Topic: Different type of layers that can be used and their effect on the model (CNN on Cifar10)

Inference:

We began with studying the effects of the core layers on the model.

For each model set we changes and modified the Dense, Dropout, Activation and Flatten parameters.

Model 1

Characteristics:

batch_size = 128

num_classes = 10

epochs = 20

dropout=0.2

num_predictions=20

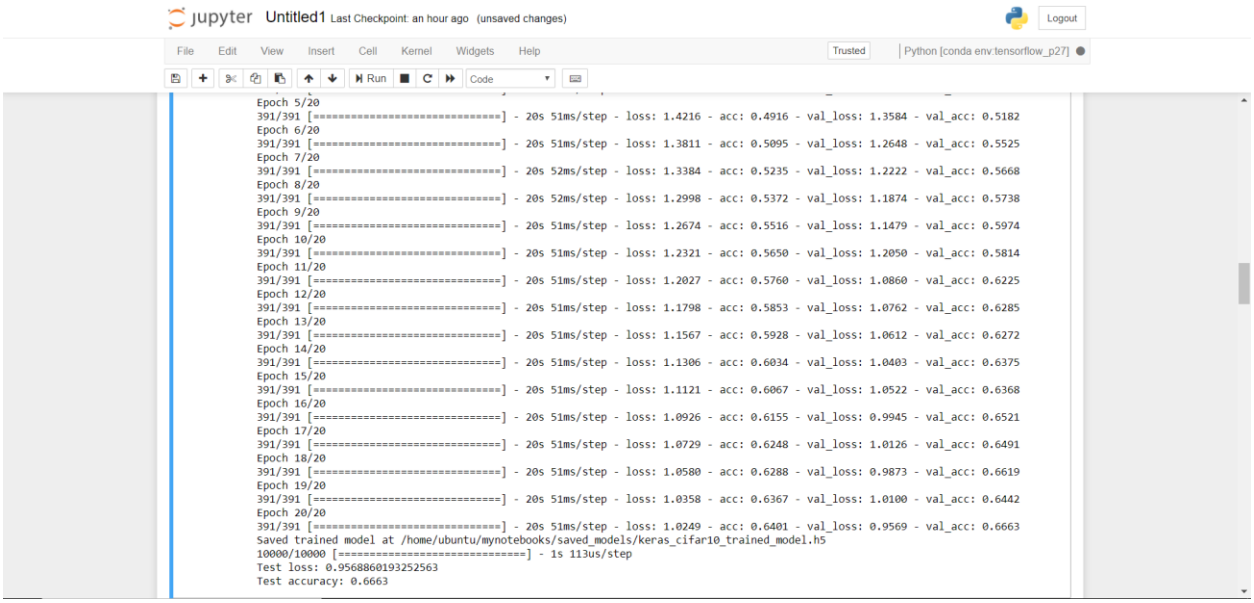
dense=128

Accuracy:

With the following parameters the validation accuracy was better than the training accuracy.

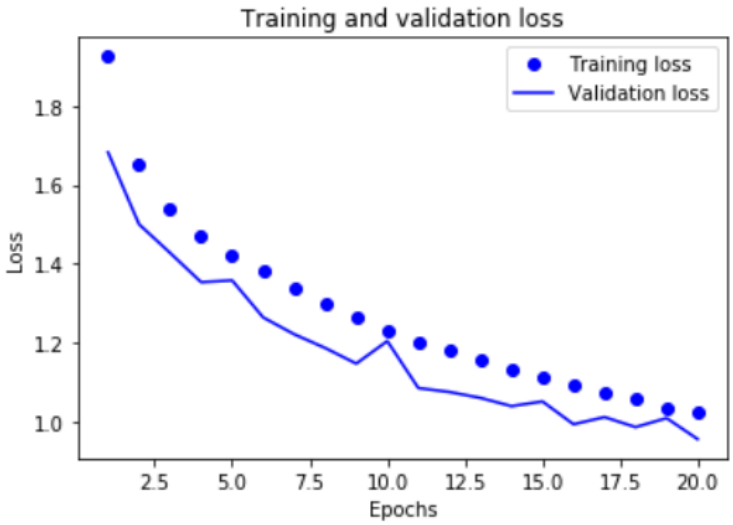
As per our research the test data cannot have a better accuracy than the training data. For this purpose we discarded this model.

