Analysis Document Reviewing the Process of Training Two Neural Networks for Predicting Diabetes.

In this study, we embarked on the development of an artificial neural network (ANN) with the objective of predicting diabetes presence in patients. The architecture was constructed using two distinct deep learning libraries, PyTorch and Keras, to assess their efficacy in handling binary classification tasks based on eight pivotal health indicators: Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function, and Age. Our dataset comprised 2,460 instances for training and 308 for evaluation, with binary outcomes indicating diabetes diagnosis (1) or absence thereof (0).

Given the variance in magnitude across the features, which inherently could skew the model's predictive capability towards the parameters with larger numerical ranges, we implemented normalization techniques. Specifically, the MinMaxScaler from the sklearn library was employed to normalize the dataset features to a range of 0 to 1, thereby standardizing their contribution to the model's output during the training phase.

One of the design considerations was the flexibility of the model architecture to accommodate an expanded set of features without requiring substantial modifications. To this end, we designed the input layer to dynamically adjust to the number of features, thus future-proofing the model against potential dataset enhancements.

The implementation of the ANN maintained a consistent feature set across both the PyTorch and Keras platforms, encompassing the chosen optimizer, network architecture, and loss function. However, a notable distinction between the two was the inclusion of an early stopping mechanism in the Keras model, a feature that optimizes training efficiency by halting the process once no further improvement is observed. This functionality, while desirable for inclusion in the PyTorch version, would necessitate a transition to PyTorch Lightning, a refactor that was beyond the scope of the current project timeframe but is identified as a primary area for future development.

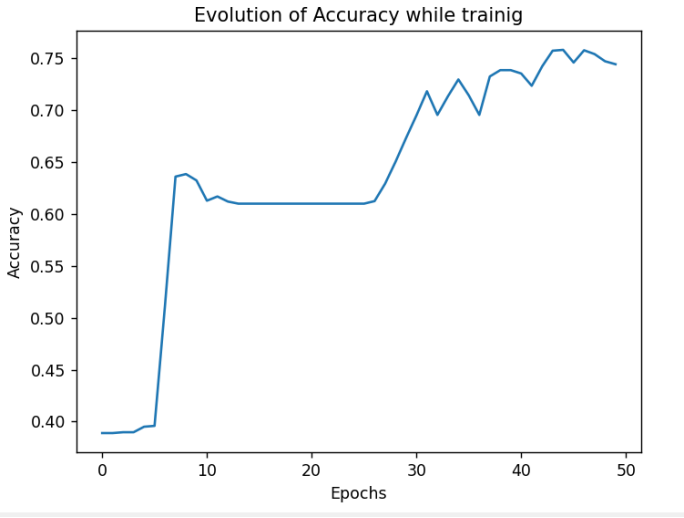
After iterative experimentation with various network structures, the chosen model architecture emerged as a medium-complexity, descending structure that exhibited superior performance in predicting diabetes. This architecture underscores the capability of neural networks to discern intricate patterns in medical data, offering promising avenues for enhancing diagnostic accuracy in healthcare settings.

8 --- 100 --- 50 --- 25 ---1

The performance metrics derived from our neural network model have shown a promising degree of efficacy, with an accuracy range of approximately 80-83%. This level of precision, while commendable, does exhibit variability across different execution instances. Such fluctuations can largely be attributed to the inherent randomness associated with the initialization of model weights prior to the commencement of the training phase. To mitigate this randomness and enhance reproducibility, our implementation within the PyTorch framework incorporates an option to set a fixed seed. By activating two lines of code at the outset of the

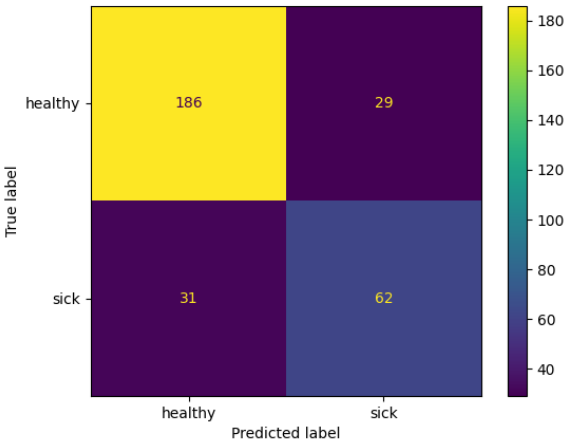
main function—currently commented out—researchers can ensure a consistent starting point for weight initialization.

