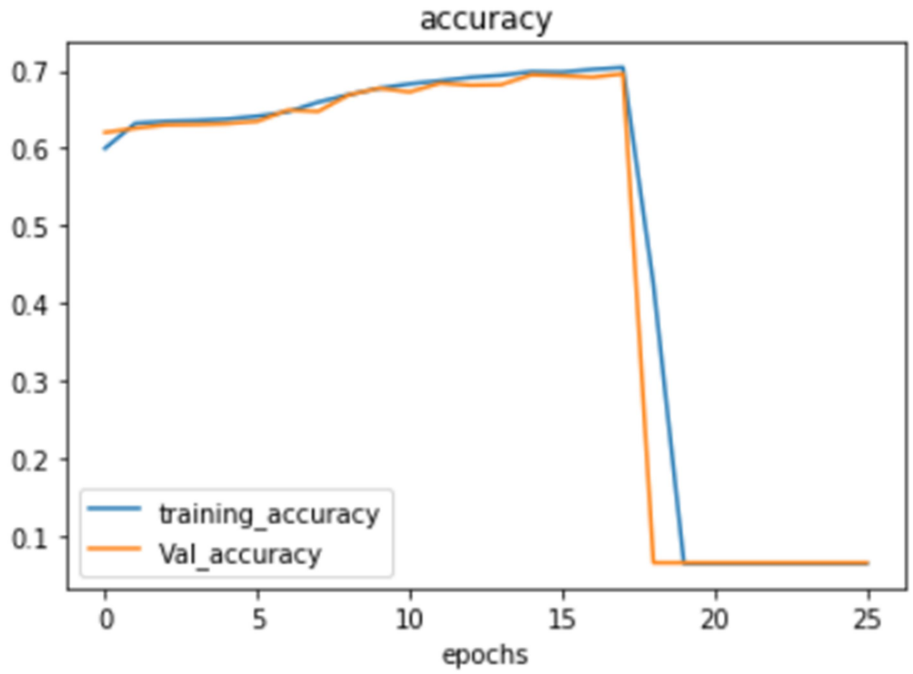
- **Result:**

  Three parameters namely Batch size, Learning rate and Number of Layers are found to be the hyperparameters which can be varied. The results of all the variations are put here along with a graph plotting Training and Testing accuracy plotted against the number of epochs.

  - **Batch size -32**

```
history = model.fit(traindata, trainlabels, epochs=100, batch_size=32, validation_data=(testdata, testlabels), verbose=2, callbac
```

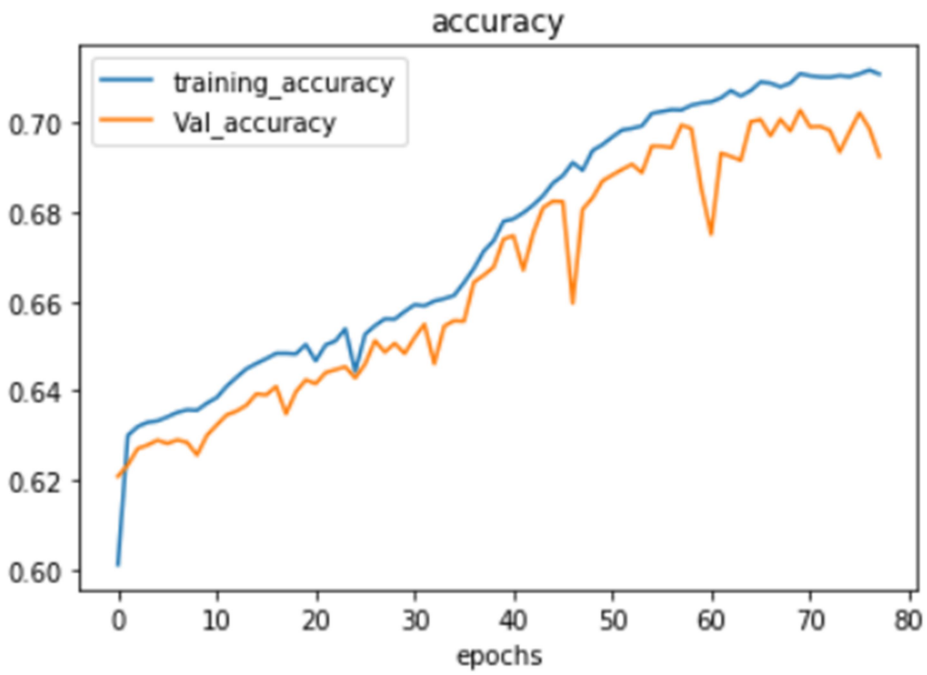
```
Epoch 22: val_accuracy did not improve from 0.69584
2075/2075 - 82s - loss: nan - accuracy: 0.0645 - val_loss: nan - val_accuracy: 0.0653 - 82s/epoch - 39ms/step
Epoch 23/100

Epoch 23: val_accuracy did not improve from 0.69584
2075/2075 - 82s - loss: nan - accuracy: 0.0645 - val_loss: nan - val_accuracy: 0.0653 - 82s/epoch - 39ms/step
Epoch 24/100

Epoch 24: val_accuracy did not improve from 0.69584
2075/2075 - 82s - loss: nan - accuracy: 0.0645 - val_loss: nan - val_accuracy: 0.0653 - 82s/epoch - 39ms/step
Epoch 25/100

Epoch 25: val_accuracy did not improve from 0.69584
2075/2075 - 82s - loss: nan - accuracy: 0.0645 - val_loss: nan - val_accuracy: 0.0653 - 82s/epoch - 39ms/step
Epoch 26/100

Epoch 26: val_accuracy did not improve from 0.69584
2075/2075 - 81s - loss: nan - accuracy: 0.0645 - val_loss: nan - val_accuracy: 0.0653 - 81s/epoch - 39ms/step
Epoch 26: early stopping
```



- **Batch size - 64**

```
history = model.fit(traindata, trainlabels, epochs=100, batch_size=64, validation_data=(testdata, testlabels), verbose=2, callbac
```

