Diabetes Classification Using a Dense Layer Neural Network

Pham Thuy Tien Le

Abstract

Diabetes is an incurable and spreading disease all over the world. Therefore, a good diabetes classifier is important for health professionals to early detect and provide their patients with proper treatment, which can prevent patients from other severe health problems. This paper aims at developing a Dense layer neural network to classify patients with diabetes and patients having no diabetes. Several techniques in data cleaning, data preprocessing and hyperparameters tuning are also employed in this paper, to improve model's performance and find the best model which results in high accuracy score. According to experiments results, the best model is a Dense layer neural network with SELU activation function, Adam optimizer, and learning rate 0.01. The model obtains an accuracy score of about 0.8, specificity score of 0.88, and precision score of 0.75 on testing data.

1. Introduction

Diabetes is one of the spreading diseases worldwide. In 2017, the number of patients suffering from this disease was about 451 million, according to the International Diabetes Federation [7]. In addition, this figure was expected to reach about 693 million by the year of 2045 [7]. Diabetes has two main types, which are Type I and Type II. While Type I is caused due to the inconsistency of blood glucose level, which affects negatively to the production of insulin, Type II is relevant to the phenomenon that body cells become resistant to insulin [7]. Although main causes of diabetes is unknown [7], genetic factors such as family history of diabetes, ethnicity, lifestyle factors including unhealthy diet, lack of physical activities, smoking habits, and other factors like age, overweight were deemed to be related to this disease [2].

Since diabetes is incurable, and it can lead to some other serious diseases such as heart diseases, nerve damage [7], blindness, and kidney diseases [12], early detection and treatment are really necessary in preventing patients from further risks of other severe health problems.

This paper aims at developing a Dense Layer Neural Network (DNN) which can support health professionals in classifying patients with diabetes. Because Pima Indians incidence rate of diabetes was high, research them can represent for global health [2]. Therefore, the Pima Indians Diabetes dataset was employed in this paper.

The organization of this paper is as follows: Sec. 2 presents literature review with several related research papers, Sec. 3 shows details of the chosen method in this paper, Sec. 4 provides details of experiments and analysis of results, and Sec. 5 summarises key points of the paper and presents discussion for future improvements.

2. Literature review

Many related research papers [2, 4, 5, 8, 10] focusing on diabetes classification were conducted. These research papers employed various Machine Learning (ML) methods and Deep Learning (DL) methods.

2.1. Related work using ML methods

According to the research carried out by Gongli *et al*. [8], Pima Indians Diabetes dataset was employed. Authors removed extreme values (outliers) based on the Interquartile range, imputed missing values using Bayesian multivariate normal model with missing values, and standardized data before performing Bayesian variable selection for Logistic regression model. This method resulted in sensitivity score of 0.805, specificity score of 0.875, and Area under the curve value (AUC) of 0.884. These scores were higher than the figures for other methods which do not conduct any preprocessing steps, or methods only impute missing values by using mean values or median values in a basic way.

The research conducted by Victor *et al*. [2] analyzed Pima Indian Diabetes dataset. In this research, three models including Naive Bayes classifier, Random Forest classifier, and J48 Decision tree (Iterative Dichotomiser 3 Decision tree) were deployed. Experiments of this research were split into three rounds. In the first round, invalid values were imputed by median values, and the

chosen three models were built on this imputed data. Then, feature selection methods such as Principle Component Analysis, K-means clustering, and Importance ranking, were executed. In the second round, the chosen three models were built with three most important features including Glucose, BMI, and Age, of the imputed data. In the final round, these three models were built with five most important features of the imputed data, which were Glucose, BMI, Age, Insulin, and Skin thickness. According to models' performance on the imputed data only, while Random Forest classifier occupied the highest values for accuracy score (nearly 0.8), precision score (over 0.89), specificity score (0.75), F-score (about 0.85), and AUC value (about 0.86), J48 Decision tree had the highest sensitivity score of 0.88. In terms of the imputed data with three chosen features and five chosen features, while Naive Bayes classifier outperformed the other two models in terms of accuracy score (around 0.77 - 0.79), F-score (around 0.84) and AUC value (0.84 - 0.86), J48 Decision tree yielded the highest sensitivity score (nearly 0.9). In addition, Random Forest classifier resulted in the highest values for specificity score (about 0.62 - 0.65) in both these cases. Besides, while J48 Decision tree tended to perform better on the imputed data with five most important features, Random Forest classifier had better performance on the imputed data with full features. There were no significant differences in Naive Bayes classifier's performance among these three cases.

