Diabetes Classification using MLP

Yuchen Liu
The University of Adelaide
yuchen.liu01@adelaide.edu.au

Abstract

Diabetes diagnosis is a critical challenge in the medical field. With the availability of numerous patient cases, deep learning networks are capable of extracting implicit patterns and providing predictive assistance for medical diagnosis. This assignment explores the application of deep learning methods to this specific problem. Following a structured pipeline of data preprocessing, network training, and performance analysis, a framework for diabetes diagnosis prediction using deep learning was developed. The code for this project is available on the GitHub page (`https://github.com/MorainingErin/COMP_SCI_7318_DLF`*).*

1. Introduction

Diabetes is a widespread and long-recognized condition affecting human populations. The prevention and early detection of diabetes are critical first steps in effective management[1]. However, identifying diabetes based on various indicators remains a critical challenge for the medical system. As a well-studied condition, diabetes presents with numerous common symptoms across patients. Therefore, by analyzing specific characteristics of a potential patient, it is possible to predict whether they may be positive or negative for the disease. While the final diagnosis must be made by a qualified doctor, pre-diagnostic information can provide valuable assistance in guiding the decision-making process.

Over the past decade, deep learning methods have demonstrated promising capabilities across a wide range of tasks. Unlike traditional machine learning methods, these methods do not require explicit feature extraction, offering greater flexibility by learning directly from data. Moreover, with the rapid development of computing devices, the depth and complexity of the deep models can be expanded, leading to substantial improvements in performance across many complex tasks.

In this assignment for the course 'Deep Learning Fundamentals'[3], deep learning techniques were applied to address the problem of diabetes prediction. As the first assignment of this course, the focus is on building and training a neural network model for a concrete classification task. Specifically, the data was first preprocessed through cleaning and splitting. A baseline result was then obtained using a simple network with a single perception layer. To further explore the classification capability of DNN, the depth of the network was increased and hyperparameter tuning was applied to enhance performance. Finally, the model's performance was demonstrated with tables and charts to illustrate its capabilities.

This report is organized as follows. The first section introduces the background and motivation for the assignment (Sec.1). The problem is defined next, along with a detailed description of the pipeline (Sec.2). Section 3 presents the performance results of both the baseline and improved models, accompanied by additional analytical experiments that demonstrate the validity and effectiveness of the improvements. The code for this assignment is available on the GitHub page (Sec.4). Finally, Section 5 summarizes the work and proposes potential directions for future improvements.

2. Method

2.1. Problem Definition

For the diabetes classification problem, the objective is to identify a classifier f, which is defined as follows:

$$f : \mathbf{x} \rightarrow y, \tag{1}$$

where $\mathbf{x}$ represents a feature vector with length of 8, and $y \in \{0, 1\}$ corresponds to the binary label indicating a negative (0) or positive (1) case.

In this work, a deep neural network is employed to fit the classifier f, with the network denoted as f_N, where N stands for 'Network'. To facilitate gradient-based optimization, the network's output differs slightly from the original definition in Eq.1:

$$f_N : \mathbf{x} \rightarrow y_N, \tag{2}$$

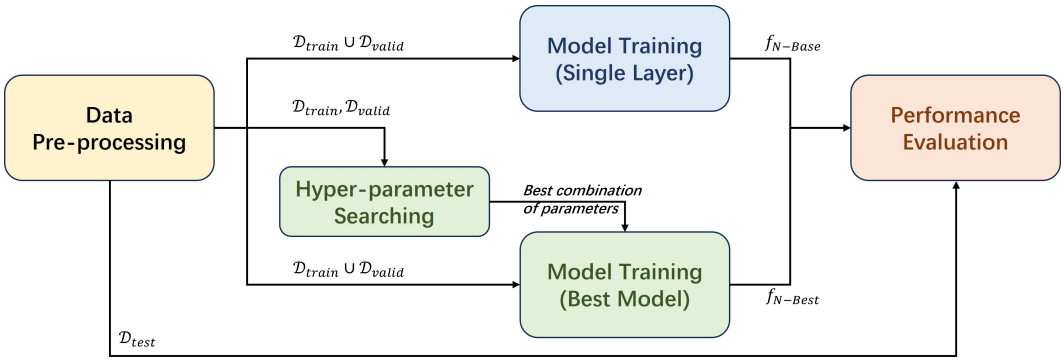

Figure 1: **The complete pipeline of the proposed methodology.** The process begins with data pre-processing, where the dataset is split into training, validation, and test sets. An initial model is trained using a single-layer architecture. Next, a hyper-parameter search is performed to determine the best combination of parameters. Based on the optimal parameters, a refined model is trained. Finally, both the baseline and best-performing models are evaluated on the test data to assess their performance.

where the output $y_N \in [0, 1]$ is a continues value between 0 and 1. This continous output helps in designing and training the deep neural network more effectively.

