We have evaluated the results of a network model that was trained with a batch size of 512 over 20 epochs during our initial analysis. Our goal was to assess the variation in the validation loss of the model under various configurations.

Hidden Layers: We played around with how many hidden layers the neural network had. The model initially used two hidden layers. We tested configurations with one and three hidden layers by changing this. The purpose of this investigation was to find out how test accuracy and validation are affected when the network architecture is altered in depth.

Hidden Units: The quantity of hidden units in every layer was also changed. We specifically tested configurations with 32, 64, and so on units. This change was made in order to track how the network's capacity affected its overall performance.

Loss Function: We investigated the application of the Mean Squared Error (MSE) loss function in addition to the Binary Cross Entropy loss function by default. Comparing these two with a regularization hyperparameter of 0.005 for MSE, We experimented with different node counts (16, 32, and 64) while keeping the three-layer architecture and the activation function "tanh" (sometimes called "Relu" for comparison).

1. Three Layers, BCE vs. MSE, and Tanh Activation (Regularization Applied) with 16 Nodes:

MSE: Using the tanh activation function and 16 nodes in each of the three hidden layers, the validation loss started at 0.17 and gradually dropped to 0.12. This shows that the model with MSE training was able to reliably reduce error during training, which led to a smaller validation loss overall.

BCE: On the other hand, the validation loss began at 0.4 and fluctuated throughout training when BCE was used as the loss function. It eventually rose to 0.58. This suggests that the BCE-trained model had difficulty convergent, leading to a higher and more variable validation loss.

2. Three layers, BCE versus MSE, and ReLU activation (Regularization Applied) with 32 nodes:

MSE: Using ReLU activation and 32 nodes in each hidden layer, the validation loss started at 0.15 and progressively dropped to about 0.13. This shows that the error was successfully minimized by the model trained with MSE and ReLU activation, leading to a comparatively low final validation loss.

BCE: With BCE as the loss function, the validation loss started out higher at 0.5 and fluctuated during training before coming down to about 0.4. The model trained with BCE, like the previous configuration, had difficulty convergent smoothly, which led to a higher and less stable validation loss than MSE.

These experiments demonstrate how various configurations, such as node counts, activation functions, and loss functions, affect the functionality of the model. When paired with suitable activation functions like tanh or ReLU, the MSE loss function selection typically produced smoother convergence and a lower final validation loss than BCE. Regularization techniques were also used to enhance the model's generalization capabilities, which resulted in a more stable and reduced validation loss in a variety of configurations.

To elaborate further:

• The MSE validation loss started out at 0.24 and decreased gradually to 0.1.
• Conversely, for BCE, the validation loss increased to 0.6 after initially starting at 0.4 and decreasing until the epoch.
• We also tested a different configuration that included 64 nodes, three layers, a ReLU activation function, and regularization techniques.
• In this instance, the validation loss for MSE started at 0.2, varied during training, and finally dropped to 0.13.
• In the same way, the validation loss for BCE began at 0.5 with this setup. changed while receiving instruction as well. elevated to 0.7 in the end.
• Ultimately, we investigated applying a tanh activation function to 64 nodes with three layers while applying techniques for regularization.
• The validation loss for MSE started at 0.2 in this configuration. during training, there were some fluctuations before the value dropped to about 0.1.
• In contrast, BCE's validation loss began at 0.5 when these settings were applied. showed some swings throughout the training phase, but eventually stabilized at that point.

Validation and Testing accuracy:
Across various neural network model configurations, the validation and testing accuracy stayed largely constant. More specifically:

Configuration: 3 layers, 32 nodes, tanh activation, regularization:
Validation Accuracy: The accuracy remained stable at 87% throughout the validation phase, despite some fluctuations. This suggests that the model performed consistently on data that had not yet been seen.
Training Accuracy: Over the course of training, the accuracy increased steadily and reached about 94%. The fact that the training and validation accuracy gaps were not too great indicates that the model did not overfit while still learning from the training set of data.

We also tested a different configuration with three layers and 64 nodes using the Tanh activation function:

Configuration: 3 layers, 64 nodes, tanh activation, and regularization

Validation Accuracy: The validation accuracy fluctuated, much like in the previous configuration, but it eventually settled at 87%. This consistency shows that the model's functionality remained stable in a variety of setups.
Training Accuracy: Up to approximately 95%, the accuracy of the training grew progressively. This shows that, probably as a result of the extra capacity offered by more nodes, the model learned well from the training set and attained a marginally higher accuracy than in the prior configuration.

Further, the neural network models' performance differed slightly in terms of the achieved accuracy on the test set in the experiments carried out with two different optimizers, Adam and RMSprop.

Adam Optimizer:
Test Set Accuracy: The model obtained an 88% test set accuracy using the Adam optimizer. Adam is an algorithm for optimizing adaptive learning rates that calculates adaptive learning rates for every parameter. It is renowned for being highly productive and successful in training deep neural networks for a variety of tasks.

RMSprop Optimizer:

Test Set Accuracy: In contrast, the model obtained a marginally lower accuracy of 86% on the test set when using the RMSprop optimizer. Moving averages of squared gradients are used by RMSprop, another adaptive learning rate optimization algorithm, to normalize the learning rates. It works especially well with noisy gradients and non-stationary objectives.

The disparity in the accuracy between the two optimizers may be related to the ways in which their individual optimization strategies adjust to the gradient updates throughout training.