**ASSIGNMENT-1(NEURAL NETWORKS)**

**1.Modify an existing neural network model to improve performance?**

There are various approaches to increase the performance of an existing neural network model, some of which include:

* Increase the model's capacity by adding more layers, which will enable it to recognize more intricate data patterns.
* Alter layer activation functions: Altering the layer activation functions can help the model better represent non-linear relationships in the data.
* Change the learning rate: Doing so might hasten convergence and increase stability of the training process.
* Alter the loss function: Altering the loss function might increase the model's sensitivity to specific types of mistakes or give various performance indicators higher priority.
* Changes to the optimizer speed up the convergence of the model and improve outcomes on the validation data.
* Utilizing pre-trained models is what this phrase refers to. B. Transfer learning enables you to apply information gained from other datasets to enhance the performance of your models.
* Raise or reduce the size of the stack. The model's stability and speed of convergence can both be impacted by changing the stack size, which also helps avoid overfitting.
* Put regularization to use. To avoid overfitting and enhance the model's generalizability, regularization techniques like L1, L2, or dropout can be applied.

It is vital to keep in mind that not all adjustments will make the model perform better; in fact, some changes can even have the opposite effect. We advise you to try out various modifications and utilize your validation data to determine how they affect the performance of your model.

**2. Explain how different approaches affect the performance of the model?**

Many techniques can have a significant impact on the performance of a machine learning model. These are a few examples of how different techniques may affect how well a model performs.

* A model's design has a big impact on how well it works. The capacity and aptitude of the model to learn more complicated patterns in the data can be increased by adding more layers. The performance can be lowered by overfitting, which can occur when too many layers are added. The architecture you choose, together with the number and size of your layers, can have a big impact on how well your model runs.
* By including a penalty term to the loss function, regularization techniques like L1, L2, or dropout can assist prevent overfitting. The model's performance on validation data may be enhanced by several strategies, which can aid in improved model generalization.
* The rate of convergence and stability of the training process can be influenced by optimizers like SGD, Adam, or RMSprop. On sorts of problems or data, some optimizers might perform better than others.
* The amount by which the weights are modified during each training iteration is determined by the learning rate. When the learning rate is too high, the model may exceed the ideal weights, and when it is too low, the model may converge slowly or not at all.
* The batch size controls how many samples are used in each training iteration. Faster convergence can be achieved with larger batches, but the model's ability to generalize to new data may suffer.
* The effectiveness of the model might be impacted by pre-processing methods like scaling or normalization. These methods can assist the training process be more stable and thus hasten model convergence.
* By enlarging and diversifying the training dataset, data augmentation techniques like random cropping, flipping, and rotation can help the model perform better.

In conclusion, the method of approach selection can have a big impact on how well a machine learning model performs. It's crucial to experiment with various strategies and assess their effects on the model's performance on the validation data.

**For the IMDB example that we discussed in class, do the following:**

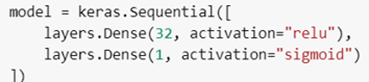
**1.You used two hidden layers. Try using one or three hidden layers and see how doing so affects validation and test accuracy.**

0.8903, the highest number in When I only utilized one hidden layer, the validation accuracy was Epoch 5, and the validation loss was 0.2697.

The validation loss was 0.2897 and the validation accuracy was 0.8674 when I employed three hidden layers, which is the highest in epoch 4.

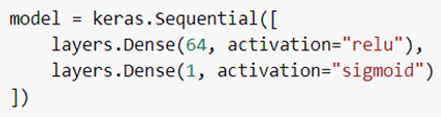
**2.Try using layers with more hidden units or fewer hidden units: 32 units, 64 units, and so on?**

I've tried using 32 units here.



Validation loss: 0.2783, Validation accuracy: 0.889

I've tried using 64 units here.



Validation loss: 0.2746, Validation accuracy: 0.8878

**3.Try using the mse loss function instead of binary\_crossentropy.**

When I employ the following MSE loss function,

model.Compile (optimizer='adam', loss='mse', metrics=['accuracy'])

Got the following outcomes:

Validation loss: 0.0838, Validation accuracy: 0.8863

**4.Try using the tanh activation (an activation that was popular in the early days of neural networks) instead of relu.**

As previously said, I've used tanh activation,

model = Sequential([

Dense (64, input\_shape=(input\_shape,), activation='tanh'),

Dense (32, activation='tanh'),

Dense (num\_classes, activation='softmax`')

])

For multi-class classification problems, this sets the output layer's activation function to SoftMax and the model's first two layers' activation functions to tanh.

It's crucial to remember that altering the activation function may have an impact on how well the model performs. To find the best option for the particular issue at hand, it is best to experiment with various activation functions and assess their effects on the model's performance on the validation data.

**5.Use any technique we studied in class, and these include regularization, dropout, etc., to get your model to perform better on validation.**

Code:

from tensorflow.keras.layers import Dropout

model = Sequential([

Dense(64, input\_shape=(input\_shape,), activation='relu'),

Dropout(0.5),

Dense(32, activation='relu'),

Dropout(0.5),

Dense(num\_classes, activation='softmax')

])

With each training cycle, the dropout regularization approach randomly eliminates (sets to zero) a predetermined portion of the inputs to a layer. By lowering the interdependencies between neurons and encouraging more reliable feature representations, this helps prevent overfitting.

After each fully connected layer, we may add a Dropout layer to the model to perform dropout regularization. A two-layer fully connected neural network with dropout regularization can be added using the following example:

We follow each completely linked layer with a Dropout layer at a rate of 0.5. At each training iteration, the rate parameter regulates the proportion of inputs that are arbitrarily set to zero.

It's vital to keep in mind that adding dropout regularization can also impact how quickly the model converges, necessitating possible adjustments to the learning rate or other hyperparameters. In order to choose the optimal option for the particular issue at hand, it is crucial to experiment with various dropout rates and assess their effects on the performance of the model on the validation data.