As shown in [10], this research used Backward Elimination (BE) method and SVM algorithm to classify patients with diabetes, employing PIMA Indians diabetes dataset. Since BE is a feature selection method, it was used to only keep the most important features such as Plasma Glucose Concentration, Insulin, BMI, Diabetes Pedigree Function, and Age, to build the SVM model. In comparison with model's performance on the original data, the SVM model with BE method achieved higher accuracy score of about 0.86 (using 90% data training).

2.2. Related work using DL methods

According to [5], this research deployed a Deep Neural Network using Stacked Autoencoders. This model was trained on Pima Indians Diabetes dataset. Stacked autoencoders were used to extract interesting features from the dataset. A Dense Neural Network was built with two layers of stacked autoencoders, and the stacked autoencoder was cascaded with a Softmax layer. In addition, backpropagation technique was used to improve model's performance. The final Deep Neural Network model obtained an accuracy score of over 0.86, precision score of over 0.9, and recall score of nearly 0.88.

The research conducted by María *et al*. [4] was about building neural network models, along with two techniques which are Data augmentation using a Variational autoencoder, and Feature augmentation using a Sparse autoencoder. Pima Indians Diabetes dataset was used in this research. Values of pregnancies variable were converted to 0 and 1, with 1 represents for patients had 1 or more times of pregnancy and 0 represents for patients have not been pregnant ever, invalid values in other features were imputed by mean values, and the data was normalized using Min-max scaling method. The Variational autoencoder method was used to reduce the imbalance of the response variable by generating more data for the less representative class. The Sparse autoencoder method was employed to extract 400 new features from the original dataset. Five neural networks were built which are Multi layer perceptron classifier, Sparse autoencoder and Multi layer perceptron classifier trained separately, Sparse autoencoder and Multi layer perceptron classifier trained jointly, Sparse autoencoder and Convolutional neural network classifier trained separately, and Sparse autoencoder and Convolutional neural network classifier trained jointly. Based on results of these models, Sparse autoencoder and Convolutional neural network classifier trained jointly outperformed the other models with an accuracy score of over 0.92.

3. Method description

In this paper, Pima Indians Diabetes dataset was employed, downloaded from https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary/diabetes. The dataset comprises 768 female patients (500 patients having no diabetes, and 268 patients with diabetes), at least 21 years old. Seven diagnostic measurements stored in this dataset are 'pregnancies' variable showing the number of times pregnant, 'glucose' variable showing plasma glucose concentration a 2 hours in an oral glucose tolerance test, 'blood_pressure' variable used for diastolic blood pressure (mm Hg), 'skin_thickness' variable showing triceps skin fold thickness (mm), 'insulin' feature storing values for 2-hour serum insulin (mu U/ml), 'bmi' feature showing body mass index, 'diabetes_pedigree_function' feature used for diabetes pedigree function, and 'age' variable stores patients age. The 'outcome' variable records values of -1 for patients having diabetes, and +1 for patients having no diabetes.

3.1. Data Splitting

The dataset was split into training set and testing set with the ratio of 8:2. The testing data then was split

into validation set and testing set with the ratio of 5:5. After splitting, the training data set 614 observations, the testing set and the validation set have 77 observations.

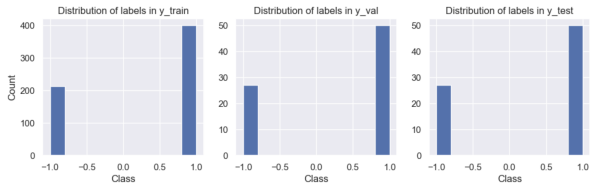

Figure 1. Distribution of classes in the training set, the validation set and the testing set. The ratio between these binary outcomes remains unchanged in all three dataset.