2.2. Pipelines

The pipelines can be summaried into three main stages: Data Preprocessing, Model Training, and Performance Evaluation. The complete pipeline is illustrated in Fig.1.

2.2.1 Data Preprocessing

Pre-processed data obtained from the website[2], denoted as $\mathcal{D}$, was utilized. The dataset is then split into three subsets: the training dataset $\mathcal{D}_{train}$, the validation dataset $\mathcal{D}_{valid}$ and the test dataset $\mathcal{D}_{test}$. The test dataset $\mathcal{D}_{test}$ is reserved exclusively for the final evaluation, and it is not used in model optimization or selection. The training dataset $\mathcal{D}_{train}$ is employed for training the network, while the validation dataset $\mathcal{D}_{valid}$ is used to tune hyperparameters and guide model selection by evaluating performance during the training process.

During data preprocessing, several missing values were identified within the dataset. Given the limited dataset size, median imputation was applied to fill in the missing values, ensuring the data remained usable for training, validation, and test purposes. The process is visualized in Fig.2 for a clearer understanding.

2.2.2 Model Training

The effectiveness of neural networks in the diabetes classification problem is explored by beginning with a single-layer network as the baseline model. This baseline consists of a linear layer, followed by a *Tanh* transformation layer, which scales the output of the linear layer to the range $[-1, 1]$. Additionally, a non-trainable scaling layer is applied to further

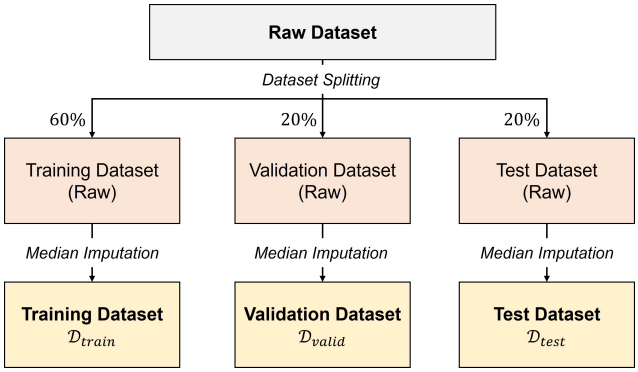

Figure 2: **Detailed overview of the data pre-processing pipeline.** The raw dataset is initially split into three subsets, and median imputation is applied to each subset to handle missing values.

rescale the output to the range $[0, 1]$. The architecture of the single-layer network is depicted in the top-left of Figure 3.

For training the network, cross-entropy loss is utilized. The baseline model is trained using both the training and validation datasets combined ($\mathcal{D}_{train} \bigcup \mathcal{D}_{valid}$). Given the network output y_N and the groundtruth y_{gt}, the cross entropy loss $\mathcal{L}$ is calculated as follows:

$$\mathcal{L} = \frac{y_{gt} \log y_N + (1 - y_{gt}) \log(1 - y_N)}{2}. \quad (3)$$

2.2.3 Performance evaluation

After completing the model training, performance is evaluated on the test dataset $\mathcal{D}_{test}$. Several evaluation metrics are computed to assess the model's performance, and a confusion matrix is generated for better visualization and comparison of results. Please refer to Sec.3.2 for more details.