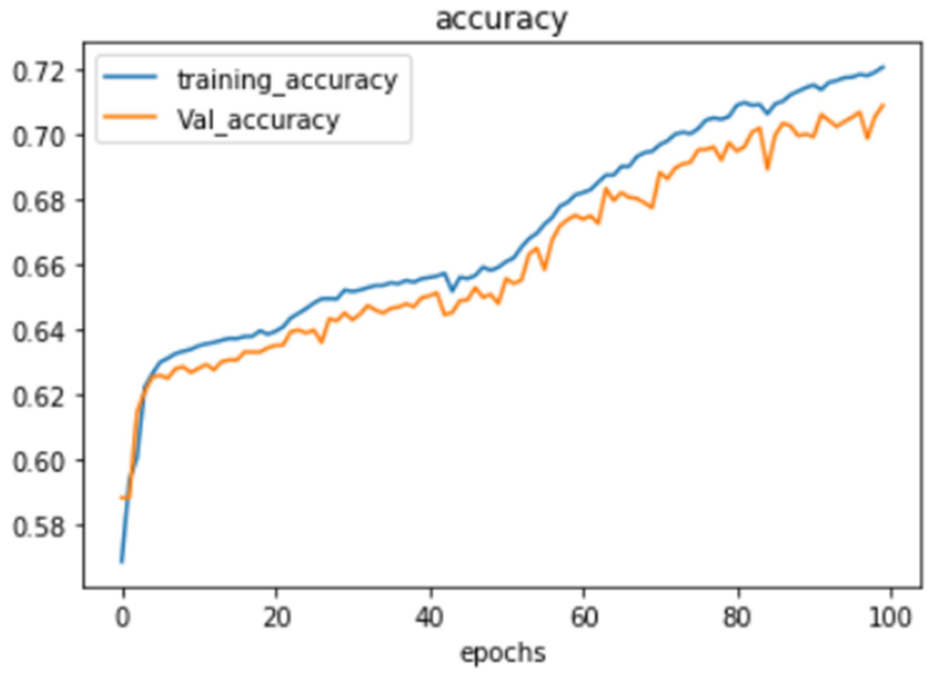
```
Epoch 74: val_accuracy did not improve from 0.70283
1038/1038 - 14s - loss: 0.9494 - accuracy: 0.7106 - val_loss: 1.0074 - val_accuracy: 0.6935 - 14s/epoch - 14ms/step
Epoch 75/100

Epoch 75: val_accuracy did not improve from 0.70283
1038/1038 - 14s - loss: 0.9499 - accuracy: 0.7103 - val_loss: 0.9909 - val_accuracy: 0.6980 - 14s/epoch - 14ms/step
Epoch 76/100

Epoch 76: val_accuracy did not improve from 0.70283
1038/1038 - 14s - loss: 0.9482 - accuracy: 0.7110 - val_loss: 0.9802 - val_accuracy: 0.7023 - 14s/epoch - 14ms/step
Epoch 77/100

Epoch 77: val_accuracy did not improve from 0.70283
1038/1038 - 14s - loss: 0.9470 - accuracy: 0.7118 - val_loss: 0.9903 - val_accuracy: 0.6987 - 14s/epoch - 14ms/step
Epoch 78/100

Epoch 78: val_accuracy did not improve from 0.70283
1038/1038 - 14s - loss: 0.9484 - accuracy: 0.7109 - val_loss: 1.0085 - val_accuracy: 0.6925 - 14s/epoch - 14ms/step
Epoch 78: early stopping
```



- **Batch size – 128**

```
history = model.fit(traindata, trainlabels, epochs=100, batch_size=128, validation_data=(testdata, testlabels), verbose=2, callb
```

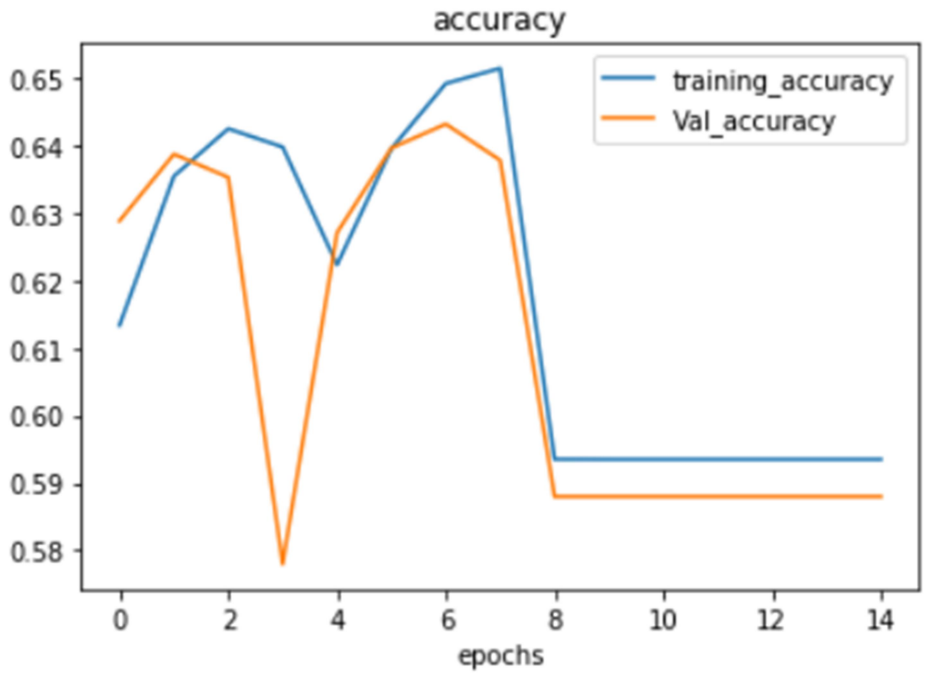
```
Epoch 96: val_accuracy did not improve from 0.70596
519/519 - 9s - loss: 0.9000 - accuracy: 0.7174 - val_loss: 0.9398 - val_accuracy: 0.7051 - 9s/epoch - 17ms/step
Epoch 97/100

Epoch 97: val_accuracy improved from 0.70596 to 0.70668, saving model to best_model.h5
519/519 - 9s - loss: 0.8968 - accuracy: 0.7182 - val_loss: 0.9369 - val_accuracy: 0.7067 - 9s/epoch - 17ms/step
Epoch 98/100

Epoch 98: val_accuracy did not improve from 0.70668
519/519 - 9s - loss: 0.8983 - accuracy: 0.7179 - val_loss: 0.9489 - val_accuracy: 0.6986 - 9s/epoch - 17ms/step
Epoch 99/100

Epoch 99: val_accuracy did not improve from 0.70668
519/519 - 9s - loss: 0.8943 - accuracy: 0.7190 - val_loss: 0.9415 - val_accuracy: 0.7054 - 9s/epoch - 17ms/step
Epoch 100/100

Epoch 100: val_accuracy improved from 0.70668 to 0.70873, saving model to best_model.h5
519/519 - 9s - loss: 0.8914 - accuracy: 0.7205 - val_loss: 0.9358 - val_accuracy: 0.7087 - 9s/epoch - 17ms/step
```



- **Learning Rate - 0.01**

```
history = model.fit(traindata, trainlabels, epochs=100, batch_size=32, validation_data=(testdata, testlabels), verbose=2, callbac
```