In this paper, Stratified sampling method was used to split the dataset into training set, testing set and validation set. Because the ratio between binary outcomes was unbalanced, Stratified sampling method kept this ratio remains unchanged in all these dataset, which can prevent these dataset from being more unbalanced after splitting. In addition, random state was defined for reproducibility in future improvements. The distribution of outcomes in all these dataset are shown in Fig. 1.

3.2. Data Preprocessing

The training dataset went through three preprocessing steps which are encoding, imputation and scaling. In encoding stage, values of the response variable were encoded as 0 for patients having no diabetes, and 1 for patients with diabetes. In imputation stage, because values of several predictors such as glucose, blood pressure, skin thickness, insulin, and bmi can not equal to 0, values of 0 in these variables were considered as missing values and imputed by their median values, which are the middle values when feature's data are arranged in order. In scaling stage, the training data was also normalized using Standard scaling method to have a distribution with mean equals to 0 and standard deviation equals to 1, based on the following Eq. 1:

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

with n represents for the total number of observations in the training set, mean is computed by Eq. 2:

$$\mu = \frac{1}{n}\sum_{i=1}^{n}(x_i) \tag{2}$$

and standard deviation is calculated as Eq. 3:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2} \tag{3}$$

3.3. Dense Layer Neural Network

DNN models were built with an input layer, several hidden layers, and the output layer having 1 neuron and using Sigmoid function. These DNN models employed Binary cross entropy loss function since this classification task has binary outcomes (0 and 1). These models was set up to run through 300 epochs with batch size of 30, however, Early stopping method was employed to prevent models from being overfitting the training data. Early stopping was defined with patience of 5, and monitored by validation loss values. In addition, DNN models also have other hyperparameters such as optimizers, learning rates, weight initializations, and activation functions, which were tuned to select the best model.

3.3.1 Activation function

Activation functions compute weighted sum of inputs and biases to define whether a neuron's input is important to be activated [16] in DNN models. In addition, activation functions are used to introduce the property of non-linearity to a deep learning model [16]. Because hyperbolic tangent activation function (Tanh) can be saturated, causing the vanishing gradient problem [16], only fours activation function considered in this paper which are Rectified linear unit (ReLU), Exponential linear unit (ELU), Leaky Rectified linear unit (LeakyReLU), and Scaled Exponential linear unit (SELU).

A ReLU activation function can alleviate the issue of vanishing gradients and increase computational simplicity since it only takes a maximum value between a pair of values, as shown in Eq. 4. According to this Eq. 4, three main problems in using this activation function are: (1) Non zero mean, (2) Negative missing and (3) Unbounded output. Non zero mean problem happens because only positive parts are remained unchanged while all negatives values are converted to 0, ReLU function is non-negative and its mean is obviously greater than 0 [16]. Therefore, this problem can affect negatively to model's convergence [16]. When negative missing problem happens, ReLU can reduce model's ability to fit and train the data because all negative values are converted to 0 while this negative part can be helpful for the model. Unbounded output problem means that output of ReLU can range from 0 to infinity [16]. In addition, this activation function becomes non differentiable at 0. A graph of ReLU activation function is shown in Fig. 2.

$$f(x) = max(0, x) \tag{4}$$

A LeakyReLU activation function is a version of ReLU. However, different from ReLU, this activation

function can mitigate the problem of negative missing by multiplying all negative values with a constant 0.01 instead of converting them to 0, as shown in Eq. 5. In particular, LeakyReLU activation function forms a straight line for negative inputs, and its non-zero slope can prevent the gradient from reaching 0 [15]. LeakyReLU has the same disadvantage with ReLU since this function also produces undefined gradients at 0 [15]. A graph of LeakyReLU activation function is shown in Fig. 2.

