# Advanced Diabetes Diagnosis Using Machine Learning and Deep Learning Approaches

Baya Loues, Said Gadri

Department of Computer Science, Faculty of Mathematics and Informatics,
University Mohamed Boudiaf of M'sila, 28000, Algeria
Laouesbaya1@gmail.com
Said.kadri@univ-msila.dz

DOI: 10.5281/zenodo.16618022

***Abstract*— Diabetes is a chronic disease that weakens the body's ability to regulate blood sugar (glucose) levels. Severe complications may occur if diabetes re-mains untreated or undiagnosed. The traditional method to identify diabetes is to visit a diagnostic center or consult a doctor in the field. However, with the advancements in machine learning (ML) and deep learning (DL) approaches, this critical problem can be addressed more efficiently. In the pre-sent work, we have designed an intelligent predictive model that enables the detection of diabetes in patients with high accuracy. For this purpose, we have used several machine learning algorithms in the first stage, including LR, LDA, KNN, CART, NB, and SVM. In the second stage, we have developed a multi-layer ANN model to improve detection accuracy. Finally, we established a comparison between ML and DL algorithms in terms of detection accuracy.**

**Keywords—** *Diabetes detection, machine learning, deep learning, Artificial neural net-works.*

## 1 Introduction

Diabetes is a chronic disease that weakens the body's ability to regulate blood sugar (glucose) levels. Severe complications may occur if diabetes remains un-treated or unidentified. The best way to identify diabetes is to visit a diabetes diagnostic center or consult a specialist doctor. However, with the advancements in machine learning (ML) and deep learning (DL) approaches, this critical problem can be addressed more efficiently. The main task of using these approaches is to design a predictive model that helps to predict the probability of diabetes in patients with a high accuracy. In simple words, the machine learning area is based on a collection of methods that enable a machine to learn meaningful features (patterns) from data without the assistance of a human (Szegedy 2015). During the last years, a lot of IA tasks such as pattern recognition, image classification, computer vision, etc., have relied essentially on Machine Learning approaches that have shown excellent results (Girshick 2015). Several algorithms have been developed in this area, including logistic regression, k-nearest neighbors, naïve Bayes, decision trees, support vector machines, and artificial neural networks, etc. The research on Artificial Neural Networks ANN can be considered the oldest discipline in ML and AI dates back to J. McCulloch and W. Pitts in 1943, however, these researches were quickly halted because of their high hardware, software, and running time requirements. Several years later, they were revived in parallel with the apparition of a new subfield called deep learning DL. This new subfield

approach greatly simplifies the feature engineering process in many vital areas such as computer vision (Szegedy 2015), (Girshick 2015), image processing (Krizhevsky 2012), (Simonyan 2014), (Gadri 2020), (Gadri 2022), object detection (Ren 2017), (Dai 2016), network optimization (Tu 2018), handwritten digits and character recognition (El-Sawy 2017) , (Gadri 2020), (Torki 2014), sensor networks (Wang 2019), sentiment analysis (Gadri 2021), (Tang 2018), system security (Gadri 2021), Diabetes Detection (Lee 2009). Among the DL methods, CNNs are of special interest. When exploiting local connectivity patterns efficiently which is the case of those used in the ImageNET competition (Pinto 2009). There are many works trying to apply CNNs to image analysis (Turaga 2010), (Abadi 2015) using a variety of methods like the rectified linear unit and deep residual learning (Theano 2016). Many other DL models such as DNNs, RNNs, auto-encoders, and transformers have been applied to various applications in-cluding diabetes detection and given promising results.