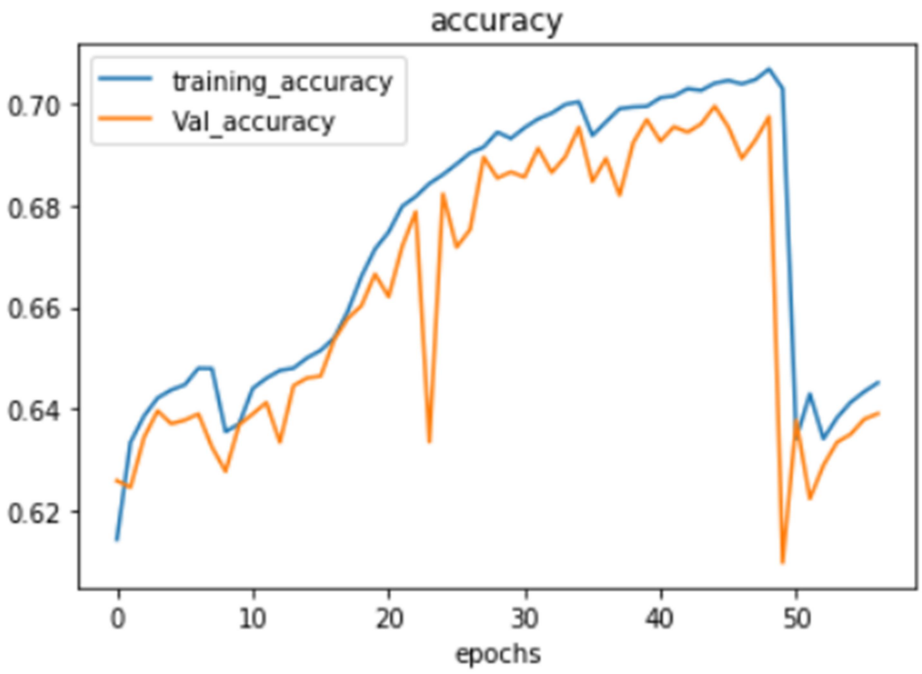
```
Epoch 11: val_accuracy did not improve from 0.64316
2075/2075 - 24s - loss: 1.5162 - accuracy: 0.5935 - val_loss: 1.5328 - val_accuracy: 0.5880 - 24s/epoch - 12ms/step
Epoch 12/100

Epoch 12: val_accuracy did not improve from 0.64316
2075/2075 - 24s - loss: 1.5162 - accuracy: 0.5935 - val_loss: 1.5299 - val_accuracy: 0.5880 - 24s/epoch - 12ms/step
Epoch 13/100

Epoch 13: val_accuracy did not improve from 0.64316
2075/2075 - 24s - loss: 1.5159 - accuracy: 0.5935 - val_loss: 1.5289 - val_accuracy: 0.5880 - 24s/epoch - 12ms/step
Epoch 14/100

Epoch 14: val_accuracy did not improve from 0.64316
2075/2075 - 25s - loss: 1.5158 - accuracy: 0.5935 - val_loss: 1.5298 - val_accuracy: 0.5880 - 25s/epoch - 12ms/step
Epoch 15/100

Epoch 15: val_accuracy did not improve from 0.64316
2075/2075 - 24s - loss: 1.5157 - accuracy: 0.5935 - val_loss: 1.5294 - val_accuracy: 0.5880 - 24s/epoch - 12ms/step
Epoch 15: early stopping
```



- **Learning Rate - 0.001**

```
history = model.fit(traindata, trainlabels, epochs=100, batch_size=32, validation_data=(testdata, testlabels), verbose=2, callbac
```

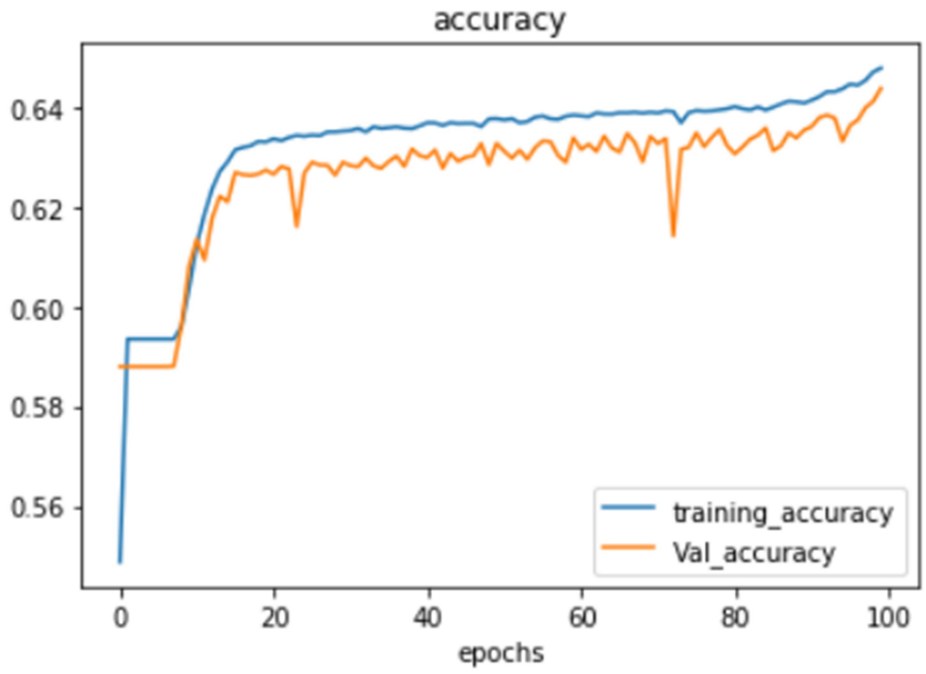
```
Epoch 53: val_accuracy did not improve from 0.69939
2075/2075 - 25s - loss: 1.2176 - accuracy: 0.6342 - val_loss: 1.2058 - val_accuracy: 0.6290 - 25s/epoch - 12ms/step
Epoch 54/100

Epoch 54: val_accuracy did not improve from 0.69939
2075/2075 - 24s - loss: 1.1822 - accuracy: 0.6383 - val_loss: 1.1888 - val_accuracy: 0.6335 - 24s/epoch - 12ms/step
Epoch 55/100

Epoch 55: val_accuracy did not improve from 0.69939
2075/2075 - 26s - loss: 1.1680 - accuracy: 0.6413 - val_loss: 1.1755 - val_accuracy: 0.6351 - 26s/epoch - 13ms/step
Epoch 56/100

Epoch 56: val_accuracy did not improve from 0.69939
2075/2075 - 24s - loss: 1.1532 - accuracy: 0.6434 - val_loss: 1.1634 - val_accuracy: 0.6380 - 24s/epoch - 12ms/step
Epoch 57/100

Epoch 57: val_accuracy did not improve from 0.69939
2075/2075 - 24s - loss: 1.1431 - accuracy: 0.6452 - val_loss: 1.1536 - val_accuracy: 0.6391 - 24s/epoch - 12ms/step
Epoch 57: early stopping
```



- **Learning Rate - 0.0001**

```
history = model.fit(traindata, trainlabels, epochs=100, batch_size=32, validation_data=(testdata, testlabels), verbose=2, callba
```

