Neural Networks

Problem Statement: Using Neural Networks, classify a movie review as favourable or negative.

Introduction:
Enhancing an existing neural network model's performance and discussing several strategies that affect the model's performance.

Data:
In this scenario, the IMDB dataset is being utilized as a binary classification, with the objective variable being binary. To get the results we required, we employed sigmoid. We will utilize this dataset to train and test our neural network models.

Model Architecture:
For this study, we employed one hidden layer with 16 units, the RELU (Rectified Linear) activation function for multiclass regression analysis, and binary cross-entropy as a loss function. A sigmoid activation function was used by one unit in the output layer. This model was trained for 20 epochs, with validation accuracy and loss measured for each one.

Techniques:

1. **Input factors that are directly proportional to accuracy.**

   The first approach we looked at was increasing the number of input units. We changed the number of units in the first hidden layer from 16 to 32 and 64 to examine how it affected the model's validation accuracy.
   We observed that increasing the number of input units improved model validation accuracy, but only marginally as compared to increasing the number of model parameters, which increased the risk of overfitting. As a result, finding a happy medium between model complexity and performance is crucial.
   Based on the current model performance, it appears that validation accuracy has reached a plateau after a few epochs. In addition to regularization strategies, we may experiment with adjusting the number of input units, which may affect the model's learning ability.
   With the present model, the first hidden layer contains 16 units. We raised the units in the hidden layers to 32 and 64 to see how it affected the validation accuracy and increasing the number of input units from 16 to 32 improved the model's validation accuracy by 0.01. Nevertheless, this improvement was minor in comparison to the increase in model parameters, which raised the danger of overfitting.

2. **Experimenting with more hidden layers:**

   To begin, we used a network model with two hidden layers instead of just one. The model's performance metrics (loss and accuracy) fluctuate with each epoch. When compared to a model with only one hidden layer, an additional hidden layer assists the model in learning more complex data representations. This might lead to better validation set performance. In contrast, having too many hidden layers may result in overfitting, in which the model memorizes the training data too well and performs poorly on fresh data.
   According to the performance metrics, the model looks to be doing well on both the training and validation sets, with a final validation accuracy of 0.8848. Yet, there is a modest increase in validation loss at the end of the training, which may indicate overfitting. To avoid overfitting, measures such as dropout or regularization may be worth investigating.

3. **Using mean square error (mse) loss function:**

   The accuracy has dropped by changing the loss function to mse; nevertheless, this is to be expected because the loss function and the accuracy metric are not directly connected. The model prioritizes mean squared error above binary cross-entropy loss. Nonetheless, it is a legitimate strategy that may surpass binary cross-entropy depending on the issue and data.
   While the mean squared error (MSE) loss function may be more appropriate for regression challenges, it does not always increase model performance. In this example, it appears that this has reduced the model's accuracy.
   The accuracy of the model is worse with the MSE loss function than with the binary cross-entropy loss function. This is most likely because MSE loss is better suited for problems with continuous goal variables, whereas the target variable in this case is binary. As a result, in this case, I propose sticking with the binary cross-entropy loss function.

4. **Tanh activation function:**

   A neural network model for a binary classification task was trained using the tanh activation function. Throughout the 20-epoch training period, the validation accuracy and loss were tracked for each epoch. The tanh (hyperbolic tangent) function is a symmetric activation function that converts -1 to 1. Because the tanh function returns 0 at its mean, it is useful for normalizing input values.
   The training logs reveal that the model achieved the greatest validation accuracy, but that the accuracy rapidly decreases in the following epochs. The training and validation losses decrease at first but increase later, suggesting that the model has begun to overfit the training data. Overall, the tanh activation function looks to perform well for the present binary classification task, while more testing with different activation functions and hyperparameters may help to improve the model's performance.

5. **Dropout technique for normalization:**

   Dropout is a technique used in neural networks to reduce overfitting. During training, Dropout randomly removes (i.e., sets to zero) a portion of the neurons in the network. This prevents any one neuron from having a large impact on the model and encourages the network to develop more resilient features. The model was trained for 20 epochs, and the training set's accuracy increased, indicating that the model was increasing its fit to the training data. Yet, the accuracy of the validation set remained generally consistent, indicating that the model was not overfitting. This is most likely because the model employs dropout. Dropout keeps any one neuron from becoming very prominent in the model, reducing overfitting. As a result of the use of dropout, the model performed well on the validation set without overfitting the training data.

Conclusion:

Lastly, we investigated a variety of methods for increasing the performance of a neural network model for a binary classification task. We investigated increasing the number of input units, adding hidden layers, changing the loss function, and using the dropout method. Each of these tactics has advantages and disadvantages, and their effectiveness is determined by the unique scenario and facts. I discovered that increasing the number of input units and hidden layers can enhance performance, but overfitting is a risk. Alternative loss functions, such as MSE loss, may be more appropriate for regression problems, but they may not improve model performance in binary classification tasks.
Using the dropout technique, on the other hand, can help to improve the model's performance on the validation set by preventing overfitting. A good starting point would be to train the model for 10-20 epochs before evaluating its performance. If the performance is insufficient, try increasing the number of epochs until the desired level of performance is reached. To find the best configuration for a particular problem, it is now required to experiment with different approaches and hyperparameters. We can ensure that our model functions optimally and is resistant to new and unknown inputs by doing so.