Deep Learning for Credit Card Default Prediction

————The Report of AI and Applications Coursework

Chenyang Lai
Department of Computer Science
University Of Exeter
cl1030@exeter.ac.uk

***Abstract*—The main conclusions of the report are:**

1) **Choosing activation functions really does make a difference in how well models perform. Specifically, LeakyReLU has been identified as particularly effective due to its construction, which enhances model generalization capabilities.**
2) **Addressing data imbalance through a combination of resampling methods, such as SMOTE and RandomUnderSampler, is crucial for improving model performance in predicting rare events like defaults.**
3) **Bayesian Optimization has proven to be a beneficial method for hyperparameter tuning, significantly improving model performance by identifying optimal settings.**
4) **The selection of appropriate performance assessment indicators, such as using k-fold cross-validation to identify the highest F1 score, is critical for evaluating model accuracy and its balance between precision and recall.**
5) **Employing a custom loss function, like Focal Loss, is advantageous for dealing with uneven data, showing superior ability in identifying less common outcomes compared to traditional binary cross entropy.**

I. Introduction

This report aims to construct a deep learning model that accurately predicts credit card user default. The process involves data preprocessing, model selection, hyperparameter tuning, and model evaluation. The main challenges include addressing data imbalance, selecting an appropriate activation function, and optimizing the model for optimal performance.

A. Related Work and Applicability of Different Deep Learning Methods

In recent years, deep learning has been extensively applied in the field of financial risk assessment [1]. The use of various activation functions, such as ReLU, LeakyReLU, Mish,Swish and PReLU, has demonstrated distinct advantages and limitations. This report evaluates the performance of different activation functions in predicting credit card defaults and explores the impact of data imbalance processing techniques, including SMOTE and RandomUnderSampler.

The application of deep learning techniques has notably improved the accuracy of credit card default predictions, especially with the use of LeakyReLU activation functions and Bayesian optimization methods. Moreover, the model's stability and generalization ability are thoroughly assessed using the k-fold cross-validation method.

The report is structured to first introduce the research background and related work, followed by a detailed methodology section covering data preprocessing, model construction, and optimization strategies. The experimental results and performance evaluation are then presented, culminating in a summary of key findings and future work directions.

II. Research Methodology

A. Data Preprocessing

In this study, the credit card dataset underwent initial normalization to mitigate the impact of varying scales. Given the issue of class imbalance within the dataset, resampling was performed using Synthetic Minority Over-sampling Technique (SMOTE) and Random Under Sampling (RUS) models. SMOTE interpolates new sample points between instances of the minority class, thereby augmenting the minority class sample count to 70% of the majority class, while Random Under Sampling reduces the majority class sample count to approach a 1:0.8 ratio. This combined strategy aims to bolster the model's predictive capability for the minority class, while preventing the predominance of the majority class samples.

B. Model Construction and Loss Function

In the realm of model architecture, a deep neural network featuring the LeakyReLU activation function was selected. LeakyReLU, a variant of the ReLU activation function, permits a non-zero gradient for neurons upon receiving negative inputs, characterized by a small, fixed slope in its negative part [2]. This attribute aids in circumventing the issue of neuron death that ReLU may encounter during the training phase, thereby enhancing the model's adaptability and generalization capacity across diverse data distributions.My model employs a sequential configuration, initiating with an input layer that incorporates 194 neurons. This layer is augmented by a kernel regularizer implementing an L2 regularization with a coefficient of 0.01. Subsequent to the initial dense layer, a Leaky Rectified Linear Unit (LeakyReLU) activation function with an alpha value of 0.02 is applied, followed by a dropout layer with a rate of approximately 0.0165 to mitigate overfitting.The architecture progresses through multiple layers, each characterized by a dense layer with varying neuron counts (299, 249, 263, 147, and 265 respectively), all employing the same L2 regularization coefficient of 0.01 and followed by identical LeakyReLU activations with an alpha value of

0.02. Each of these layers is succeeded by dropout layers with distinct rates (approximately 0.3994, 0.4964, 0.2611, 0.1303, and 0.2071, in order), aiming to further prevent overfitting by randomly omitting a fraction of neurons during training. The culmination of this model is an output layer consisting of a single neuron, which utilizes a sigmoid activation function to yield a probability, indicating the likelihood of the input belonging to a particular class. Initially, a custom weighted binary cross-entropy loss function was explored to address potential model bias due to class imbalance [3]. This weighting strategy aimed to amplify the model's focus on minority class samples by increasing their weight. However, upon comparison with the focal loss function, focal loss was found to be more effective**(see Table I)** in addressing class imbalance by diminishing the influence of easily classified samples and augmenting attention towards difficult-to-classify samples, thereby effectively balancing the learning process.