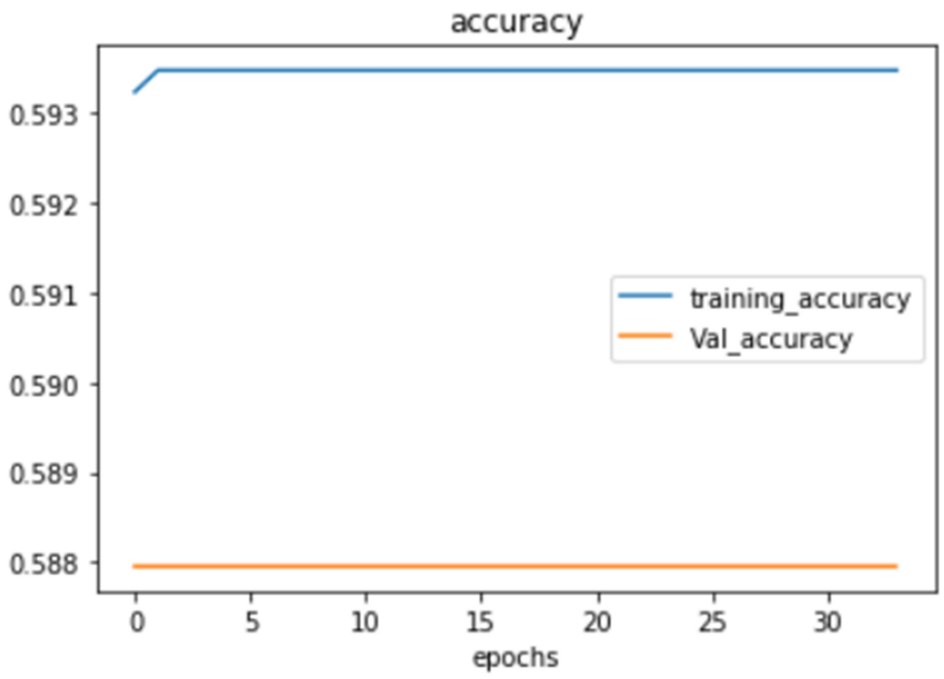
```
Epoch 96: val_accuracy did not improve from 0.63840
2075/2075 - 24s - loss: 1.1305 - accuracy: 0.6446 - val_loss: 1.1518 - val_accuracy: 0.6364 - 24s/epoch - 12ms/step
Epoch 97/100

Epoch 97: val_accuracy did not improve from 0.63840
2075/2075 - 24s - loss: 1.1300 - accuracy: 0.6444 - val_loss: 1.1633 - val_accuracy: 0.6374 - 24s/epoch - 12ms/step
Epoch 98/100

Epoch 98: val_accuracy improved from 0.63840 to 0.63991, saving model to best_model.h5
2075/2075 - 24s - loss: 1.1287 - accuracy: 0.6454 - val_loss: 1.1460 - val_accuracy: 0.6399 - 24s/epoch - 12ms/step
Epoch 99/100

Epoch 99: val_accuracy improved from 0.63991 to 0.64117, saving model to best_model.h5
2075/2075 - 24s - loss: 1.1266 - accuracy: 0.6471 - val_loss: 1.1461 - val_accuracy: 0.6412 - 24s/epoch - 12ms/step
Epoch 100/100

Epoch 100: val_accuracy improved from 0.64117 to 0.64377, saving model to best_model.h5
2075/2075 - 24s - loss: 1.1251 - accuracy: 0.6478 - val_loss: 1.1420 - val_accuracy: 0.6438 - 24s/epoch - 12ms/step
```



- **Learning Rate - 0.05**

```
history = model.fit(traindata, trainlabels, epochs=100, batch_size=32, validation_data=(testdata, testlabels), verbose=2, callba
```

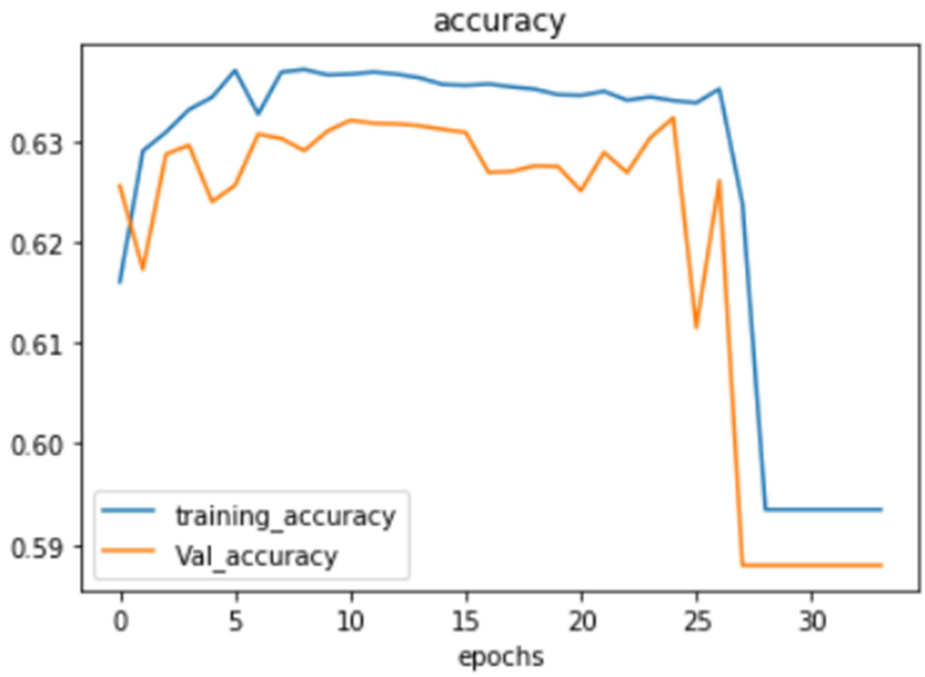
```
Epoch 30: val_accuracy did not improve from 0.58796
2075/2075 - 25s - loss: 1.5156 - accuracy: 0.5935 - val_loss: 1.5319 - val_accuracy: 0.5880 - 25s/epoch - 12ms/step
Epoch 31/100

Epoch 31: val_accuracy did not improve from 0.58796
2075/2075 - 25s - loss: 1.5153 - accuracy: 0.5935 - val_loss: 1.5314 - val_accuracy: 0.5880 - 25s/epoch - 12ms/step
Epoch 32/100

Epoch 32: val_accuracy did not improve from 0.58796
2075/2075 - 25s - loss: 1.5155 - accuracy: 0.5935 - val_loss: 1.5353 - val_accuracy: 0.5880 - 25s/epoch - 12ms/step
Epoch 33/100

Epoch 33: val_accuracy did not improve from 0.58796
2075/2075 - 25s - loss: 1.5152 - accuracy: 0.5935 - val_loss: 1.5319 - val_accuracy: 0.5880 - 25s/epoch - 12ms/step
Epoch 34/100

Epoch 34: val_accuracy did not improve from 0.58796
2075/2075 - 25s - loss: 1.5155 - accuracy: 0.5935 - val_loss: 1.5325 - val_accuracy: 0.5880 - 25s/epoch - 12ms/step
Epoch 34: early stopping
```



