HW2_113034572_王奕鈞

1. A. Selected hyperparameters and test values

Hyperparameters	test values
Learning Rate	0.001, 0.01, 0.1
Hidden Units	128, 256, 512

B. Validation set accuracy results

Learning Rate	Hidden Units	Validation Accuracy (%)
0.001	128	77.78
0.001	256	80.25
0.001	512	80.25
0.01	128	79.01
0.01	256	**81.48**
0.01	512	80.25
0.1	128	50.62
0.1	256	50.62
0.1	512	50.62

C. Analysis and Discovery：

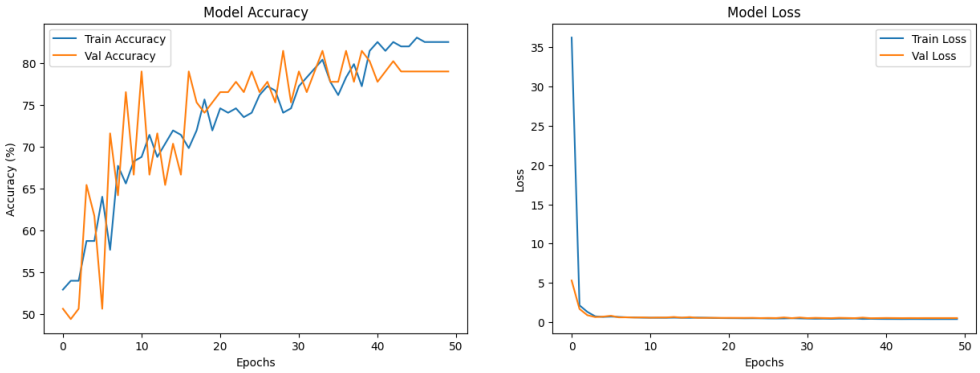
1. **Learning Rate:**
   - When the learning rate is set to 0.01, the model performs best, with a validation accuracy of 81.48%.
   - A learning rate of 0.1 leads to a significant drop in accuracy, which may cause training instability due to excessive gradient update amplitude.
2. **Hidden Units:**
   - Increasing the number of hidden neurons from 128 to 256 improves model performance.
   - The effect of increasing from 256 to 512 is limited and may have reached a performance bottleneck or lead to the risk of overfitting.

D. Optimal hyperparameter combination and model improvement

- Best combination: Learning Rate = 0.01, Hidden Units = 256
- Validation set accuracy: 81.48%
- Test set accuracy: 70.97%

- Compared with the original model, the test set accuracy increased by about 3%, showing that appropriate adjustment of hyperparameters can effectively improve generalization ability and model performance.