TABLE I
PERFORMANCE METRICS OF DIFFERENT LOSS FUNCTION
FL= FOCAL LOSS
WBC=WEIGHTED BINARY CROSSENTROPY

Activation Function	Accuracy	Precision	Recall	F1	Loss Func
PReLU(5)	0.800015	0.554954	0.535	0.545	FL
LeakyReLU(6)	0.794667	0.496910	0.600	0.544	FL
Swish(3)	0.790667	0.520107	0.579	0.548	FL
PReLU(6)	0.799333	0.54375	0.519	0.531	WBC
LeakyReLU(6)	0.7935	0.484498	0.595	0.534	WBC
Swish(3)	0.8015	0.509223	0.577	0.541	WBC

C. Optimization Strategy

A Bayesian optimization strategy was employed for automatic hyperparameter adjustment [4]. This involved defining an objective function that takes standardized training data, undergoes resampling through SMOTE and RandomUnderSampler, and then constructs and trains the neural network model. The F1 score on the validation set served as the return value of the objective function, with the optimization goal being to maximize this score. A search space for a series of hyperparameters was defined, including the number of neurons in each hidden layer, dropout rate, learning rate, and batch size. The optimal hyperparameters obtained through Bayesian optimization were used to construct the final model, significantly improving model performance and enhancing its generalization capability. The F1 score was selected as the basis for model optimization and early stopping.

D. Experimental Setup

The experiment employed k-fold cross-validation to evaluate the model's stability and reliability. This methodology aided in assessing the model's performance across different data slices, ensuring the universality of the outcomes.

III. EXPERIMENTAL RESULTS AND ANALYSIS

My experiments focused on comparing the performance of different activation functions—PReLU, ReLU, LeakyReLU (with various (α)values), Mish, and Swish configurations with different hidden layers**(see Table II)**. The results indicate that, although all activation functions performed similarly, LeakyReLU (with ($\alpha = 0.02$) exhibited relatively better performance, especially in configurations with 14 layers.

TABLE II
PERFORMANCE METRICS OF VARIOUS ACTIVATION FUNCTIONS
(0.01)MEANS$(\alpha = 0.01)$

Activation Function	layers	accuracy	precision	recall	F1
PReLU	8	0.805667	0.530303	0.548917	0.539
PReLU	10	0.804667	0.543660	0.534727	0.539
PReLU	12	0.800015	0.554954	0.535474	0.54
PReLU	14	0.793500	0.536075	0.554892	0.545
ReLU	6	0.798333	0.510410	0.567588	0.537
ReLU	8	0.804500	0.572680	0.488424	0.532
ReLU	10	0.802833	0.528908	0.553398	0.541
ReLU	12	0.794167	0.516908	0.559373	0.537
LeakyReLU(0.01)	6	0.801167	0.561037	0.501120	0.529
LeakyReLU(0.01)	8	0.802333	0.490391	0.590739	0.536
LeakyReLU(0.01)	10	0.795000	0.541508	0.530993	0.536
LeakyReLU(0.01)	12	0.789167	0.512718	0.572069	0.541
LeakyReLU(0.01)	14	0.797667	0.481372	0.607916	0.537
LeakyReLU(0.02)	6	0.796167	0.503984	0.566841	0.533
LeakyReLU(0.02)	8	0.803667	0.521769	0.563854	0.542
LeakyReLU(0.02)	10	0.807000	0.520337	0.554145	0.536
LeakyReLU(0.02)	12	0.799500	0.545666	0.526512	0.536
LeakyReLU(0.02)	14	0.794667	0.496910	0.600448	0.544
Mish	4	0.803000	0.489484	0.573562	0.528
Mish	6	0.799500	0.544697	0.536968	0.540
Mish	8	0.796167	0.536585	0.542196	0.539
Mish	10	0.792667	0.535474	0.535474	0.535
Swish	6	0.800667	0.557063	0.521285	0.538
Swish	8	0.790667	0.520107	0.579537	0.548
Swish	10	0.795333	0.513476	0.554892	0.533

A. Hyperparameter Settings and Model Evaluation

My model employed various hyperparameter settings, including learning rate, batch size, optimal accuracy on the validation set, optimal precision, optimal recall, and F1 score. After using Bayesian optimization, the learning rate is set to 0.106089675367281 and the batch size is set to 128. Based on finding the optimal parameter settings, I set the F1 score Callback. The LeakyReLU activation function showed good robustness across multiple hyperparameter configurations. After adjusting with the focal loss function through its parameters ($\gamma = 1.2$) and ($\alpha = 0.6$), I observed that the model's precision and recall reached a balanced state in most cases. To further optimize model performance, I experimented with different optimizers, including Adamax, Adam, SGD, and RMSprop**(see Table III)**, ultimately finding that Adamax allowed the model to achieve the best values in various parameters.