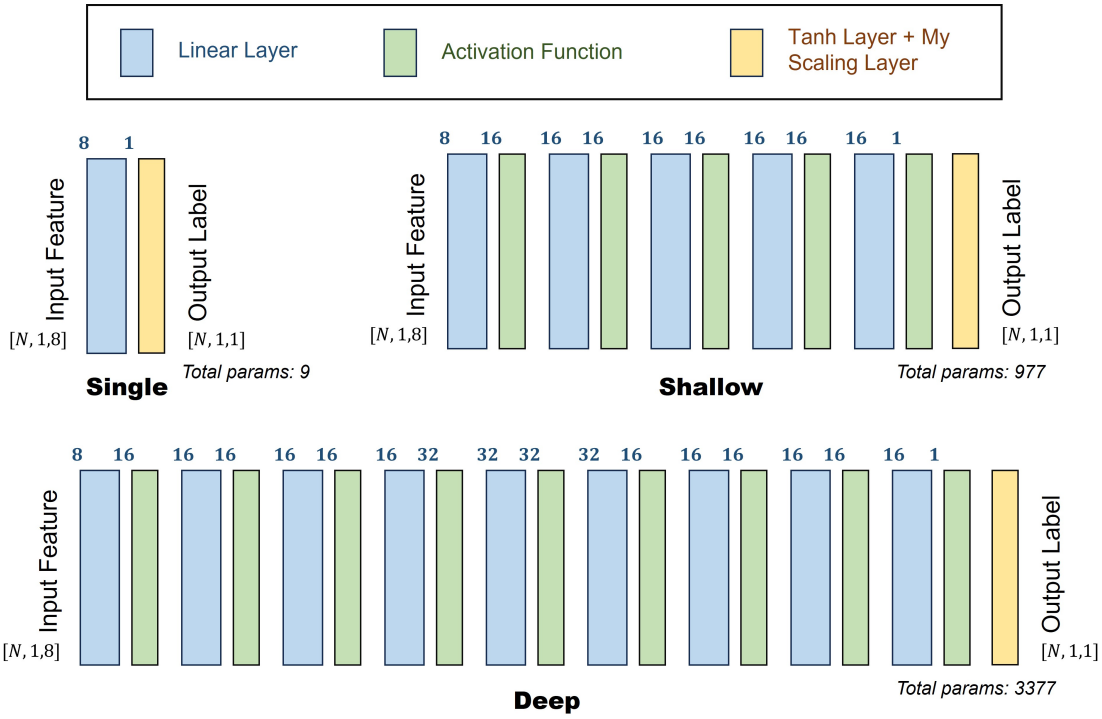

Figure 3: **Overview of the three network architectures used in this study.** The architectures include a Single layer network, a Shallow network, and a Deep network. The input and output channels of linear layers are marked on their top.

2.3. Improvements

After establishing the baseline model, performance was further enhanced by experimenting with deeper network architectures and optimizing hyperparameters. Two new network architectures, *shallow* and *deep*, were designed, both consisting of additional densely connected layers. The architectural details are shown in Fig.3. To further optimize model training, various hyperparameters were explored, as summarized in Tab.1.

Hyper-parameters	**Possible values**
Activation Function	ReLU, ELU, SeLU, Leaky ReLU
Batch Size	8, 16, 32
Learning Rate	0.1, 0.01, 0.001, 0.0001
Optimizer	Adam, RmsProp, NAdam

Table 1: **Hyperparameters explored in the model optimization process.** The table lists the possible values for the key hyperparameters, including activation functions, batch sizes, learning rates, and optimizers.

Coarse-to-fine random search. Given the large hyperparameter space, extensive training is required to find the best combination via Grid Search. To efficiently balance quality and time, a coarse-to-fine random search scheme was implemented . Initially, 30 training trails were conducted with randomly selected hyperparameter combinations. From the top three hyperparameter combinations, all unique hyperparameter values were extracted and then used to perform an additional 20 training runs within a refined search space, ensuring a more focused second random search. Fig.4 illustrates this two-step random search process. By employing this approach, the number of training trials needed to identify the optimal hyperparameter combination was effectively reduced.

Training Dataset. To prevent data leakage, only the training and validation datasets were used for model selection. Specifically, the training process was conducted on the training dataset $\mathcal{D}_{train}$, and the best one was selected based on performance on the validation dataset $\mathcal{D}_{valid}$. Once the optimal hyperparameters were identified, the model was retrained using both the training and validation datasets $\mathcal{D}_{train} \bigcup \mathcal{D}_{valid}$, and its final performance was evaluated on the test dataset $\mathcal{D}_{test}$, following the same procedure applied to the baseline model.

3. Experiments

In this section, the experimental setup is detailed, and the effectiveness of the improvement methods is demonstrated.

