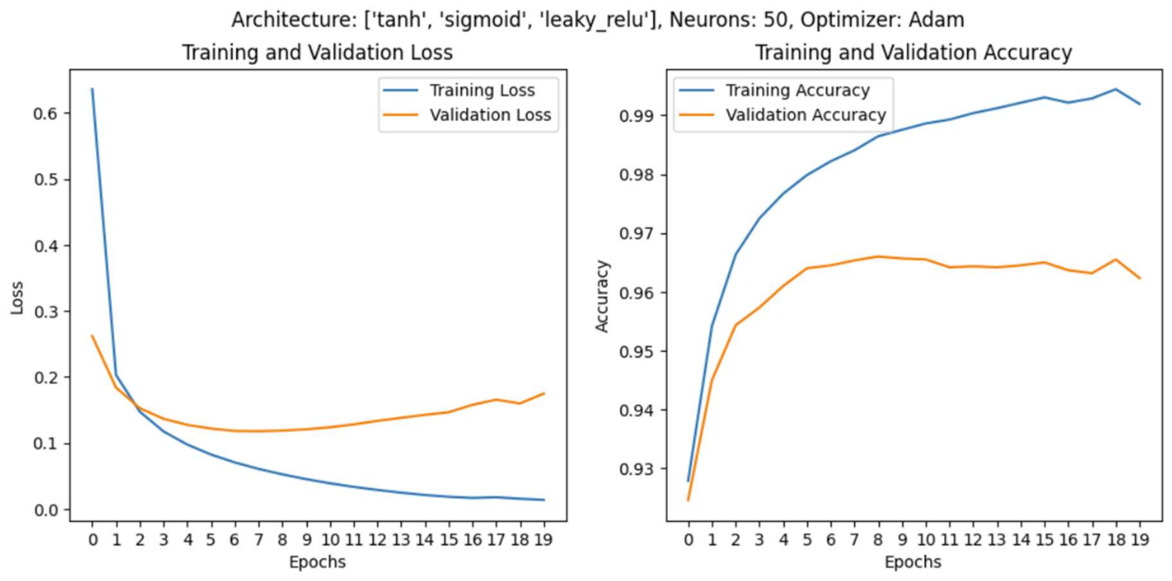
Problem 2

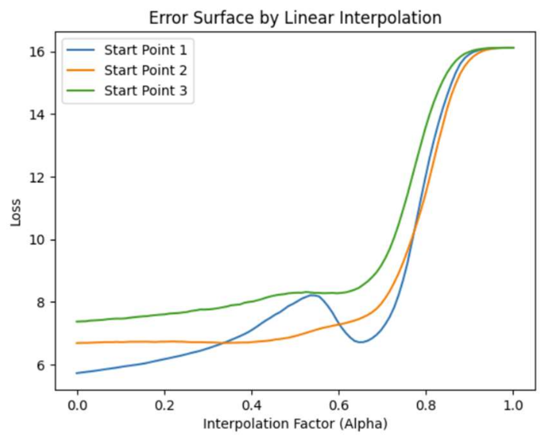
Description: For this question, I extended the 2-layer logistic regression to 6 architectures stated in the question. The neuron choices are 50 or 100. The optimizers are Adam or RMSProp. Then I plotted the loss & accuracy curves as well as linear interpolation plot for each setting. Finally, I discussed the impact of different hyperparameters on learning rates and activation functions.

Results

Architecture 1: 1st layer: tanh, 2nd layer: sigmoid, 3rd layer: leaky ReLU

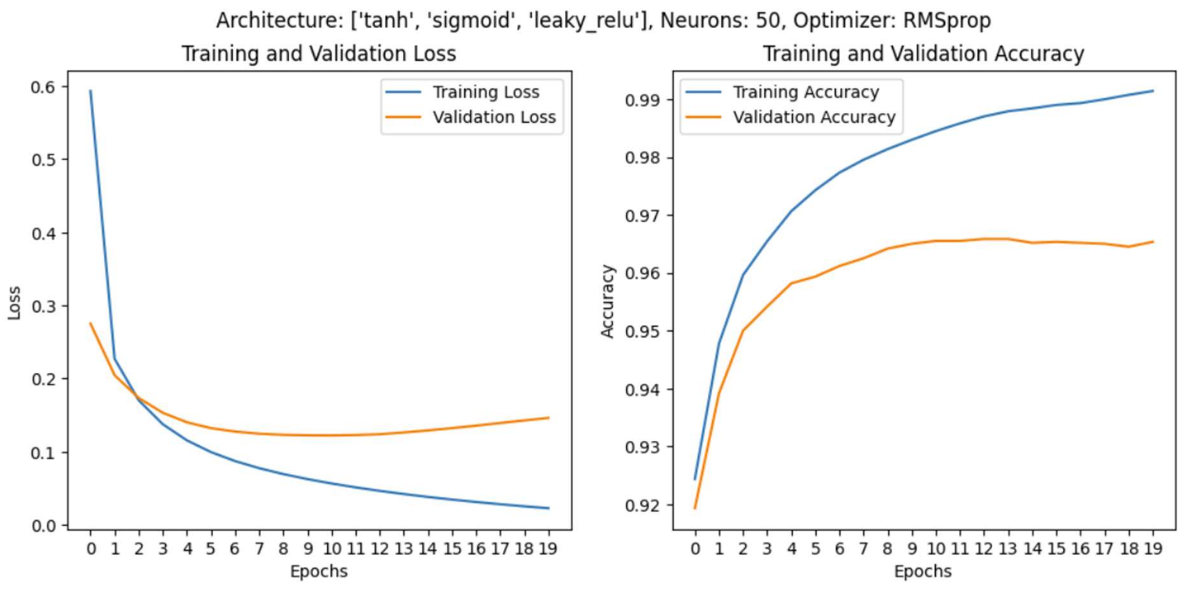
Adam as optimizer with 50 neurons:

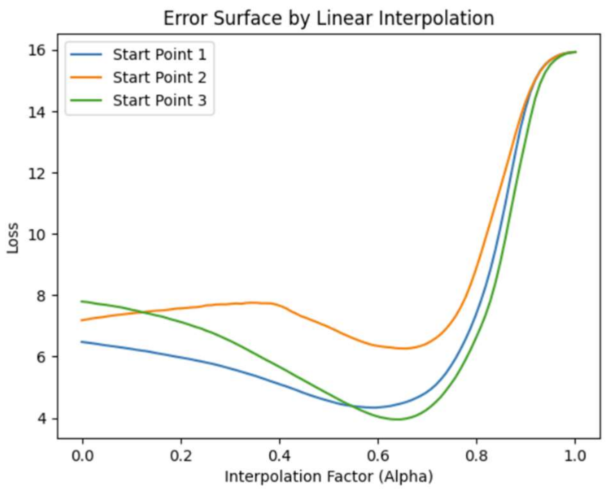




- Final Validation Accuracy: 96.54%

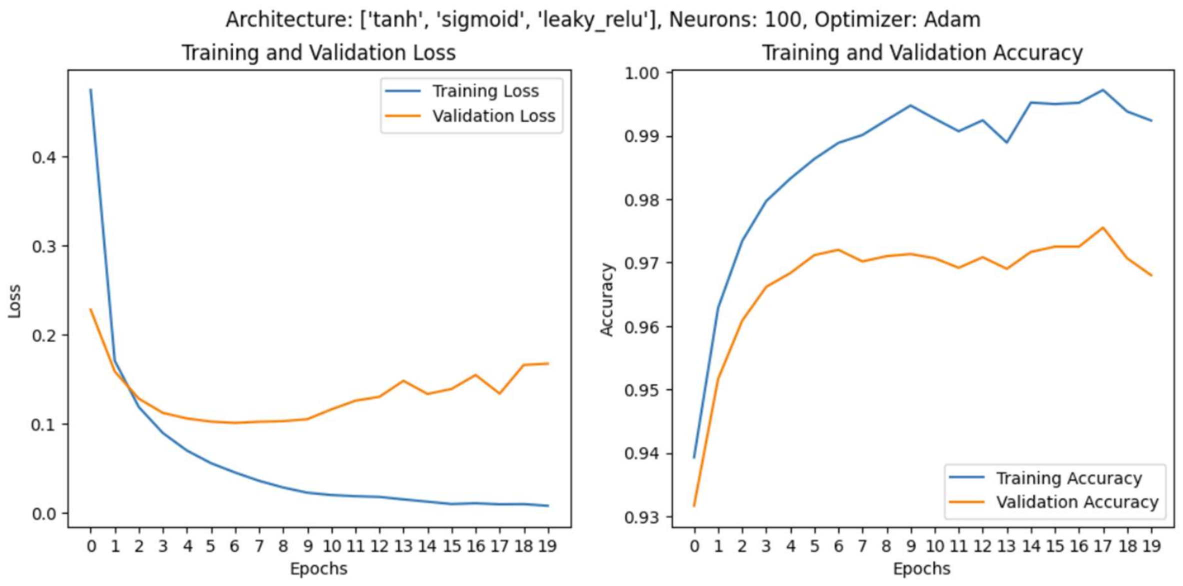
RMSProp as optimizer with 50 neurons:

