CS512 – Computer Vision Spring 2023

Assignment 3 Report

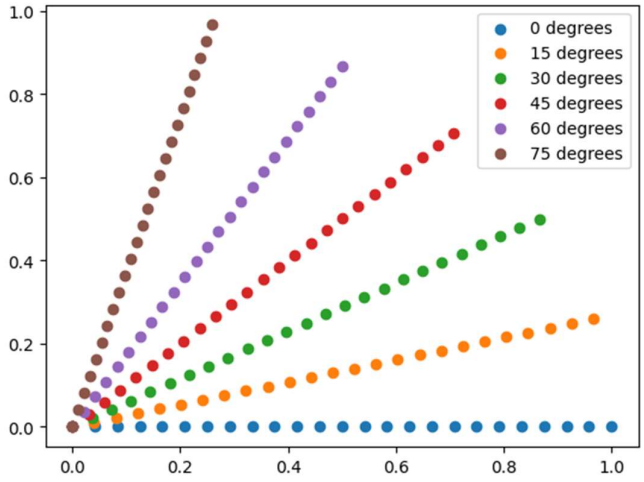
Name: Sai Manohar Vemuri

CWID: A20514848

Question 1. Line fitting and robust estimation

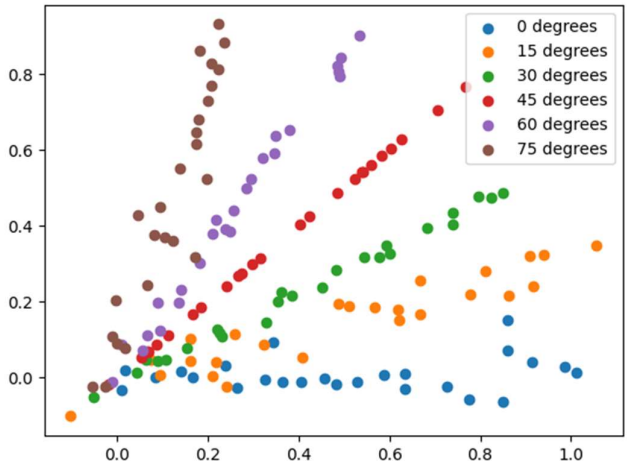
2.A

Generated 6 lines of distance 1 between 0 and 90 degree angle from the origin.



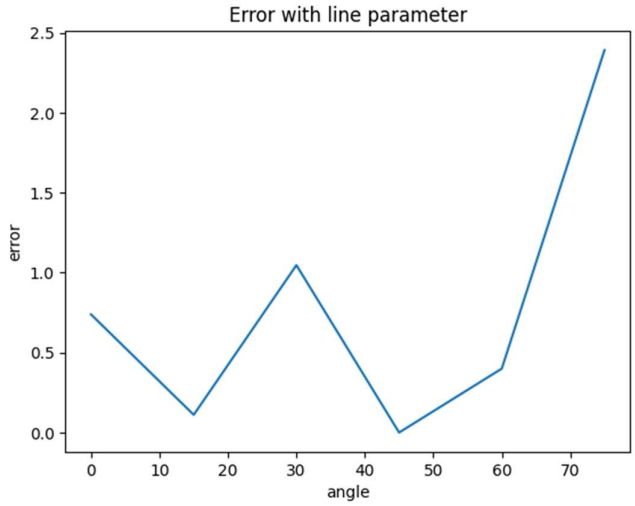
2.B

We added the gaussian noise to the above generated line with the mean zero and standard deviation of 5% of the length of the line. We randomly generated the noise values using the numpy np.random.normal() function.



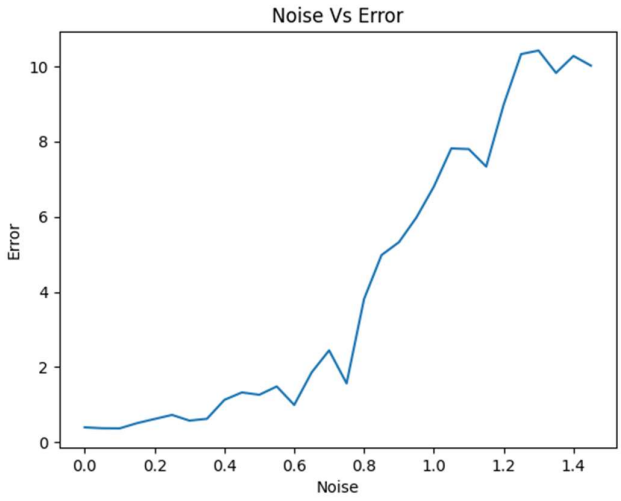
2.C

In this question, we fit the linear regression line using the noise points set. For each and every angle we are plotting the line and calculating the error.



