REPORT ASSIGNMENT-2 AML

SARADINDU MEGHANA KARLAPALEM

INTRODUCTION:

According to my research tensorflow is used because TensorFlow is a regularly used and quite well deep learning library after assembling the essential libraries to create a neural network While there are other popular libraries, such as PyTorch TensorFlow is a common choice amongst developers because to its strong implementation and support This is owing to the robust documentation constant development comunity and number of pre-built models and datasets accesible As such TensorFlow is strong tool for developing deep learning models that is suitable for both novice and expert programmers.

LIST OF IMPORTS ARE:

firstly we need to import keras from tensorflow and also from tensorflow leras import layer from keras.layers import Dense from kent.layers import Dropout

After that we imported keras, keras.layers, Dense and Dropouts. In implementation of this each an everything is important.

Keras is a greater API for developing machine learning models that facilitates the process of developing TensorFlow 2 deep learning models Keras' core structures are layers and models with the Sequential model having the most basic Dense layers are used to set the amount of hidden units in the neural network whilst Dropout is a regularisation approach that is used to arbitrarily eliminate connectivity Keras makes it simple to stack layers and customize the amount of hidden units and activation functions.

model = keras.Sequential() ----- Sequential model is the most easiest model that pile up the layers in the sequences. model.add(Dense(64,activation='tanh')) piling up the layers is easy using the. add function. Also 64 which tells the Number of hidden units and also we have use the activation function which is tanh.

Contents of neural networks:

INPUT LAYER:

A neural network is composed of three major components:

The input layer serves as where the data is received. The input data in this case which represented like vector of IMDB data.

The hidden layers are in charge of performing calculations on the input data Each buried layer has dense units or neurons that analyze input data before passing it on to the next layer We can construct as many hidden units as we need to tackle paraticular issue.

The output layer generates the neural network's final output Depending on the situation the output layer may comprise one or many dense units The output layer's activation function is decided by the issue's specific nature such as binary or multi-class clasification, regresion or other forms of classifiication.

Here in the given project we implemented one layered approach.

.model = keras.Sequential([layers.Dense(64, activation="tanh"), layers.Dense(1, activation="sigmoid")])

As per the code which model started as sequenital. And we just piled them up with the one layer with 64 dense unit. And the activation function is tanh.

And also as per the instruction which have given I impleneted the tanh instead of relu.

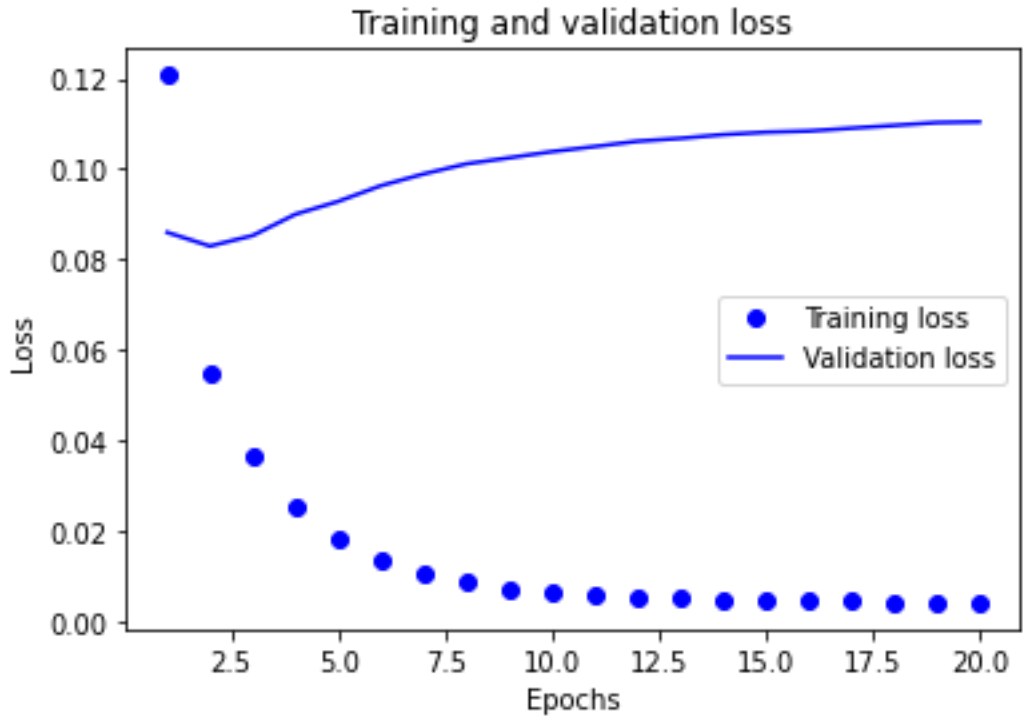
model.compile(optimizer="adagrad", loss="MSE", metrics=["accuracy"])