# 2 The Proposed Model

In the present work, we have developed an automatic binary classifier that per-mits to identify people affected by diabetes based on some characteristics (fea-tures). Thus, it is a binary classification into two (02) given classes (onset of dia-betes as 1 or not as 0). All of the input variables that describe each patient are numerical. We note also that to build the best predictive model, we have used two approaches: the classic ML approach, and DL approach. Initially, we proceeded with classification task using many ML algorithms including LR, LDA, KNN, CART, NB, and SVM. Then we proposed an ANN model composed of many sim-ple layers: one input layer (100 neurons), four hidden layers (50, 30, 18,4 neurons), and finally an output layer (01 neuron). As programming tools, we have used Python, Tensorflow, and Keras which are the most used in this field.

# 3 Experimentation and Obtained Results

## 3.1 Used dataset

In our experiments on scikit-learn toolkit and ANNS, we used the pima Indians onset of diabetes dataset which is a standard machine learning dataset from the UCI Machine Learning repository. It describes patient medical record data for Pima Indians and whether they had an onset of diabetes within five years. Thus, our problem is a binary classification problem (onset of diabetes as 1 or not as 0). All of the input variables that describe each patient are numerical. This makes it easy to use directly with neural networks that expect numerical input and output values, and ideal to use neural networks in Python, Tensorflow, and Keras. The dataset includes data from 768 women with 8 characteristics, in particular: Number of times pregnant, Plasma glucose concentration 2 hours in an oral glucose tolerance test, Diastolic blood pressure (mm Hg), Triceps skin fold thickness (mm), 2-hour serum insulin (mu U/ml), Body mass index (weight in kg/(height in m)^2), Diabetes pedigree function, Age (years), The last column of the dataset indicates if the person has been diagnosed with diabetes (1) or not (0).

### 3.2 Programming Tools

Python: Python is currently one of the most popular languages for scientific applications. It has a high-level interactive nature and a rich collection of scientific libraries which makes it a good choice for algorithmic development and exploratory data analysis. It is increasingly used in academic establishments and also in industry. It contains a famous module called scikit-learn tool integrating a large number of ML algorithms for supervised and unsupervised problems such as decision trees, logistic regression, Naïve Bayes, KNN, ANN, etc. This package of algorithms allows to simplify ML for non-specialists working on a general purpose.

Tensorflow: TensorFlow is a multipurpose open-source library for numerical computation using data flow graphs. It offers APIs for beginners and experts to develop for desktop, mobile, web, and cloud. TensorFlow can be used from many programming languages such as Python, C++, Java, Scala, R, and Runs on a variety of platforms including Unix, Windows, iOS, and Android. We note also that Tensorflow can be run on single machines (CPU, GPU, TPU) or distributed machines of many 100s of GPU cards

Keras: Keras is the official high-level API of TensorFlow which is characterized by many important characteristics: Minimalist, highly modular neural networks library written in Python, Capable of running on top of either TensorFlow or Theano, Large adoption in the industry and research community, Easy productization of models, Supports both convolutional networks and recurrent networks and combinations of the two, Supports arbitrary connectivity schemes (including multi-input and multi-output training), Runs seamlessly on CPU and GPU.

### 3.3 Evaluation

To validate the different ML algorithms, and obtain the best model, we have used the cross-validation method consisting of: splitting our dataset into 10 parts, train on 9 and test on 1, and repeat for all combinations of train/test splits. For the ANN model, we have used two parameters which are: loss value and accuracy metric.

Accuracy metric: This is a ratio of the number of correctly predicted instances divided by the total number of instances in the dataset multiplied by 100 to give a percentage (e.g., 90% accurate).

Loss value: used to optimize an ML algorithm or DL model. It must be calculated on training and validation datasets. Its simple interpretation is based on how well the ML algorithm or the DL-built model is doing in these two datasets. It gives the sum of errors made for each example in the training or validation set.

*Obtained Results*

To build the best predictive model, we have performed two tasks:

1. Applying many ML algorithms, including Logistic Regression LR, Linear Discriminant Analysis LDA, K-nearest Neighbors KNN, Decision Tree (CART variant), Gaussian Naïve Bayes NB, Support Vector Machine SVM. For this purpose, we used scikit-learn library of Python containing the most known learning algorithms.

