Early Detection of Diabetes Using Perceptron Networks

Thi Kim Oanh Nguyen
a1879781

September 29, 2023

1 Introduction

1.1 Problem Statement

Diabetes, a chronic metabolic disorder characterized by elevated blood glucose levels, poses a significant global health challenge. According to recent estimates by the International Diabetes Federation (IDF) [1], diabetes affects over 537 million adults worldwide, with a staggering USD $966 billion spent on global health expenditure. Furthermore, the IDF predicts a steady rise in diabetes prevalence, indicating that by 2030, one in nine adults (643 million people) will be affected by diabetes [1]. By 2045, the number of adults with diabetes will rise to one in every eight (783 million people), exceeding USD $1,054 billion in healthcare costs. Timely detection is essential to mitigate its impact on individuals and reduce the enormous economic burden on healthcare systems worldwide.

Traditional diagnostic methods often rely on clinical tests that may not capture subtle early warning signs or consider multiple risk factors simultaneously. To address this gap, the application of machine learning techniques, specifically deep learning with perceptron networks, presents an opportunity to revolutionize diabetes diagnosis. The problem involves the development of a predictive model that uses diverse patient attributes, including medical history, physiological measurements, and demographic information. The primary objective is to accurately identify individuals at risk of developing diabetes well before clinical symptoms manifest.

Our methodology in this research is to create a simple perceptron model and experiment with different hyperparameters to achieve the best performance result. Perceptrons are relatively simple neural network architectures, making them computationally efficient and interpretable. When presented with patient data, a perceptron can rapidly classify an individual as either at risk of diabetes or not, allowing for timely interventions.

The model achieved promising results, with a training accuracy of 83.70%, a validation accuracy of 80.52%, and a test accuracy of 81.17%. This demonstrates the potential for early detection of diabetes using perceptron networks.

This paper proceeds as follows: Section 2 describes the method, Section 4 presents the experimental results, and Section 5 discusses future directions of our research.

1.2 Related Work

Some research papers have also employed the PIMA dataset for diabetes prediction. In Naz and Ahuja's study (2020) [2], Deep Learning demonstrated the highest accuracy at 98.07% when compared to various algorithms, including Artificial Neural Network (ANN), Naive Bayes (NB), and Decision Tree (DT). However, the research did not provide details about the process of hyperparameter selection and the construction of the deep learning model to achieve this result. Ayon and Islam (2019) [3] implemented a deep learning model with four hidden layers, achieving an impressive accuracy score of 98.04%. Nevertheless, the paper does not offer an explanation of the hyperparameters used to attain this result. Despite these limitations, Ayon and Islam's research serves as a valuable reference for our own work and potential future investigations.

2 Method Description

We use PyTorch to implement a simple perceptron model and experiment with different model architectures and hyperparameters to identify the combination of hyperparameters that results in the optimal model and achieve a high accuracy score.

The dataset consists of 768 instances, 8 feature variables and 1 target variable. Table 1 provides a description of each feature with its type.

Variables	Description
Pregnancies	The number of times a person has been pregnant (integer)
Glucose	Glucose concentration levels (integer)
Blood Pressure	Blood pressure measurements (integer)
Skin Thickness	Skinfold thickness measurements, estimating body fat (integer)
Insulin	Insulin levels in the blood (integer)
BMI	A measure of body fat based on height and weight (float)
Diabetes Pedigree Function	The likelihood of diabetes based on family history (float)
Age	The age of the individuals in years (integer)
Outcome	The target variable to indicate whether an individual has diabetes or not

Table 1: Description of features and the target variable

Figure 1 shows that there are 500 instances labeled as 0 (indicating individuals without diabetes) and 268 instances labeled as 1 (indicating individuals with diabetes). This significant difference in counts between the two labels indicates an imbalance in the dataset.