Further elucidating the model's performance dynamics, we have generated a graphical representation of the accuracy evolution throughout the training process. This visualization serves as a critical tool for understanding the model's learning trajectory over time, offering insights into the consistency and reliability of its predictive capabilities. Such analysis not only reinforces the model's effectiveness but also identifies potential areas for optimization, laying the groundwork for future enhancements to further elevate diagnostic accuracy within the domain of diabetes prediction.



The observed performance graph indicates an initial rapid learning phase, attributed to the model quickly identifying clear patterns within the dataset. However, a notable plateau in accuracy between epochs 10 and 30 suggests the learning rate may be too high for the model to capture more subtle patterns during this period. To address this, we introduced a dynamic learning rate adjustment function. This mechanism reduces the learning rate when the model's improvement stalls, allowing for finer pattern detection and potentially enhancing overall predictive accuracy. Further details on this adaptive approach will be discussed later in the document.

The confusion matrices for the final model:



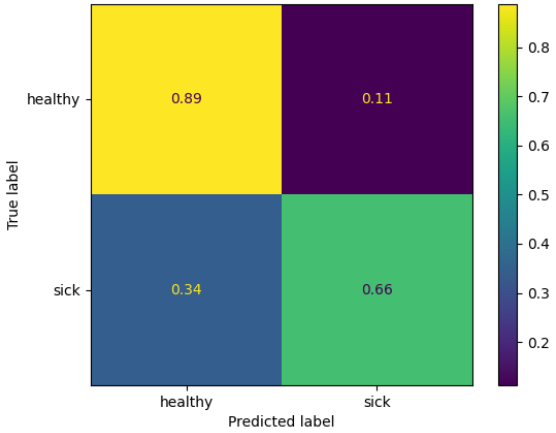


Fig1 Fig2

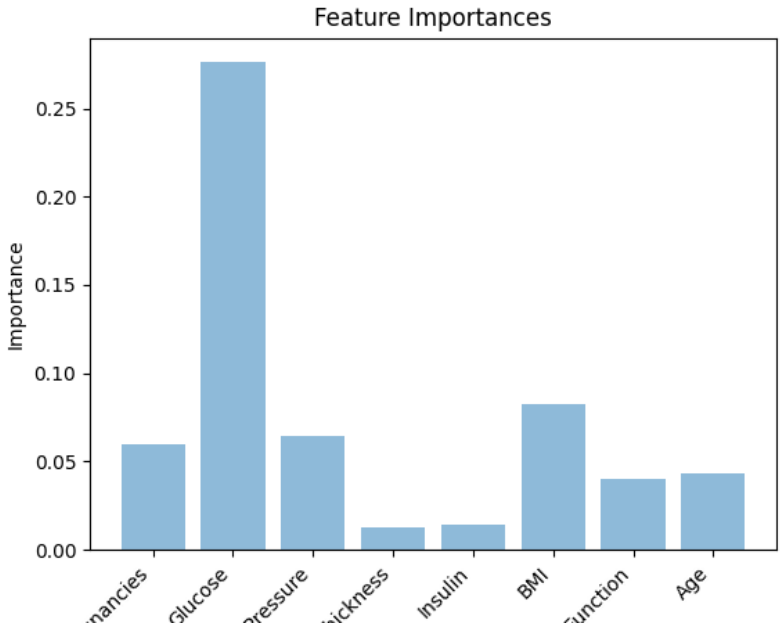
Analyzing the Confusion Matrices (Fig1 and Fig2) reveals a disparity in the model's predictive accuracy for healthy versus sick patients. Specifically, the model demonstrates a higher proficiency in identifying healthy patients, achieving an 89% accuracy rate, compared to a 66% accuracy for sick patients. This discrepancy is attributed primarily to the composition of the training and testing datasets, which are skewed towards healthy patients (1499 healthy vs. 960 sick in training, and 215 healthy vs. 93 sick in testing). This imbalance likely biases the model towards recognizing patterns associated with healthy cases more effectively than those for sick cases.