2. Designing an ANN (Convolutional Neural Network) model according to the following process:

- We proposed a model composed of six fully connected layers; the first layer has 100 neurons and expects 8 input variables. The second hidden layer has 50 neurons, the third hidden layer has 30 neurons, the fourth hidden layer has 18 neurons, the fifth hidden layer has 4 neurons, and finally, the output layer has 1 neuron to predict the class (onset of diabetes or not).
- The six fully connected layers are defined using the dense class of Keras. We can specify the number of neurons in the layer as the first argument, the initialization method as the second argument, and specify the activation function using the activation argument.
- We initialize the network weights to a small random number generated from a uniform distribution (‘uniform‘), in this case between 0 and 0.05 which is the default uniform weight initialization in Keras. Or ‘normal’ for small random numbers generated from a Gaussian distribution.
- We will also use the rectifier (‘relu‘) activation function on the first five layers and the sigmoid function on the output layer.
- We use a sigmoid on the output layer to ensure our network output is between 0 and 1 and easy to map to either a probability of class 1 or snap to a hard classification of either class with a default threshold of 0.5
- We compile the model using the efficient numerical libraries of Keras under the covers (the so-called backend) such as TensorFlow. The backend automatically chooses the best way to represent the network for training and making predictions to run on your hardware, such as CPU or GPU or even distributed.
- When compiling, we must specify some additional properties required when training the network. Remember training a network means finding the best set of weights to make predictions for this problem.
- We must specify the loss function to use to evaluate a set of weights, the optimizer used to search through different weights for the network, and any optional metrics we would like to collect and report during training. Here, we will use logarithmic loss, which for a binary classification problem is defined in Keras as “binary_crossentropy“.
- We will also use the efficient gradient descent algorithm “adam” for no other reason than it is an efficient default.
- Finally, because it is a classification problem, we will collect and report the classification accuracy as the metric.
- Execute the model on some data.
- We can train or fit our model on our loaded data by calling the fit () function on the model, the training process will run for a fixed number of iterations through the dataset called epochs, which we must specify using the n-epochs argument. We can also set the number of

instances that are evaluated before a weight update in the network is performed, called the batch size, and set using the batch_size argument. For this problem, we will run for a small number of iterations (350) and use a relatively small batch size of 25. These can be chosen experimentally by trial and error.

- We have trained our neural network on the entire dataset and evaluated its performance on a part of the same dataset (the test dataset) using the evaluate() function. This will generate a prediction for each input and output pair and collect scores, including the average loss and any metrics you have configured, such as accuracy.

The following tables summarize the obtained results on ML algorithms and ANN model

TABLE 1. The Average Accuracy after applying various ML algorithms.

| **Algorithm** | Accuracy |
|---|---|
| **LR** | 0.76 |
| **LDA** | 0.76 |
| **KNN** | 0.725 |
| **CART** | 0.72 |
| **NB** | 0.76 |
| **SVM** | 0.64 |

TABLE 2. Description of the Proposed ANN Model.

| **Layer (Type)** | **Output shape** | **Parameters** |
|---|---|---|
| **Dense_1 (Dense)** | (None, 100) | 900 |
| **batch_normalization_1 (Batch** | (None, 100) | 400 |
| **Dense_2 (Dense)** | (None, 50) | 5050 |
| **batch_normalization_1 (Batch** | (None, 50) | 200 |
| **Dense_3 (Dense)** | (None, 30) | 1530 |
| **batch_normalization_2 (Batch** | (None, 30) | 120 |
| **Dense_4 (Dense)** | (None, 18) | 558 |
| **batch_normalization_3 (Batch** | (None, 18) | 72 |
| **Dense_5 (Dense)** | (None, 4) | 76 |
| **batch_normalization_4 (Batch** | (None, 4) | 16 |
| **Dense_6 (Dense)** | (None, 1) | 5 |
| **Total parameters** | 8927 | |
| **Trainable parameters** | 8523 | |
| **Non-trainable parameters** | 404 | |

