1.

Hyper-parameters		Training set		Validation set		Test set
Learning Rate	Epoch	Loss	Accuracy	Loss	Accuracy	Accuracy
0.00001	100	0.542	69.3%	0.677	59.3%	58.06%
0.0001	500	0.369	85.7%	0.603	70.4%	74.19%
0.0005	1000	0.130	95.8%	1.336	71.6%	58.06%

2. Even after running for 1000 epochs, the performance of parameter set 3 on the test set is only equal to that of parameter set 1, indicating that parameter set 3 performs very poorly. The third parameter set is obviously to be overfitting, as shown in Figure 2, the validation loss increases dramatically after epoch 200, suggesting that the learning rate may be too large, which makes it difficult for the model to converge properly. Although parameter set 2 has the best performance among these three parameter sets, Figure 2 reveals a gap between the training and validation accuracies, confirming the presence of overfitting. To improve test set accuracy, I suggest trying a learning rate of 0.00005 with 500 epochs while also tuning the other hyperparameters simultaneously.

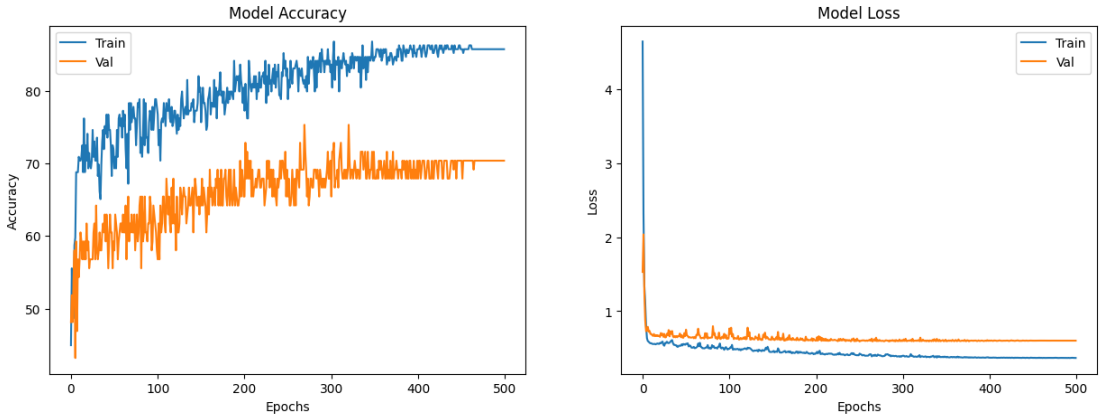

Figure 1. Accuracy and Loss of parameter set 2 (learning rate=1e-4, epoch=500)

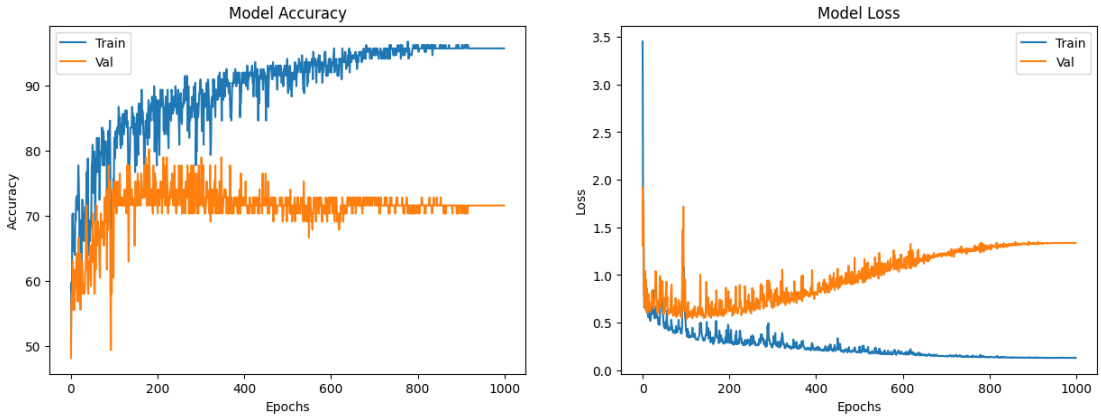

Figure 2. Accuracy and Loss of parameter set 3 (learning rate=5e-4, epoch=1000)

3.

In Lab 2, the discrepancy in accuracy between the training and test datasets can often be attributed to overfitting. When a model learns the noise and specific patterns of the training data too well, it performs excellently on that dataset while failing to generalize to unseen data. This gap may also stem from differences in data distributions; if the test data comes from a slightly different distribution than the training data, the model's performance may drop because it has not learned to handle those variations. Lastly, insufficient training data can lead to a model that does not capture the underlying patterns comprehensively, causing a disparity in performance between the training and test sets.

4.

There are three methods for feature selection: filter, wrapper, and embedded. Filter methods only assess the correlation between variables and the predicted values to eliminate the least relevant features. Since this approach does not consider the relationships among the variables, it may select features that, while highly correlated with the target, actually contain redundant information. Wrapper methods evaluate every subset of features using machine learning models and search strategies. Embedded methods combine the approaches of Filter and Wrapper methods by performing feature selection simultaneously during model training. It begins by training the model using Filter techniques to accelerate the process while retaining the advantage of Wrapper methods in accounting for the interactions among features, and they incorporate penalty mechanisms to prevent overfitting.

When applying data mining and machine learning to high-dimensional datasets, one faces the "curse of dimensionality." This challenge arises because data become sparse as dimensions increase, negatively affecting algorithms designed for lower-dimensional spaces. Also, too many features can lead to overfitting, resulting in poorer performance with new data. High-dimensional data also increases demands on memory and computational resources. Feature selection can significantly improve learning performance, boost efficiency, reduce memory needs, and help develop models with better generalization capabilities. (Li *et al.*, 2017)

5. Based on a review paper about deep neural networks and tabular data (Borisov *et al.*, 2022), I list some deep learning models below which are more suitable for tabular data than ANNs:

- TabNet

  Tabular data often contain a large number of features, some of which may be

irrelevant or even noisy. TabNet employs a sequential attention mechanism to dynamically select the most useful features at each decision step and uses sparse feature masks to reduce the influence of redundant information. This allows the model to focus on the key information, enhancing both prediction performance and interpretability.

- NODE
  An ensemble of differentiable oblivious decision trees, transforms the traditional decision tree concept into a fully differentiable model. This design allows NODE to be trained end-to-end using gradient descent while naturally handling categorical data without extensive preprocessing. Its structure, which leverages the same splitting function across nodes at the same level, enables efficient parallelization and competitive performance on tabular datasets .
- TabTransformer
  Focuses on transforming categorical features by applying self-attention mechanisms to generate contextual embeddings. This approach captures the complex interactions among features and mitigates missing or noisy data issues. The contextual embeddings are combined with numerical features using a multilayer perceptron, improving robustness and performance on heterogeneous tabular data.

References

Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., and Kasneci, G. (2022), "Deep neural networks and tabular data: A survey," *IEEE transactions on neural networks and learning systems*, Vol., No., pp.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., and Liu, H. (2017), "Feature selection: A data perspective," *ACM computing surveys (CSUR)*, Vol. 50, No. 6, pp. 1-45.