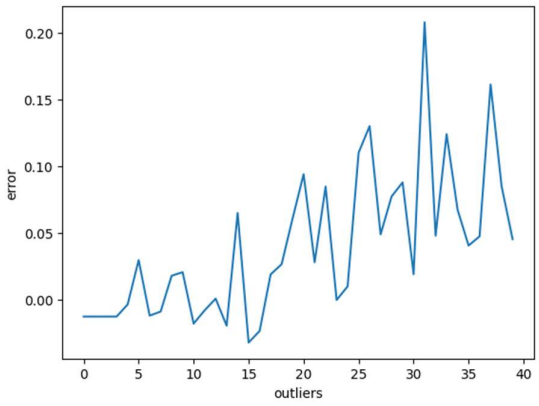
2.D

A line is considered with a known angle and distance 1. Noise is varied with the function of error and the result is plotted.



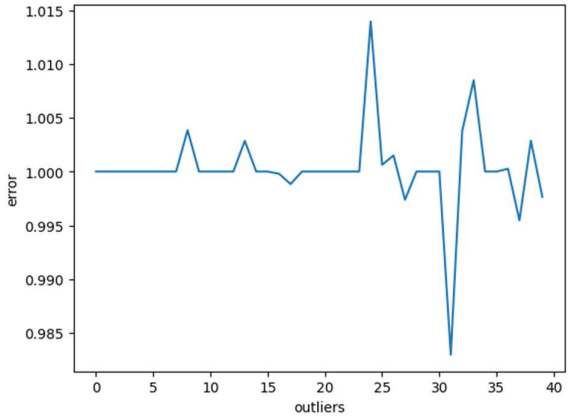
2.E

One line is chosen and we are varying the outlier percentage and its corresponding error is calculated by computing the line parameters using linear regression.



2.F

In this instead of using linear regression inbuilt function, we are using cv2.fitLine to fit the line to perform robust estimation and with the CV_DIST_HUBER, distance is used. The below graph indicates the error as a function of outlier percentage. With robust estimation we are able to reduce the outliers to some extent.



3. Image Classification 1

Dataset: All the data which we have used for the training is uploaded to google drive. This includes CatDog dataset and CIFAR – 10. The catdog dataset contains 3 folders: train, test and validation

https://drive.google.com/drive/folders/1ZCPhUiDjI-Qx_NuI41jXyuPkPw6KZW97?usp=share_link

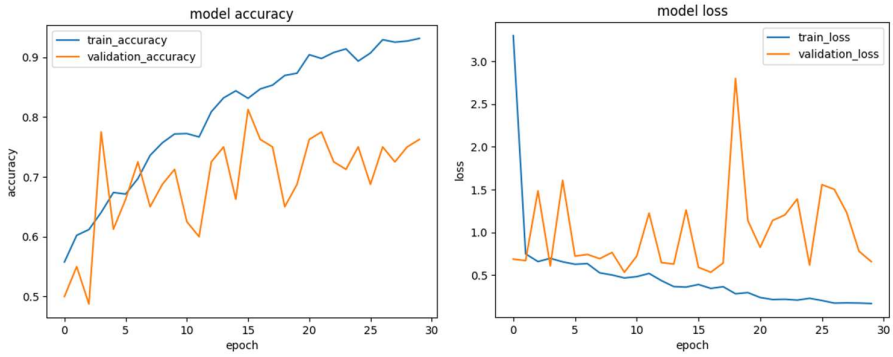
3.A

We have taken 2000 images each of cats and dogs classes and divided them for the training, validation and testing with the split ratio of 80:10:10.