TABLE 3. Loss and accuracy Metrics obtained through the Application of the proposed model.

| **ANN (100, 50, 30, 18, 4,1)** | | |
|---|---|---|
| **Training set** | Loss: 0.27 | - Acc: 0.89 |
| **Test set** | Loss: 0.25 | - Acc: 0.91 |

**Curves**


Fig. 1. Training loss vs validation loss of the ANN model

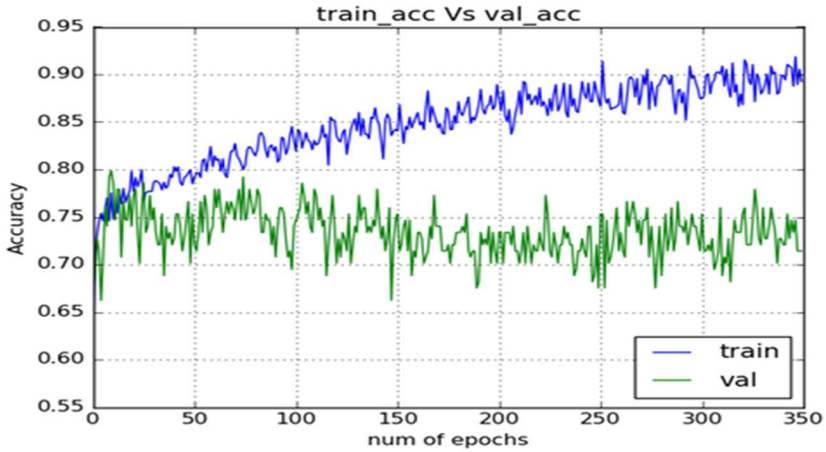


Fig. 2. Training Accuracy vs validation Accuracy of the ANN model

### 3.4. Discussion

Table 1 summarizes the obtained results when applying the different ML algorithms including LR, LDA, KNN, DT (CART), NB, and SVM. We observe that LR, LDA, KNN, CART, and NB give high classification accuracy (> 70%), while SVM gives a relatively low value of accuracy (64%) compared to the previous algorithms.

Table 2 describes the detailed architecture of the proposed ANN model, which is composed of the following layers: the first layer has 100 neurons and expects 8 input variables. The second hidden layer has 50 neurons, the third hidden layer has 30 neurons, the fourth hidden layer has 18 neurons, the fifth hidden layer has 4 neurons, and finally, the output layer has 1 neuron to predict the class (onset of diabetes or not).

In the same way, Figure 1 shows the evaluation of training loss and validation loss over time and in function of the number of epochs. It begins very high for the training set and ends very low because of the large number of samples, but its variation for the validation set is not very quick and appears relatively stable.

Similarly, figure 2 plots the evolution of training accuracy and validation accuracy in function of the number of epochs. Contrary to the loss value, the accuracy starts very low and ends very high. This property is clearer with the training set because of its large size.

We can see also in figure 1 and 2 representing loss value and accuracy value successively that curves do not continue but there exist some pics, that because the two values of loss and accuracy don't progress continuously over epochs, but increase and decrease until stabilizing on a specific value.

## 4. Conclusion and Future Work

In recent years, object recognition mainly uses machine learning (ML), which performs well. Significant progress has been made with deep learning, a new sub-field of ML that uses neural networks to find patterns in large data sets. This ap-proach helps solve complex problems like medical image classification and fraud detection. Results show high accuracy, often above 90%, with digit recognition exceeding 97%. In the present paper, we have designed an intelligent predictive model that enables the detection of diabetes in patients with high accuracy. The proposed model has been trained and validated on the Pima Indian dataset. In the first stage, we used many ML algorithms including LR, LDA, KNN, DT (CART), NB, and SVM to build our model. The obtained accuracy was relatively good. In the second stage, we built a multi-layer ANN model to perform the same task of binary. The achieved performance using the ANN model is very surprising and overpasses the performance of the ML model built in the first stage. As perspectives of this work, we propose to improve these results by improving the architecture of ANN built model by changing some parameters such as the number of layers, the number of neurons in each layer, the number of training epochs, and the size of data batches. Another axe of improvement is to apply other models such as CNNs, auto encoders or combine many models to achieve high performance.

