Project Report

Introduction

The project was to download the CIFAR-10 dataset and run two different algorithms on it to be accurately able to classify the image. The dataset contains a total of 60,000 images in which their are 50,000 training images and 10,000 test images of size 32x32x3. The 10 classification labels of the dataset are Airplane, Automobile, Cat, Bird, Frog, Dog, Deer, Horse, Ship and Truck. The classes in the dataset are mutually exclusive.

Methods

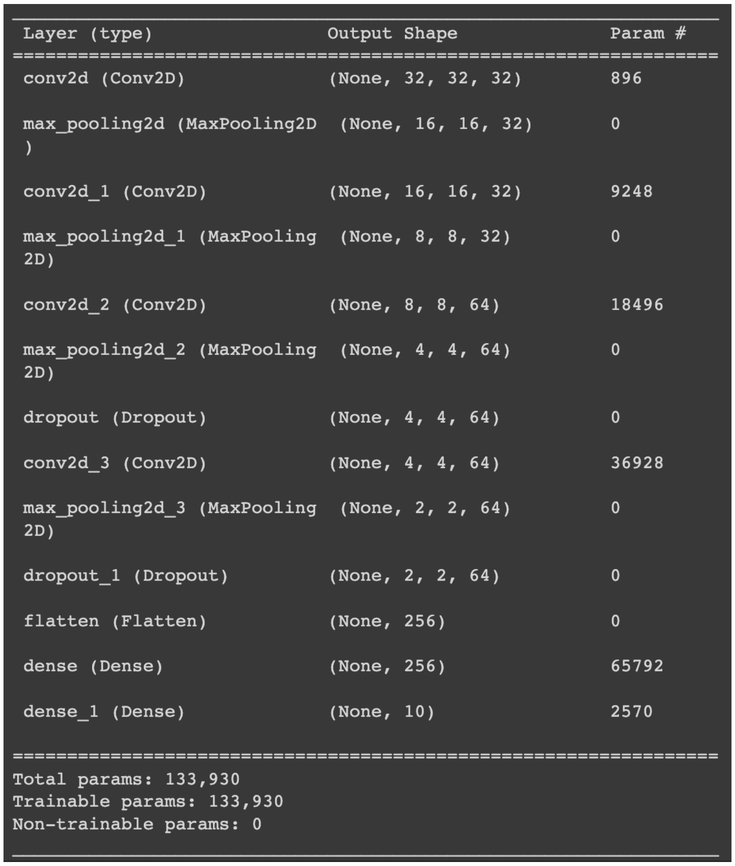
I have used two methods the first one is Deep Neural Network with Convolution Layer and the second one being the Deep Neural Network without Convolution Layer.

I. Deep Neural Network with Convolution Layer

Convolution is mostly used when working with image data. The convolution consists of convolution, pooling, and then a fully connected deep neural network layers. Convolution means two perform dot product between two matrices one being the image matrix and the other the kernel matrix whose side we provide and then the next step is to pool the matrix in which we use max pooling i.e. we extract the most important features from the image after performing convolution.

Fig 1. shows the model structure applied by me. My model contains four convolution layers and four max pooling layers. I have also added a dropout layer to improve accuracy. The dropout layer randomly drops 25% of the connections. The next step is to flatten the layers and then use a small Dense network. I have added two dense layers on with ReLu activation function and the second layer with only 10 neurons and used the softmax activation function to predict the class which had highest prediction.

Further, I used the Adam optimizer for the model to learn from the back propagation. The learning rate I chose is 0.001 after some trial and error. Then I used 10 epochs on the model and the model returned an validation accuracy of 73.74%.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        896

 max_pooling2d (MaxPooling2D  (None, 16, 16, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 16, 16, 32)        9248

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 32)         0
 2D)

 conv2d_2 (Conv2D)           (None, 8, 8, 64)          18496

 max_pooling2d_2 (MaxPooling  (None, 4, 4, 64)         0
 2D)

 dropout (Dropout)           (None, 4, 4, 64)          0

 conv2d_3 (Conv2D)           (None, 4, 4, 64)          36928

 max_pooling2d_3 (MaxPooling  (None, 2, 2, 64)         0
 2D)