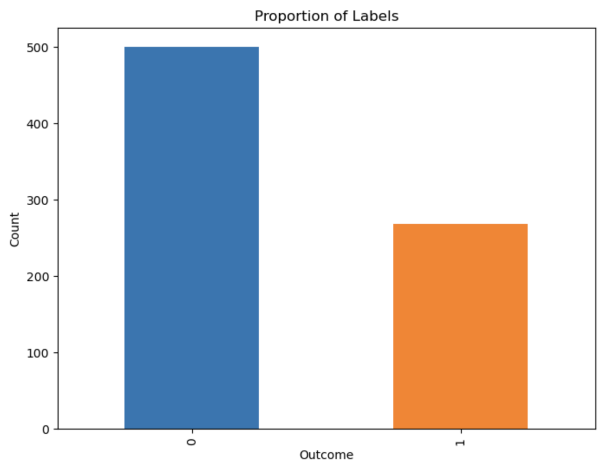
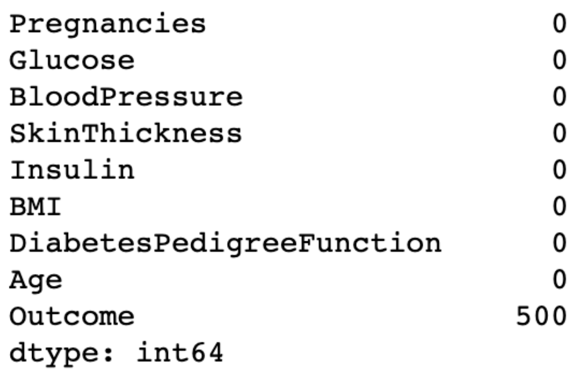


Figure 1: Proportion of label 0 and 1

2.1 Data Preprocessing

We check whether the dataset has duplicate rows or not. Then, we replace all occurrences of the value 0 in the selected columns with NaN and fill NaN values with the mean (average) value of the non-NaN elements within the same group. Figure 2 indicates the count of 0 values after data preprocessing.



```
Pregnancies                   0
Glucose                       0
BloodPressure                 0
SkinThickness                 0
Insulin                       0
BMI                           0
DiabetesPedigreeFunction      0
Age                           0
Outcome                     500
dtype: int64
```

Figure 2: Count of 0 values after data processing

We calculate the quartiles (Q1 and Q3), the interquartile range (IQR), upper and lower limits for detecting outliers. Outliers are replaced with the corresponding value from either UpperLimit or Lower-

Limit depending on whether it is above or below the limit, as shown in Figure 3.

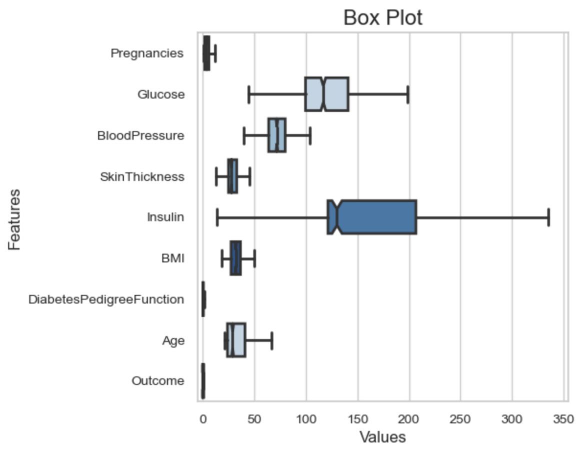


Figure 3: Boxplot of outliers after data preprocessing

The dataset is split into training, validation, and test sets using a stratified split strategy to ensure that the data is divided in a way that maintains the class distribution. We also transform the data so that it has a mean of 0 and a standard deviation of 1.

2.2 Building and Training the Model

Defining the Perceptron Model
We define custom architectures by subclassing **torch.nn.Module**. One of the critical decisions in designing the model architecture is determining the appropriate number of layers. A more complex and deep model with many layers is not always better. When dealing with limited training data, it might lead to overfitting. Therefore, we implement a single fully connected (linear) layer followed by a sigmoid activation function.

Loss Function and Optimizer
The Binary Cross-Entropy Loss is chosen as the loss function. Stochastic Gradient Descent (SGD) is selected as the optimizer with a learning rate of 0.01.

Training the Model
The training loop iterates for 100 epochs. An epoch is one complete pass through the entire training dataset. The training data (features and labels) are loaded into PyTorch tensors. Input features and labels are converted to the same data type to ensure compatibility. The forward pass computes the model's predictions for the input data. The Binary Cross-Entropy Loss is calculated by comparing the model's predictions (outputs) to the actual labels (labels). The loss quantifies how well or poorly the model is performing on the current batch of data. Backpropagation calculates gradients with respect to the model's parameters. The optimizer (SGD) then updates the model's parameters using the computed gradients to minimize the loss. This is an initial model, we will experiment different hyperparameters in the Experiments and Analysis section.