The differential accuracy rates underscore a critical area for improvement, especially considering the clinical importance of accurately identifying sick patients. Enhancing the model's sensitivity to sick patients is essential for its application in healthcare settings, where the cost of false negatives (failing to detect diabetes) is significantly higher than false positives. To address this, augmenting the dataset with more sick cases or employing techniques to balance the dataset could be pivotal strategies. Such adjustments would not only aim to equalize the model's performance across both patient categories but also optimize its utility in real-world diagnostic applications.

Parameters Importance:

In response to the practical relevance of identifying key predictive factors for diabetes, we generated a graph to determine which parameters most significantly influence the model's predictions. Considering the variability in model outputs across different runs, this graph represents an average from multiple iterations. This approach provides a clearer understanding of the paramount features for diabetes prediction, offering crucial insights for both model

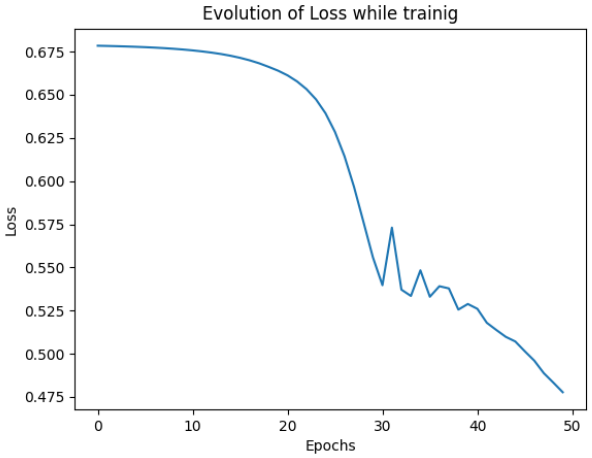
optimization and clinical application.



The analysis revealed that glucose levels stand out as the most critical parameter in predicting diabetes, aligning with medical understanding of the disease. Following glucose, pregnancies, blood pressure, and the diabetes pedigree function emerge as significant, with their influences consistently ranging from 0.03 to 0.07. This insight into parameter importance underscores the nuanced interplay of biological factors in diabetes prognosis.

Loss function and evolution

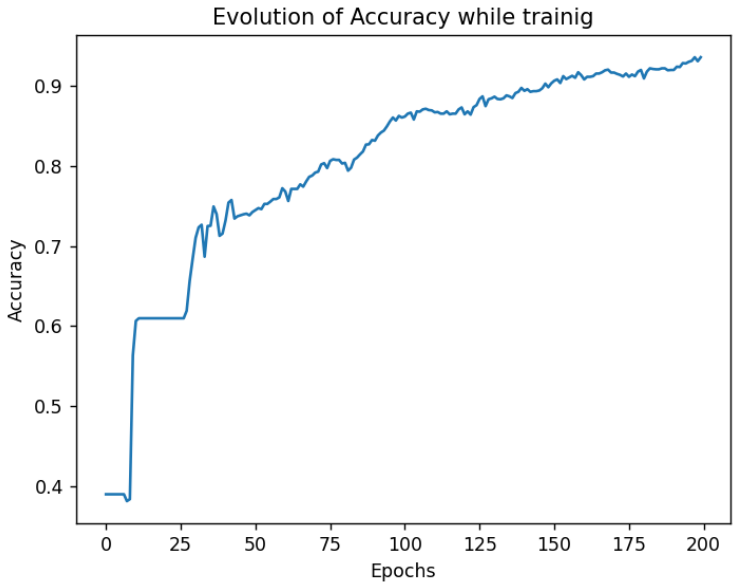
Our optimal model employs binary cross-entropy as the loss function across both implementations. Despite exploring various loss functions, binary cross-entropy consistently facilitated achieving accuracies near the 80% mark, making it our preferred choice. Additionally, we've visualized the loss evolution throughout the training phase, further illustrating the model's learning progress.



