## **ICS 661**

## **Advanced AI**

Fall 2024

## *Assignment 1 Report*

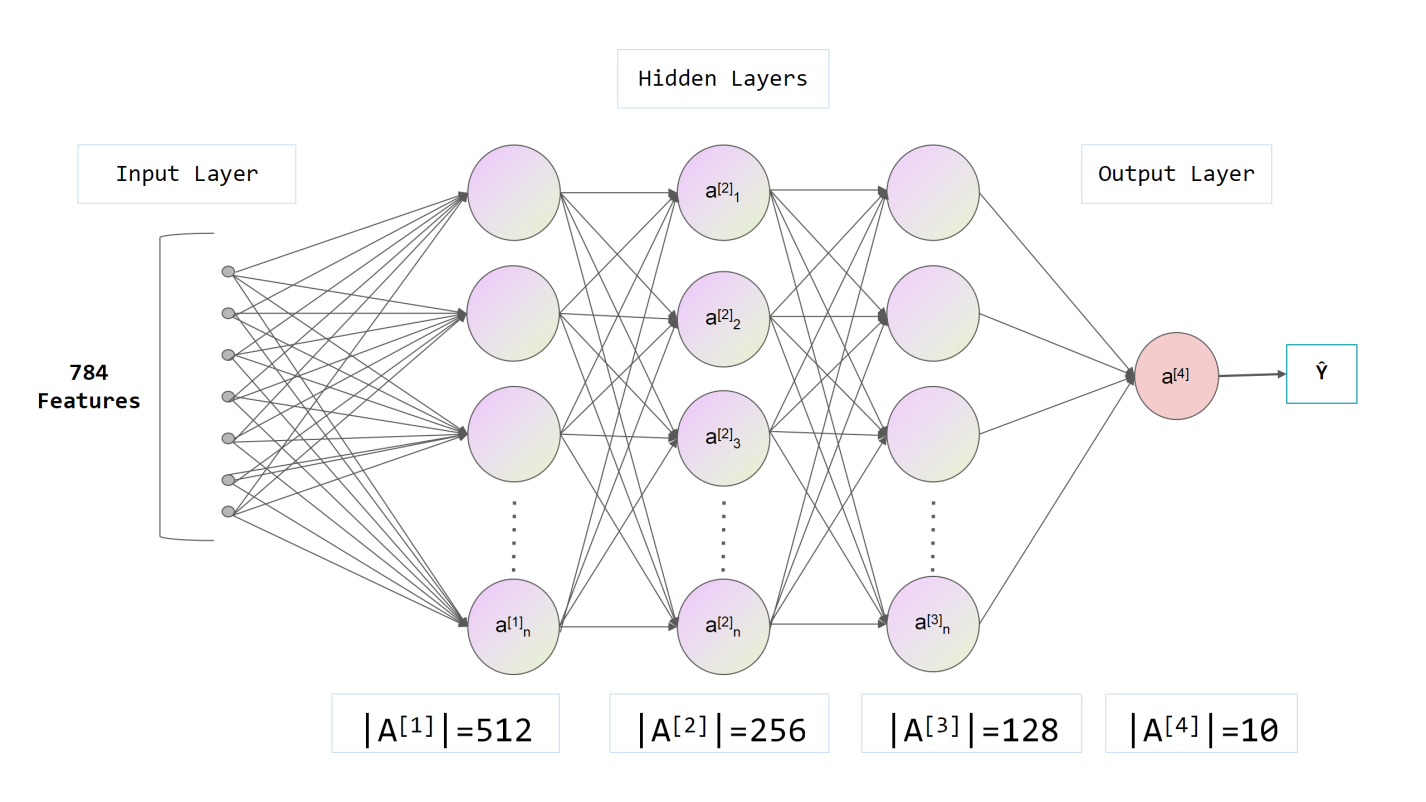
*Presented by: Haohan Yuan*

*UH ID: 30117604*

**Section 1: Task Description**

In this task, we develop and evaluate a Multilayer Perceptron (MLP) model using the provided train and test datasets. The MLP architecture is a fundamental feedforward neural network, well-suited for tasks involving classification and regression. For this task, we implemented an MLP with three hidden layers, trained on the dataset and tested for performance evaluation. Additionally, we explored the impact of different factors on the model's performance, such as varying optimizers, adjusting dropout rates, and modifying the number of neurons in the hidden layers. Through these experimental results, we provide insights into the optimization and robustness of the MLP model.