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0.01x & \text{if } x < 0 \end{cases} \tag{5}$$

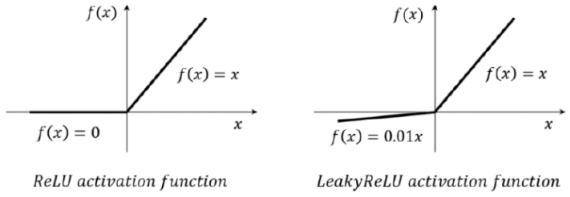

Figure 2. ReLU activation function and LeakyReLU activation function [9]. In ReLU, all negative values are converted to 0 while they are multiplied by 0.01 in LeakyReLU function.

An ELU activation function mitigates a discontinuity at 0 problem of ReLU and LeakyReLU activation function by creating smoothness and continuity around 0 [15]. This activation function creates a smooth curve for negative values by modifying them by a positive α slowly, as shown in Eq. 6. Additionally, ELU has some advantages in increasing the speed of learning by pushing the mean of activation function closer to 0, avoiding neuron death and mitigating bias shift effect, which can be observed in employing ReLU activation function [6]. A graph of ELU activation function is shown in Fig. 3.

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0 \end{cases} \tag{6}$$

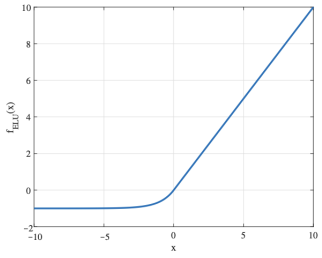

Figure 3. ELU activation function [13]. It forms a smooth line going through negative values instead of converting them to 0.

Like ELU, a SELU activation function can mitigate vanishing gradient problem, decrease training time, and push the mean of activation function closer to 0 [14]. SELU also modifies negative values in a restricted manner with a scale parameter λ (greater than 1) by multiplying this scale value λ with the output of ELU activation function to ensure that a slope of the line going through positive inputs larger than one, as shown in Eq. 7. In addition, SELU is a self-normalizing function, in which a layer keeps its previous layer's mean and variance. A graph of SELU activation is shown in Fig. 4.

$$f(x) = \lambda\{ \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0 \end{cases} \tag{7}$$

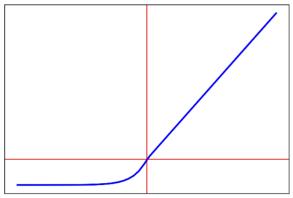

Figure 4. SELU activation function [1]. This function creates a line with slope greater than one for positive inputs.

3.3.2 Sigmoid function

Because the classification task in this paper is binary classification, Sigmoid function was employed to generate probabilities that a patients can have diabetes since patients with diabetes are encoded as 1 and patients having no diabetes encoded as 0, as shown in Eq. 8. The threshold value of 0.5 is used for diabetes classification in this paper. Patients with the probability of having diabetes less than 0.5 are categorized in 'No Diabetes' class, otherwise, patients with this probability greater than or equal to 0.5 are categorized as 'Diabetes' class. However, Sigmoid function also has its own disadvantages such as non zero mean problem (like ReLU activation function), and vanishing gradient problem.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{8}$$

3.3.3 Optimizer

Optimizers are used in DNN models to compute optimal parameters which can minimize the loss functions. Three optimizers considered in this paper are Stochastic Gradient Descent (SGD), Adaptive Moment Estimation (Adam), and Nesterov-accelerated Adaptive Moment Estimation (Nadam). SGD minimizes the loss

function through calculating the gradients. However, SGD can result in local minina instead of global minima, depending on the value of its learning rate. While Adam is a combination of SGD with momentum and Root Mean Squared Propagation (RMSProp), Nadam employs SGD with Nesterov acceleration. SGD with momentum and Nesterov momentum can prevent models from being trapped in local minima, instead, they can reach the global minima and find out optimal parameters. Besides, learning rates are used to control the speed in updating parameters estimates. Two values of learning rates considered in this paper were 0.01 and 0.001.