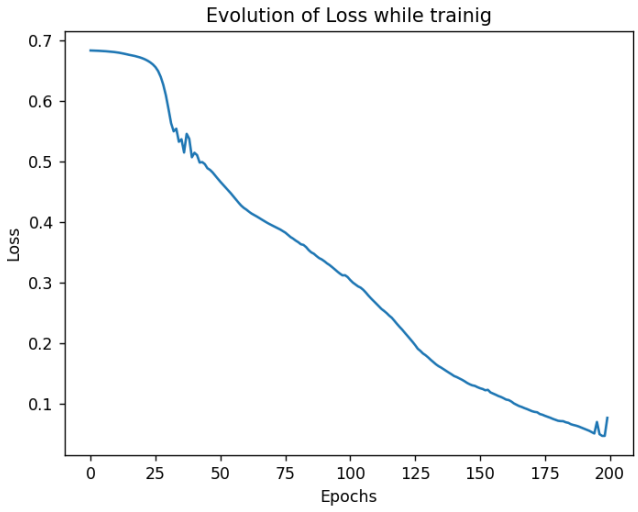
The graph indicates that initially, the model's learning rate is modest until a significant inflection point is reached, leading to a rapid decrease in loss. This suggests the model identifies a crucial pattern or feature that substantially improves its predictive capability. The observed oscillation in loss post-decline likely stabilizes as the model assimilates new patterns. While extending training epochs might intuitively seem beneficial, we will later illustrate that this leads to overfitting, adversely affecting the model's generalization and, consequently, its accuracy on unseen data.

Number of Epochs

In the development of our model, extensive experimentation with the optimal number of training epochs was conducted. Ultimately, it was determined that a more concise training duration, specifically 50 epochs, yielded the most robust and consistent accuracy outcomes across multiple test iterations for this dataset. This decision was supported by analytical comparisons, where extended training periods, such as 200 epochs, were systematically evaluated. The subsequent analyses, particularly focused on the PyTorch implementation—which lacks an early stopping mechanism—reveal the rationale behind selecting a reduced epoch count. These findings are illustrated through graphs generated from the PyTorch model, demonstrating the advantages of this approach in terms of model performance and reliability.



The graph depicting model performance over 200 epochs clearly shows an increasing trend in accuracy, highlighting an improved rate of learning compared to the model trained for fewer epochs. However, the accuracy on the test dataset ranges between 73% and 76%, which is notably lower than the accuracy achieved by the model trained with just 50 epochs. This discrepancy, despite near-perfect accuracy on the training set, suggests a classic case of overfitting. Consequently, the model with the higher number of epochs fails to generalize well to unseen data, underscoring the importance of selecting an appropriate epoch count to balance learning and generalization capabilities.