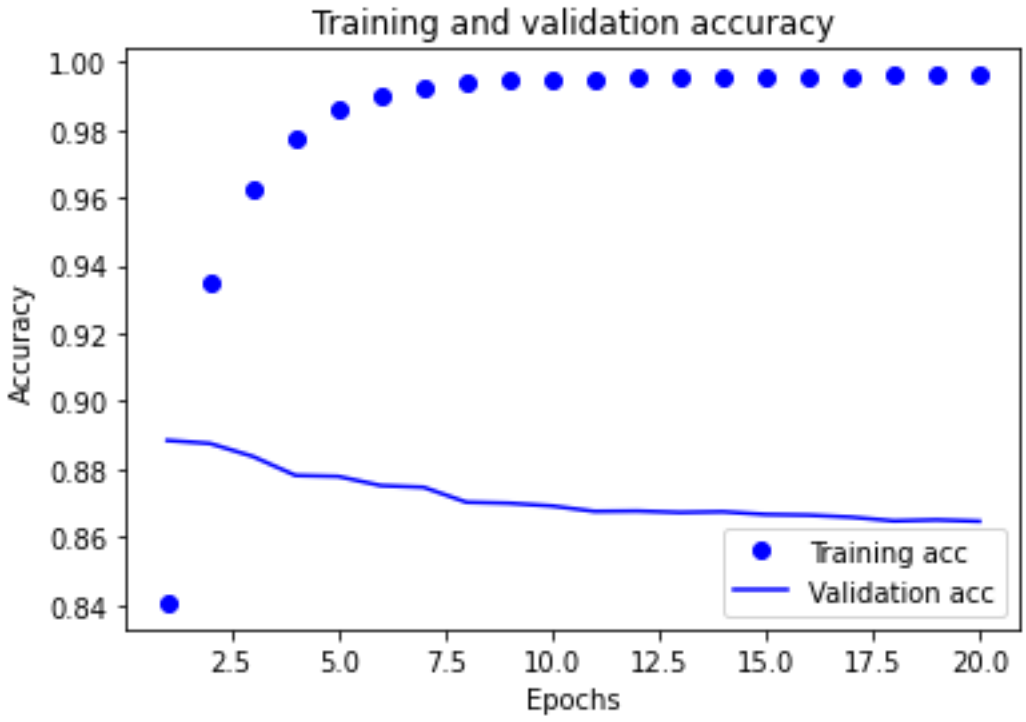
The code that we got uses an optimizer which is known as adagard with MSE loss. I was in a dilemma here in the starting imbd data uses a loss of binary cross entrophy which is probilistic loss and what it will happen when we change the regression loss. For the minimization of error optimizers plays a very crucial role and for that we have differenty optimizers. Among the different approaches we have adam is known as good optimizer. In the above project I have used adgrad. The second and third reference links explain these in a very detailed manner.