3.3.4 Weight Initialization

Weight initialization is about defining initial values for DNN model's parameters before these models are trained on the training data. Inappropriate initial weights can lead to two common problems such as Vanishing gradient problem and Exploding gradient problem. Different weight initializations can be constant values such as 0, Uniform distribution, Xavier initialization, and He initialization [3]. In addition, choosing appropriate weight initializations depends on the chosen activation functions [11]. Uniform distribution can be suitable for Sigmoid and Tanh activation function [3]. To Xavier initialization including Xavier uniform and Xavier normal, this methods can be also appropriate for Sigmoid and Tanh activation function [3]. He initialization also comprises He normal and He uniform, this approach can work well with ReLU activation function [3]. Several weight initialization such as zeros, random normal, he normal, lecun normal, glorot normal, and glorot uniform were considered in this paper.

3.4. Model assessment

After fitting the final DNN model on the training set, this model was applied on the testing set to assess its performance and make predictions. Sensitivity score, specificity score, accuracy score, precision score and f-1 score are main metrics to evaluate the model's performance in this paper. These metrics are calculated as following:

$$\text{Sensitivity (TPR)} = \frac{\text{True Positive}}{\text{True Positive + False Negative}} \tag{9}$$

$$\text{Specificity} = \frac{\text{True Negative}}{\text{True Negative + False Positive}} \tag{10}$$

$$\text{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total Positive + Total Negative}} \tag{11}$$

$$\text{Precision (PPV)} = \frac{\text{True Positive}}{\text{True Positive + False Positive}} \tag{12}$$

$$\text{f1} = \frac{2(\text{PPV} * \text{TPR})}{\text{PPV} + \text{TPR}} \tag{13}$$

4. Experiments and results

A baseline model was created with 1 input layer and 1 output layer, using Sigmoid function, SGD optimizer, learning rate of 0.01, 300 epochs, batch size of 30. Early stopping method was also employed. The baseline model was trained on both the original training data and the preprocessed training data. When the baseline model was trained on the original training data, it converged after 19 epochs with maximum accuracy score on the training set of about 0.1612 and maximum accuracy score on the validation set approximately 0. In comparison with model's performance on the original training data, the baseline model built on the preprocessed training data converged after 53 epochs, and obtained higher maximum training accuracy score of about 0.79 and maximum validation score of nearly 0.73. Therefore, data cleaning is an important step in improving model's performance, and the preprocessed training data was used to build DNN models and analyze further.

DNN models were built with an input layer, three hidden layers with the number of neurons were 30, 15 and 8 respectively, an output layer with Sigmoid function, 300 epochs, batch size of 30, and Early Stopping method. Several hyperparameters such as weight initialization, activation function, optimizer and learning rate were tuned to find out the best DNN model, keeping other parameters remain unchanged. The maximum validation accuracy score of DNN models built on each combination of these hyperparameters are shown in Fig. 5. According to Fig. 5, three DNN models yielded the highest maximum validation accuracy score were: (1) A DNN model with SELU activation function, Lecun_normal weight initializer, Adam optimizer and learning rate 0.001 (called Model 1), (2) A DNN model with SELU activation function, He_normal weight initializer, Nadam optimizer and learning rate 0.001 (called Model 2), and (3) A DNN model with ELU activation function, Lecun_normal weight initializer, Adam optimizer and learning rate 0.01 (called Model 3). Although all of their maximum validation accuracy scores were approximately 0.766, Model 1 converged with validation accuracy score of 0.7403, which was the same with the figure for Model 3, and higher than the figure for

Model 2 (0.7013). However, the difference between training accuracy score and validation accuracy score of Model 1 was only about 0.04, while the figure for Model 3 was 0.06, which means that Model 3 can be more likely to meet overfitting problem. Therefore, Model 1 was chosen to be the best DNN model, then it was trained and assessed performance on the testing data.

