**Improving the IMDB Neural Network Model**

**Introduction**

It refers to a computational model inspired by the structure and function of the human brain. These models are composed of interconnected layers of artificial neurons, which process information and learn from data to perform tasks like image recognition, natural language processing, and more.

**Objective**

The purpose of this project is to experiment with different modifications to a neural network model to improve its performance on the IMDB movie review dataset. The original model used two hidden layers with 16 units each, relu activation, and the binary\_crossentropy loss function. Several changes will be made to the architecture, activation functions, loss functions, and regularization techniques, and the impact on validation and test accuracy will be analyzed.

**Modifications of Nerual network Model:**

**Modifications**

**Goal:** To understand how Modifications in the model affects the model's performance.

1. **Hidden layers:**

Hidden layers in a neural network act as intermediaries, processing and transforming data from the input to the output. They help the network recognize complex patterns and relationships within the data, improving its ability to make accurate predictions. The number of hidden layers and neurons affects the network's performance more layers and neurons can capture more intricate details but may also cause the model to become too focused on the training data, reducing its effectiveness on new data. Striking the right balance is key to optimizing the network's performance.

Code1 has two hidden layers ## Code1 is original code

Code2 has three hidden layers ## Code2 is Modified Code

1. **Activation Function:**

An activation function in a neural network helps the model learn complex patterns by adding non-linearity to the output of each neuron. Without it, the network could only handle simple, linear problems. Popular activation functions include ReLU, which passes through positive values and zeros out negatives, sigmoid, which compresses values between 0 and 1, and tanh, which scales outputs between -1 and 1. The activation function you choose can significantly affect how well the network learns and makes predictions.

Code1 has “relu” activation function

Code2 has “tanh” activation function

1. **Loss function:**

In a neural network, a loss function measures how well the model's predictions match the actual results. It calculates the difference between the predicted output and the true target, guiding the network on how much it needs to adjust its weights during training. Common loss functions include mean squared error (MSE) for regression tasks and binary crossentropy or categorical crossentropy for classification tasks. The goal is to minimize the loss, which improves the model's accuracy by making better predictions.

Code1 has “Binary\_cross entropy”

Code2 has “Mean Squared error”

**4. Epochs:**

An epoch in a neural network refers to one full pass of the model through the entire training dataset. Since the data is usually processed in smaller chunks, an epoch means the model has seen and learned from all the data once. Models typically train over multiple epochs to gradually improve their accuracy, adjusting weights with each pass. While more epochs can help the model learn better, too many can lead to overfitting, where the model does great on the training data but struggles with new, unseen data.

Code1 has 20 epochs

Code2 has 30 epochs

1. **Regularization:**

Regularization is a method in machine learning that helps prevent overfitting, where a model learns the training data too well but struggles to perform on new data. It does this by adding a penalty to the loss function, making sure the model doesn’t become overly complex or dependent on specific features. Common regularization techniques include L1 and L2, which add penalties based on the size of the model's weights, and dropout, which randomly deactivates certain neurons during training to make the model more adaptable. Overall, regularization helps the model perform better on new, unseen data.

Code1 has no Regularization

Code2 has Dropout with dropout rate of 0.5 to prevent overfitting

**Observations:**

Code1 (Test Loss: 0.30898192524909973, Test Accuracy: 0.8751199841499329)

Code2 (Test Loss: 0.12793733179569244, Test Accuracy: 0.8597599864006042)

**Test Loss:** A lower test loss indicates that the model's predictions are closer to the actual values. It suggests that the model is performing well in minimizing the error between the predicted and actual labels.

**Test Accuracy:** A higher test accuracy shows that the model is correctly classifying a greater percentage of instances from the test dataset.

Finally, from our code Code2 (modifed code) has less test loss and also low test accuracy than Code1. So the model is overfitting to the training data.

Code2 has more complexity due to additional layers and units, which enables to capture the patterns in the data.

**Summary of output:**

In summary, the results suggest a trade-off between test loss and test accuracy. While lowering test loss is a desirable outcome, achieving high test accuracy is equally important for model performance on unseen data. The observations indicate that increasing model complexity and training capacity (as in Code 2) can improve loss but may not always lead to better accuracy. This highlights the need for careful tuning, including experimentation with different architectures, regularization techniques, and validation strategies to achieve a balanced model that generalizes well.

**Conclusion:**

In this report, we examined two neural network models using the IMDB dataset, focusing on how different design choices affect performance. Code 1 featured a straightforward architecture with two hidden layers, achieving a test accuracy of 87.5% and a test loss of 0.308, which showed good generalization to new data. On the other hand, Code 2 was more complex, using three hidden layers, more hidden units, the mean squared error (MSE) loss function, the tanh activation function, and dropout for regularization. While it had a lower test loss of 0.1279, its test accuracy dropped to 85.6%. These results highlight the importance of balancing model complexity and generalization to avoid overfitting. The study stresses the need for ongoing experimentation with different configurations and careful monitoring of performance to achieve a well-functioning model. Future efforts could include further hyperparameter tuning and exploring advanced techniques to improve accuracy and reliability.