**To improve the accuracy and reduce the losses, you can try the following strategies**

1. Adjust the model architecture: Experiment with different network architectures, such as increasing the number of hidden layers, changing the number of neurons in each layer, or trying different activation functions. This can help the model learn more complex patterns in the data.
2. Hyperparameter tuning: Try different learning rates, batch sizes, and optimizers. Adjusting these hyperparameters can have a significant impact on model performance. You can use techniques like grid search or random search to find the optimal set of hyperparameters.
3. Data augmentation: If you have a limited dataset, you can apply data augmentation techniques to artificially increase the size and diversity of your training data. This can help the model generalize better and improve its performance.
4. Regularization techniques: Implement regularization techniques such as L1 or L2 regularization, dropout, or batch normalization to prevent overfitting and improve generalization.
5. Increase the training data: If possible, gather more labeled data to train the model. More data can provide the model with a broader understanding of the patterns and improve its performance.
6. Evaluate the class distribution: Check if the classes in your dataset are balanced or imbalanced. If there is a class imbalance, consider using techniques like oversampling or undersampling to balance the class distribution.
7. Early stopping: Monitor the validation loss during training and implement early stopping to stop training if the validation loss does not improve for a certain number of epochs. This prevents overfitting and saves computational resources.

Remember that improving model performance is an iterative process. It may require trying different combinations of strategies and experimenting with various approaches to find the best results.

**When the model is not learning or improving its performance over time it could be an indication of a few possible issues:**

1. Underfitting: The model may not have enough complexity to capture the patterns in the data. You can try increasing the model's capacity by adding more hidden layers or increasing the number of neurons in each layer.
2. Data-related issues: It is also possible that the dataset itself may not contain enough information or variability for the model to learn from. You can consider collecting more diverse and representative data or applying data augmentation techniques to introduce more variations.
3. Hyperparameter tuning: The hyperparameters of the model, such as learning rate and optimizer, may not be suitable for the given dataset. Experimenting with different hyperparameter values or using techniques like learning rate schedules or adaptive learning rate optimizers (e.g., AdamW, RMSprop) can help improve the model's performance.
4. Class imbalance: If the dataset has imbalanced class distribution, the model may struggle to learn the minority class effectively. Applying techniques like oversampling, undersampling, or using class weights can help address this issue.

It is important to analyze the data, model architecture, and hyperparameters to identify potential areas for improvement. Additionally, monitoring the training process, analyzing learning curves, and evaluating the model's performance on unseen data can provide further insights into its effectiveness.

**Here are some suggestions to further improve the model's performance:**

1. Try different activation functions: Instead of using ReLU in all hidden layers, you can experiment with different activation functions such as sigmoid or tanh. Different activation functions may help the model learn different types of patterns and improve its representation capabilities.
2. Adjust the dropout rate: The dropout layers can help regularize the model and prevent overfitting. You can try different dropout rates, such as 0.2 or 0.3, and see if it improves the model's generalization performance.
3. Explore different optimization algorithms: The choice of the optimization algorithm can also affect the model's performance. You can try different optimizers such as SGD with momentum, RMSprop, or AdamW and see if it leads to better convergence and results.
4. Collect more data: If possible, consider collecting more training data to increase the diversity and amount of information available for the model to learn from. More data can help the model generalize better and improve its performance.
5. Experiment with different hyperparameters: Apart from the architecture, try different learning rates, batch sizes, and number of epochs. Fine-tuning these hyperparameters can have a significant impact on the model's performance.
6. Perform feature engineering: Analyze the dataset and consider performing feature engineering techniques to extract more informative features or engineer new features that might be relevant to the problem.