In comparison with the accuracy score of the baseline model on the testing data (0.7403), this best DNN model achieved higher testing accuracy score of around 0.7922. This increase can happen due to several reasons. First, while the baseline model only had 1 input layer, and 1 output layer of 1 neuron, the best DNN model used 3 additional hidden layers with 30, 15, 8 neurons respectively, which can help to extract more information from the training data, leading to better model performance. Second, the baseline model only used Sigmoid function for the output layer, however, this function can meet non zero center and vanishing gradient problem, hence it can be difficult to find the optimal parameters and affect negatively to model's performance. Besides Sigmoid function, the best DNN model also used SELU activation function for hidden layers, which can alleviates vanishing gradient problem, hence its performance was better than the baseline model. Third, the best DNN model had higher testing accuracy because it employed Adam optimizer, which was contributed by SGD with momentum, while the baseline model only used SGD. Due to the momentum, SGD with momentum can reach global minima instead of being stuck in local minima, which is a problem observed in using SGD. Therefore, SGD with momentum can result in optimal parameters, leading to better model's performance.

Learning rate	Optimizer	Activation function	Weight initialization	Max validation accuracy
0.010	<class 'keras.src.optimizers.sgd.SGD'>	elu	zeros	0.649351
0.010	<class 'keras.src.optimizers.adam.Adam'>	LeakyReLU	zeros	0.649351
0.001	<class 'keras.src.optimizers.nadam.Nadam'>	elu	zeros	0.649351
0.001	<class 'keras.src.optimizers.adam.Adam'>	relu	zeros	0.649351
0.010	<class 'keras.src.optimizers.adam.Adam'>	selu	zeros	0.649351
...	...	...	...	...
0.010	<class 'keras.src.optimizers.adam.Adam'>	LeakyReLU	lecun_normal	0.766234
0.001	<class 'keras.src.optimizers.adam.Adam'>	relu	he_normal	0.766234
0.001	<class 'keras.src.optimizers.adam.Adam'>	selu	lecun_normal	0.766234
0.001	<class 'keras.src.optimizers.nadam.Nadam'>	selu	he_normal	0.766234
0.010	<class 'keras.src.optimizers.adam.Adam'>	elu	lecun_normal	0.766234

Figure 5. The maximum validation accuracy for each combination of activation function, weight initialization, optimizer and learning rate for DNN models

In terms of this best DNN model's performance, its accuracy score was about 0.79, precision score was 0.72, specificity score was 0.86, sensitivity score was nearly 0.67, f1 score was around 0.69, and AUC value was nearly 0.85. Compared to the related work using ML methods shown in Sec. 2, this best DNN model obtained pretty similar specificity score and AUC value with the method conducted by Gongli *et al*. [8], however, its sensitivity score was much lower. In comparison with all methods used in the research of Victor *et al*. [2], while the best DNN model of this paper yields much higher specificity score, its sensitivity score, precision score and f1 score were much lower. Besides, since accuracy score of related work using DL methods shown in Sec. 2 ranged from about 0.86 to 0.92, these scores were higher than the figure for the best DNN model of this paper.

4.1. Code section

Code chunk, employed to execute all experiments in this paper, was uploaded to https://github.com/Tien-le98/Assignment1_DL. This code chunk used to implement data loading, data cleaning, data splitting, data preprocessing, building models, tuning hyperparameters and evaluating model's performance.

5. Discussion and conclusion

Experiments conducted in this paper present that data cleaning plays a vital role in building models, which can significantly affect model's performance. In addition, tuning hyperparameters can contribute in choosing the best DNN model which can result in better overall accuracy, meaning that better model's performance. Depends on the dataset, features, and relationship between features and the outcome variable, choosing appropriate hyperparameters can promote model's performance.

However, there are several points which should be improved in future research in order to increase values of model's assess metrics. First, the imbalance between classes of the response variable should be mitigated through gathering more observations or employing suitable techniques such as Data augmentation to increase the number of observations in less representative class, because the unbalanced outcome variable can have negative effects on this model's performance. For example, lack of observations in the class 'Diabetes' can be a reason for the low sensitivity score. Second, because there are only 768 observations included in this dataset, which can be small for a DNN model to perform well, more observations should be collected because increasing sample size can also improve model's performance. Third, during training the DNN model, there are some cases that overfitting problem occurred because training accuracy score is much higher than validation accuracy score. In future research, other regularization techniques such as Dropout, L1 and L2 regularization method should be considered to minimize this overfit-

ting problem. Last, existing features in Pima Indian Diabetes dataset seem not to have strong relationship with the outcome variable, which may not contribute much to this classification tasks. Hence, in future research, more relevant features should be included to promote model's performance in explaining the outcome variable.