Dropout: Providing a dropout decreased the efficiency of the training data.



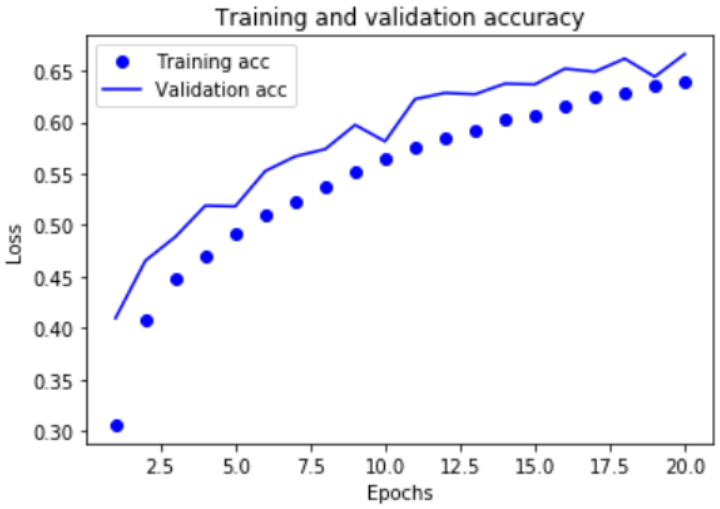
```
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



```
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



Model 2

Inference:

The model with no dropout and two denser layers: 128 and 256

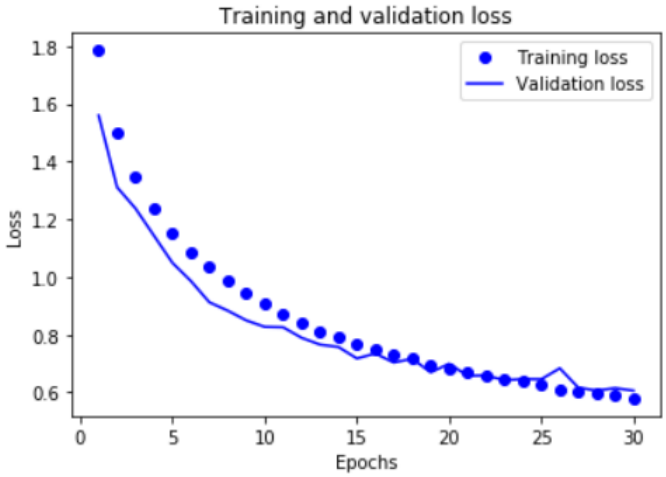
For this model, the accuracy was the same in both the training and validation sets.

We concluded from this observation that the training data leaked into the validation data.

Characteristics:

```
batch_size = 32
num_classes = 10
epochs = 30
data_augmentation = True
num_predictions = 20
```

```
Epoch 24/30
1563/1563 [==============================] - 25s 16ms/step - loss: 0.6370 - acc: 0.7787 - val_loss: 0.6458 - val_acc: 0.7780
Epoch 25/30
1563/1563 [==============================] - 25s 16ms/step - loss: 0.6252 - acc: 0.7823 - val_loss: 0.6456 - val_acc: 0.7810
Epoch 26/30
1563/1563 [==============================] - 25s 16ms/step - loss: 0.6104 - acc: 0.7865 - val_loss: 0.6835 - val_acc: 0.7673
Epoch 27/30
1563/1563 [==============================] - 25s 16ms/step - loss: 0.6032 - acc: 0.7900 - val_loss: 0.6168 - val_acc: 0.7875
Epoch 28/30
1563/1563 [==============================] - 25s 16ms/step - loss: 0.5953 - acc: 0.7921 - val_loss: 0.6062 - val_acc: 0.7916
Epoch 29/30
1563/1563 [==============================] - 25s 16ms/step - loss: 0.5888 - acc: 0.7955 - val_loss: 0.6149 - val_acc: 0.7875
Epoch 30/30
1563/1563 [==============================] - 25s 16ms/step - loss: 0.5790 - acc: 0.7999 - val_loss: 0.6059 - val_acc: 0.7935
Saved trained model at /home/ubuntu/mynotebooks/saved_models/keras_cifar10_trained_model.h5
10000/10000 [==============================] - 1s 115us/step
Test loss: 0.6059001004219056
Test accuracy: 0.7935
```