- **Layer n=1**

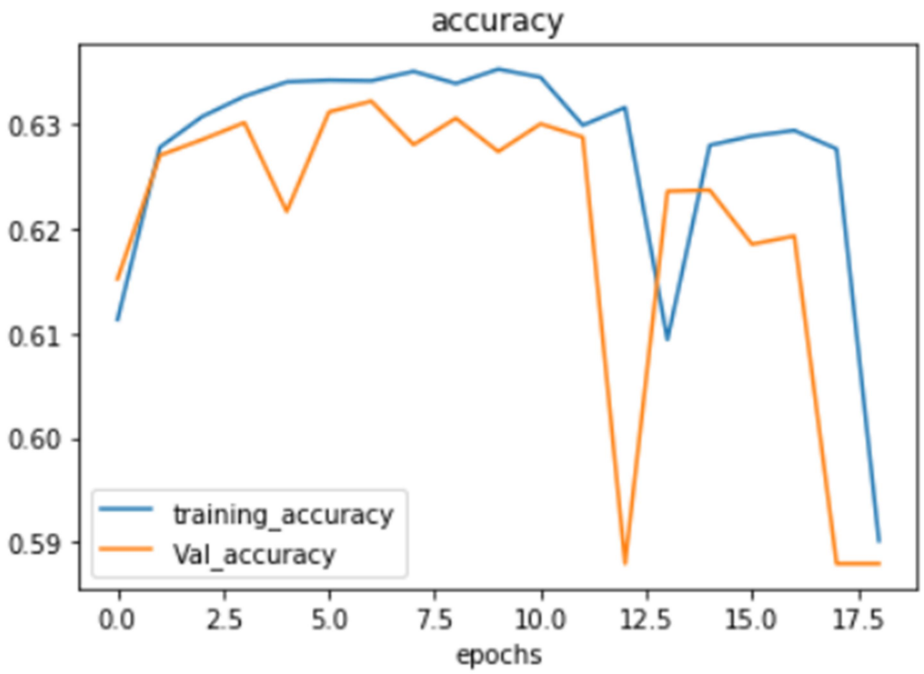
```
history = model.fit(traindata, trainlabels, epochs=70, batch_size=32, validation_data=(testdata, testlabels), verbose=2, callbacl
```

```
Epoch 30: val_accuracy did not improve from 0.63231
2075/2075 - 11s - loss: 1.5162 - accuracy: 0.5935 - val_loss: 1.5310 - val_accuracy: 0.5880 - 11s/epoch - 5ms/step
Epoch 31/70

Epoch 31: val_accuracy did not improve from 0.63231
2075/2075 - 11s - loss: 1.5154 - accuracy: 0.5935 - val_loss: 1.5345 - val_accuracy: 0.5880 - 11s/epoch - 5ms/step
Epoch 32/70

Epoch 32: val_accuracy did not improve from 0.63231
2075/2075 - 11s - loss: 1.5156 - accuracy: 0.5935 - val_loss: 1.5323 - val_accuracy: 0.5880 - 11s/epoch - 5ms/step
Epoch 33/70

Epoch 33: val_accuracy did not improve from 0.63231
2075/2075 - 11s - loss: 1.5155 - accuracy: 0.5935 - val_loss: 1.5293 - val_accuracy: 0.5880 - 11s/epoch - 5ms/step
Epoch 34/70

Epoch 34: val_accuracy did not improve from 0.63231
2075/2075 - 11s - loss: 1.5153 - accuracy: 0.5935 - val_loss: 1.5306 - val_accuracy: 0.5880 - 11s/epoch - 5ms/step
Epoch 34: early stopping
```



- **Layer n=2 Minimal Inbuilt Layers of LSTM**

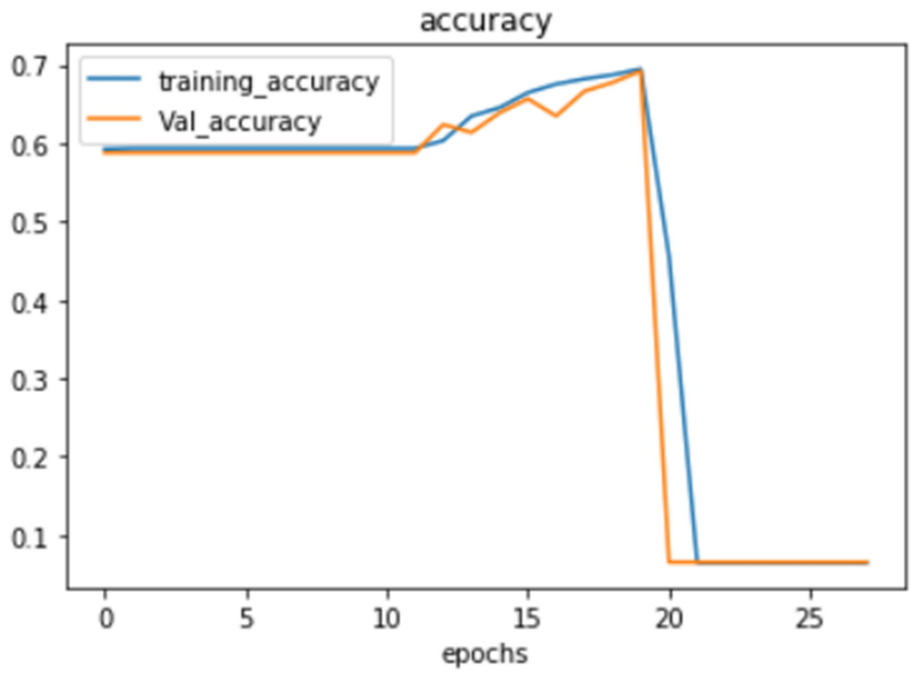
```
history = model.fit(traindata, trainlabels, epochs=100, batch_size=32, validation_data=(testdata, testlabels), verbose=2, callba
2075/2075 - 21s - loss: 1.3306 - accuracy: 0.6280 - val_loss: 1.3277 - val_accuracy: 0.6237 - 21s/epoch - 10ms/step
Epoch 16/100

Epoch 16: val_accuracy did not improve from 0.63219
2075/2075 - 22s - loss: 1.3209 - accuracy: 0.6289 - val_loss: 1.3322 - val_accuracy: 0.6185 - 22s/epoch - 10ms/step
Epoch 17/100

Epoch 17: val_accuracy did not improve from 0.63219
2075/2075 - 21s - loss: 1.3144 - accuracy: 0.6294 - val_loss: 1.3307 - val_accuracy: 0.6193 - 21s/epoch - 10ms/step
Epoch 18/100

Epoch 18: val_accuracy did not improve from 0.63219
2075/2075 - 21s - loss: 1.3158 - accuracy: 0.6276 - val_loss: 1.5562 - val_accuracy: 0.5880 - 21s/epoch - 10ms/step
Epoch 19/100

Epoch 19: val_accuracy did not improve from 0.63219
2075/2075 - 21s - loss: 593983242240.0000 - accuracy: 0.5901 - val_loss: 1.7277 - val_accuracy: 0.5880 - 21s/epoch - 10ms/ste
p
Epoch 19: early stopping
```