Evaluating the Model
The primary evaluation metric used is accuracy. A high accuracy score implies that the model is effective at distinguishing between individuals with and without diabetes based on the selected features.

During training and evaluation, the code records training loss, training accuracy, validation loss, and validation accuracy. We visualize the learning curves to see how the model's performance evolved during training.

2.3 Ensuring Reproducibility

Setting random seeds at the beginning of the implementation of the model ensures that random processes, like weight initialization and data splitting, are consistent across runs. Checking for GPU availability and selecting the device accordingly makes the code adaptable to different hardware.

3 Method Implementation

The link for the implementation of this method is as follows: `https://github.com/OanhOlivia/Deep-Learning/blob/main/a1879781_DL_Ass1_Final.ipynb`.

4 Experiments and Analysis

After implementing the initial model as described in the Method Description section, the obtained results indicate room for improvement. The training accuracy stands at 69.78%, suggesting that the model may not perfectly capture the intricacies of the training data. Furthermore, the validation accuracy, which reaches approximately 64.29%, lags slightly behind the training accuracy. This performance gap implies a hint of overfitting. The test accuracy, coming in at about 68.83%, aligns closely with the validation accuracy. To enhance overall model performance and reduce overfitting, it is beneficial to potentially fine-tune hyperparameters, all while keeping the number of training epochs.

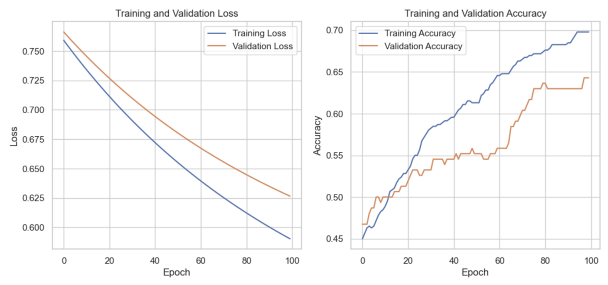


Figure 4: The learning curve of the model with SGD optimizer

The choice of optimizer significantly impacts model performance, making it one of the most crucial factors. Therefore, we will use the Adam optimizer instead of SGD. Adam has several advantages over SGD. Firstly, Adam adapts the learning rate for each parameter individually based on the past gradients and helps in better convergence, while SGD uses a fixed learning rate for all parameters. Additionally, Adam includes a form of regularization through the first and second moments of the gradients, helping in generalization, preventing overfitting to some extent. Finally, Adam is better at escaping local minima compared to SGD with its momentum and adaptive learning rates. In the case of SGD, which uses a fixed learning rate, it can get stuck in local minima more easily.

Change to the Adam Optimizer

This change led to a significant improvement in testing accuracy, which increased from 68.83% to 78.57%. However, the gap between training and validation accuracy (82.17% and 75.97%) did not decrease; in fact, it increased when compared to the SGD model (69.78% and 64.29%). This suggests that overfitting did not improve significantly with the optimizer change.

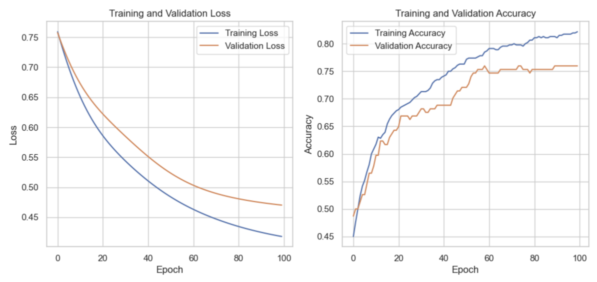


Figure 5: The learning curve of the model with Adam optimizer before Kaiming initialization

Adam is designed to work effectively with adaptive learning rates and weight updates. When combined with proper weight initialization, such as Kaiming initialization, it can significantly enhance training performance, leading to a more efficient learning process and helping to mitigate issues related to poor initialization and overfitting. The presence of inadequate initialization states may be a contributing factor to the persistence of overfitting in the model. To explore this further, we will incorporate Kaiming initialization and assess if this combination with the Adam optimizer improves the model's performance.