 dropout_1 (Dropout)         (None, 2, 2, 64)          0

 flatten (Flatten)           (None, 256)               0

 dense (Dense)               (None, 256)               65792

 dense_1 (Dense)             (None, 10)                2570

=================================================================
Total params: 133,930
Trainable params: 133,930
Non-trainable params: 0
_________________________________________________________________
```

Fig 1: Model with Convolutional Layer

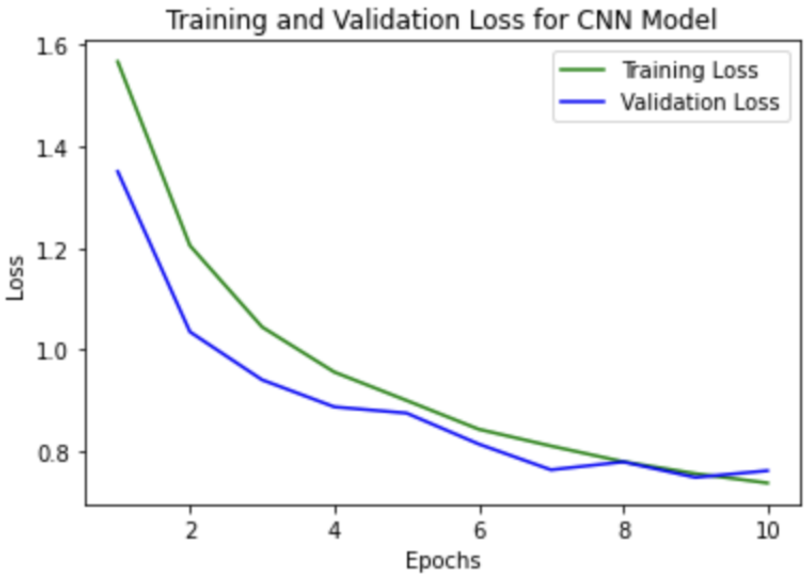
II. Deep Neural Network without Convolutional Layer

Deep Neural Network is basically an artificial neural network the ain aim of a neural network is to mimic a human brain functionality. So, it works comparatively better then other methods but it does not work very good with image data. To solve the problem using a Deep Neural Network first we flatten the image after flattening the images we pass it to further neurons reducing it to 10 neurons which represent the 10 classes of the dataframe. The activation function I used was ReLu which penalises the negative values and makes them zero and the positive values remain the same. The last layer uses a softmax function activation as for two classes we use the sigmoid activation and for more then two classes we use the softmax activation function.

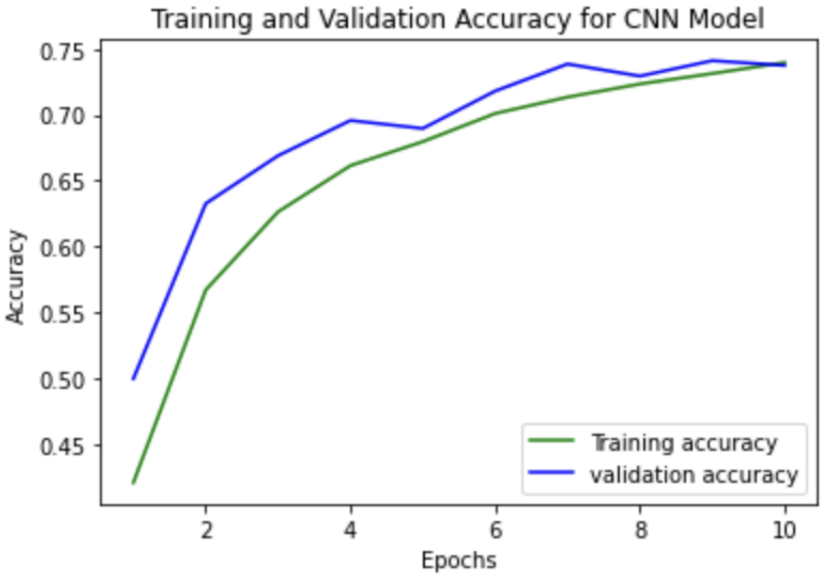
The model included one flattened layer and five hidden layers and one output layers.

Result

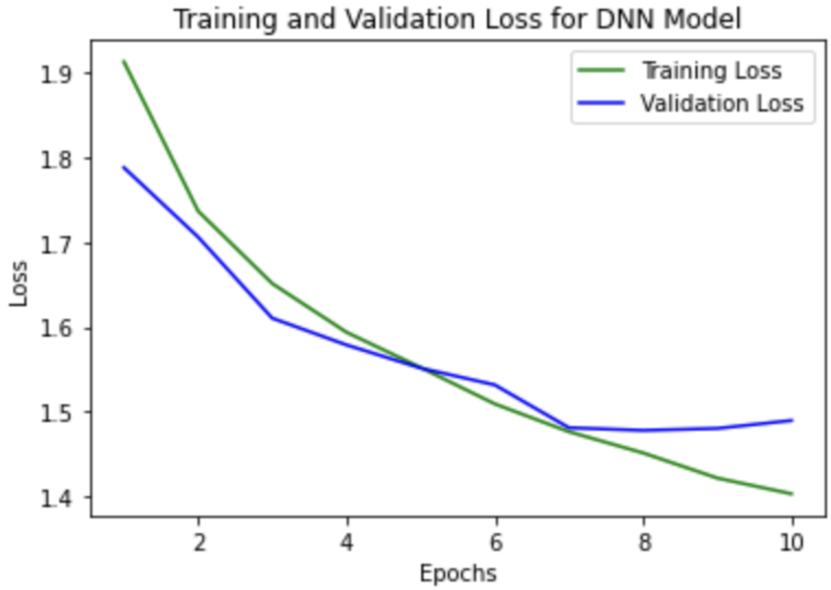
- The following graph shows the training and validation loss for the deep neural network with CNN layer. We can see that from the graph that the training loss and validation loss decrease at a steady rate but we can observe that after 10 epochs the validation loss started increasing more then the training loss. From the graph we can also tell that the graph has a good learning rate.