**Section 2: Model Description**

****

*Fig. 1 The architecture of the developed model.*

As shown in Fig.1, the developed MLP model consists of an input layer with 784 features, followed by four hidden layers with 512, 256, 128, and 64 neurons, respectively. Each hidden layer includes Batch Normalization, ReLU activation, and a Dropout layer with a dropout rate of 0.2. The output layer has 10 neurons, corresponding to the number of target classes. Specifically, we use Batch Normalization and Dropout to improve training stability and prevent overfitting.

**Section 3: Experiment Settings**

**3.1 Dataset Description**

|  |  |  |
| --- | --- | --- |
|  | train.csv | test.csv |
| Number of Instances | 60000 | 10000 |

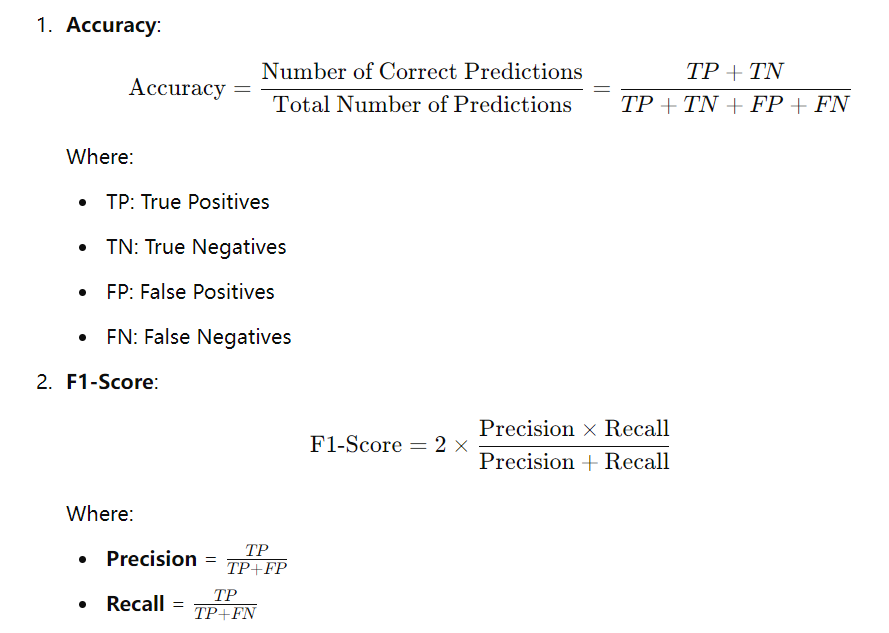
*Tab. 1 Statistics of the dataset.*

The dataset consists of two files: train.csv with 60,000 instances and test.csv with 10,000 instances. Each instance contains 785 comma-separated values. The first element is the label, representing a class from 0 to 9, while the remaining 784 elements are the features of the data instance, which are integers.

**3.2 Detailed Experimental Setups**

The model is implemented using the PyTorch framework. The optimizer used is AdamW with a learning rate of 1e-02, and no warm-up is applied. Both training and testing are performed with a batch size of 32. The architecture includes hidden layers with neuron counts of 512, 256, 128, and 64, with a dropout rate of 0.2 applied after each layer. The activation function used throughout the model is ReLU.