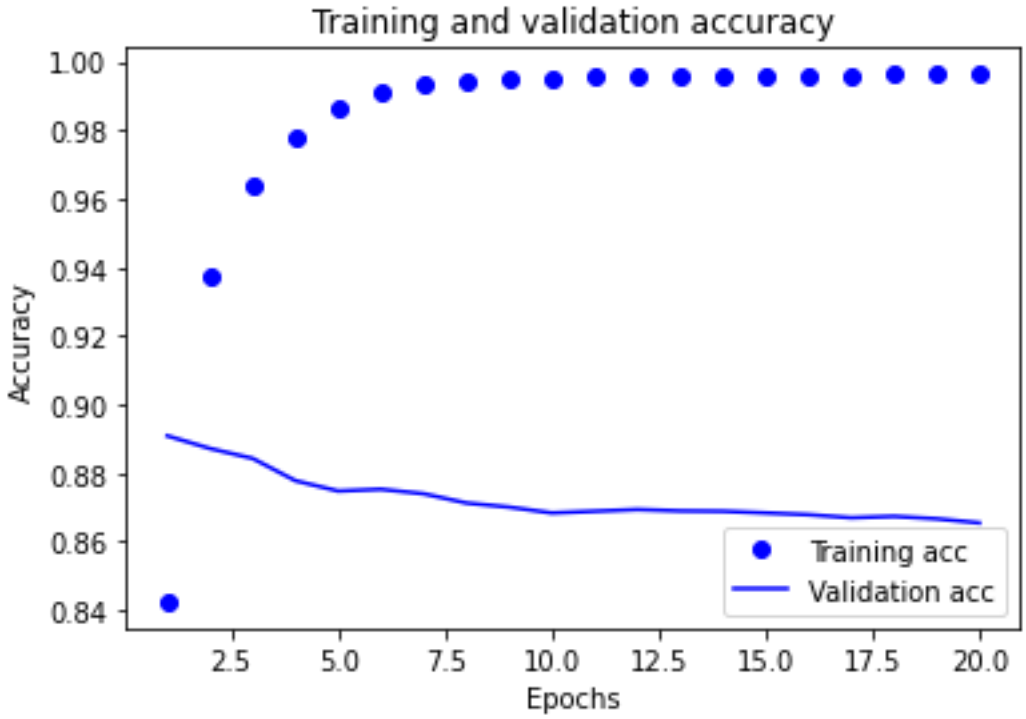
x_val = x_train[:10000] partial_x_train = x_train[10000:] y_val = y_train[:10000] partial_y_train = y_train[10000:] Training the data history = model.fit(partial_x_train, partial_y_train, epochs=20, batch_size=256, validation_data=(x_val, y_val))

The line which is above of the code represents the neural network with 20 epoch and the batch size 256 and parallely it compare with the validation data. In this project in which I have used L1 and L2 regulizers which does not give much importance on the total validation accuracy.

THE PLOTS:


Training and validation loss
Loss
Epochs
Training loss
Validation loss
0.12
0.10
0.08
0.06
0.04
0.02
0.00
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Training and validation accuracy
Accuracy
Epochs
Training acc
Validation acc
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.84
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0




```
782/782 [==============================] - 2s 3ms/step - loss: 0.1253 -
accuracy: 0.8618
[0.12526674568653107, 0.861840009689331]
```

CASE-2:

The other ipynb that I have submitted In which I have taken the128 as a hidden unit then the code is

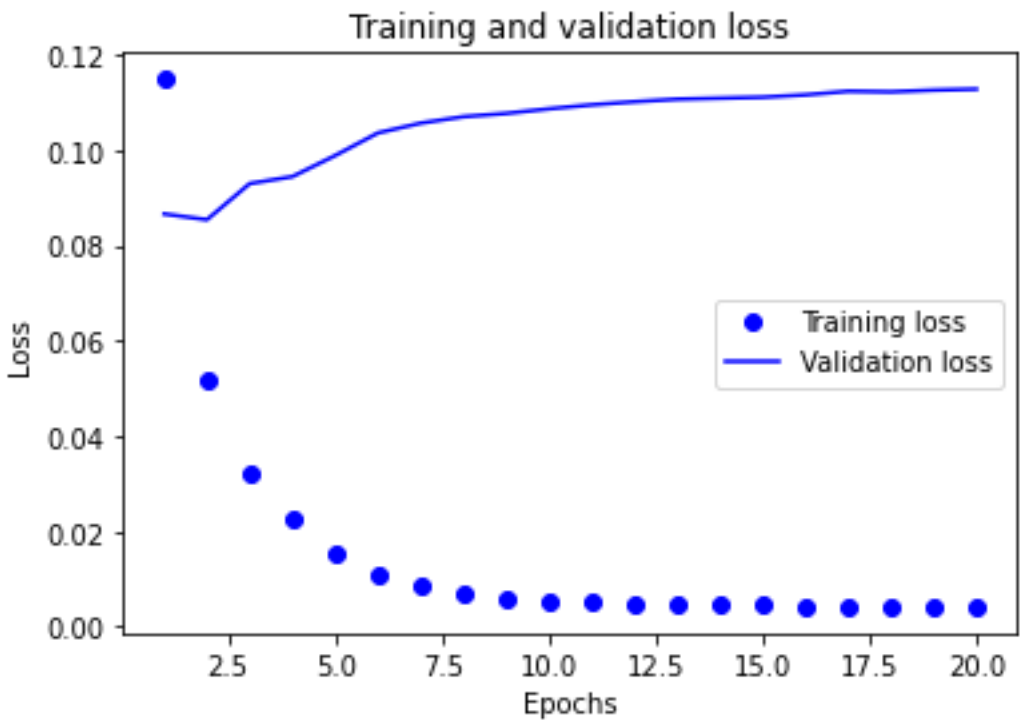
model.add(Dense(128,activation='tanh'))