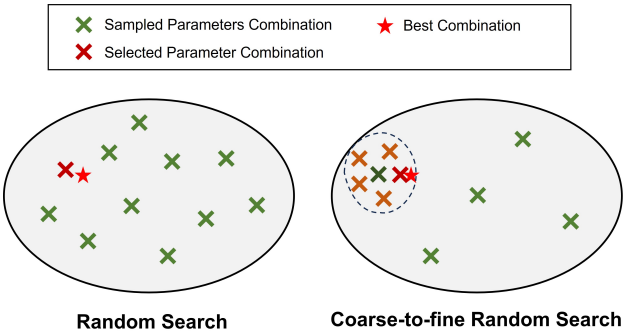

Figure 4: **Illustration of the coarse-to-fine random search process.** The diagram compares traditional random search (left) with my two-step random search approach (right). In my case, a coarse-to-fine strategy is employed, first sampling parameters broadly and then focusing on promising regions, which allows for more efficient identification of the best combination of parameters.

Model	Accuracy	Precision	Recall	F1-score
Baseline	53.90%	48.04%	53.90%	50.04%
Improved	**71.43%**	**70.29%**	**71.43%**	**70.00%**

Table 2: **Quantitative performance comparison between the baseline and improved models.** The improved model shows enhancements across all evaluation metrics, reflecting the effectiveness of the hyperparameter tuning and deeper network architecture applied.

3.1. Experimental Details

The diabetes dataset was obtained from the official source [5]. For this study, the preprocessed version of the dataset[2] was utilized. The scaled version was avoided as it was based on statistics from the entire dataset, potentially leading to data leakage. The dataset was randomly divided into three subsets: 60% for training, 20% for validation, and 20% for test. Histograms of each feature dimension are also provided in the code notebook on GitHub to offer a general overview of the input data.

All experiments were conducted on a GPU platform (RTX2080Ti), and the neural network implementation was carried out using PyTorch[4]. Each model was trained for 200 epochs.

3.2. Main Experiments

Model performance was evaluated using four metrics on the test dataset: accuracy, precision, recall and f1 score. For the final comparison, two models were assessed: the original baseline model, which has no modifications and uses the default training parameters (referred to as 'Baseline'), and the model selected through the hyperparameter search (referred to as 'Improved'). The quantitative results are presented in Table 2. Additionally, confusion matrices for both models are provided in Fig.5 to offer a more detailed visualization of their performance.

3.3. Analytical Experiments

To further demonstrate the effectiveness of the pipeline, two additional experiments were conducted for a deeper analysis.

First, the effectiveness of the two-step random search scheme is evaluated. The rationale behind this design is to refine the hyperparameter search from a coarse level to a finer one. In Fig.6, a box plot of the validation accuracy from both search phases is presented. The second phase clearly achieves a higher average accuracy and a smaller variance compared to the first phase, indicating that the second step is refining the search around the optimal hyperparameter combinations.

Next, the overfitting tendency of the training scheme is analyzed. The results of the first and second search are plotted in Fig.7, with training accuracy on the X-axis and validation accuracy on the Y-axis. In most cases, the validation accuracy is only slightly lower than the training accuracy, suggesting that overfitting is not a significant issue in my experiments.

4. Code

The code for this project is available on my GitHub page (`https://github.com/MoraingErin/COMP_SCI_7318_DLF`). For clarity and ease of use, the code is provided in Jupyter Notebook format, with an accompanying PDF version attached for reference.

5. Conclusion

In this assignment, a series of experiments were conducted to explore the effectiveness of deep learning methods in a binary classification problem. Starting with a single-layer network as the baseline model, data preprocessing was applied before modifying the network architecture and training parameters to improve performance. The baseline and improved models were evaluated both quantitatively and qualitatively through various experiments.

Limitations: Although the improvement techniques led to better performance compared to the single-layer model, there remains significant room for further accuracy improvements. The primary limitation is likely the relatively small dataset size, which hampers the effectiveness of data-driven techniques like deep learning. To address this, future work may explore techniques such as ad-hoc feature extraction or synthetic data generation to enhance performance.