3.B

In this we have created a model using 3 Convolution layers and a dense layer with 256 filters which gives the output between 0 and 1. We used Relu activation function for the hidden layers and sigmoid activation function for the output layer. Input shape is 128x128x3 image. The below image shows the model summary. We have used the Adam optimizer with the learning rate of 1e-3 and the binary cross entropy loss.

```
Model: "model_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_7 (InputLayer)         [(None, 128, 128, 3)]     0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 128, 128, 32)      896
_________________________________________________________________
batch_normalization_18 (Batc (None, 128, 128, 32)      128
_________________________________________________________________
dropout_24 (Dropout)         (None, 128, 128, 32)      0
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 64, 64, 64)        18496
_________________________________________________________________
batch_normalization_19 (Batc (None, 64, 64, 64)        256
_________________________________________________________________
dropout_25 (Dropout)         (None, 64, 64, 64)        0
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 32, 32, 128)       73856
_________________________________________________________________
batch_normalization_20 (Batc (None, 32, 32, 128)       512
_________________________________________________________________
dropout_26 (Dropout)         (None, 32, 32, 128)       0
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 16, 16, 128)       0
_________________________________________________________________
flatten_6 (Flatten)          (None, 32768)             0
_________________________________________________________________
dense_12 (Dense)             (None, 256)               8388864
_________________________________________________________________
dropout_27 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_13 (Dense)             (None, 1)                 257
=================================================================
Total params: 8,483,265
Trainable params: 8,482,817
Non-trainable params: 448
_________________________________________________________________
```

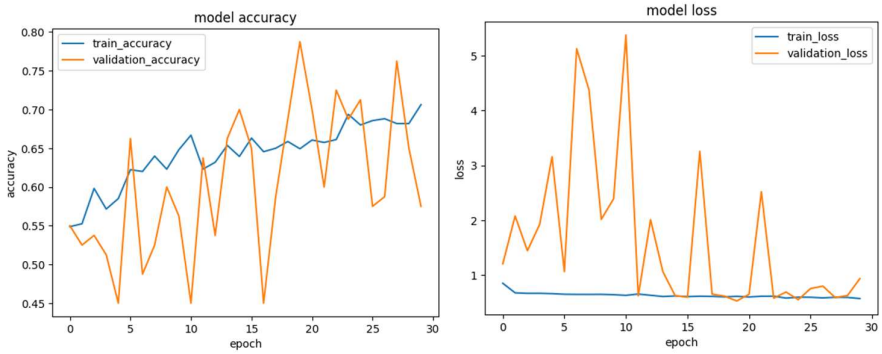


Model Accuracy and Loss

We ran for 30 epochs and the loss and accuracy were plotted in the above graph. We can see that both validation and train accuracy were increasing while loss was decreasing but still near the 1. We have used the Adam optimizer with the learning rate of 1e-3 and the binary cross entropy loss.

3.C

In this question we have used image generator and performed augmentation. And tested the model performance on the validation dataset. The below graph indicates the model performance.

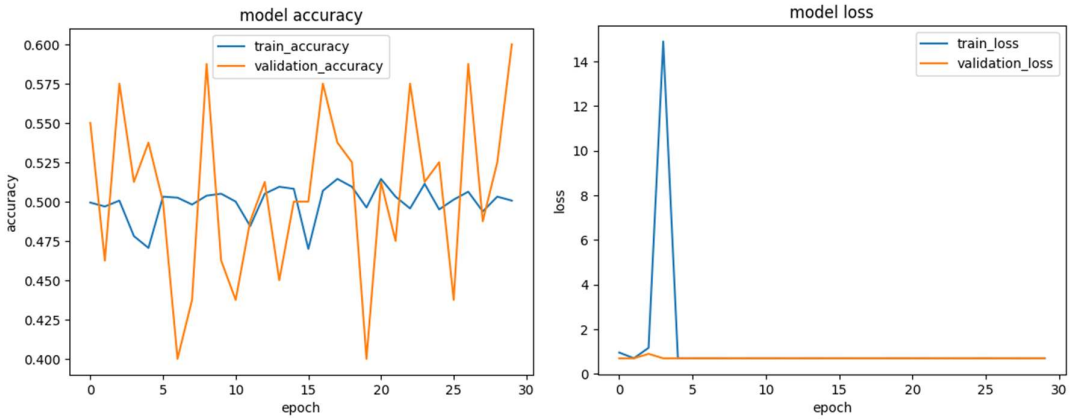


We can see the fluctuations in the performance over the 30 epochs on the validation set and loss was above 1.