TABLE III
PERFORMANCE METRICS OF DIFFERENT OPTIMIZER

Activation Func	Accuracy	Precision	Recall	F1	Optimizer
PReLU	0.800015	0.554954	0.535474	0.545	Adamax
PReLU	0.804333	0.510934	0.575803	0.541	Adam
PReLU	0.807031	0.492835	0.590739	0.537	SGD
PReLU	0.795477	0.541699	0.528753	0.535	RMSprop
LeakyReLU	0.794667	0.496910	0.600448	0.544	Adamax
LeakyReLU	0.801667	0.485088	0.595220	0.534	Adam
LeakyReLU	0.792833	0.535451	0.541449	0.538	SGD
LeakyReLU	0.797333	0.520921	0.557879	0.539	RMSprop

B. Performance Analysis

The experimental results showed that, under a 6-layer neural network structure using the LeakyReLU activation function, the model achieved an average validation set accuracy of 0.7883, an average validation set precision of 0.5264, and an average validation set recall of 0.5484, resulting in an average F1 score of 0.535. These results were obtained under a learning rate of 0.0112 and using k-fold cross-validation (10 folds). Compared to the configuration using PReLU, LeakyReLU demonstrated better performance on certain metrics, especially when utilizing the k-fold cross-validation method.

Through the compare of the application of k-fold cross-validation and normal split validation **(see Table IV and Table V)**, the model with the LeakyReLU activation function demonstrated its robustness in handling imbalanced datasets. These experimental findings affirm that, in the context of credit card default prediction, the LeakyReLU activation function can effectively enhance the overall performance of the model under specific hyperparameter configurations.

TABLE IV
PERFORMANCE OF THE THREE MODELS UNDER NORMAL SPLIT

Activation Function	Val Accuracy	Val Precision	Val Recall	F1
PReLU	0.800015	0.554954	0.535474	0.545
LeakyReLU	0.794667	0.49691	0.600448	0.544
Swish	0.790667	0.520107	0.579537	0.548

TABLE V
PERFORMANCE OF THE THREE MODELS UNDER K-FOLD CROSS-VALIDATION (10FOLDS)

Activation Function	Val Accuracy	Val Precision	Val Recall	F1
PReLU	0.75591	0.462059	0.530605	0.49
LeakyReLU	0.788345	0.526442	0.548399	0.535
Swish	0.740375	0.439783	0.545682	0.48

IV. CONCLUSION

This comprehensive analysis has revealed that the Leaky Rectified Linear Unit (LeakyReLU) may have advantages in credit card default prediction tasks. By using a neural network architecture consisting of 14 layers and using the Bayesian optimization to get the best hyperparameter, particularly by pre-processing of data through SMOTE and RUS and optimizing the model through k-fold cross-validation. Although the Parametric Rectified Linear Unit (PReLU) and Swish activation functions have demonstrated strong performance under certain configurations,I believe LeakyReLU provides the best overall performance.

In conclusion, the experimental results indicate that, within the chosen activation function and hyperparameter configuration, LeakyReLU is a viable and effective solution for credit card default prediction tasks, particularly when dealing with datasets that have imbalanced classes. Future research will explore the applicability of LeakyReLU in different deep learning models and financial prediction tasks.

REFERENCES

[1] K. Peng and G. Yan, "A survey on deep learning for financial risk prediction," *Quantitative Finance and Economics*, vol. 5, no. 4, pp. 716–737, 2021. [Online]. Available: https://doi.org/10.3934/QFE.2021032

[2] A. Maniatopoulos and N. Mitianoudis, "Learnable leaky relu (lelelu): An alternative accuracy-optimized activation function," *Information*, vol. 12, no. 12, p. 513, 2021. [Online]. Available: https://doi.org/10.3390/info12120513

[3] Y. S. Aurelio, G. M. De Almeida, C. L. de Castro, and A. P. Braga, "Learning from imbalanced data sets with weighted cross-entropy function," *Neural processing letters*, vol. 50, pp. 1937–1949, 2019. [Online]. Available: https://doi.org/10.1007/s11063-018-09977-1

[4] A. H. Victoria and G. Maragatham, "Automatic tuning of hyperparameters using bayesian optimization," *Evolving Systems*, vol. 12, no. 1, pp. 217–223, 2021. [Online]. Available: https://doi.org/10.1007/s12530-020-09345-2