- **Layer n=2 Excessive Inbuilt LSTM layers**

```
history = model.fit(traindata, trainlabels, epochs=40, batch_size=32, validation_data=(testdata, testlabels), verbose=2, callbac
```

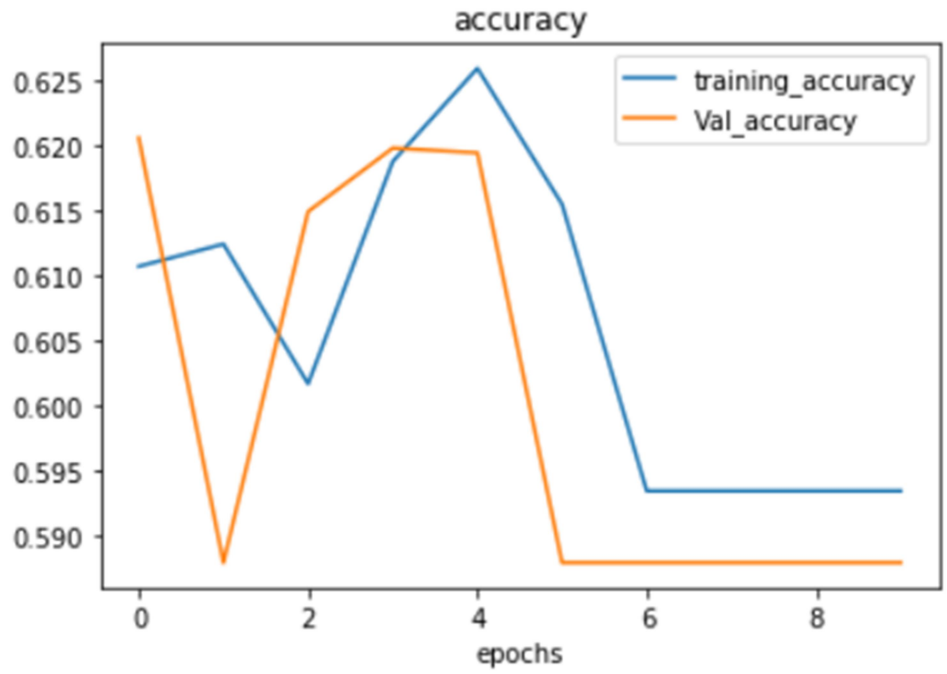
```
Epoch 24: val_accuracy did not improve from 0.69216
2075/2075 - 25s - loss: nan - accuracy: 0.0645 - val_loss: nan - val_accuracy: 0.0653 - 25s/epoch - 12ms/step
Epoch 25/40

Epoch 25: val_accuracy did not improve from 0.69216
2075/2075 - 25s - loss: nan - accuracy: 0.0645 - val_loss: nan - val_accuracy: 0.0653 - 25s/epoch - 12ms/step
Epoch 26/40

Epoch 26: val_accuracy did not improve from 0.69216
2075/2075 - 24s - loss: nan - accuracy: 0.0645 - val_loss: nan - val_accuracy: 0.0653 - 24s/epoch - 12ms/step
Epoch 27/40

Epoch 27: val_accuracy did not improve from 0.69216
2075/2075 - 24s - loss: nan - accuracy: 0.0645 - val_loss: nan - val_accuracy: 0.0653 - 24s/epoch - 12ms/step
Epoch 28/40

Epoch 28: val_accuracy did not improve from 0.69216
2075/2075 - 24s - loss: nan - accuracy: 0.0645 - val_loss: nan - val_accuracy: 0.0653 - 24s/epoch - 12ms/step
Epoch 28: early stopping
```



- **Layer n=3**

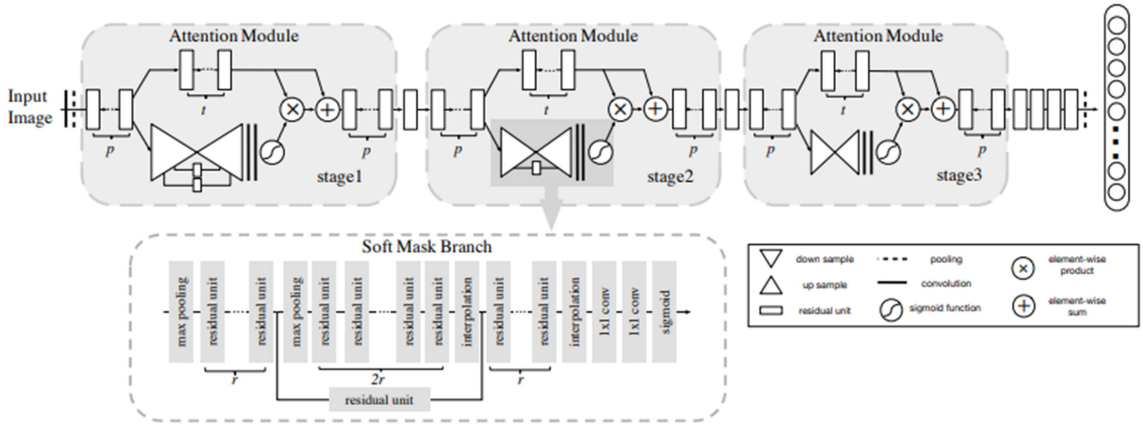
```
history = model.fit(traindata, trainlabels, epochs=100, batch_size=32, validation_data=(testdata, testlabels), verbose=2, callbac
```

```
Epoch 6: val_accuracy did not improve from 0.62062
2075/2075 - 33s - loss: 1.3979 - accuracy: 0.6155 - val_loss: 1.5310 - val_accuracy: 0.5880 - 33s/epoch - 16ms/step
Epoch 7/100

Epoch 7: val_accuracy did not improve from 0.62062
2075/2075 - 32s - loss: 1.5170 - accuracy: 0.5935 - val_loss: 1.5303 - val_accuracy: 0.5880 - 32s/epoch - 16ms/step
Epoch 8/100

Epoch 8: val_accuracy did not improve from 0.62062
2075/2075 - 32s - loss: 1.5164 - accuracy: 0.5935 - val_loss: 1.5305 - val_accuracy: 0.5880 - 32s/epoch - 16ms/step
Epoch 9/100

Epoch 9: val_accuracy did not improve from 0.62062
2075/2075 - 32s - loss: 1.5159 - accuracy: 0.5935 - val_loss: 1.5340 - val_accuracy: 0.5880 - 32s/epoch - 16ms/step
Epoch 10/100

Epoch 10: val_accuracy did not improve from 0.62062
2075/2075 - 32s - loss: 1.5159 - accuracy: 0.5935 - val_loss: 1.5331 - val_accuracy: 0.5880 - 32s/epoch - 16ms/step
Epoch 10: early stopping
```



## RESIDUAL ATTENTION NETWORK:

- **Methodology:**




Residual Attention Network is a convolutional neural network using an attention mechanism which can incorporate state-of-the-art feed forward network architecture in an end-to-end training fashion. Residual Attention Network is built by stacking Attention Modules which generate attention-aware features. The attention-aware features from different modules change adaptively as layers go deeper. Inside each Attention Module, bottom-up top-down feedforward structure is used to unfold the feedforward and feedback attention process into a single feedforward process. It is a convolutional network that adopts a mixed attention mechanism in a “very deep” structure.