3.D

In this question we have used VGG16 model with imagenet weights. And we have added the dense layer with 64 filters and a output layer with 1 filter and sigmoid activation function which give the output between 0 and 1. We have used the Adam optimizer with the learning rate of 1e-3 and the binary cross entropy loss.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense (Dense)                (None, 64)                524352
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 15,239,105
Trainable params: 15,239,105
Non-trainable params: 0
_________________________________________________________________
```



As we can see that loss is not reducing for the validation dataset and accuracy is fluctuating near 0.5 for the train dataset and for the validation dataset, we can see sudden changes in the accuracy.

4. Image Classification 2

4.A

We used the CIFAR-10 dataset and unpickled with the pickle library and performed the one hot encoding for the labels.

Question 4.a

In [161]:
```
import pickle
```

In [162]:
```
cifar_DIR=r'..\data\cifar-10-batches-py\data_batch_1'
cifar_read = open(cifar_DIR, "rb")
cifar_data = pickle.load(cifar_read,encoding='latin1')
X=cifar_data['data'].reshape(-1,3,32,32).transpose((0,2,3,1))
y=cifar_data['labels']
```

In [ ]:
```
```

In [163]:
```
from sklearn.preprocessing import OneHotEncoder
X=np.array(X)
y=np.array(y)
print(X.shape)
print(y.shape)
label_encoder = OneHotEncoder(sparse=False)
y_ = label_encoder.fit_transform(y.reshape(-1,1))
print(y_.shape)
```

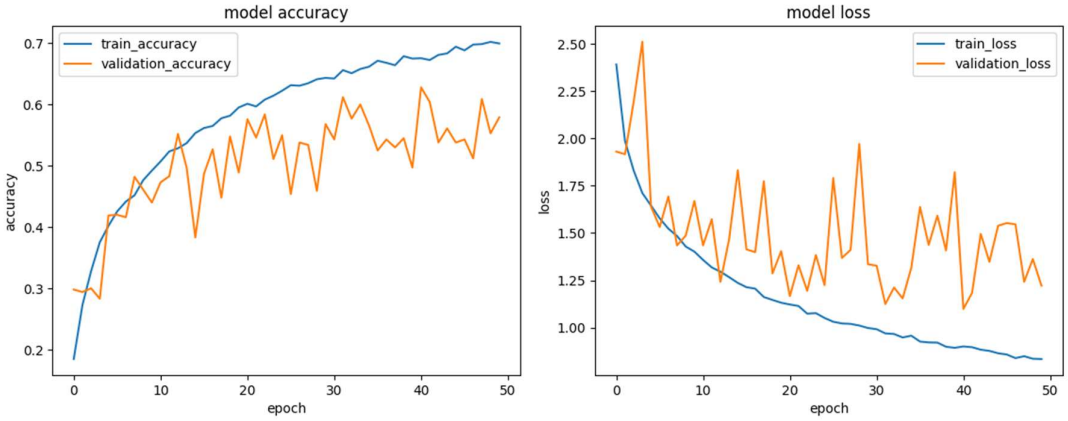
```
(10000, 32, 32, 3)
(10000,)
(10000, 10)
```

```
C:\Users\Manohar Vemuri\.conda\envs\main\lib\site-packages\sklearn\preprocessing\_encoders.py:868: FutureWarning: `sparse` was
renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless you leave `sparse` to i
ts default value.
  warnings.warn(
```

4.B

We have built a model with 3 convolution blocks and 2 dense layers. The input shape, we are passing is 32x32x3 image with a batch size of 64. The output layer consists of 10 classes and used a softmax activation function. we have used the categorical cross entropy loss and adam optimizer with the learning rate of 1e-3. Below diagram represents the summary of the model.

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 32, 32, 3)]       0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 32, 32, 64)        1792
_________________________________________________________________
batch_normalization_6 (Batch (None, 32, 32, 64)        256
_________________________________________________________________
dropout_8 (Dropout)          (None, 32, 32, 64)        0
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 16, 16, 32)        18464
_________________________________________________________________
batch_normalization_7 (Batch (None, 16, 16, 32)        128
_________________________________________________________________
dropout_9 (Dropout)          (None, 16, 16, 32)        0
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 8, 8, 32)          0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 8, 8, 16)          4624
_________________________________________________________________
batch_normalization_8 (Batch (None, 8, 8, 16)          64
_________________________________________________________________
dropout_10 (Dropout)         (None, 8, 8, 16)          0
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 4, 4, 16)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 64)                16448
_________________________________________________________________
dropout_11 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_5 (Dense)              (None, 10)                650
=================================================================
Total params: 42,426
Trainable params: 42,202
Non-trainable params: 224
_________________________________________________________________
```