**3.3 Evaluation Metrics**



*Fig. 2 Evaluation Metrics.*

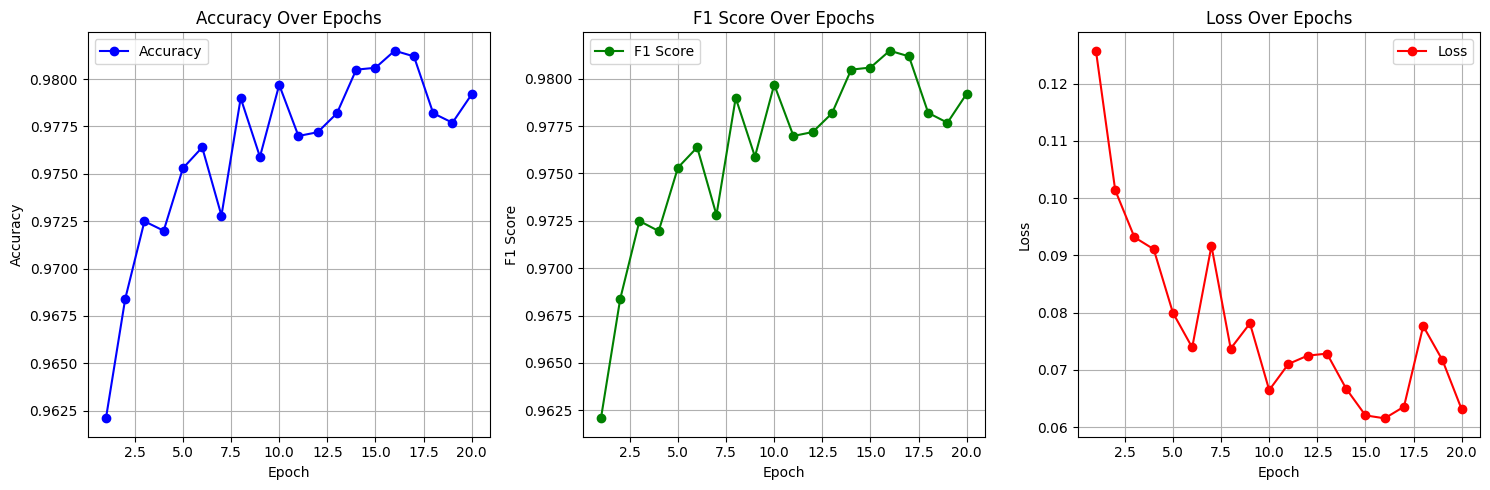
To evaluate the model's performance, we used two metrics: **Accuracy**, **Precision**, **Recall** and **F1-Score**. The formulas for both are shown in Fig. 2. Accuracy is the ratio of correct predictions (true positives and true negatives) to the total predictions, giving an overall measure of performance. The F1-score is the harmonic mean of Precision and Recall, helping balance false positives and false negatives. This is particularly useful for imbalanced datasets, as it considers both how many predicted positives are correct (Precision) and how many actual positives are identified (Recall).

**3.4 Source Code**

Google Colab Link: <https://colab.research.google.com/drive/1twJGHih09QoLkCdlEdalQZCA-iJkKFt9?usp=sharing>

Github Link: <https://github.com/Akeshh/ICS661-Assignment1>

**3.5 Training Convergence Plot**



*Fig. 3 Training Convergence Plot*

As shown in the Fig.3, the training loss reached its minimum at the 16th epoch, while the accuracy and F1-score on the test set also peaked. However, from the 17th to the 20th epoch, the loss started to increase and fluctuate, with both accuracy and F1-score decreasing slightly compared to their earlier values. This shows that the model began to overfit during this period.