- **Result:**

  The Residual Attention Network algorithm with the Attention modules have been implemented for the Image Classification of the Festival Image Dataset collected and mentioned above. The Variations performed and their output have been showcased below.

  - **Batch size 32**

```
Epoch 1/4
1555/1555 [==============================] - 9598s 6s/step - loss: 1.3881 - accuracy: 0.6673 - val_loss: 1.0661 - val_accuracy:
0.7227
Epoch 2/4
1555/1555 [==============================] - 9507s 6s/step - loss: 0.9154 - accuracy: 0.7438 - val_loss: 1.1477 - val_accuracy:
0.6456
Epoch 3/4
1555/1555 [==============================] - 9484s 6s/step - loss: 0.7054 - accuracy: 0.7912 - val_loss: 2.4883 - val_accuracy:
0.6000
Epoch 4/4
1555/1555 [==============================] - 9469s 6s/step - loss: 0.5511 - accuracy: 0.8303 - val_loss: 2.1864 - val_accuracy:
0.5418
Time taken by above cell is 634.3130537986756.
<ipython-input-6-31679718aee6>:70: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future versio
n. Please use `Model.evaluate`, which supports generators.
  val_scores = model.evaluate_generator(val_generator, verbose=0)
<ipython-input-6-31679718aee6>:71: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future versio
n. Please use `Model.evaluate`, which supports generators.
  test_scores = model.evaluate_generator(test_generator, verbose=0)
validation loss: 2.186354160308838
validation accuracy: 0.5417947173118591
Test loss: 2.2296741008758545
Test accuracy: 0.5345627665519714
```

  - **Batch size 64**

```
777/777 [==============================] - 9281s 12s/step - loss: 1.4775 - accuracy: 0.6660 - val_loss: 1.2513 - val_accuracy:
0.6596
Epoch 2/4
777/777 [==============================] - 9237s 12s/step - loss: 0.9077 - accuracy: 0.7553 - val_loss: 2.0352 - val_accuracy:
0.4540
Epoch 3/4
777/777 [==============================] - 9226s 12s/step - loss: 0.7077 - accuracy: 0.8065 - val_loss: 1.0999 - val_accuracy:
0.7023
Epoch 4/4
777/777 [==============================] - 9203s 12s/step - loss: 0.5441 - accuracy: 0.8489 - val_loss: 2.3418 - val_accuracy:
0.6214
Time taken by above cell is 615.7969138463338.
<ipython-input-6-61045c967e82>:70: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future versio
n. Please use `Model.evaluate`, which supports generators.
  val_scores = model.evaluate_generator(val_generator, verbose=0)
<ipython-input-6-61045c967e82>:71: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future versio
n. Please use `Model.evaluate`, which supports generators.
  test_scores = model.evaluate_generator(test_generator, verbose=0)
validation loss: 2.3417811393737793
validation accuracy: 0.621406614780426
Test loss: 2.4006564617156982
Test accuracy: 0.6159223914146423
```

  - **Batch size 128**

```
Epoch 1/4
388/388 [==============================] - 9083s 23s/step - loss: 1.5632 - accuracy: 0.6575 - val_loss: 17.9195 - val_accuracy:
0.0612
Epoch 2/4
388/388 [==============================] - 9053s 23s/step - loss: 0.9118 - accuracy: 0.7602 - val_loss: 1.4859 - val_accuracy:
0.6009
Epoch 3/4
388/388 [==============================] - 9060s 23s/step - loss: 0.6892 - accuracy: 0.8216 - val_loss: 1.6597 - val_accuracy:
0.5196
Epoch 4/4
388/388 [==============================] - 9040s 23s/step - loss: 0.4889 - accuracy: 0.8806 - val_loss: 2.2753 - val_accuracy:
0.6124
Time taken by above cell is 603.9361920595169.
<ipython-input-6-d11fc2f6b36b>:70: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future versio
n. Please use `Model.evaluate`, which supports generators.
  val_scores = model.evaluate_generator(val_generator, verbose=0)
<ipython-input-6-d11fc2f6b36b>:71: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future versio
n. Please use `Model.evaluate`, which supports generators.
  test_scores = model.evaluate_generator(test_generator, verbose=0)
validation loss: 2.2753312587738037
validation accuracy: 0.6124269366264343
Test loss: 2.343256711959839
Test accuracy: 0.6005544662475586
```

- **Learning Rate 0.0001**

```
777/777 [==============================] - 9281s 12s/step - loss: 1.4775 - accuracy: 0.6660 - val_loss: 1.2513 - val_accuracy:
0.6596
Epoch 2/4
777/777 [==============================] - 9237s 12s/step - loss: 0.9077 - accuracy: 0.7553 - val_loss: 2.0352 - val_accuracy:
0.4540
Epoch 3/4
777/777 [==============================] - 9226s 12s/step - loss: 0.7077 - accuracy: 0.8065 - val_loss: 1.0999 - val_accuracy:
0.7023
Epoch 4/4
777/777 [==============================] - 9203s 12s/step - loss: 0.5441 - accuracy: 0.8489 - val_loss: 2.3418 - val_accuracy:
0.6214
Time taken by above cell is 615.7969138463338.

<ipython-input-6-61045c967e82>:70: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future versio
n. Please use `Model.evaluate`, which supports generators.
  val_scores = model.evaluate_generator(val_generator, verbose=0)
<ipython-input-6-61045c967e82>:71: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future versio
n. Please use `Model.evaluate`, which supports generators.
  test_scores = model.evaluate_generator(test_generator, verbose=0)

validation loss: 2.3417811393737793
validation accuracy: 0.621406614780426
Test loss: 2.4006564617156982
Test accuracy: 0.6159223914146423
```

- **Learning Rate 0.001**

```
Epoch 1/4
777/777 [==============================] - 9458s 12s/step - loss: 79.6592 - accuracy: 0.5414 - val_loss: 1.5538 - val_accu
racy: 0.5871
Epoch 2/4
777/777 [==============================] - 9355s 12s/step - loss: 1.3769 - accuracy: 0.6381 - val_loss: 1.3384 - val_accur
acy: 0.6614
Epoch 3/4
777/777 [==============================] - 9230s 12s/step - loss: 1.1726 - accuracy: 0.6771 - val_loss: 1.2815 - val_accur
acy: 0.6787
Epoch 4/4
777/777 [==============================] - 9226s 12s/step - loss: 1.0144 - accuracy: 0.7158 - val_loss: 1.0175 - val_accur
acy: 0.7191
Time taken by above cell is 621.1546146472296.

<ipython-input-6-a01a02508139>:70: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future v
ersion. Please use `Model.evaluate`, which supports generators.
  val_scores = model.evaluate_generator(val_generator, verbose=0)
<ipython-input-6-a01a02508139>:71: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future v
ersion. Please use `Model.evaluate`, which supports generators.
  test_scores = model.evaluate_generator(test_generator, verbose=0)

validation loss: 1.0175334215164185
validation accuracy: 0.7191696821212769
Test loss: 1.0434980392456055
Test accuracy: 0.7049961090087891
```