Add the Kaiming Initialization

It can be seen that a significant improvement in the model's performance is observed. The test accuracy increases from 78.57% to 81.17%. Additionally, the training and validation loss also exhibit improvements. By Epoch 100, the training loss was 0.41, and the validation loss was 0.45, compared to the training loss of 0.42 and the validation loss of 0.47 in the previous result. While the initial training and validation losses were higher with Adam and Kaiming, the model improved more efficiently during training, resulting in lower final losses and better generalization. One of the most significant improvements is the reduction in overfitting. With the introduction of Kaiming initialization and the Adam optimizer, this gap has considerably narrowed (83.70% for the training accuracy and 80.52% for the validation accuracy).

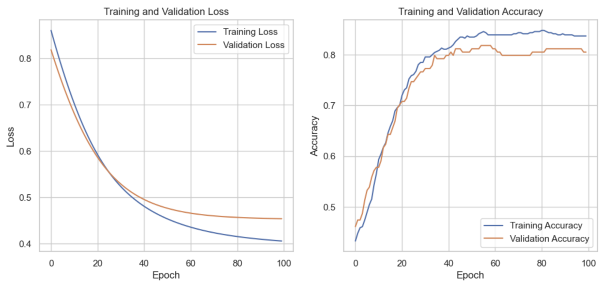
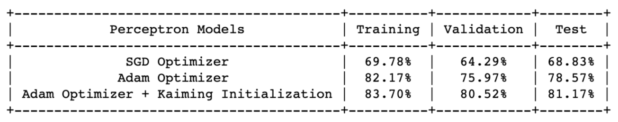


Figure 6: The learning curve of the model with Adam optimizer after Kaiming initialization



```
+------------------------------------------+----------+------------+--------+
|              Perceptron Models           | Training | Validation |  Test  |
+------------------------------------------+----------+------------+--------+
|                SGD Optimizer             |  69.78%  |   64.29%   | 68.83% |
|               Adam Optimizer             |  82.17%  |   75.97%   | 78.57% |
| Adam Optimizer + Kaiming Initialization  |  83.70%  |   80.52%   | 81.17% |
+------------------------------------------+----------+------------+--------+
```

Figure 7: Comparison of three models

5 Reflection on Project

5.1 Summary

We implemented a simple Perceptron model using Kaiming initialization. Binary Cross-Entropy Loss (BCELoss) was chosen, suitable for binary classification tasks. We used the Adam optimizer with a learning rate of 0.01 and conducted training over 100 epochs.

The initial model achieved a training accuracy of 69.78%, a validation accuracy of 64.29%, and a test accuracy of 68.83%, with signs of overfitting. Switching to the Adam optimizer substantially improved test accuracy to 78.57%. However, it widened the gap between training and validation accuracy, indicating persistent overfitting. Incorporating Kaiming weight initialization, along with the Adam optimizer, led to significant improvements. Test accuracy increased to 81.17%, and the gap between training and validation accuracy narrowed substantially, suggesting reduced overfitting.

5.2 Future Work

Future research directions include further hyperparameter tuning to optimize learning rate and the number of epochs. Additionally, exploring more advanced neural network architectures or model ensembles could enhance performance. Incorporating regularization techniques such as dropout or L2 regularization may improve model generalization. Experimentation with alternative optimizers like RMSprop or Nadam could also yield valuable insights. Addressing class imbalance through techniques like oversampling or undersampling is another avenue for investigation. Lastly, research focused on model interpretability, particularly in real-world applications, will be essential for practical deployment.

References

[1] Diabetes Australia 2023, *About Diabetes Globally*, Diabetes Australia, viewed 28 September 2023, https://www.diabetesaustralia.com.au/about-diabetes/diabetes-globally/.

[2] H. Naz and S. Ahuja, "Deep learning approach for diabetes prediction using PIMA Indian dataset," *Journal of Diabetes and Metabolic Disorders*, vol. 19, no. 1, pp. 391–403, 2020.

[3] S. Ayon Islam and M. Milon Islam, "Diabetes Prediction: A Deep Learning Approach," *International Journal of Information Engineering and Electronic Business*, vol. 11, no. 2, pp. 21–27, 2019.