Model 3

Inference:

We increased the dense value to 256 to create a denser network and increased the number of epochs to space out the data.

Since in the previous model, the validation accuracy was higher than the training accuracy, we ruled out adding the dropout parameter. We concluded that the model required more training data to give a better test prediction.

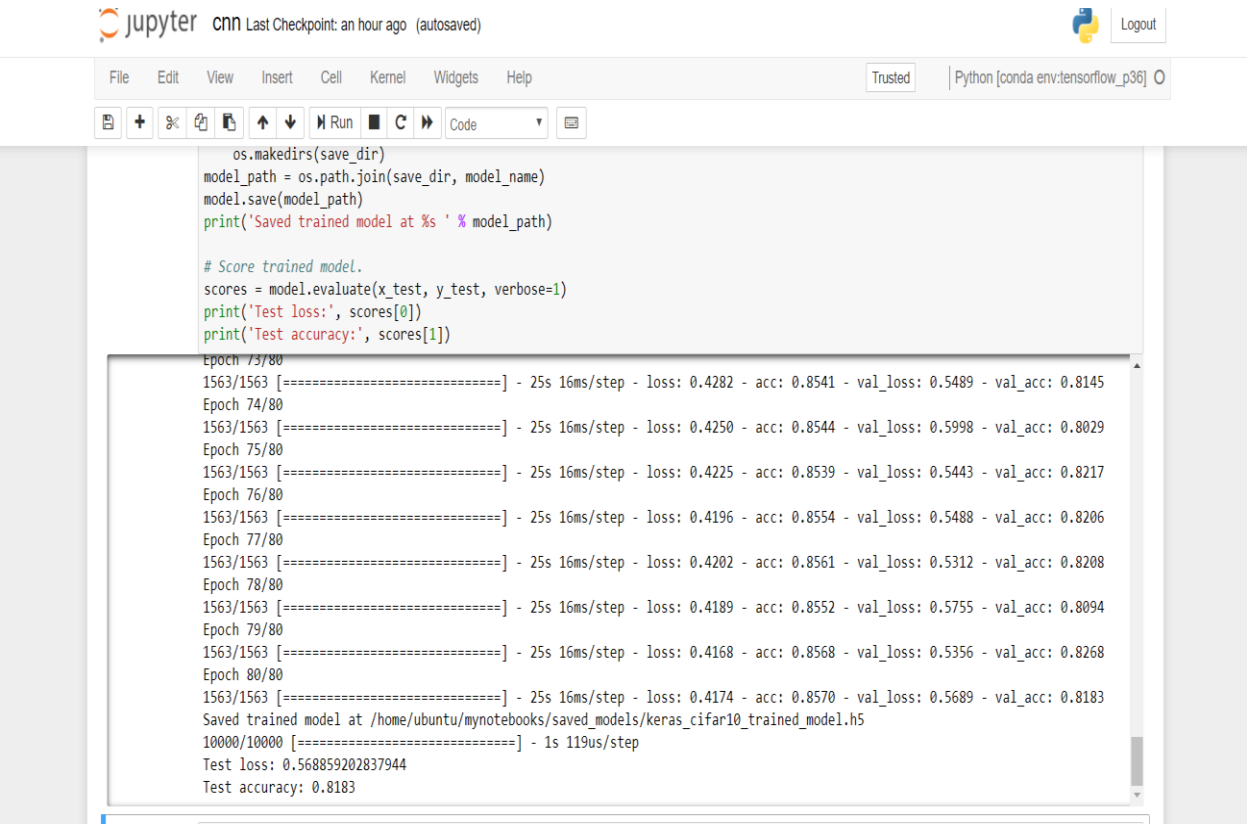
This resulted in better accuracy and lesser validation loss on the validation data.