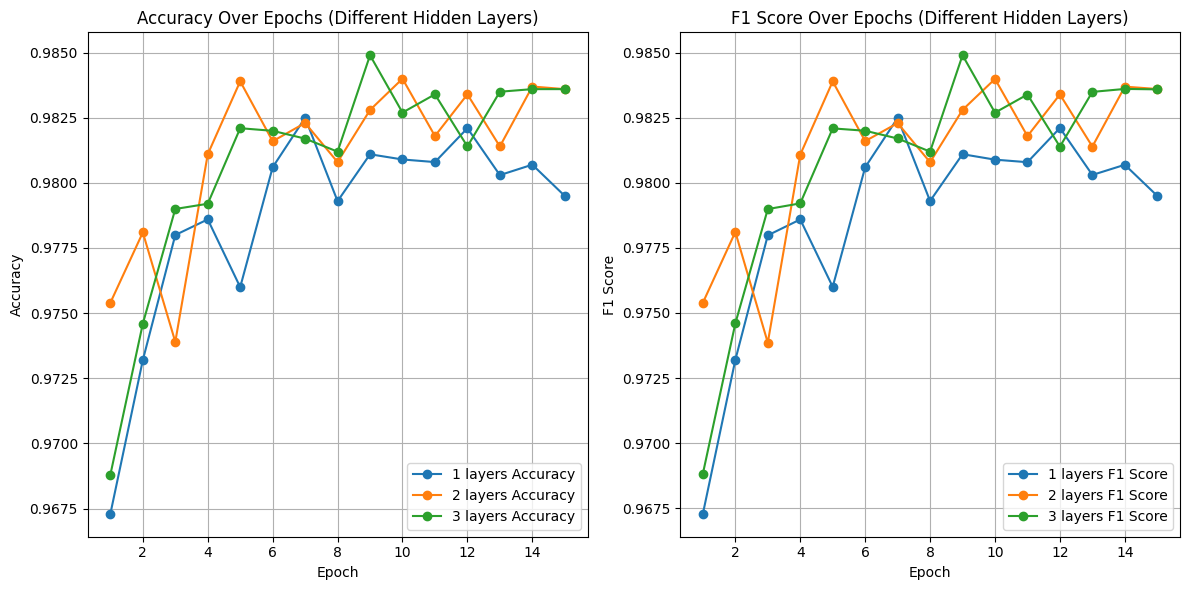
**3.6 Model Performance**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy (%) | Precision (%) | Recall (%) | F1-Score (%) |
| Our Model | 98.1498 | 98.1534 | 98.1498 | 98.1479 |

*Tab. 2 Performance results based on the given dataset.*

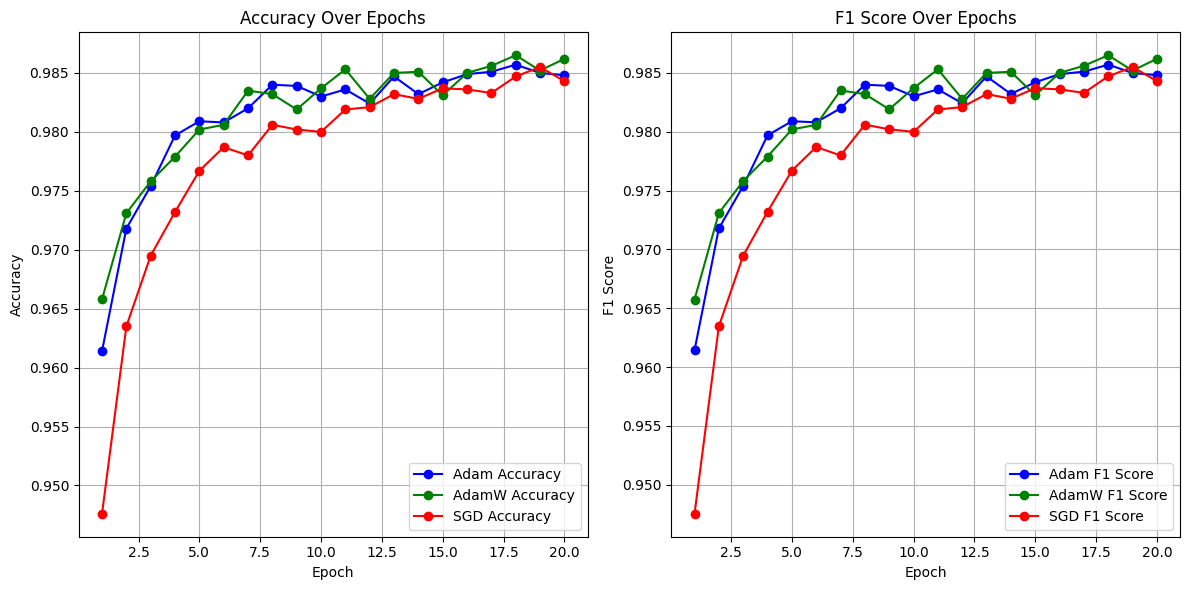
As shown in Table 2, our model demonstrates strong performance on the dataset. Specifically, the model achieved an accuracy of 98.1498%, a precision of 98.1534%, a recall of 98.1498%, and an F1-score of 98.1479%. Moreover, We found that the F1-score and accuracy were very close at all measured points. After analyzing the class distribution, we found that the number of instances for each class was roughly the same. We infer that this balanced data distribution is the reason for the little difference between the two scores.