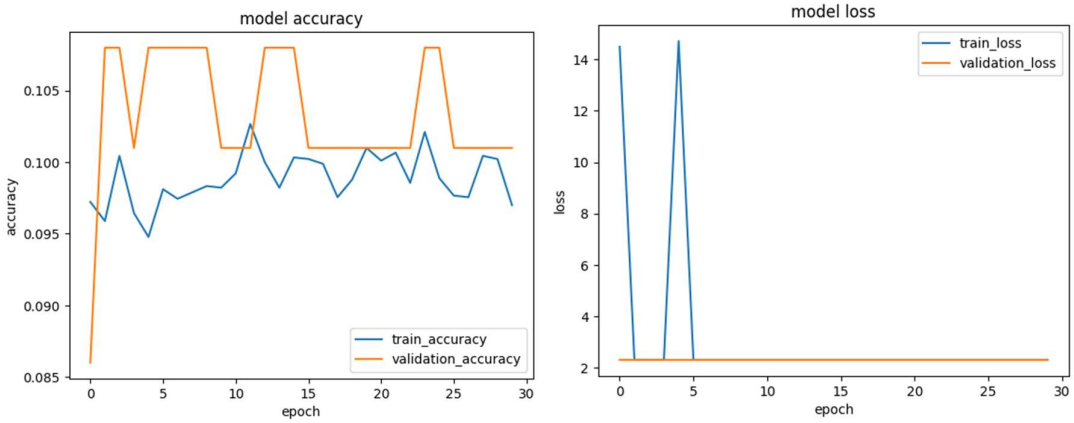


The above diagram represents the model performance over 50 eopchs. And we can see that model is converging where loss is reducing and accuracy is improving for both training and validation dataset.

4.C

In this question we have replaced the convolution blocks we have defined with the inception blocks as mentioned in the question. We have used 3 inception blocks with each consisting of 3 convolution layers and a maxpooling layer. And we have added 2 dense layers. The output layer consists of 10 nodes which results the class probabilities. The below diagram represents the model architecture.

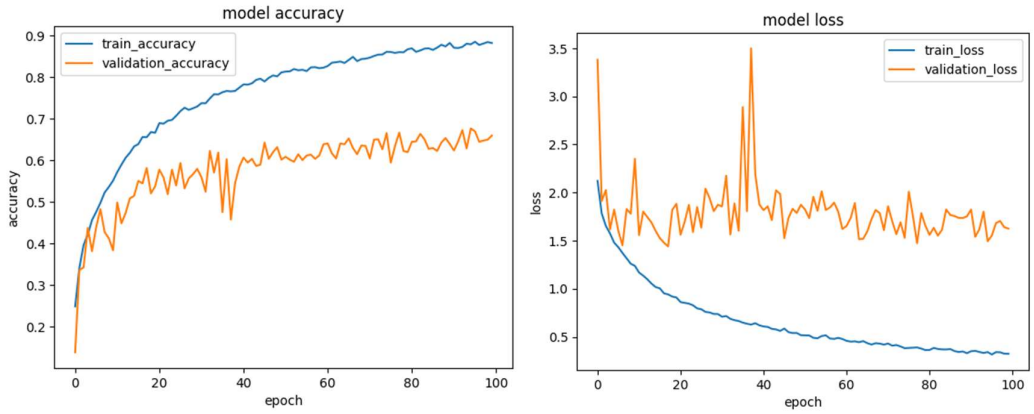
We have used adam optimizer with the learning rate of 1e-3 and categorical cross entropy loss function.



The above diagram indicates that the model is not performing well as the model loss is not decreasing and the accuracy is not crossing above 0.2.