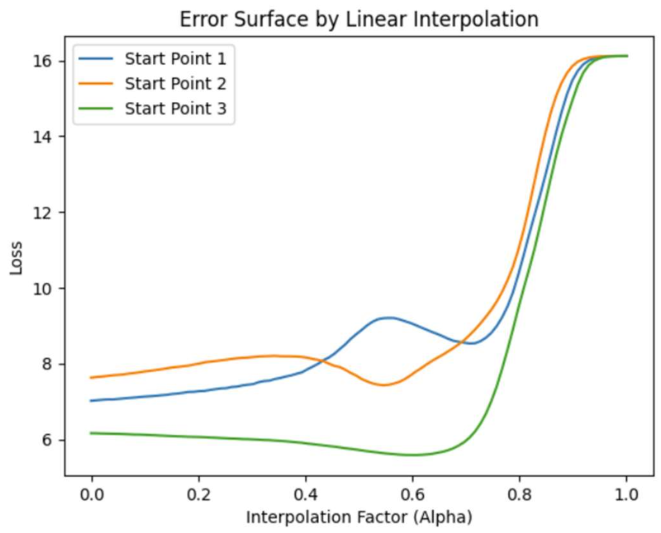




- Final Validation Accuracy: 96.86%

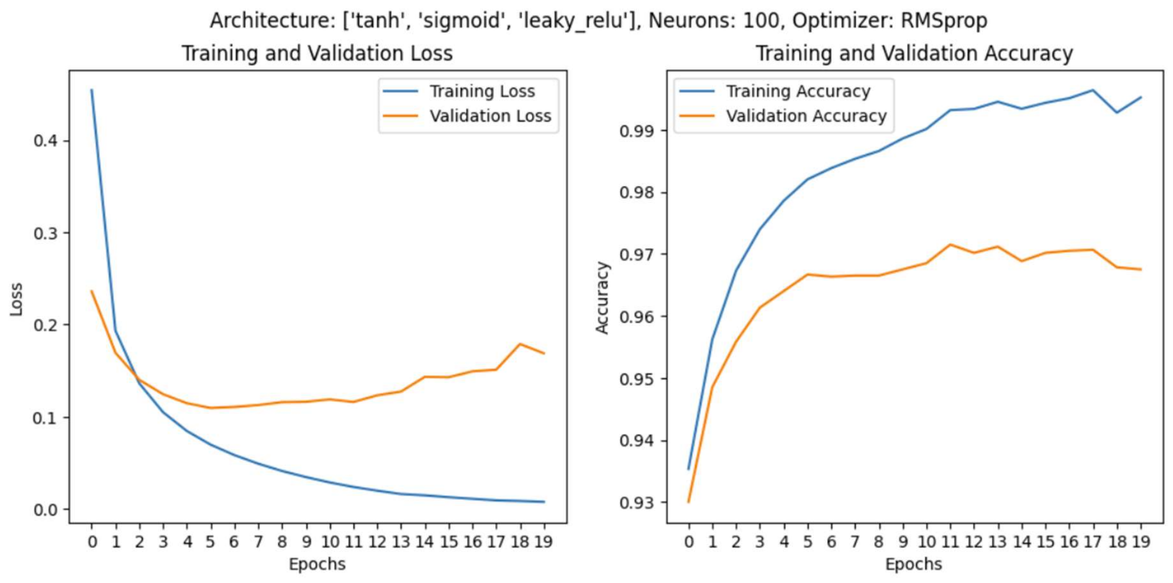
Adam as optimizer with 100 neurons:

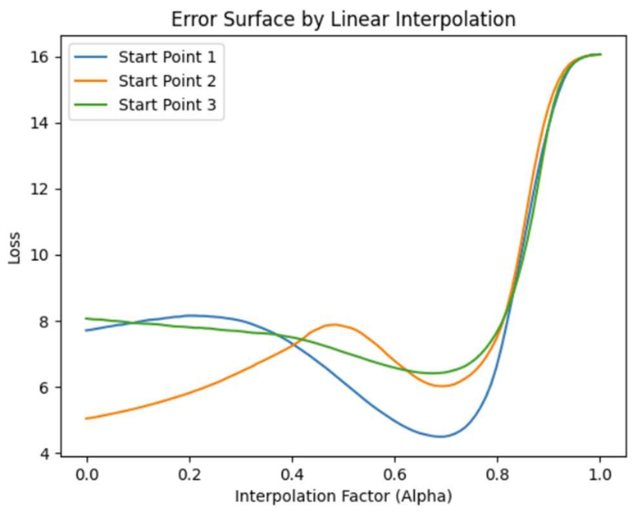




- Final Validation Accuracy: 97.10%

RMSProp as optimizer with 100 neurons:

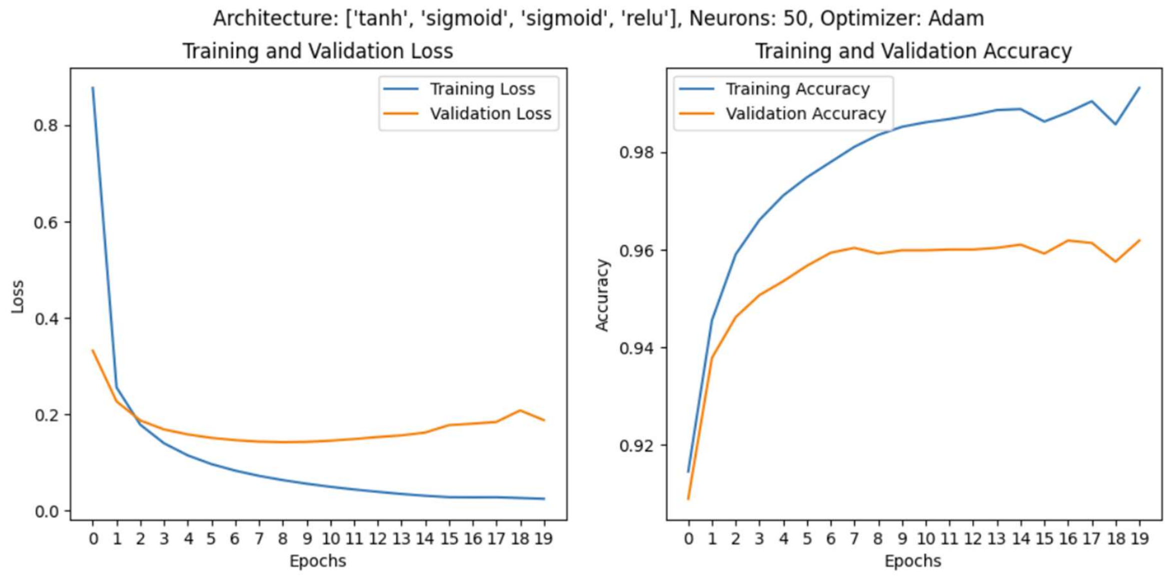


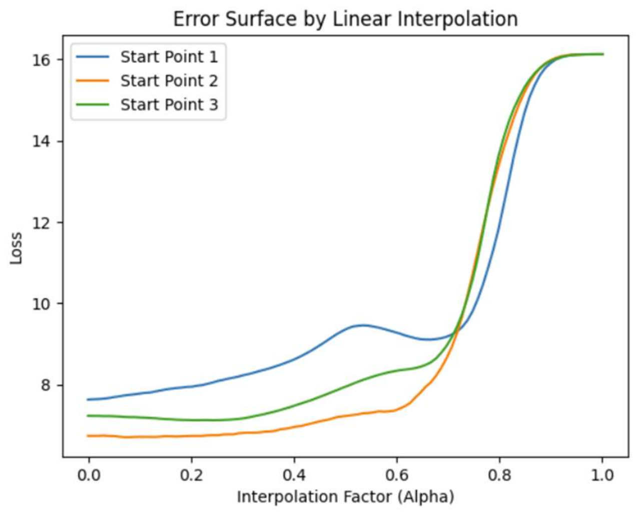


- Final Validation Accuracy: 97.28%

Architecture 2: 1st layer: tanh, 2nd layer: sigmoid, 3rd layer: sigmoid, 4th layer: ReLU

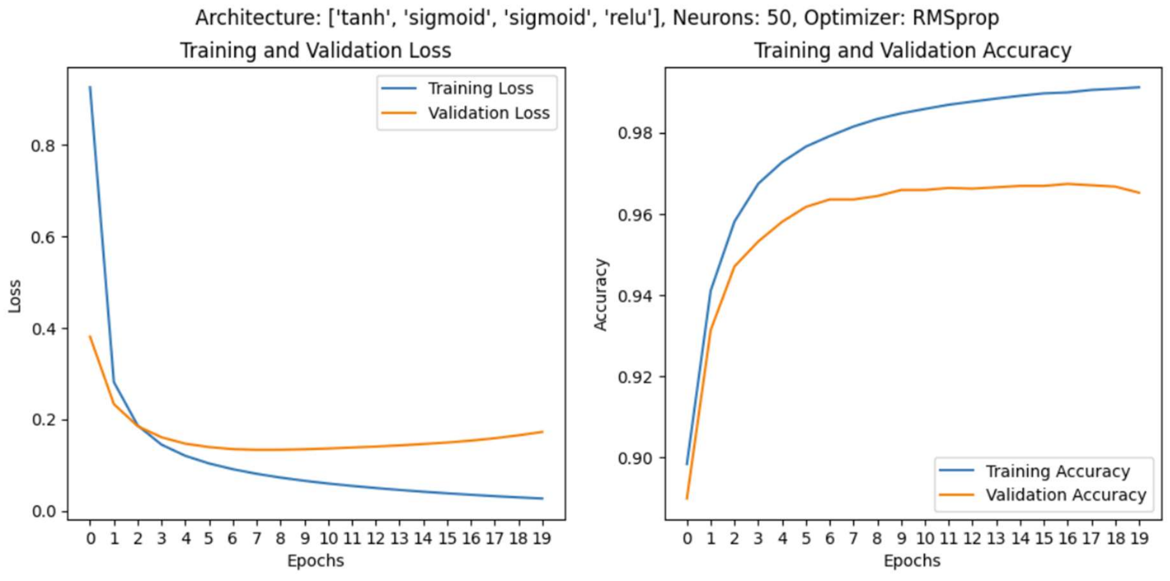
Adam as optimizer with 50 neurons:

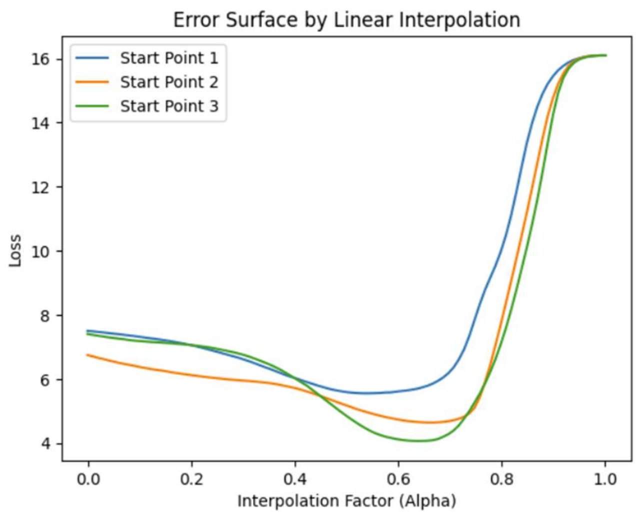




- Final Validation Accuracy: 96.52%

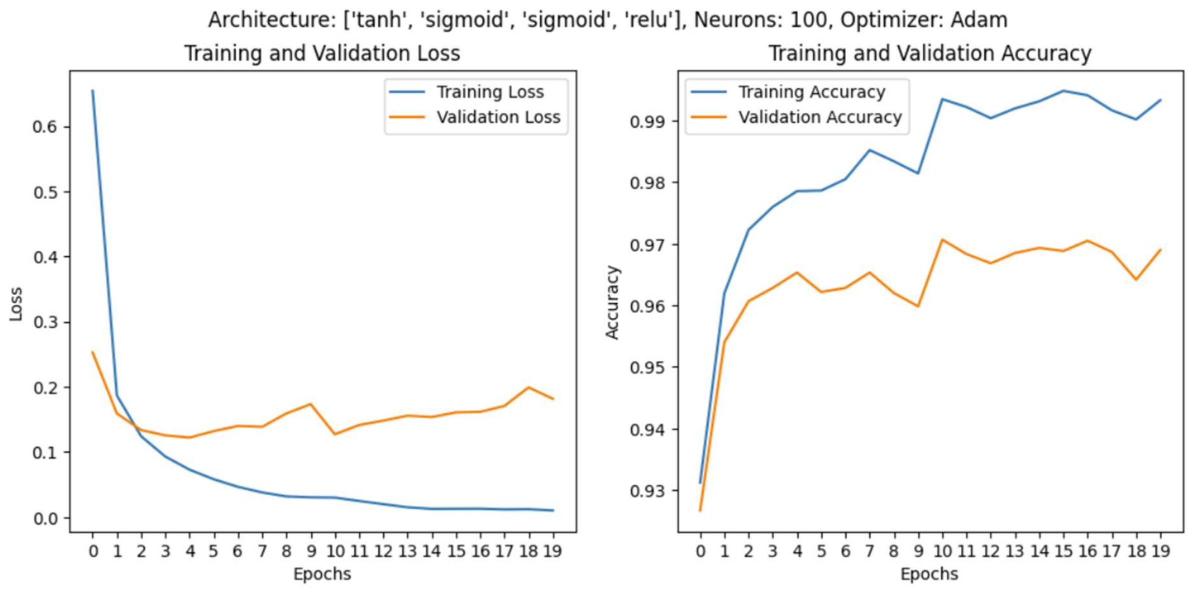
RMSProp as optimizer with 50 neurons:

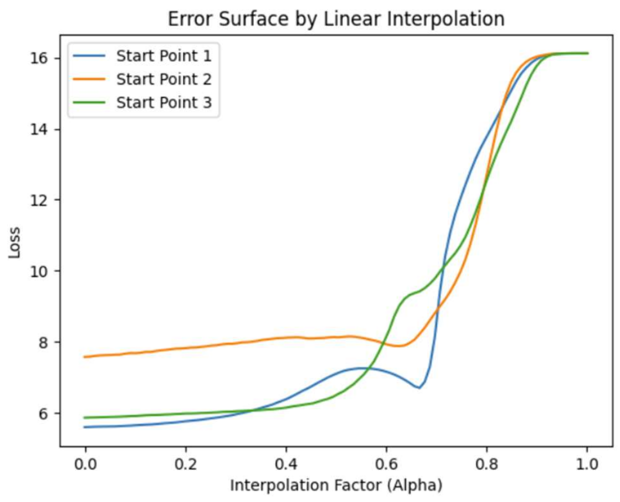




- Final Validation Accuracy: 96.71%

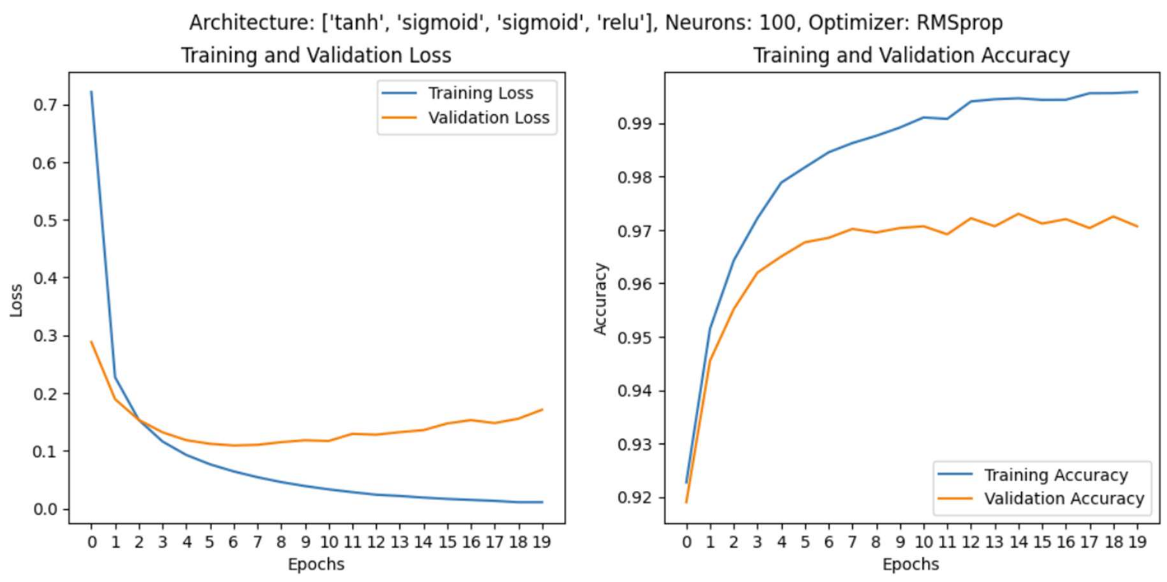
Adam as optimizer with 100 neurons:

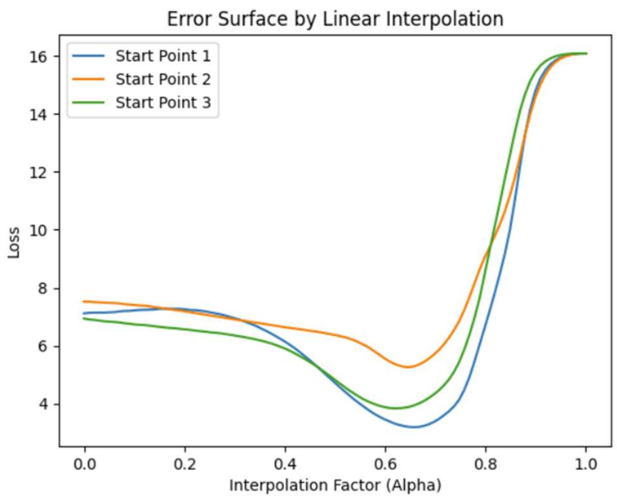




- Final Validation Accuracy: 96.95%

RMSProp as optimizer with 100 neurons:

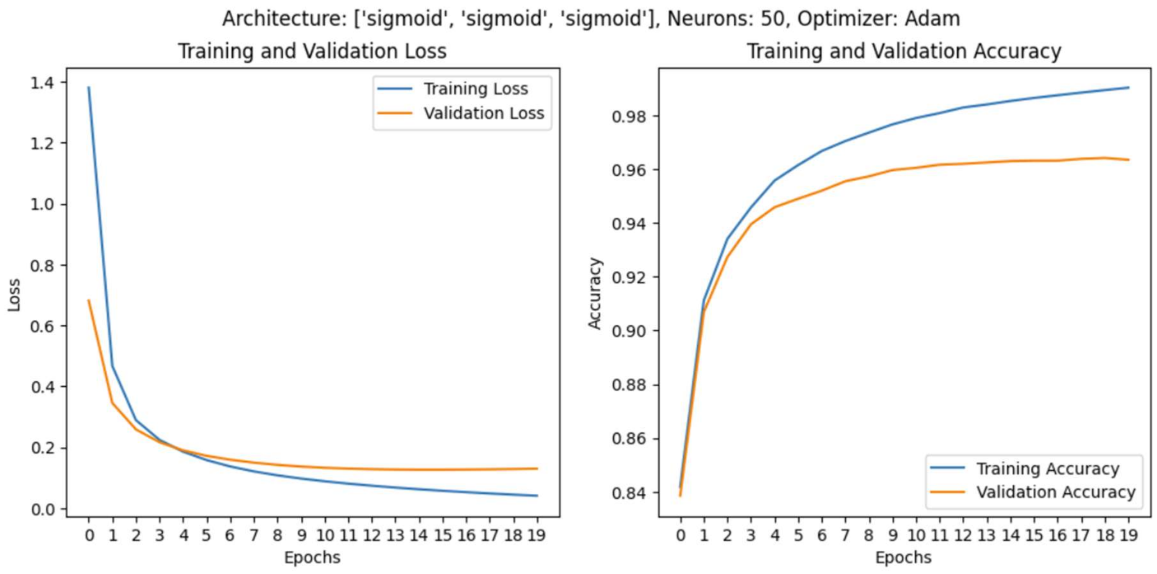


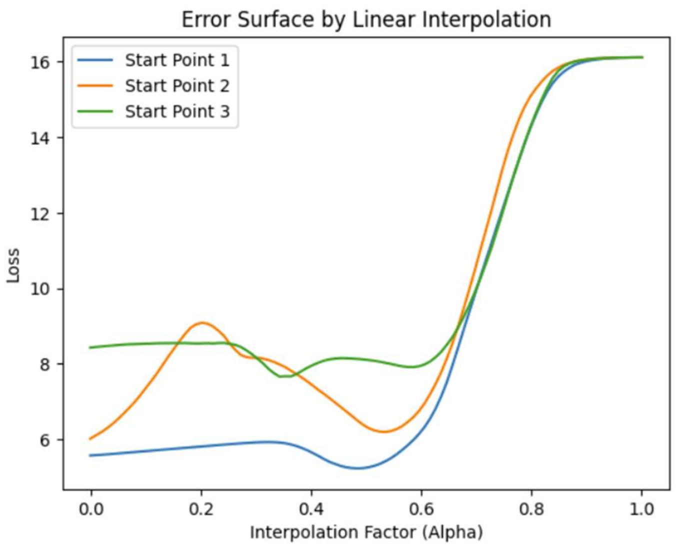


- Final Validation Accuracy: 97.36%

Architecture 3: 3 layers, all sigmoid

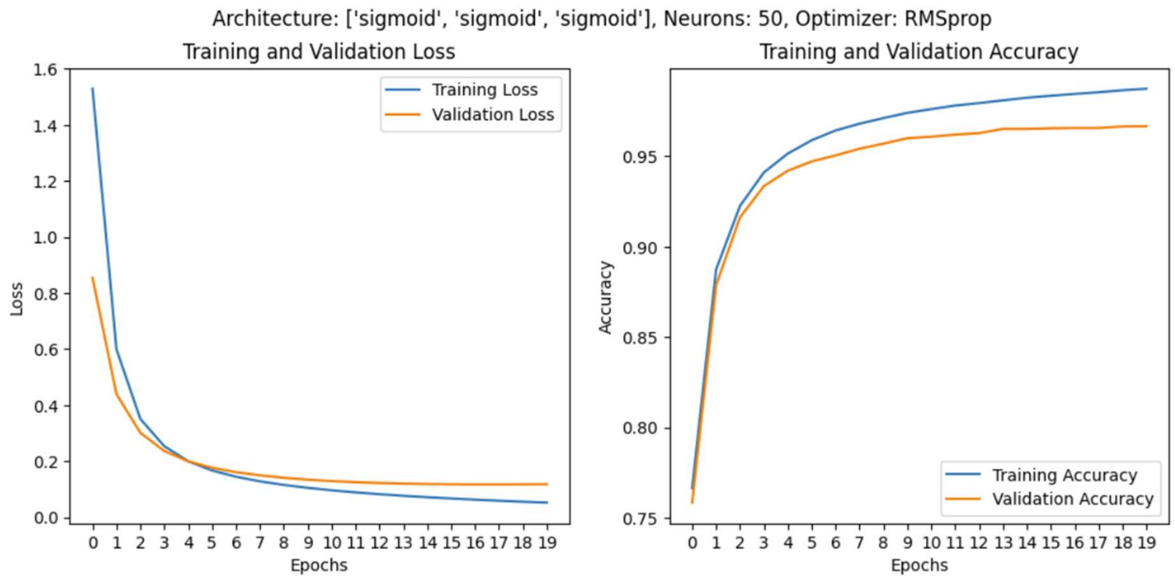
Adam as optimizer with 50 neurons:

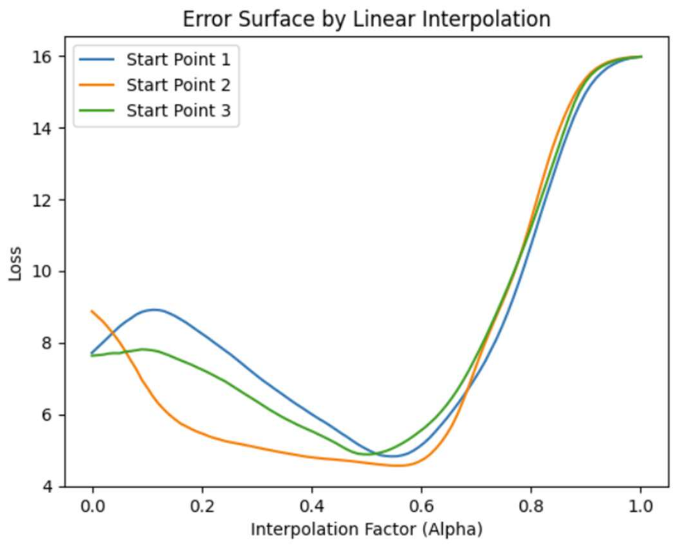




- Final Validation Accuracy: 96.90%

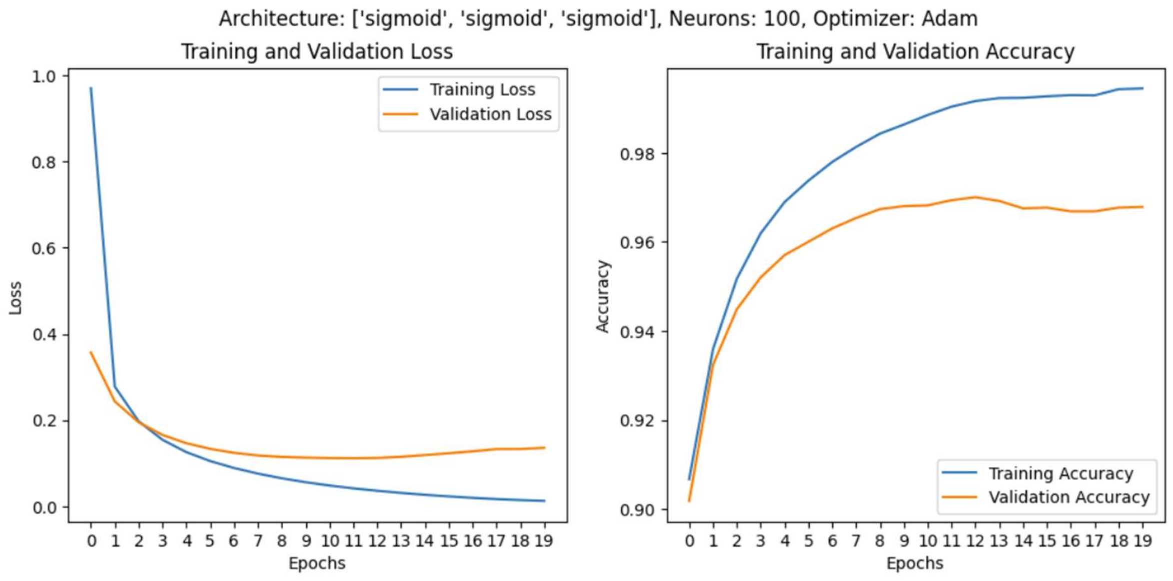
RMSProp as optimizer with 50 neurons:

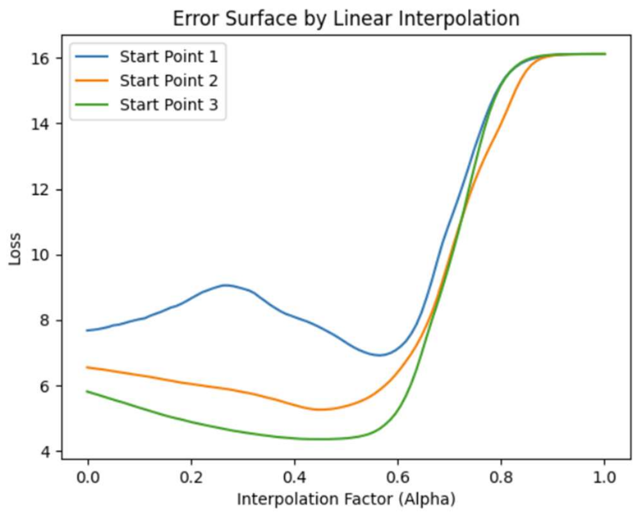




- Final Validation Accuracy: 96.84%

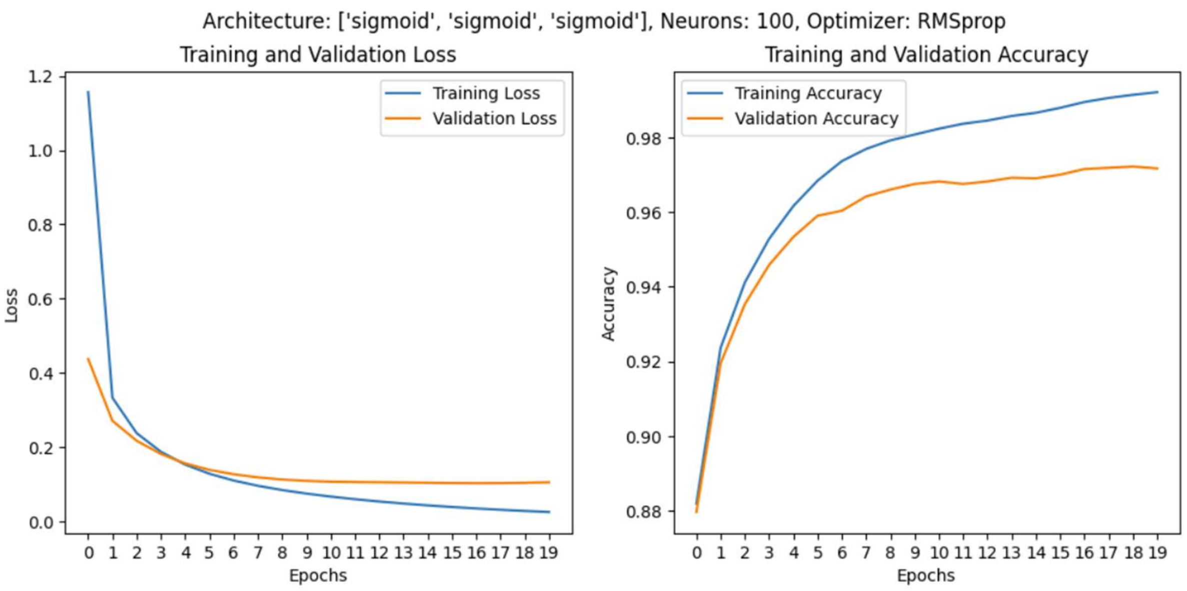
Adam as optimizer with 100 neurons:

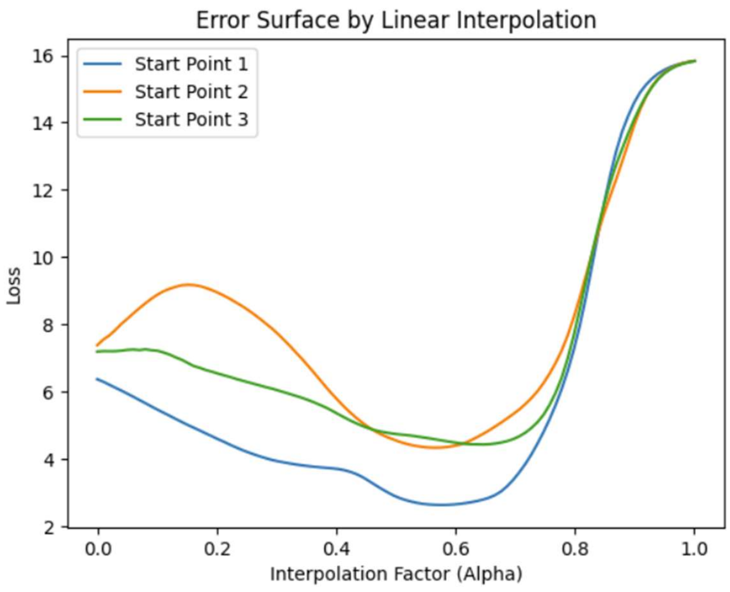




- Final Validation Accuracy: 97.25%

RMSProp as optimizer with 100 neurons:

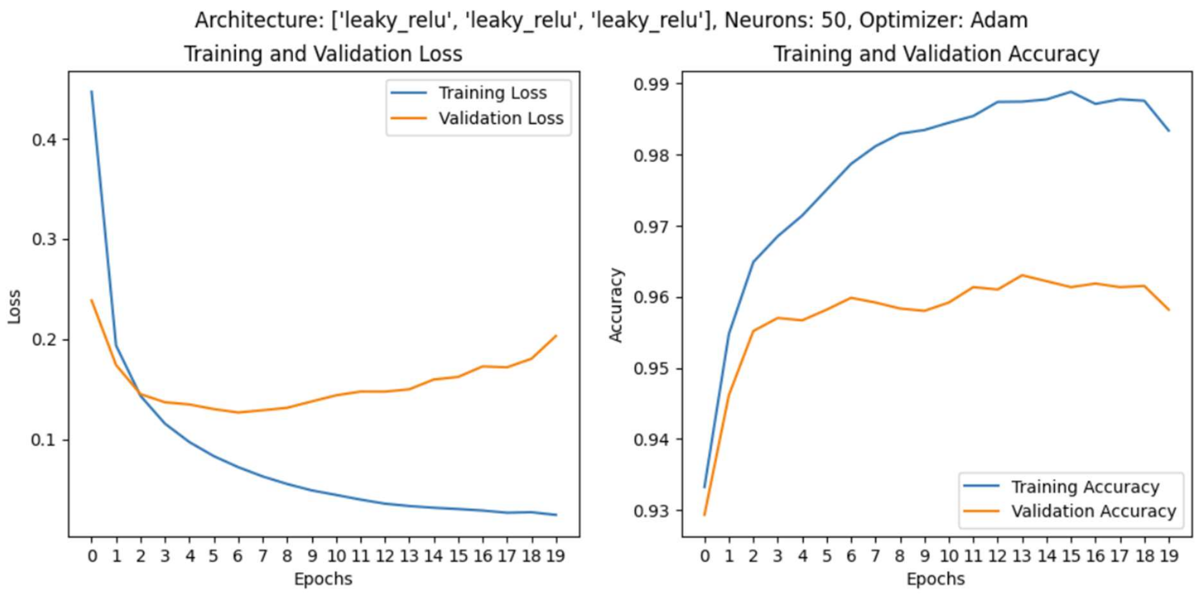


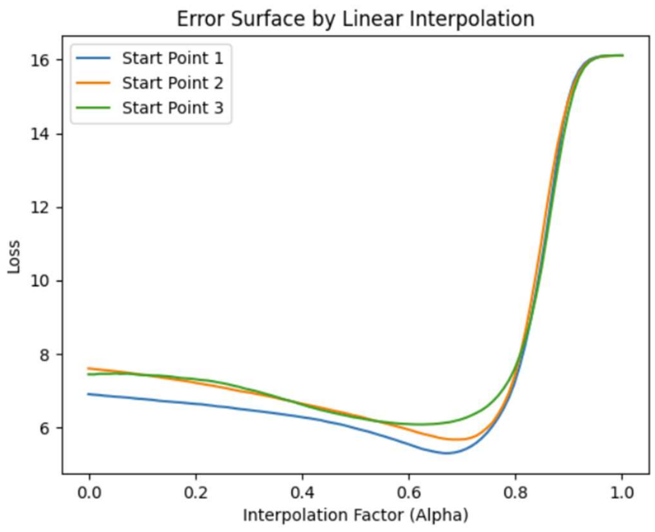


- Final Validation Accuracy: 97.63%

Architecture 4: 3 layers, all leaky ReLU

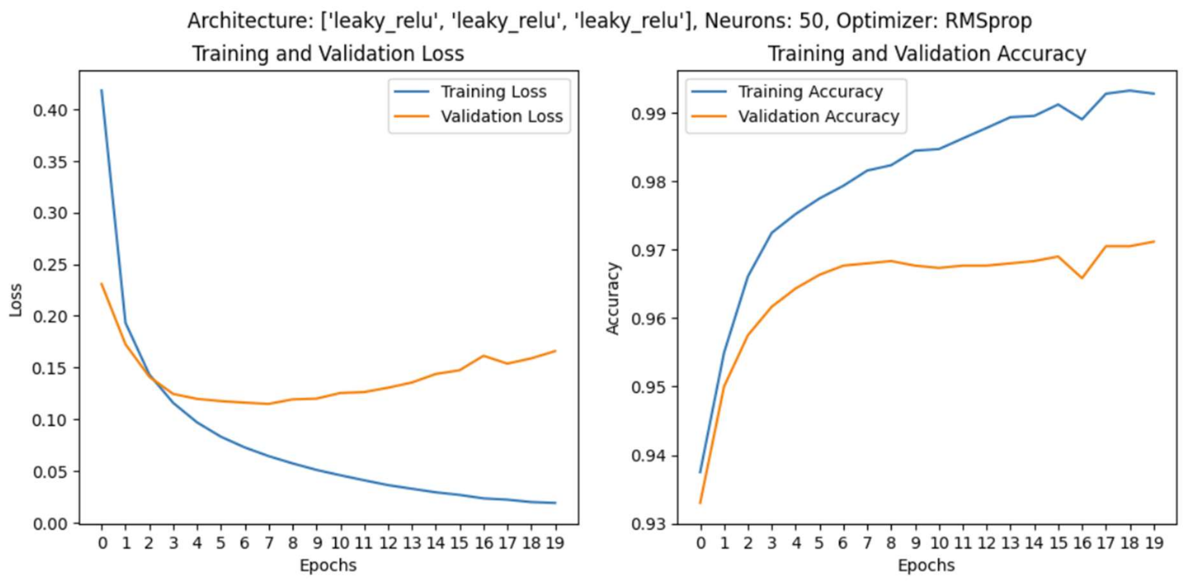
Adam as optimizer with 50 neurons:

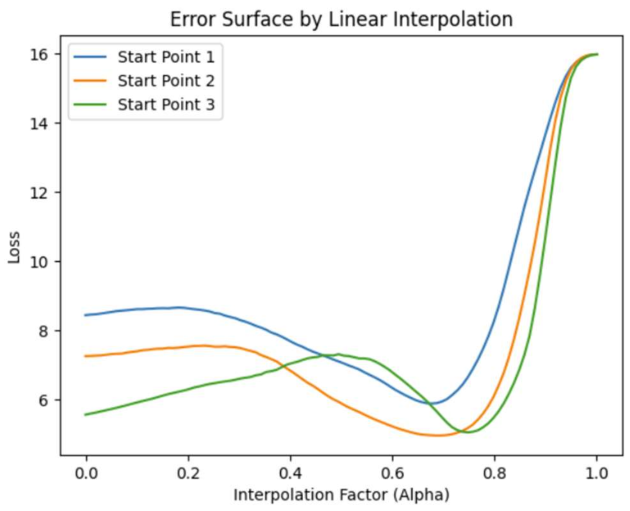




- Final Validation Accuracy: 96.45%

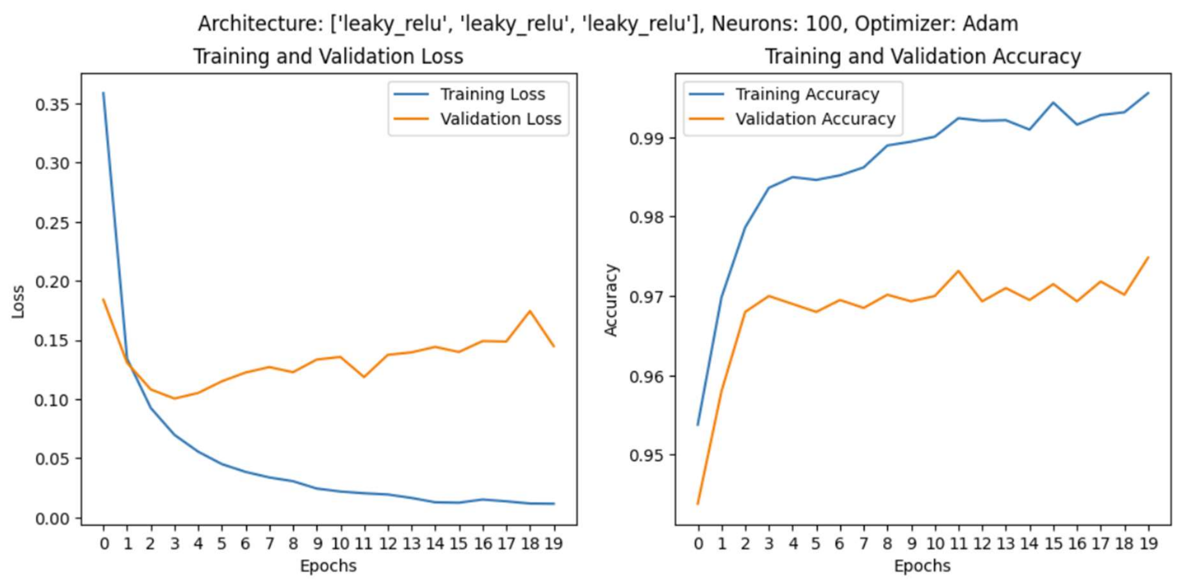
RMSProp as optimizer with 50 neurons:

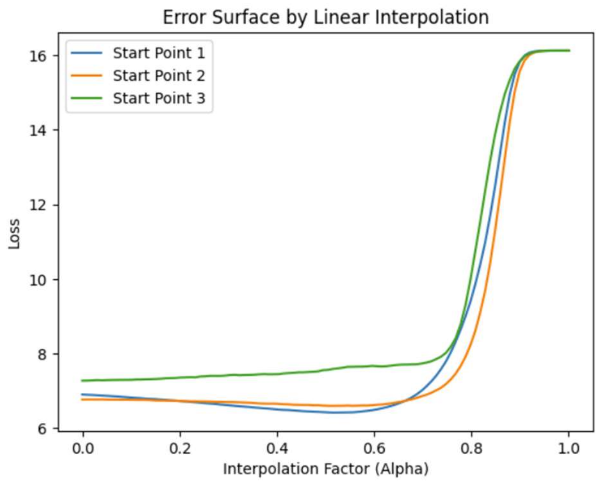




- Final Validation Accuracy: 96.97%

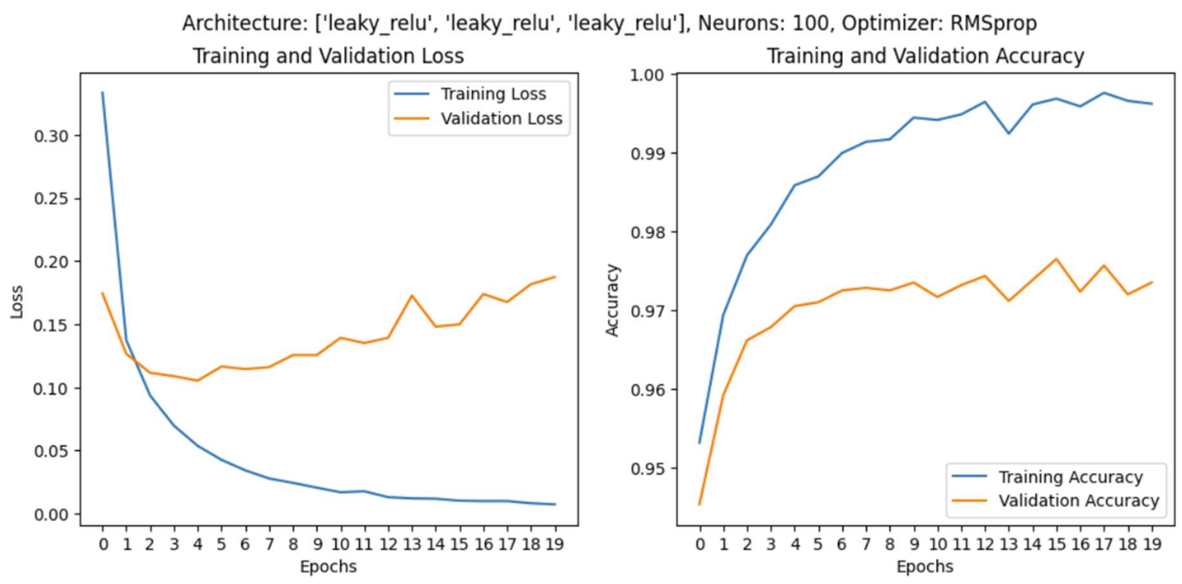
Adam as optimizer with 100 neurons:

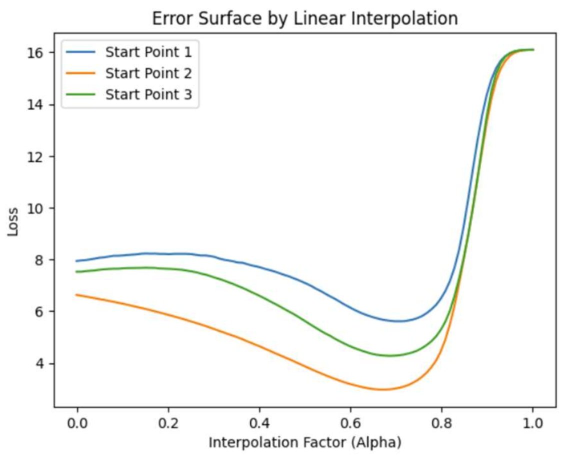




- Final Validation Accuracy: 97.55%

RMSProp as optimizer with 100 neurons:

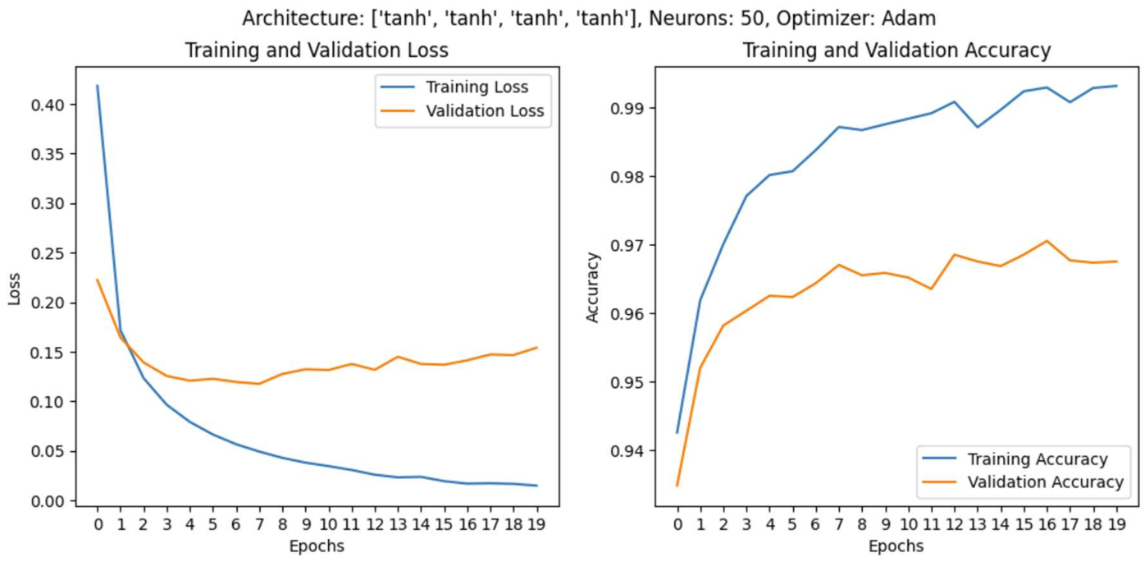


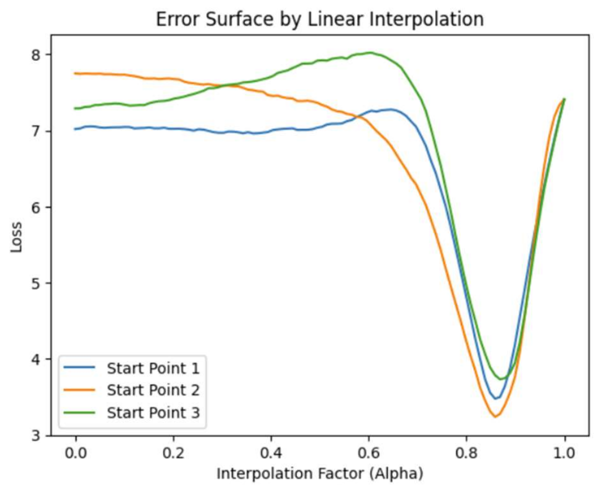


- Final Validation Accuracy: 97.75%

Architecture 5: 4 layers, all tanh

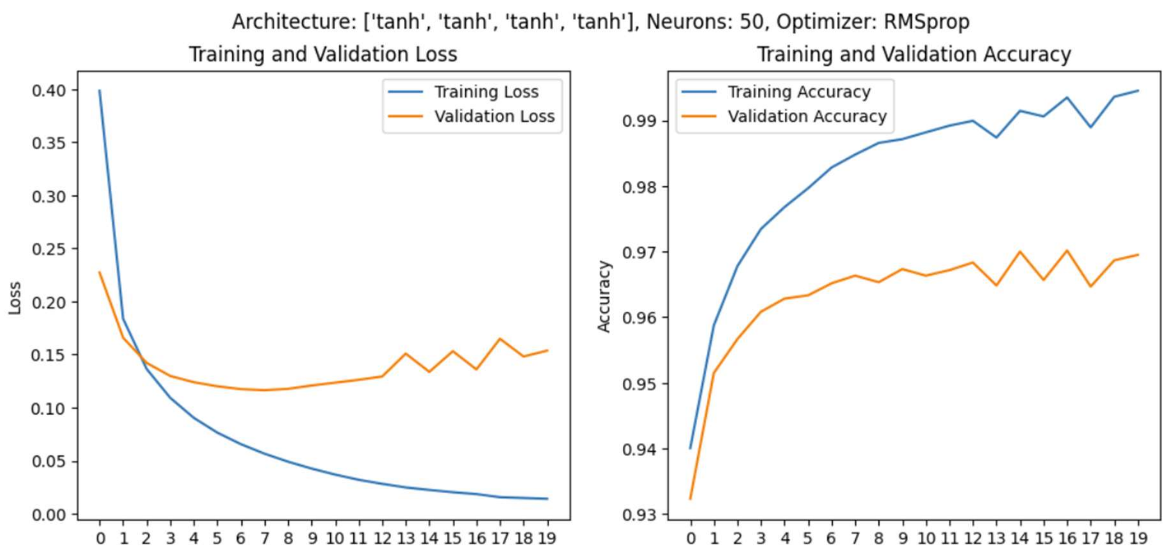
Adam as optimizer with 50 neurons:

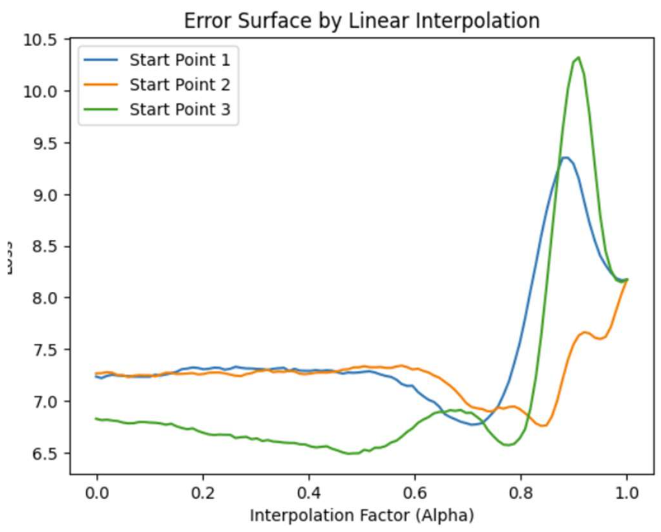




- Final Validation Accuracy: 96.77%

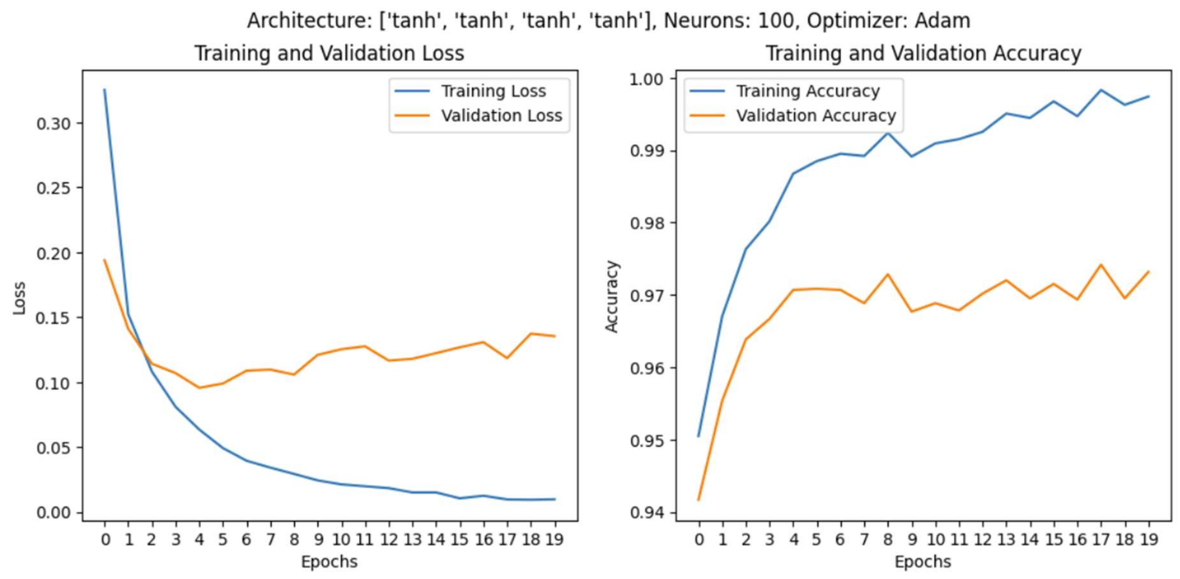
RMSProp as optimizer with 50 neurons:

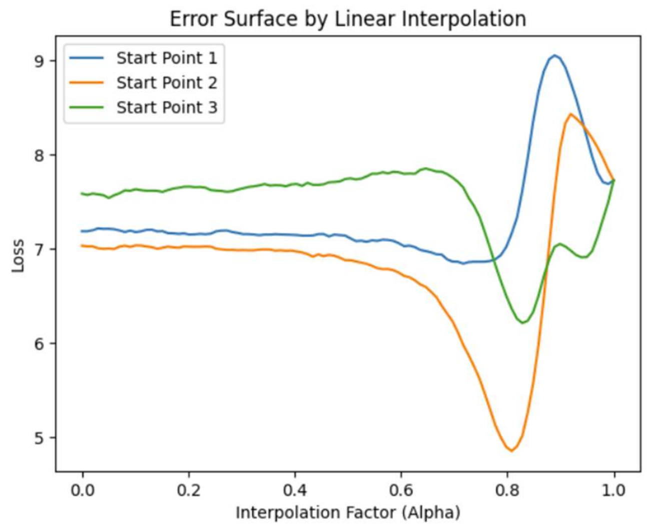




- Final Validation Accuracy: 97.19%

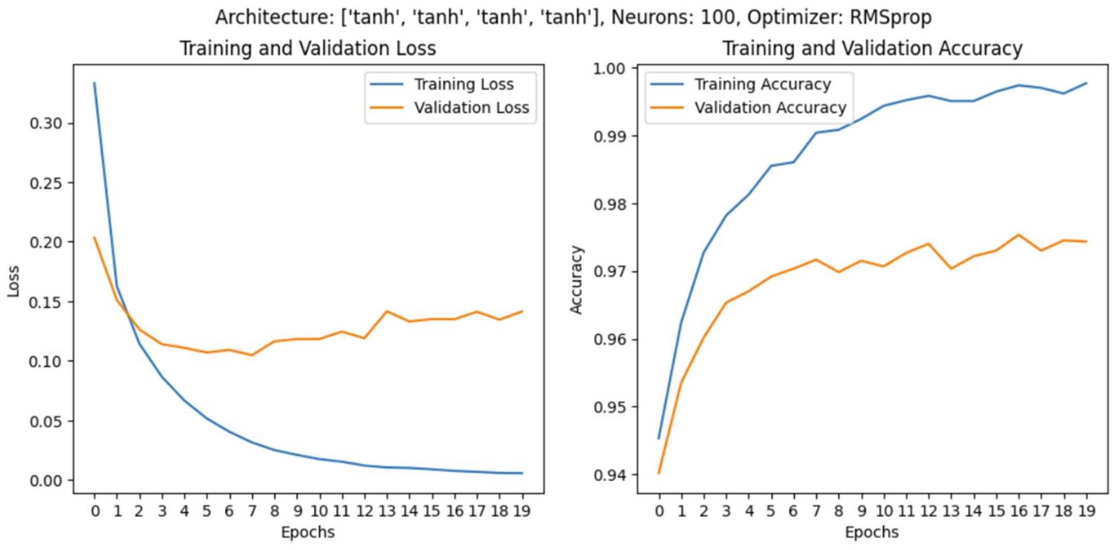
Adam as optimizer with 100 neurons:

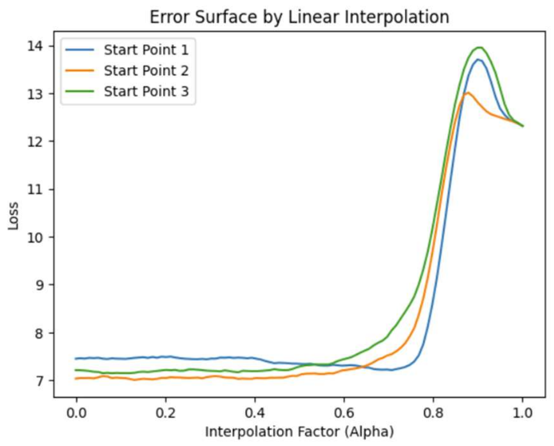




- Final Validation Accuracy: 97.57%

RMSProp as optimizer with 100 neurons:

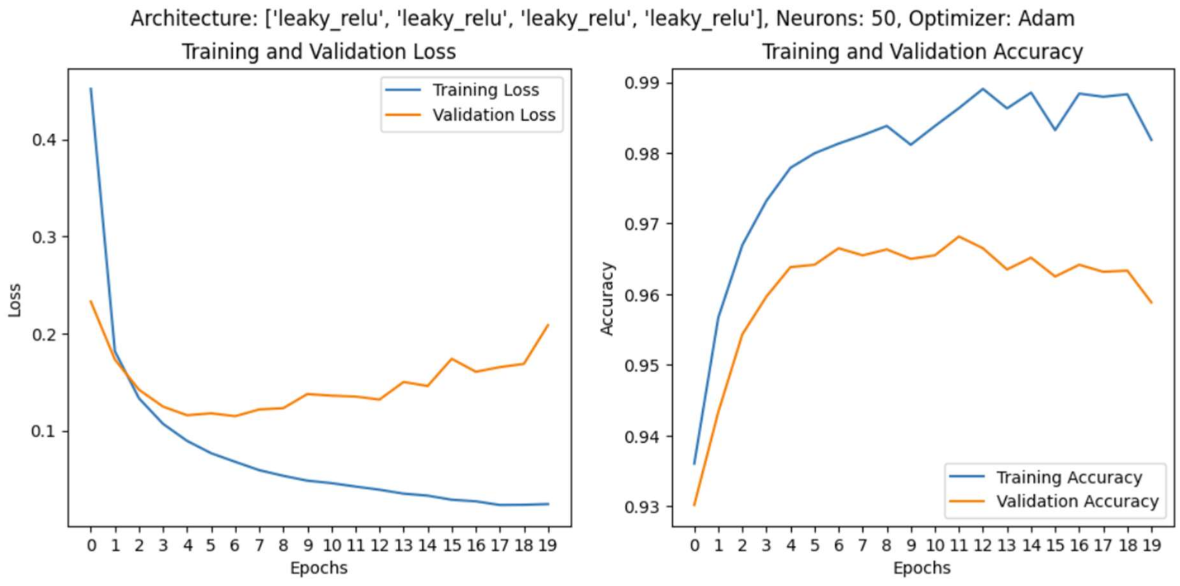


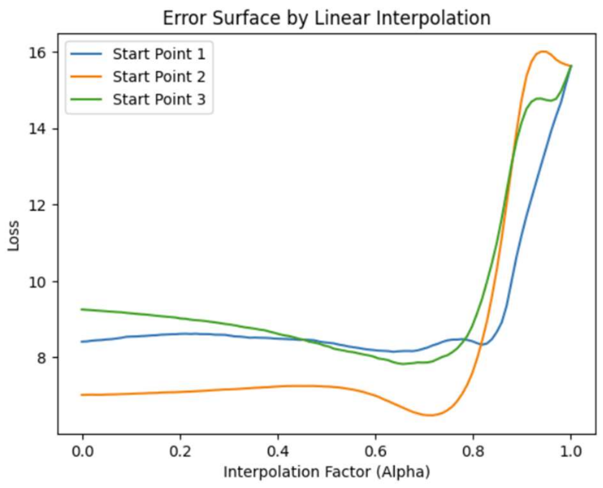


- Final Validation Accuracy: 97.32%

Architecture 6: 4 layers, all leaky ReLU

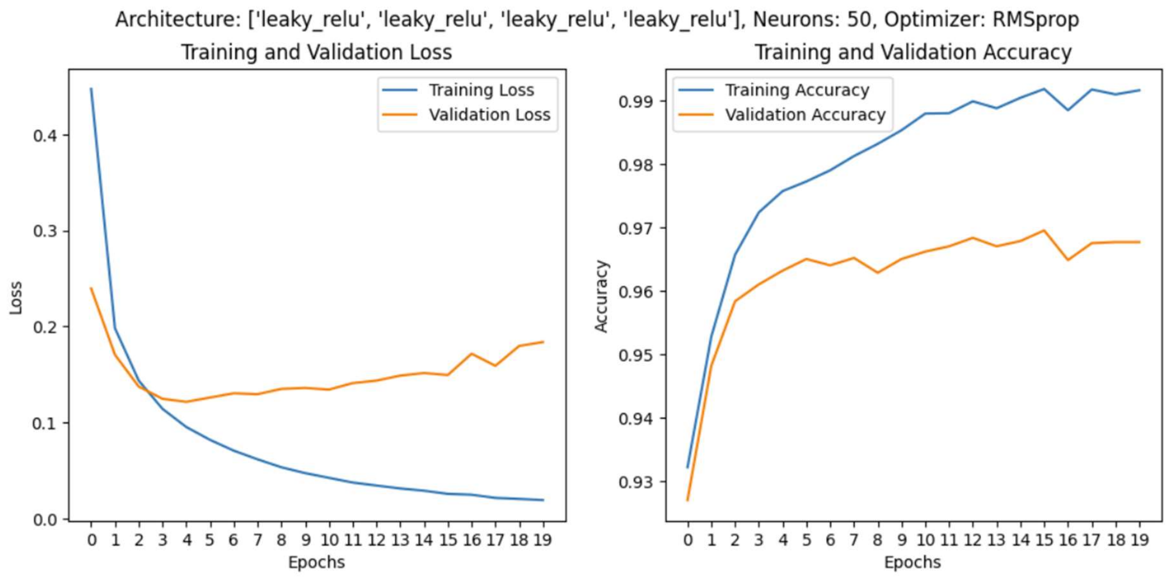
Adam as optimizer with 50 neurons:

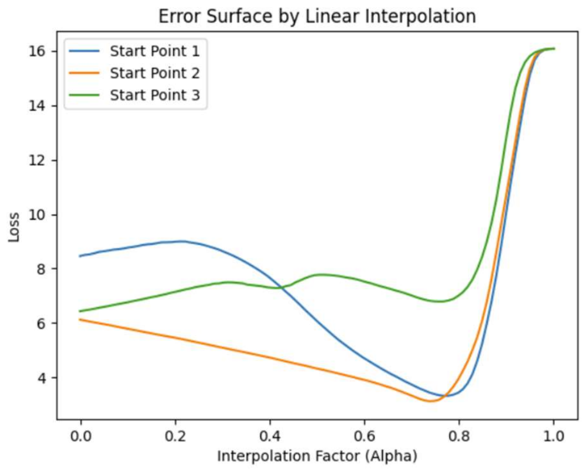




- Final Validation Accuracy: 96.27%

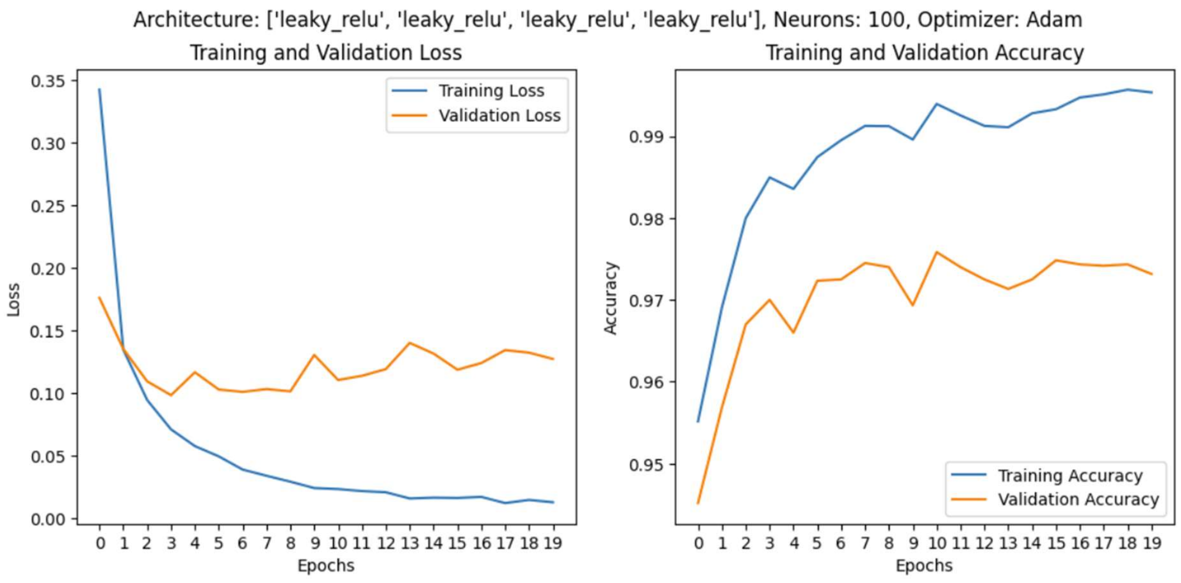
RMSProp as optimizer with 50 neurons:

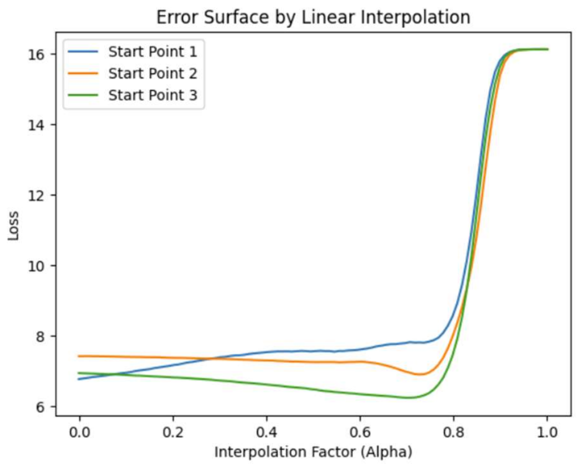




- Final Validation Accuracy: 96.95%

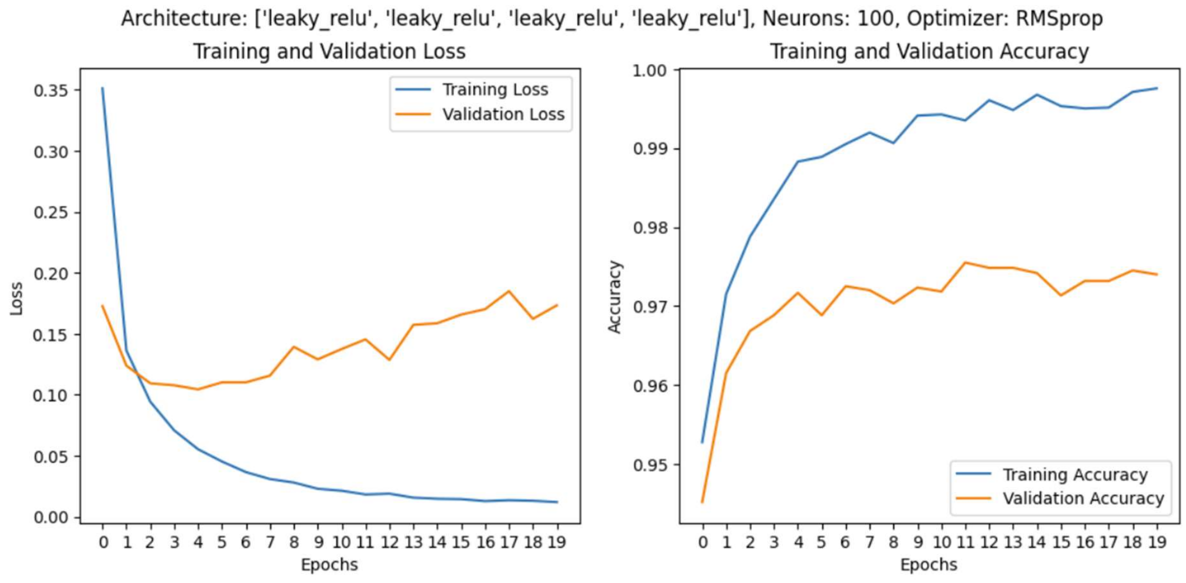
Adam as optimizer with 100 neurons:

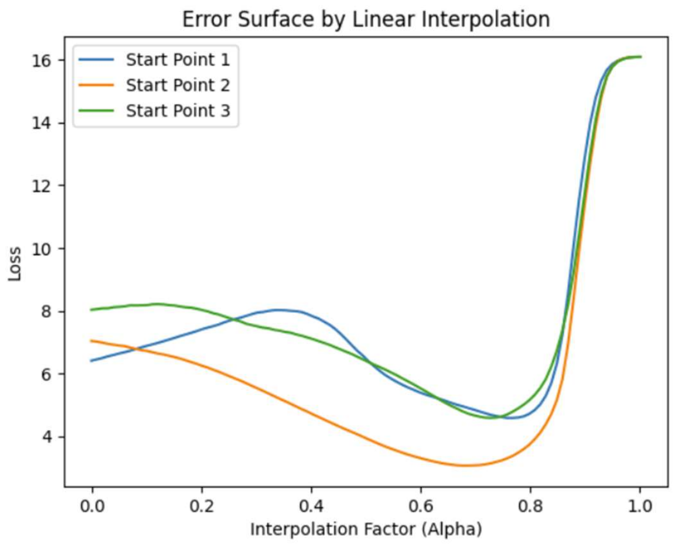




- Final Validation Accuracy: 97.57%

RMSProp as optimizer with 100 neurons:





- Final Validation Accuracy: 97.78%

About the Error Surface

For the 2D error surface by linear interpolation, we interpolated between a few different randomly initialized starting weights and the final optimized weight to visualize the error surface.

Note: For the linear interpolation plots, I use the formula below:

$$interpolated\ weights = (1-\alpha) * start\ weight + \alpha * final\ weight$$

Overall Accuracy Results on Validation data

Architecture	50 Neurons (Adam/RMSProp)	100 Neurons (Adam/RMSProp)
1	**0.9654 / 0.9686**	**0.9710 / 0.9728**
2	**0.9652 / 0.9671**	**0.9695 / 0.9736**
3	**0.9690 / 0.9684**	**0.9725 / 0.9763**
4	**0.9645 / 0.9697**	**0.9755 / 0.9775**
5	**0.9677 / 0.9719**	**0.9757 / 0.9732**
6	**0.9627 / 0.9695**	**0.9757 / 0.9778**

Conclusion & Analysis

1. Impact of Neurons per Layer: Increasing the number of neurons from 50 to 100 consistently improved the validation accuracy across all architectures. This suggests that higher neuron counts allow the model to capture more complex patterns in the data, leading to better generalization.

2. Optimizer Performance: The RMSProp optimizer slightly outperformed Adam in most configurations, particularly with higher neuron counts (100). This indicates that RMSProp may be more effective for these architectures and the problem dataset, potentially due to better handling of adaptive learning rates in layered structures.

3. Architecture and Activation Function Influence: Among the architectures tested, the 4-layer tanh (architecture 5) and 4-layer leaky ReLU (architecture 6) achieved the highest validation accuracy with 100 neurons, indicating that deeper architectures with non-linear activations (tanh, leaky ReLU) help in capturing complex non-linear relationships, which may contribute to better overall model performance.

Different Learning Rate

For this part, I choose the architecture 1 with 50 neurons and Adam as optimizer to train with different learning rates [0.0001, 0.001, 0.01, 0.1, 0.5]. We want to discover how the learning rate affects the training performance over time.

Result

Learning rate	Validation accuracy	Testing accuracy	Running time (in second)
0.0001	**0.9523**	**0.9506**	**460.09**
0.001	**0.9648**	**0.9648**	**456.17**
0.01	**0.9624**	**0.9642**	**436.34**
0.1	**0.1141**	**0.1134**	**434.87**
0.5	**0.1141**	**0.1134**	**432.27**

Conclusion

1. Optimal Learning Rate Range: Learning rates of 0.001 and 0.01 achieved the highest validation and testing accuracy, suggesting that these values strike a good balance between convergence speed and stability for this architecture and optimizer combination.

2. Effect of Very Low Learning Rate (0.0001): While the lowest learning rate (0.0001) provided relatively high accuracy, it took the longest time to converge, indicating that very low learning rates can increase training duration without significant accuracy benefits over slightly higher learning rates.

3. Instability with High Learning Rates (0.1 and 0.5): Both the 0.1 and 0.5 learning rates resulted in significantly lower accuracy scores, which indicates that these high values lead to unstable training, potentially causing the model to diverge or fail to find an optimal solution.

4. Marginal Time Differences Across Learning Rates: The running time slightly decreased as the learning rate increased, with only marginal differences across learning rates, suggesting that higher learning rates reduce the number of required iterations but may compromise accuracy if set too high.