References

[1] Diego Aquino-Brítez, Andrés Ortiz, Julio Ortega, Javier León, Marco Formoso, John Q. Gan, and Juan José Escobar. Optimization of deep architectures for eeg signal classification: An automl approach using evolutionary algorithms. *Sensors*, 21, 2021. 4

[2] Victor Chang, Jozeene Bailey, Qianwen Ariel Xu, and Zhili Sun. Pima indians diabetes mellitus classification based on machine learning (ml) algorithms. *Neural Computing Applications*, 35, 2023. 1, 6

[3] Paul Fergus and Carl Chalmers. *Applied deep learning: tools, techniques, and implementation*. Springer, 2022. 5

[4] María Teresa García-Ordása, Carmen Benavidesb, José Alberto Benítez-Andradesb, Héctor Alaiz-Moretóna, and Isaías García-Rodrígueza. Diabetes detection using deep learning techniques with oversampling and feature augmentation. *Computer Methods and Programs in Biomedicine*, 202, 2021. 1, 2

[5] Kannadasan K, Damodar Reddy Edla, and Venkatanareshbabu Kuppili. Type 2 diabetes data classification using stacked autoencoders in deep t neural networks. *Clinical Epidemiology and Global Health*, 7, 2019. 1, 2

[6] Daeho Kima, Jinah Kimb, and Jaeil Kim. Elastic exponential linear units for convolutional neural networks. *Neurocomputing*, 406, 2020. 4

[7] Souad Larabi-Marie-Sainte, Linah Aburahmah, Rana Almohaini, and Tanzila Saba. Current techniques for diabetes prediction: Review and case study. *Applied Sciences*, 9, 2019. 1

[8] Gongli Li, Yueze Liu, Han Li, Ruikuan Yao, and Chenyang Li. Mcmc impute missing values and bayesian variable selection for logistic regression model to predict pima indian diabetes. *Journal of Physics: Conference Series*, 1865, 2021. 1, 6

[9] Z. Li, W.T. Nash, S.P. O'Brien, Y. Qiu, R.K. Gupta, and N. Birbilis. cardigan: A generative adversarial network model for design and discovery of multi principal element alloys. *Journal of Materials Science Technology*, 125, 2022. 4

[10] F Maulidina, Z Rustam, S Hartini, V V P Wibowo, I Wirasati, and W Sadewo. Feature optimization using backward elimination and support vector machines (svm) algorithm for diabetes classification. *Journal of Physics: Conference Series*, 1821, 2021. 1, 2

[11] Umberto Michelucci. *Applied deep learning with TensorFlow 2: learn to implement advanced deep learning techniques with Python*. Apress, 2022. 5

[12] Radhanath Patra and Bonomali khuntia. Analysis and prediction of pima indian diabetes dataset using sdknn classifier technique. *IOP Conference Series. Materials Science and Engineering*, 1070, 2021. 1

[13] Shuran Sheng, Peng Chen, Yuxuan Yao, Lenan Wu, and Zhimin Chen. Atomic network-based doa estimation using low-bit adc. *Electronics*, 10, 2021. 4

[14] Tao Yang, Yadong Wei, Zhijun Tu, Haolun Zeng, Michel A. Kinsy, Nanning Zheng, and Pengju Ren. Design space exploration of neural network activation function circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38, 2019. 4

[15] Brosnan Yuen, Minh Tu Hoang, Xiaodai Dong, and Tao Lu. Universal activation function for machine learning. *Scientific Reports*, 11, 2021. 4

[16] Yuan Zhou, Dandan Li, Shuwei Huo, and Sun-Yuan Kung. Shape autotuning activation function. *Expert Systems With Applications*, 171, 2021. 3