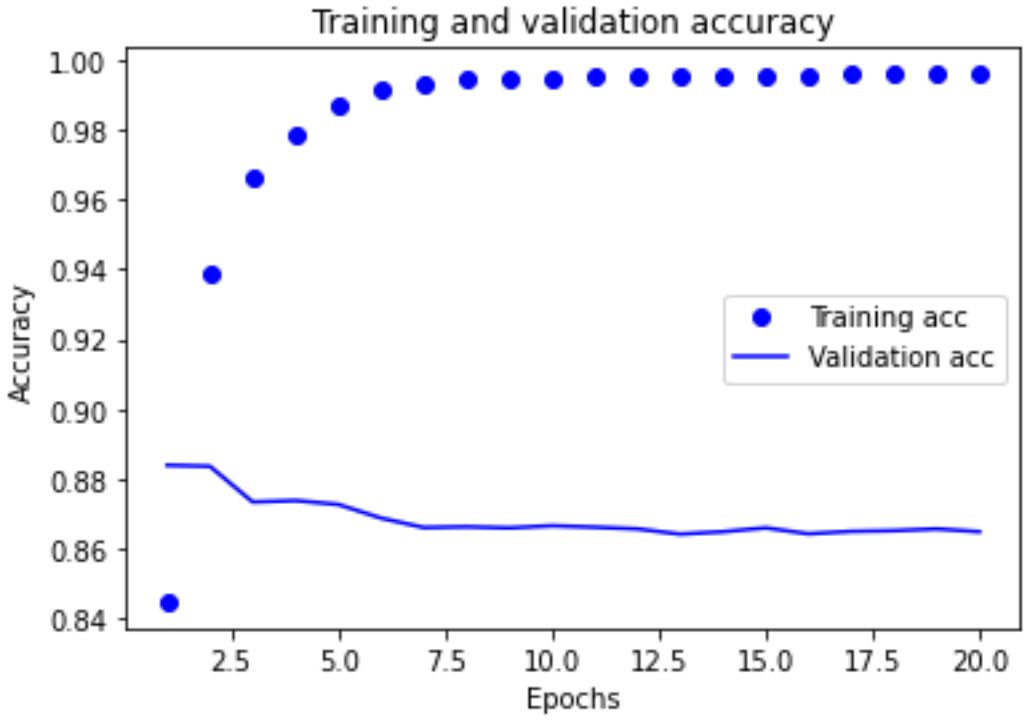
for one layered approach

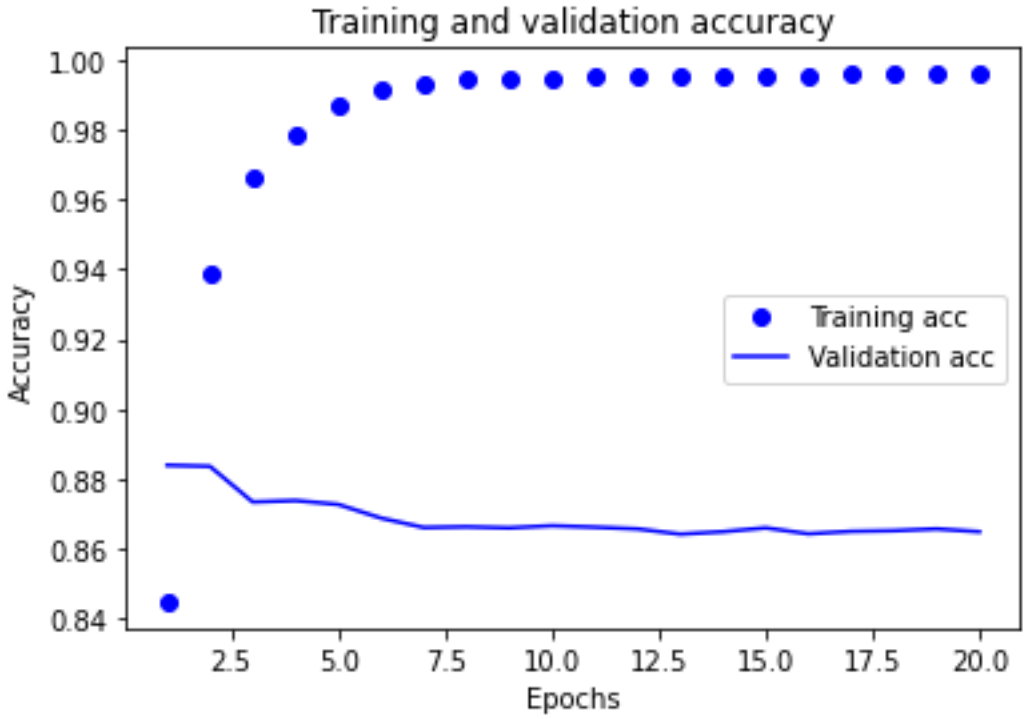
.model = keras.Sequential([layers.Dense(128, activation="tanh"), layers.Dense(1, activation="sigmoid")])

Here as per the code started as sequential. And we just build them up with the one layer with 128 hidden unit. And the activation function is tanh

THE PLOT:


Training and validation loss
Loss
Epochs
Training loss
Validation loss
0.12
0.10
0.08
0.06
0.04
0.02
0.00
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Training and validation accuracy
Accuracy
Epochs
Training acc
Validation acc
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.84
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0




```
782/782 [==============================] - 4s 5ms/step - loss: 0.1861 -
accuracy: 0.8544
[0.18611685931682587, 0.8544399738311768]
```

CONCLUSION

In the case-1

ACCURACY	0.86184
HIDDEN LAYERS	1
HIDDEN UNITS	64

In the case-2

ACCURACY	0.8544
HIDDEN LAYERS	1
HIDDEN UNITS	128

This is a 1 layered neural network with tanh as the activation function so instead relu. By instead rmsprop, adam used as the optimiser and the both L1 and L2 regularizers are employed. Moreover a dropout layer with dropout rate of 0.5 is utilized which indicates that 50% of the inputs are deleted at random while training.

Tanh is a non-linear activation function that demolishes input values between -1 and 1 It is often utilized in neural networks particularly in buried layers since it

enables the network to learn more sophisticated correlations between inputs and outputs Tanh on the other hand might suffer from the diminishing gradient problem when the input values are very high or very tiny which can slow down the training process

Adam is an optimizer that updates the network's weights after training using adaptive learning rates and momentum It is a popular optimizer since it iterates rapidly than some other optimizers such as rmsprop, particularly in deep neural networks.

To prevent overfitting, L1 and L2 regularizers add a penalty function to the loss function dependent on the size of the weights By punishing big weights L1 regulariSation fosters sparseness whereas L2 regularization supports tiny weights by penalizing large weight values squared The network can learn more robust and generalizable model for the input data by applying both L1 and L2 regularization

Dropout is a regularization approach that removes a percentage of the inputs at random during training Overfiting may be mitigated by encouraging the network to acquire more robust features that aren't dependent on specific inputs A dropout rate of 0.5 indicates that 50% of the inputs will be rejected at random during training which is a common number for dropout

Reference:

1. https://keras.io/api/losses/
2. https://keras.io/api/optimizers/
3. https://keras.io/about
4. https://keras.io/documentation/
5. https://keras.io/applications/