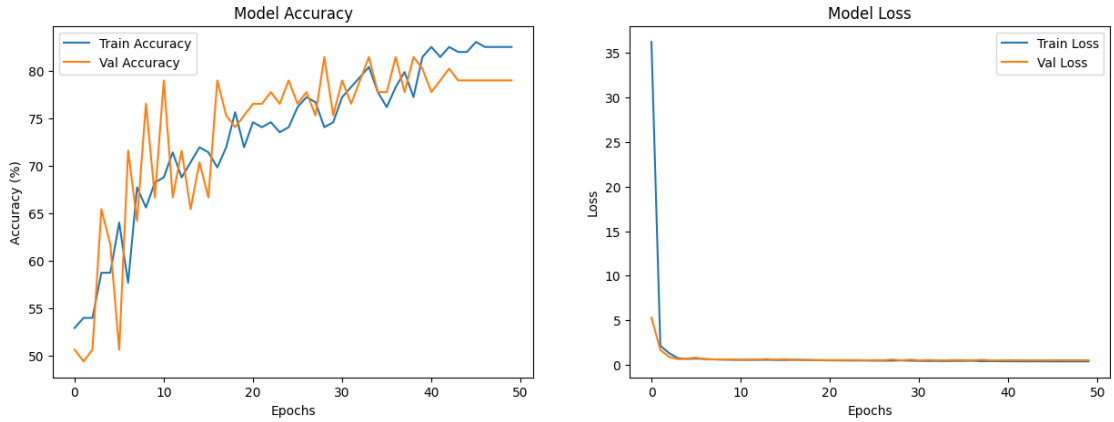
E. Visualize results



1. **Left: Model Accuracy (accuracy)**
   - Training accuracy (blue line) steadily increases as training progresses, eventually reaching about 85%.
   - Verification accuracy (orange line) increased during fluctuations and finally stabilized at about 78-81%, showing that the model has good generalization ability.
   - The trends of the two are close, and there is no excessive deviation → not overfitting.
2. **Right: Model Loss (loss value)**
   - The initial loss dropped rapidly and leveled off after the fifth epoch.
   - Train and Val Loss fit closely, further proving that model training is stable and good.
3. As shown in the figure, under the optimal hyperparameter combination (Learning Rate = 0.01, Hidden Units = 256), the accuracy of model training and verification is steadily improved, and the gap between the two is not large, indicating that the model is not overfitted and has good generalization ability. The loss value curve quickly converged after the fifth epoch, and the model training was stable, finally reaching an accuracy of 81.48% on the verification set.

2.

Based on the results, learning rate and hidden units significantly impacted model performance. With a learning rate of 0.01 and 256 hidden units, the model achieved the highest validation accuracy of 81.48%, and test accuracy reached 70.97%, showing a 3% improvement over the baseline. Increasing hidden units to 512 or reducing them to 128 yielded no further benefit. Notably, a learning rate of 0.1 caused all models to drop to 50.62% validation accuracy, indicating unstable training. As shown in the plots, proper hyper-parameters resulted in stable loss convergence (right plot) and steadily increasing accuracy (left plot), with training and validation curves aligning, suggesting better generalization.



3.

The discrepancy between training and test accuracy may be caused by overfitting. When a model learns too many details from the training data, it may struggle to generalize effectively to unseen test data. Additionally, insufficient training data or differences in distribution between training and test sets can also lead to accuracy gaps. Furthermore, excessive model complexity, such as having too many hidden neurons, may result in strong performance on training data but poor adaptability to new data. To reduce this gap, techniques such as regularization (e.g., L1/L2 regularization), increasing training data (e.g., data augmentation), or simplifying the model (e.g., reducing the number of layers) can be applied.

In this experiment, our final model achieved a training accuracy of 83%, while the test accuracy was only 70.97%, indicating that the model adapted well to the training data but struggled with unseen test data, likely due to overfitting. As shown in

the plots, although the trends of training and validation accuracy were similar, the test accuracy did not reach the same level, further supporting the possibility of overfitting. Additionally, differences in data distribution between training and test sets may have also contributed to this discrepancy. Therefore, to enhance the model's generalization ability, we recommend applying regularization, increasing the training dataset, or adjusting the model structure to mitigate the effects of overfitting.

4.

Feature selection is a crucial data preprocessing step in machine learning that aims to select the most representative features to enhance model performance, reduce the risk of overfitting, and simplify model complexity. Common feature selection methods include:

- Filter Method: Uses statistical tests (such as correlation coefficients) to evaluate the relationship between features and target variables, filtering out the most relevant features.
- Wrapper Method: Builds models with different subsets of features to find the optimal combination. Although effective, this approach can be computationally expensive and does not specifically consider variable types.
- Intrinsic Method: Some models inherently perform feature selection during training, such as Lasso regularization and decision trees' feature importance.

Effective feature selection not only improves model performance but also reduces training time and resource consumption, enhancing the model's generalization capability and preventing overfitting or underfitting. Additionally, it strengthens the interpretability of variables, making it particularly valuable in applications requiring high interpretability.

References：Brownlee, J. (2019). How to choose a feature selection method for machine learning. Machine Learning Mastery, 10, 1-7.

5.

Although Artificial Neural Networks (ANNs) are highly capable and versatile, they are not always the best choice for tabular data due to their limited ability to exploit the inherent structure of such data. ANNs often require extensive feature engineering to perform well, and they are prone to overfitting and high computational costs when handling structured data.

A more suitable alternative is TabNet, which is specifically designed for tabular data. TabNet leverages sequential attention mechanisms to focus on the most relevant features, enabling end-to-end learning from raw data without heavy preprocessing. Its sparse attention reduces computation, and the model offers built-in interpretability through feature selection masks, making it ideal for applications requiring transparency, such as healthcare or finance. Overall, TabNet often outperforms traditional ANNs on tabular tasks, offering improved efficiency and effectiveness.

References：Arik, S. Ö., & Pfister, T. (2021, May). Tabnet: Attentive interpretable tabular learning. In Proceedings of the AAAI conference on artificial intelligence (Vol. 35, No. 8, pp. 6679-6687).