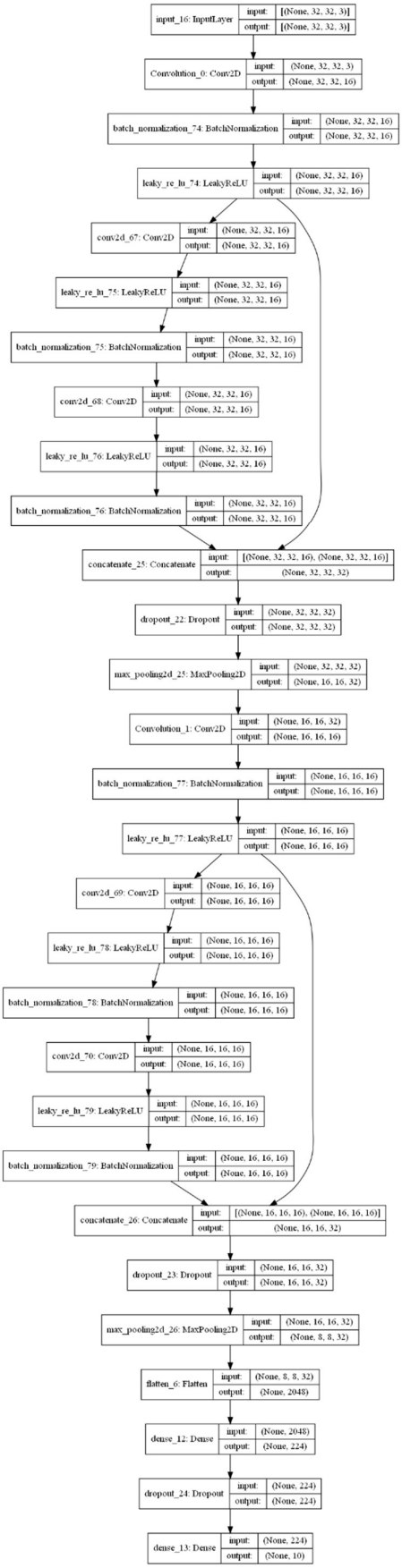
4.D

In this question we have replaced the convolution blocks we have defined with the resnet blocks as mentioned in the question. We have used 2 resnet blocks with each consisting of 2 convolution layers. And we have added 2 dense layers. The output layer consists of 10 nodes which results the class probabilities. The below diagram represents the model performance.



We ran for over 100 epochs and the above diagram indicates that the model is converging very slowly but the validation loss is not reducing as it was fluctuating near 1.5.

The below diagram represents the model architecture.


input_16: InputLayer
input: [(None, 32, 32, 3)]
output: [(None, 32, 32, 3)]
Convolution_0: Conv2D
input: (None, 32, 32, 3)
output: (None, 32, 32, 16)
batch_normalization_74: BatchNormalization
input: (None, 32, 32, 16)
output: (None, 32, 32, 16)
leaky_re_lu_74: LeakyReLU
input: (None, 32, 32, 16)
output: (None, 32, 32, 16)
conv2d_67: Conv2D
input: (None, 32, 32, 16)
output: (None, 32, 32, 16)
leaky_re_lu_75: LeakyReLU
input: (None, 32, 32, 16)
output: (None, 32, 32, 16)
batch_normalization_75: BatchNormalization
input: (None, 32, 32, 16)
output: (None, 32, 32, 16)
conv2d_68: Conv2D
input: (None, 32, 32, 16)
output: (None, 32, 32, 16)
leaky_re_lu_76: LeakyReLU
input: (None, 32, 32, 16)
output: (None, 32, 32, 16)
batch_normalization_76: BatchNormalization
input: (None, 32, 32, 16)
output: (None, 32, 32, 16)
concatenate_25: Concatenate
input: [(None, 32, 32, 16), (None, 32, 32, 16)]
output: (None, 32, 32, 32)
dropout_22: Dropout
input: (None, 32, 32, 32)
output: (None, 32, 32, 32)
max_pooling2d_25: MaxPooling2D
input: (None, 32, 32, 32)
output: (None, 16, 16, 32)
Convolution_1: Conv2D
input: (None, 16, 16, 32)
output: (None, 16, 16, 16)
batch_normalization_77: BatchNormalization
input: (None, 16, 16, 16)
output: (None, 16, 16, 16)
leaky_re_lu_77: LeakyReLU
input: (None, 16, 16, 16)
output: (None, 16, 16, 16)
conv2d_69: Conv2D
input: (None, 16, 16, 16)
output: (None, 16, 16, 16)
leaky_re_lu_78: LeakyReLU
input: (None, 16, 16, 16)
output: (None, 16, 16, 16)
batch_normalization_78: BatchNormalization
input: (None, 16, 16, 16)
output: (None, 16, 16, 16)
conv2d_70: Conv2D
input: (None, 16, 16, 16)
output: (None, 16, 16, 16)
leaky_re_lu_79: LeakyReLU
input: (None, 16, 16, 16)
output: (None, 16, 16, 16)
batch_normalization_79: BatchNormalization
input: (None, 16, 16, 16)
output: (None, 16, 16, 16)
concatenate_26: Concatenate
input: [(None, 16, 16, 16), (None, 16, 16, 16)]
output: (None, 16, 16, 32)
dropout_23: Dropout
input: (None, 16, 16, 32)
output: (None, 16, 16, 32)
max_pooling2d_26: MaxPooling2D
input: (None, 16, 16, 32)
output: (None, 8, 8, 32)
flatten_6: Flatten
input: (None, 8, 8, 32)
output: (None, 2048)
dense_12: Dense
input: (None, 2048)
output: (None, 224)
dropout_24: Dropout
input: (None, 224)
output: (None, 224)
dense_13: Dense
input: (None, 224)
output: (None, 10)