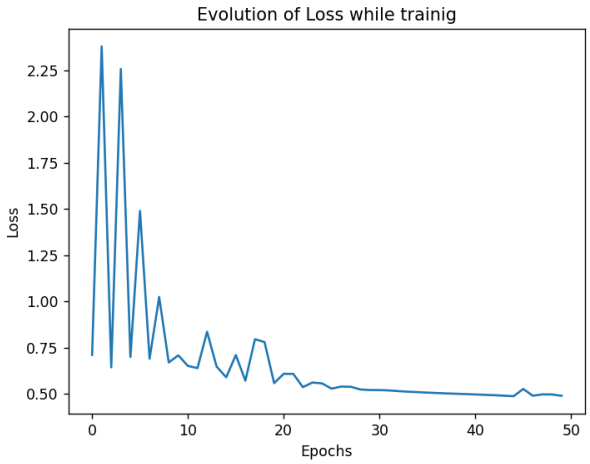


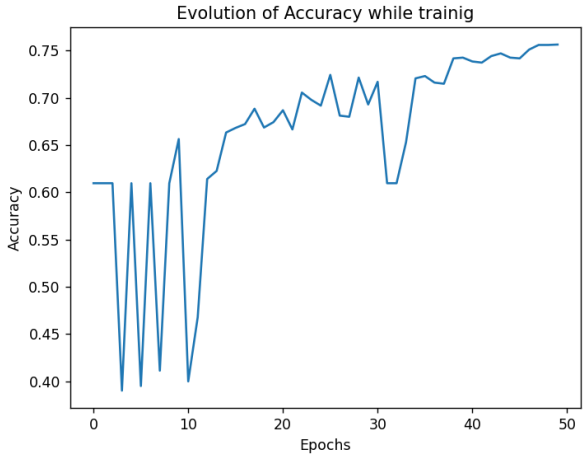
The loss evolution for the model trained over 200 epochs, as opposed to 50, further corroborates the phenomenon of overfitting. Despite the training loss diminishing to minimal levels, the accuracy on the test dataset does not exceed 76%. Through repeated experimentation, it became evident that training for 50 epochs optimizes performance for our dataset. This situation highlights the potential benefits of an early stopping mechanism, which could preempt overfitting by halting training when improvements plateau.

While the Keras framework facilitated the integration of early stopping through a built-in function, enabling the model to cease training typically around 100 epochs, the PyTorch implementation lacked this direct functionality. The observed discrepancy in optimal epoch counts between the PyTorch and Keras models—50 versus approximately 100 epochs, respectively—underscores the nuanced differences in optimizer and loss function behaviors across these libraries. Implementing early stopping in PyTorch, a feature on our agenda for future enhancements, could refine our control over model training dynamics and further align the performance metrics of models developed across both platforms.

Learning Rate

Through iterative experimentation, we established a learning rate of 0.0001 as optimal for our model. While this value may appear modest, it was determined to be the most effective after observing the impact of various learning rates on model performance. To enhance training dynamics further, we incorporated the "ReduceLROnPlateau" function from the PyTorch library. This adaptive mechanism judiciously lowers the learning rate in response to training metrics, facilitating finer adjustments to the learning process and potentially extending it to a lower bound of 1e-6. To illustrate the significance of choosing an appropriate learning rate, we have generated graphs that demonstrate the consequences of employing higher learning rates on the accuracy and loss during the training phase, underscoring the critical role of learning rate optimization in achieving model efficacy.