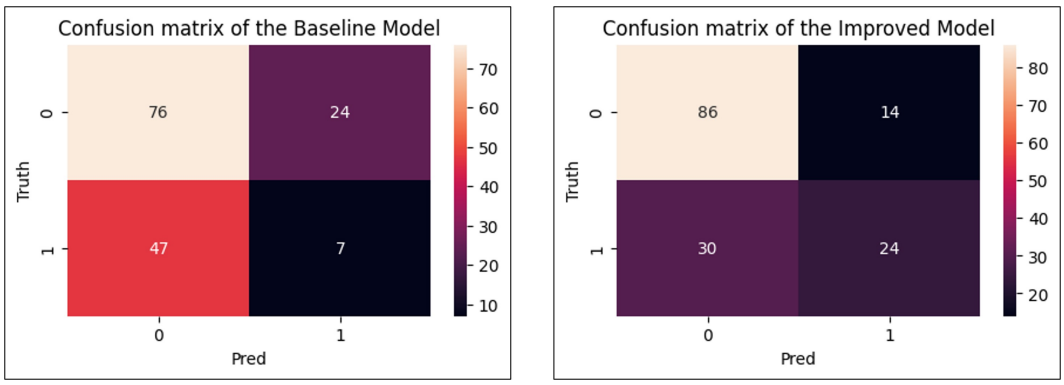

Figure 5: **Confusion matrix comparison between the baseline model (left) and the improved model (right).** The baseline model shows higher false negatives, while the improved model demonstrates a more balanced distribution between true positives and true negatives.

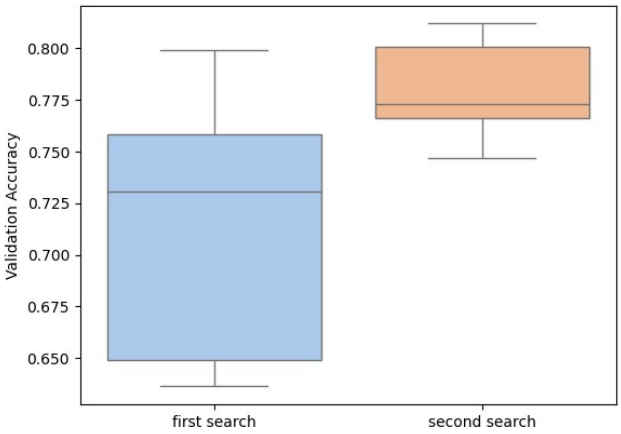

Figure 6: **Comparison of validation accuracy across the two search stages.** The boxplot illustrates the validation accuracy distributions obtained from the first coarse search and the second fine-tuned search. The second search demonstrates a noticeable improvement in accuracy and reduced variance, indicating the effectiveness of the coarse-to-fine random search strategy.

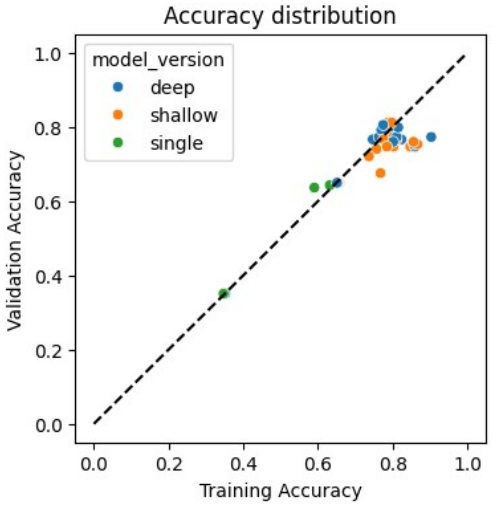

Figure 7: **Accuracy distribution plot comparing training and validation accuracy.** The plot illustrates the training and validation accuracy of the models. Points close to the diagonal line indicate minimal overfitting, suggesting that the models generalize well in the experiments.

References

[1] Diabetes Australia. Managing type 2 diabetes. https://www.diabetesaustralia.com.au/managing-diabetes/type-2/, 2024. [Accessed: 01-10-2024].

[2] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

[3] The University of Adelaide. Course outlines: COMP SCI 7318 - Deep learning fundamentals. https://www.adelaide.edu.au/course-outlines/110026/1/sem-2/2020/, 2024. [Accessed 01-10-2024].

[4] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. 2019.

[5] Jack W Smith, James E Everhart, WC Dickson, William C Knowler, and Robert Scott Johannes. Using the adap learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the annual symposium on computer application in medical care*, page 261, 1988.