- **Learning Rate 0.005**

```
Epoch 1/3
777/777 [==============================] - 9322s 12s/step - loss: 11537687248896.0000 - accuracy: 0.3033 - val_loss: 71625
63072.0000 - val_accuracy: 0.4476
Epoch 2/3
777/777 [==============================] - 9320s 12s/step - loss: 4295102976.0000 - accuracy: 0.3867 - val_loss: 307312665
6.0000 - val_accuracy: 0.5796
Epoch 3/3
777/777 [==============================] - 9325s 12s/step - loss: 20750086144.0000 - accuracy: 0.3844 - val_loss: 67592376
32.0000 - val_accuracy: 0.5623

Time taken by above cell is 619.9312503576278.

<ipython-input-6-60bb35c68f9e>:70: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future v
ersion. Please use `Model.evaluate`, which supports generators.
  val_scores = model.evaluate_generator(val_generator, verbose=0)
<ipython-input-6-60bb35c68f9e>:71: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future v
ersion. Please use `Model.evaluate`, which supports generators.
  test_scores = model.evaluate_generator(test_generator, verbose=0)

validation loss: 1981361664.0
validation accuracy: 0.57960688771009445
Test loss: 1765053056.0
Test accuracy: 0.38448235837221146
```

- **Dropout 0.6**

```
Epoch 1/4
777/777 [==============================] - 9458s 12s/step - loss: 79.6592 - accuracy: 0.5414 - val_loss: 1.5538 - val_accu
racy: 0.5871
Epoch 2/4
777/777 [==============================] - 9355s 12s/step - loss: 1.3769 - accuracy: 0.6381 - val_loss: 1.3384 - val_accur
acy: 0.6614
Epoch 3/4
777/777 [==============================] - 9230s 12s/step - loss: 1.1726 - accuracy: 0.6771 - val_loss: 1.2815 - val_accur
acy: 0.6787
Epoch 4/4
777/777 [==============================] - 9226s 12s/step - loss: 1.0144 - accuracy: 0.7158 - val_loss: 1.0175 - val_accur
acy: 0.7191
Time taken by above cell is 621.1546146472296.

<ipython-input-6-a01a02508139>:70: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future v
ersion. Please use `Model.evaluate`, which supports generators.
  val_scores = model.evaluate_generator(val_generator, verbose=0)
<ipython-input-6-a01a02508139>:71: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future v
ersion. Please use `Model.evaluate`, which supports generators.
  test_scores = model.evaluate_generator(test_generator, verbose=0)

validation loss: 1.0175334215164185
validation accuracy: 0.7191696821212769
Test loss: 1.0434980392456055
Test accuracy: 0.7049961090087891
```

- **Dropout 0.4**

```
Epoch 1/4
388/388 [==============================] - 9083s 23s/step - loss: 1.5632 - accuracy: 0.6575 - val_loss: 17.9195 - val_accuracy:
0.0612
Epoch 2/4
388/388 [==============================] - 9053s 23s/step - loss: 0.9118 - accuracy: 0.7602 - val_loss: 1.4859 - val_accuracy:
0.6009
Epoch 3/4
388/388 [==============================] - 9060s 23s/step - loss: 0.6892 - accuracy: 0.8216 - val_loss: 1.6597 - val_accuracy:
0.5196
Epoch 4/4
388/388 [==============================] - 9040s 23s/step - loss: 0.4889 - accuracy: 0.8806 - val_loss: 2.2753 - val_accuracy:
0.6124
Time taken by above cell is 603.9361920595169.

<ipython-input-6-d11fc2f6b36b>:70: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future versio
n. Please use `Model.evaluate`, which supports generators.
  val_scores = model.evaluate_generator(val_generator, verbose=0)
<ipython-input-6-d11fc2f6b36b>:71: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future versio
n. Please use `Model.evaluate`, which supports generators.
  test_scores = model.evaluate_generator(test_generator, verbose=0)

validation loss: 2.2753312587738037
validation accuracy: 0.6124269366264343
Test loss: 2.343256711959839
Test accuracy: 0.6005544662475586
```

- **Dropout 0.2**

```
Epoch 1/3
777/777 [==============================] - 9322s 12s/step - loss: 11537687248896.0000 - accuracy: 0.3033 - val_loss: 71625
63072.0000 - val_accuracy: 0.4476
Epoch 2/3
777/777 [==============================] - 9320s 12s/step - loss: 4295102976.0000 - accuracy: 0.3867 - val_loss: 307312665
6.0000 - val_accuracy: 0.5796
Epoch 3/3
777/777 [==============================] - 9325s 12s/step - loss: 20750086144.0000 - accuracy: 0.3844 - val_loss: 67592376
32.0000 - val_accuracy: 0.5623

Time taken by above cell is 619.9312503576278.

<ipython-input-6-60bb35c68f9e>:70: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future v
ersion. Please use `Model.evaluate`, which supports generators.
  val_scores = model.evaluate_generator(val_generator, verbose=0)
<ipython-input-6-60bb35c68f9e>:71: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future v
ersion. Please use `Model.evaluate`, which supports generators.
  test_scores = model.evaluate_generator(test_generator, verbose=0)

validation loss: 1981361664.0
validation accuracy: 0.57960688771009445
Test loss: 1765053056.0
Test accuracy: 0.38448235837221146
```

## DISCUSSION:

From the above executed set of algorithms, we observe that the accuracy for the chosen set of algorithms (Transfer Learning, Attention Learning and LSTM) compares as follows:

**Transfer Learning > Attention Learning > LSTM**

Thus, Transfer Learning (In this case, with Resnet50) proves to be the best algorithm for the case with the benefits of saving a lot of time and also reducing the usage of the computational resources.

Transfer Learning serves as the best for Image Classification using the currently generated Festival Image Dataset because, the intuition behind transfer learning for image classification is that **if a model is trained on a large and general enough dataset, this model will effectively serve as a generic model of the visual world**. In this case our Algorithm inherits a model which has been trained from ImageNet Dataset which contains a general image set that is enough to classify all the Festival Images effectively.

## REFERENCE:


- Alsabei, A., Alsayed, A., Alzahrani, M., & Al-Shareef, S. (2021). Waste Classification by Fine-Tuning Pre-trained CNN and GAN. International Journal of Computer Science & Network Security, 21(8), 65-70.
- Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., ... & Tang, X. (2017). Residual attention network for image classification. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3156-3164).
- Tatsunami, Y., & Taki, M. (2022). Sequencer: Deep LSTM for Image Classification. arXiv preprint arXiv:2205.01972.