**3.7 Ablation Studies**

****

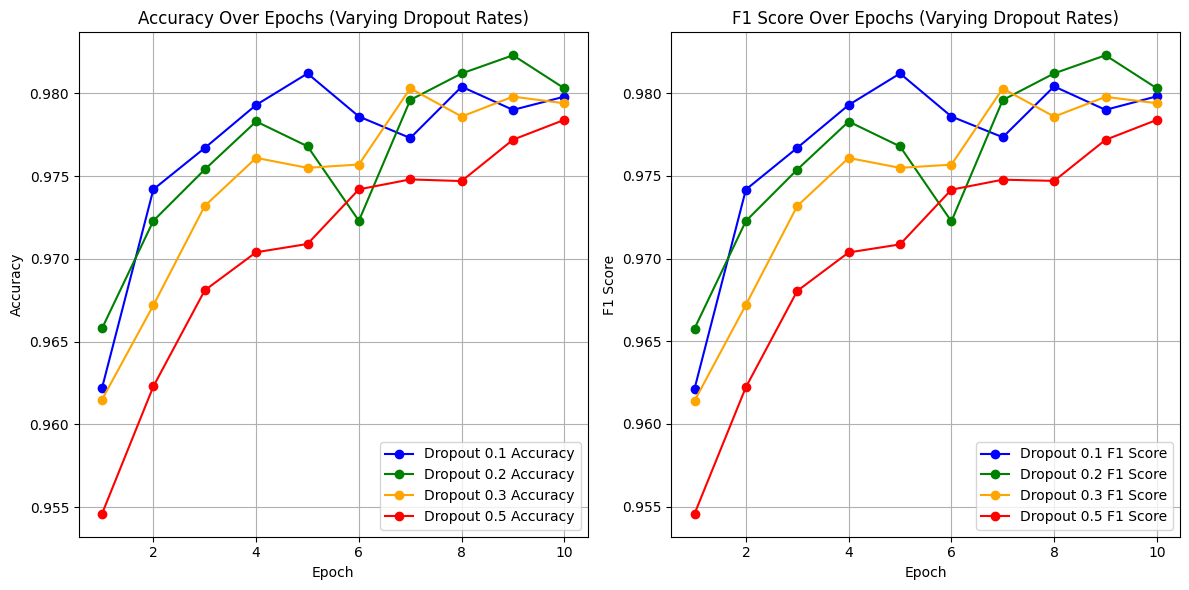
*Fig. 4 Ablation studies based on the number of model hidden layers.*

Fig. 4 illustrates the impact of modifying the number of hidden layers on the model's performance. Specifically, we tested three configurations: a single hidden layer with 256 neurons, two hidden layers with 512 and 256 neurons, and three hidden layers with 512, 256, and 128 neurons. The figure shows that the model performed the worst with only one hidden layer, while performance was more stable with two hidden layers. The best performance was achieved with three hidden layers.



*Fig. 5 Performance Analysis based on the optimizer.*

As shown in Fig. 5, we compared the model's performance when using different optimizers: Adam, AdamW, and SGD. The results show that the model performed significantly worse with SGD compared to the other two optimizers. In contrast, the model using the AdamW optimizer achieved the highest performance during testing.



*Fig. 6 Performance Analysis based on the dropout rates.*

As shown in Fig. 6, the model's performance is significantly affected by changes in dropout rates. Specifically, when the dropout rate is set to 0.5, the model performs the worst. On the other hand, the best performance is achieved with a dropout rate of 0.2.