Neural Networks

Problem Statement: Classify a movie review as either positive or negative using Neural Networks.

Introduction:
Improving the performance of an existing neural network model and describing different techniques that impact the model's performance.

Data:
The IMDB dataset is being used as a binary classification in this case, with the goal variable being binary. We used sigmoid to acquire the results we needed. This dataset will be used to train and assess our neural network models.

Model Architecture:
The first model we used for this research included one hidden layer with 16 units, the RELU (Rectified Linear) activation function for multiclass regression analysis, and binary cross-entropy as a loss function. One unit in the output layer had a sigmoid activation function. We trained this model for 20 epochs, measuring validation accuracy and loss for each one.

Techniques:

1. **Input units directly proportional to the accuracy.**

   The first method we investigated was increasing the number of input units. We adjusted the number of units in the first hidden layer from 16 to 32 and 64 to see how it influenced the validation accuracy of the model.
   We discovered that increasing the number of input units improved model validation accuracy, but the improvement was minor when compared to increasing the number of model parameters, which also raised the danger of overfitting. As a result, striking a balance between model complexity and performance is critical.
   Based on the current model performance, it seems that the validation accuracy is still plateauing after a few epochs. In addition to regularization techniques, we can try changing the number of input units, which may impact the model's capacity to learn.
   In the current model, we have 16 units in the first hidden layer. We have increased the units to 32 and 64 in the hidden layers and observed how it has an impact on the validation accuracy and increasing the number of input units from 16 to 32 improved the model's validation accuracy by 0.01. However, this improvement was relatively small compared to the increase in the number of model parameters, which also increased the risk of overfitting.

2. **Trying more hidden layers**

   To begin, we employed a network model with two hidden layers rather than simply one. At each epoch, the model's performance measures (loss and accuracy) change. In comparison to the model with only one hidden layer, an additional hidden layer helps the model learn more complicated data representations. This may result in improved performance on the validation set. Having too many hidden layers, on the other hand, might lead to overfitting, in which the model memorizes the training data too well and performs badly on new data.
   With a final validation accuracy of 0.8855, the model appears to be doing well on both the training and validation sets, according to the performance metrics. Nevertheless, toward the conclusion of the training, there is a minor rise in validation loss, which may imply overfitting. To avoid overfitting, it may be worthwhile to investigate approaches such as dropout or regularization.

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

   By changing the loss function to mse, the accuracy has decreased; nevertheless, this is to be expected because the loss function and the accuracy metric are not directly related. The model optimizes for mean squared error rather than binary cross-entropy loss. Yet, it is a valid method that, depending on the issue and data, may outperform binary cross-entropy.

While utilizing the mean squared error (MSE) loss function for regression difficulties may be more suitable, it does not always improve model performance. That appears to have lowered the model's accuracy in this case.
The model's accuracy is poorer with the MSE loss function than with the binary cross-entropy loss function. This is most likely because MSE loss is better suited for issues where the goal variable is continuous, but the target variable in this case is binary. As a result, I recommend staying with the binary cross-entropy loss function for this situation.

4. **Tanh activation function:**

The tanh activation function was used to train a neural network model for a binary classification challenge. The validation accuracy and loss were tracked for each epoch during the training period of 20 epochs. The tanh (hyperbolic tangent) function is a symmetric activation function that transfers input values from -1 to 1. The tanh function returns 0 at its mean, making it handy for standardizing input numbers.
The training logs show that the model attained the highest validation accuracy, but subsequently, the accuracy reduces gradually in consecutive epochs. The training and validation losses drop initially but grow subsequently, indicating that the model has begun to overfit to the training data. Overall, the tanh activation function appears to perform quite well for the current binary classification problem, although more testing with various activation functions and hyperparameters may assist to enhance the model's performance.

5. **Dropout technique for normalization**

Dropout is a strategy used in neural networks to minimize overfitting. Dropout randomly drops out (i.e., sets to zero) part of the neurons in the network during training. This keeps any one neuron from being very significant in the model and drives the network to acquire more robust properties.
The result shows that the model was trained for 20 epochs and that the training set's accuracy rose, showing that the model was improving its fit to the training data. Nonetheless, the validation set's accuracy remained largely consistent, showing that the model was not overfitting. This is most likely owing to the model's usage of dropout. Dropout prevents any one neuron from becoming very significant in the model, hence preventing overfitting. As a result, the application of dropout allowed the model to perform well on the validation set without overfitting the training data.

Conclusion:

Finally, we looked into many approaches for improving the performance of a neural network model for a binary classification job. We considered increasing the number of input units, employing additional hidden layers, modifying the loss function, and employing the dropout approach. Each of these strategies has benefits and drawbacks, and their performance is dependent on the individual situation and data.
We found that increasing the number of input units and hidden layers can improve performance, but there is a danger of overfitting. Other loss functions, such as MSE loss, may be more suited to regression issues, but they may not necessarily increase model performance in binary classification tasks. On the other hand, adopting the dropout strategy can assist to enhance the model's performance on the validation set by preventing overfitting. A decent beginning point would be to train the model for 10-20 epochs and then evaluate its performance. If the performance is not adequate, try increasing the number of epochs until the required level of performance is achieved.
To discover the ideal configuration for a specific issue, it is now necessary to experiment with various methodologies and hyperparameters. By doing so, we can ensure that our model performs optimally and is resistant to new and unknown inputs.