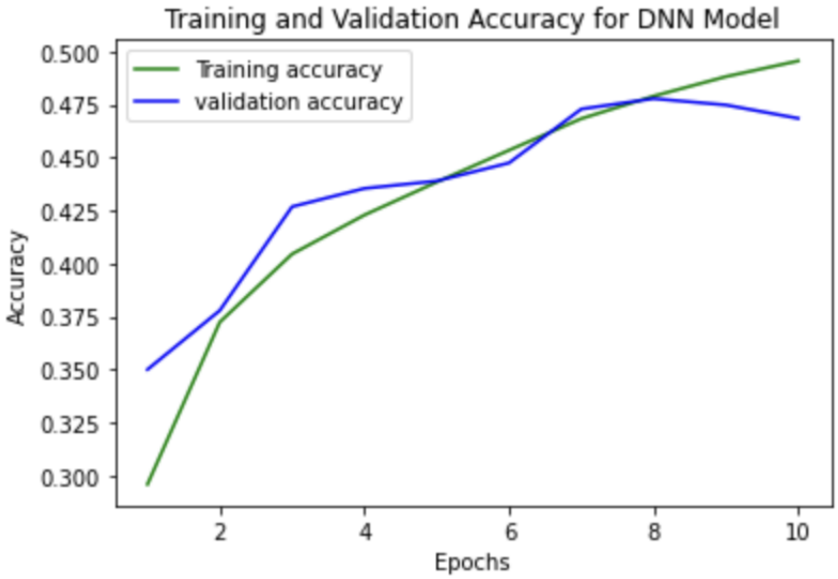
- The figure below shows the training and validation accuracy as the training and validation accuracy are gradually increasing we can tell that the data is not overfitting the model.



- The following figure shows the training and validation loss for the Deep neural network without Convolutional layer. From the graph we can see that the training and validation loss are high compared to the DNN with Convolution layer. Also, the validation loss is greater then the training loss for half the epochs.

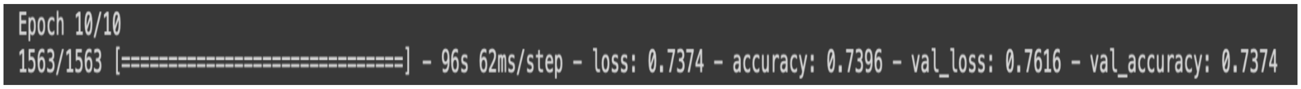


- The figure below is shows the training and validation accuracy for the DNN model. The model seems to overfit the data as the validation accuracy has started increasing.



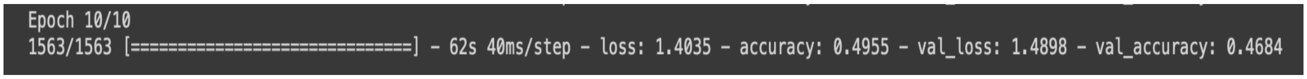
Comparison

- The graphs and from the following Fig 2 and Fig 3 we can say that using the convolution layer helps very much to improve the model. The features extracted by convolution are much better then the features being identified by the deep neural network as image is a matrix data therefore matrix operations are better to work with matrices i.e. convolution. This can be easily seen by the accuracy in Fig 2 being 0.7374 and accuracy in Fig 3 being 0.4684.
- We can also observe the run time of the epoch and it is clear that the model with convolution layer takes longer to run compared to the other model due to the increase in the number of computations which are part of the convolution layer.