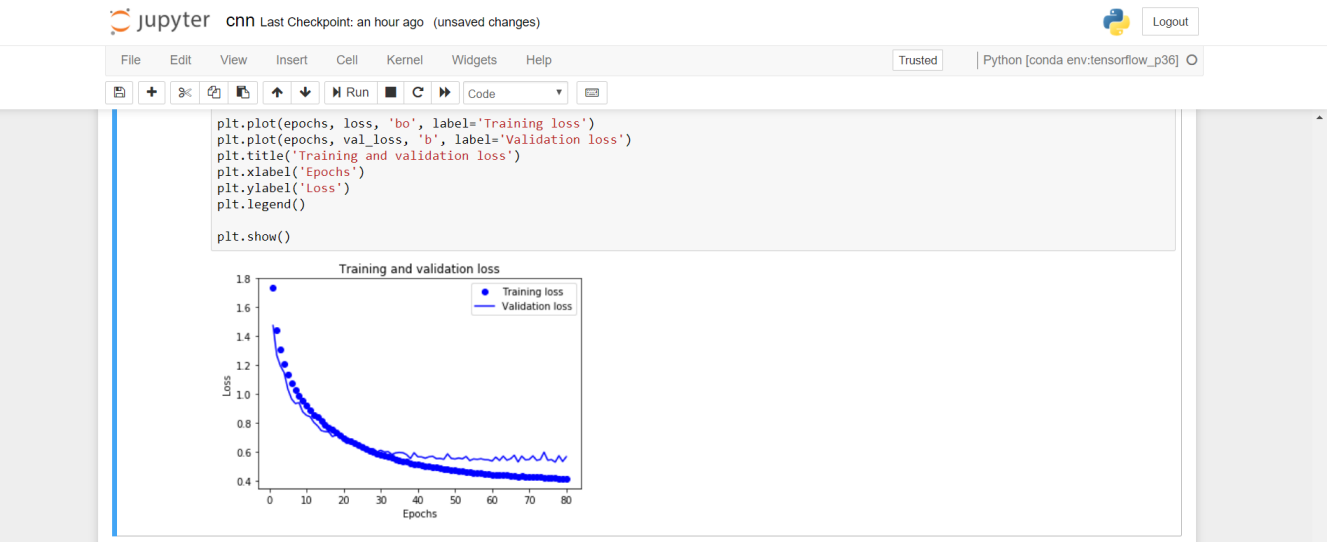
Characteristics:

```
batch_size = 32
num_classes = 10
epochs = 80
data_augmentation = True
num_predictions = 20
save_dir = os.path.join(os.getcwd(), 'saved_models')
model_name = 'keras_cifar10_trained_model.h5'
```

No dropout

Dense 256





Final Model 4

Inference:

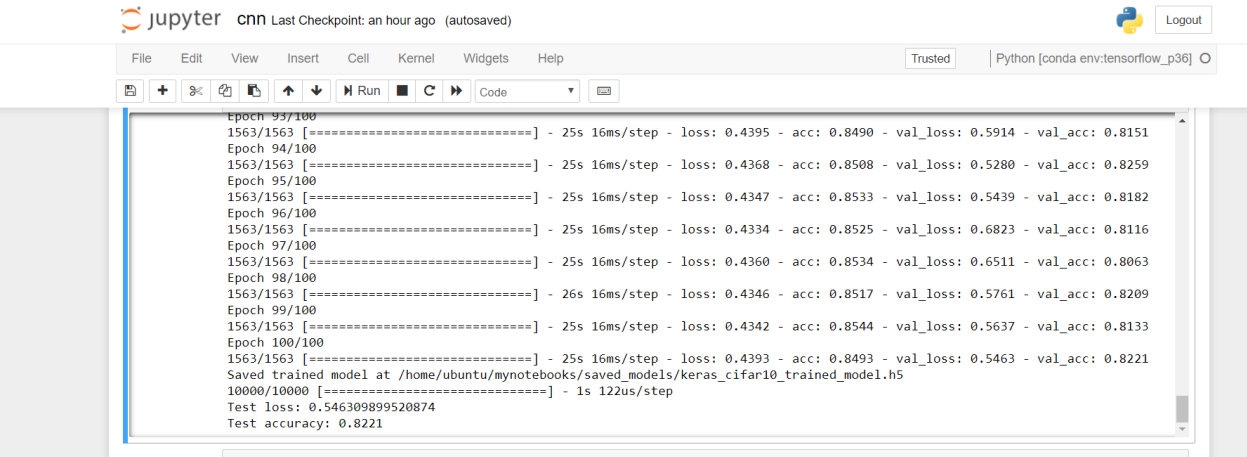
To test the impact of the dense layer we increased the number of epochs and dense layers but kept the batch size same as the previous one.

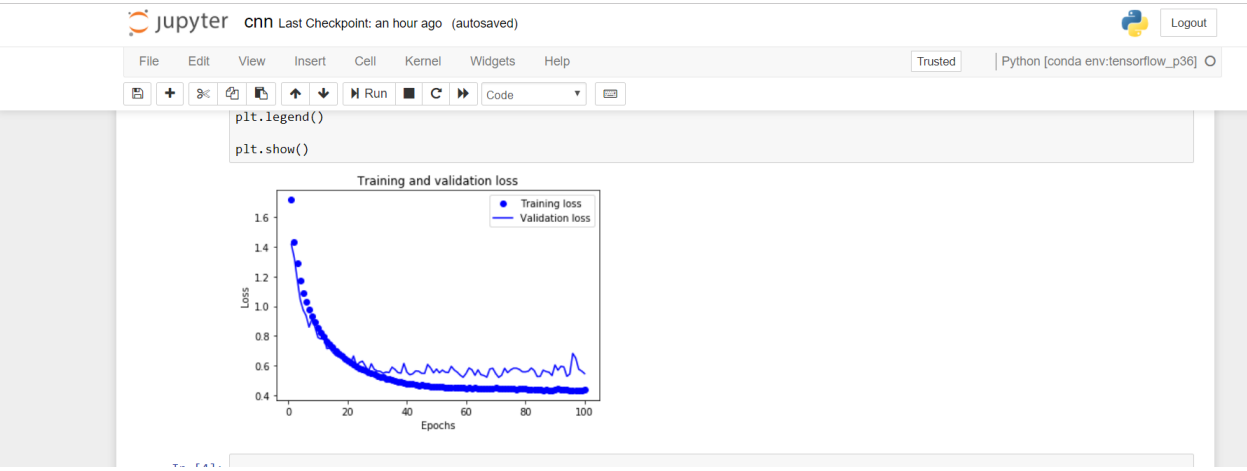
This gave us a similar model to model 2.

The graph of training loss to validation was slightly better as we added a denser layer.

Characteristics:

```
batch_size = 32
num_classes = 10
epochs = 100
data_augmentation = True
num_predictions = 20
save_dir = os.path.join(os.getcwd(), 'saved_models')
model_name = 'keras_cifar10_trained_model.h5'
```





Conclusion:

Through the numerous models we trained we derived that for the given dataset, we cannot use Sigmoid as an activation function for this dataset. As sigmoid function causes the problem of gradient descent. According to the reference book: Deep Learning with Python, sigmoid and tanh causes gradient descent and for a dataset with 10 classes they aren't the best activation functions.

We tested the model without the flattening function. This resulted in an error. The purpose of having the flattening function is to get the desired shape. Hence we give it before the denser layer.

Since the dataset provided isn't large enough dropping out data at random was causing underfitting or overfitting. Hence to achieve the best model we concluded my not adding the dropout function we were getting the best accuracy.

Final model consists of the core layers with values: dense layer as 512., epochs 100, activation = relu and no dropout.