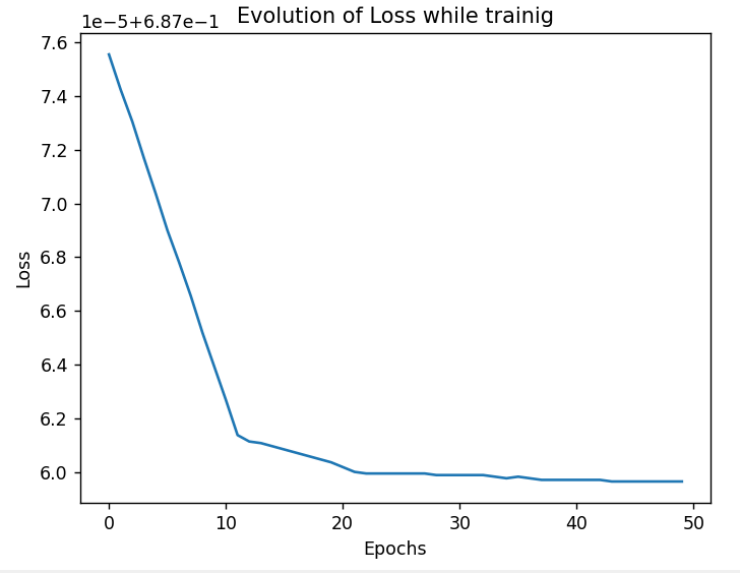




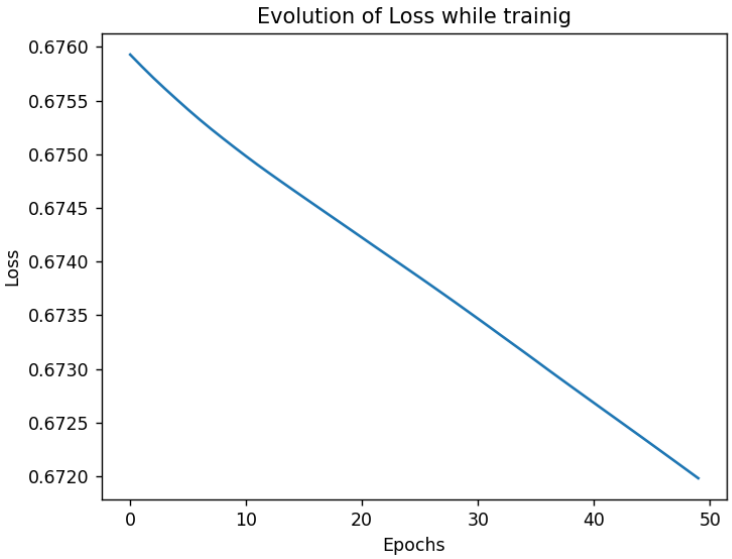
The analysis of two preceding graphs demonstrates that initiating the training with a learning rate of 0.1 results in considerable fluctuations in both accuracy and loss metrics during the early stages. However, the introduction of the learning rate reduction mechanism, "ReduceLROnPlateau", significantly stabilizes the learning curve by adjusting the learning rate downwards, thereby facilitating a more consistent and orderly training process. After extensive experimentation, it has been conclusively determined that a starting learning rate of 0.0001 is most conducive for our model, ensuring optimal learning efficiency and model performance from the outset.

Optimizers

After an exhaustive exploration of the array of potent optimizers available within both the PyTorch and Keras libraries, our investigation was directed towards identifying the most suitable optimizer for our specific dataset. Despite encountering several optimizers that delivered commendable outcomes, we collectively concluded that the optimizers, RPROP (Resilient Propagation) for the Pytorch library and the RMSprop (Root Mean Square Propagation) for the Keras library, emerged as the most compatible with our data, consistently facilitating superior accuracy levels. To provide a comprehensive perspective on this decision, we have prepared a series of graphs illustrating the performance disparities of various prominent optimizers when applied to our dataset, underscoring the unique suitability of RPROP in enhancing our model's predictive accuracy.



The Stochastic Gradient Descent (SGD) optimizer, despite its widespread use and popularity, was found to be ill-suited for our dataset. The graphical analysis revealed that while the loss initially decreased rapidly from a high starting point under SGD optimization, it plateaued at a loss value of 6.0, which is substantially higher than the minimal loss of 0.4 achieved by our current model. This significant level of loss resulted in the SGD optimizer categorizing all data instances as either entirely healthy or sick, demonstrating its inability to effectively discriminate between the two classes in our dataset.



The Adam optimizer, another widely recognized and utilized optimizer, exhibited a steadier and more gradual decrease in loss according to our graphical analysis. However, the graph highlighted a critical limitation: the model's learning rate under the Adam optimizer was

markedly slow. This sluggish learning progression led to an inability to accurately differentiate between healthy and sick patients within our dataset. Consequently, like with the SGD optimizer, the Adam optimizer also resulted in the model making overly generalized predictions, categorizing the entire dataset as either completely healthy or entirely sick. This outcome further underscores the necessity of selecting an optimizer that not only minimizes loss but also maintains an appropriate pace of learning to ensure nuanced and accurate model predictions.

Conclusion.

In conclusion, the model developed through this research demonstrates a high degree of efficacy in predicting diabetes from the selected dataset. Its performance, underpinned by the strategic choice of the RPROP optimizer and an initial learning rate of 0.0001, highlights its potential applicability in clinical settings. The successful normalization of feature scales and the careful calibration of epochs underscore the model's robustness and reliability.

Looking ahead, we identify several avenues for further enhancement of the model's architecture and functionality. Foremost among these is the integration of an early stopping mechanism within the PyTorch framework. This addition would refine the training process by automatically determining the optimal number of training epochs, thereby obviating the need for manual estimation and potentially improving the model's generalizability.

Moreover, the exploration of additional predictive parameters could enrich the model's input, potentially unveiling new insights into the complex interplay of factors contributing to diabetes. Such enhancements, combined with ongoing optimization and validation, could significantly bolster the model's utility as a diagnostic tool, offering a scalable and efficient solution for early diabetes detection.

This study lays the groundwork for future investigations into AI-driven healthcare solutions, emphasizing the importance of precise model calibration and the exploration of diverse data inputs. With continued refinement, there is a promising pathway for deploying this model in real-world clinical scenarios, contributing to the proactive management and early detection of diabetes.