```
Epoch 10/10
1563/1563 [==============================] - 96s 62ms/step - loss: 0.7374 - accuracy: 0.7396 - val_loss: 0.7616 - val_accuracy: 0.7374
```

Fig 2: Epoch 10 of Deep neural network with convolution layer model



```
Epoch 10/10
1563/1563 [==============================] - 62s 40ms/step - loss: 1.4035 - accuracy: 0.4955 - val_loss: 1.4898 - val_accuracy: 0.4684
```

Fig 3: Epoch 10 of Deep neural network model

Code

```
# Importing Libraries
"""

from keras.layers import Dense, Conv2D, MaxPool2D, Flatten, Dropout
from sklearn.metrics import classification_report, confusion_matrix
from tensorflow.keras.optimizers import Adam
from keras.models import Sequential
from keras.datasets import cifar10
import matplotlib.pyplot as plt
import tensorflow as tf
import seaborn as sns
import numpy as np
import keras
import cv2

"""# Loading Dataset"""

num_classes = 10
```

```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

print("x_train shape:", x_train.shape)
print(x_train.shape[0])
print(x_test.shape[0])

y_train = keras.utils.np_utils.to_categorical(y_train, num_classes)
y_test = keras.utils.np_utils.to_categorical(y_test, num_classes)

plt.figure(figsize = (2, 2))
plt.imshow(x_train[0])
plt.title(y_train[0])

x_train = np.array(x_train) / 255.0
x_test = np.array(x_test) / 255.0

"""# CNN Model"""

model = Sequential()

model.add(Conv2D(32, (3, 3), padding="same", activation="relu", input_shape=(32, 32, 3)))
model.add(MaxPool2D())

model.add(Conv2D(32, (3, 3), padding="same", activation="relu"))
model.add(MaxPool2D())

model.add(Conv2D(64, (3, 3), padding="same", activation="relu"))
model.add(MaxPool2D())
model.add(Dropout(0.2))

model.add(Conv2D(64, (3, 3), padding="same", activation="relu"))
model.add(MaxPool2D())
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256, activation="relu"))
model.add(Dense(10, activation="softmax"))
model.summary()

opti = Adam(learning_rate=0.001)
model.compile(optimizer=opti, loss="categorical_crossentropy", metrics=["accuracy"])

history_cnn = model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test))

model2 = Sequential()

model2.add(Flatten())
model2.add(Dense(1024, activation="relu"))
model2.add(Dense(512, activation="relu"))
model2.add(Dense(512, activation="relu"))
model2.add(Dense(256, activation="relu"))
model2.add(Dense(256, activation="relu"))
model2.add(Dense(10, activation="softmax"))

opti = Adam(learning_rate=0.001)
model2.compile(optimizer=opti, loss="categorical_crossentropy", metrics=["accuracy"])

history_dnn = model2.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test))

loss_train_cnn = history_cnn.history['loss']
loss_val_cnn = history_cnn.history['val_loss']
epochs = range(1, 11)
plt.plot(epochs, loss_train_cnn, 'g', label='Training Loss')
plt.plot(epochs, loss_val_cnn, 'b', label='Validation Loss')
plt.title('Training and Validation Loss for CNN Model')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

loss_train_dnn = history_dnn.history['loss']
loss_val_dnn = history_dnn.history['val_loss']
epochs = range(1, 11)
plt.plot(epochs, loss_train_dnn, 'g', label='Training Loss')
plt.plot(epochs, loss_val_dnn, 'b', label='Validation Loss')
plt.title('Training and Validation Loss for DNN Model')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

acc_train_cnn = history_cnn.history['accuracy']
acc_val_cnn = history_cnn.history['val_accuracy']
epochs = range(1,11)
plt.plot(epochs, acc_train_cnn, 'g', label='Training accuracy')
plt.plot(epochs, acc_val_cnn, 'b', label='validation accuracy')
plt.title('Training and Validation Accuracy for CNN Model')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

acc_train_dnn = history_dnn.history['accuracy']
acc_val_dnn = history_dnn.history['val_accuracy']
epochs = range(1,11)
plt.plot(epochs, acc_train_dnn, 'g', label='Training accuracy')
plt.plot(epochs, acc_val_dnn, 'b', label='validation accuracy')
plt.title('Training and Validation Accuracy for DNN Model')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```