# References

(Abadi 2015) Martín Abadi et al. (2015). TensorFlow: large-scale machine learning on heterogeneous systems. Available: http://tensorflow.org/.

(Dai 2016) Jifeng Dai et al. (2016). R-FCN: Object detection via region-based fully convolutional networks.

(El-Sawy 2017) Ahmed El-Sawy et al. (2017). Arabic Handwritten Characters Recognition using Convolutional Neural Network. *WSEAS Transactions on Computer Research*, 5, 11–19.

(Gadri 2020) Said Gadri. (2020). Efficient Arabic handwritten character recognition based on ML and DL approaches. *Journal of Advanced Research in Dynamical & Control Systems*, 12(07-Special Issue), 9–17.

(Gadri 2020) Said Gadri and Erich Neuhold. (2020). Building Best Predictive Models Using ML and DL. *ICAISC*, Zakopane, Poland. Springer.

(Gadri 2021) Said Gadri. (2021). Developing an efficient predictive model based on ML and DL to detect diabetes. *International Journal of Computing and Informatics*, 45(3).

(Gadri 2021) Said Gadri et al. (2022). Sentiment Analysis: Efficient Model via ML and DL. *ICO 2021*, LNNS, vol. 371. Springer.

(Gadri 2022) Said Gadri and Nour El-houda Adouane. (2022). Efficient Traffic Signs Recognition Based on CNN. *ICO 2021*, LNNS, vol. 371. Springer.

(Girshick 2015) Ross Girshick. (2015). Fast R-CNN. *IEEE ICCV*, pp. 1440–1448.

(Krizhevsky 2012) Alex Krizhevsky et al. (2012). ImageNet classification with deep CNNs. *Neural Information Processing Systems*, pp. 1097–1105.

(Lee 2009) Honglak Lee et al. (2009). Convolutional deep belief networks. *ICML*, pp. 609–616.

(Pinto 2009) Nicolaa Pinto et al. (2009). High-throughput screening of biologically inspired visual representation. *PLoS Computational Biology*, 5(11), e1000579.

(Ren 2017) Shaoqing Ren et al. (2017). Faster R-CNN. *IEEE TPAMI*, 39(6), 1137–1149.

(Simonyan 2014) Karen Simonyan and Andrew Zisserman. (2014). Very deep CNNs for large-scale image recognition.

(Szegedy 2015) Christian Szegedy et al. (2015). Going deeper with convolutions. *CVPR*, pp. 1–9.

(Tang 2018) Zhuo Tang et al. (2018). Self-adaptive Bell-LaPadula model. *IEEE TIFS*, 13(8), 2047–2061.

(Theano 2016) Theano Development Team. (2016). Theano: Python framework for fast computation. *arXiv:1605.02688*.

(Torki 2014) Marwan Torki et al. (2014). Arabic Handwritten Alphabet Recognition: Dataset Study. *arXiv:1411.3519*.

(Tu 2018) Ya Tu et al. (2018). Semi-supervised learning with GANs for signal modulation classification. *Computational Materials and Continua*, 55(2), 243–254.

(Turaga 2010) Srinivas C Turaga et al. (2010). Convolutional networks for affinity graphs in image segmentation. *Neural Computation*, 22(2), 511–538.

(Wang 2019) Jin Wang et al. (2019). Energy-efficient routing algorithm with mobile sink. *Sensors*, 19(7), 1494.