UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI

FUNDAMENTAL AND APPLIED SCIENCE DEPARTMENT

Research and Development

GROUP PROJECT

By:

Nguyễn Minh Đức BI11-057

Bùi Lê Anh Duy BI11-068

Chu Minh Quân BI11-227

Title:

Enhancing Loss Functions to Address Imbalanced Data in Deep Neural Networks

Instructors: Prof. Dr. Nghiêm Thị Phương
ICTLab

Hanoi, March 2023

To whom it may concern,

I, Nghiem Thi Phuong, certify that the group project report of Mr. Nguyen Minh Duc, Mr. Bui Le Anh Duy, and Mr. Chu Minh Quan is qualified to be presented in the Group Project Jury 2022-2023.

Hanoi, Monday, March 6, 2023

Instructor's Signature

Table of Contents

ACKNOWLEDGEMENTS

We would like to express our gratitude to all those who gave us the possibility to complete this thesis in specific as well as academic support in general that we received during the time of our studying at the University of Science and Technology of Hanoi.

Foremost, we would like to send our honest thanks to our supervisor, Nghiem Thi Phuong Ph.D. for guiding us through our research.

LIST OF ABBREVIATIONS

Mean False Error	MFE
Mean Squared False Error	MSFE
Mean Squared Error	MSE
Weighted Cross-Entropy	WCE
Focal Loss	FL
Deep Neural Network	DNN
Fully Convolutional Network	FCN
Convolutional Neural Network	CNN

ABSTRACT

In the last decade, thanks to the many advances of the research community and an ever-increasing amount of data generated, deep learning has found itself becoming exponentially more popular and adored by both academia and industry. Nevertheless, research is often done on curated datasets with a balanced distribution, not unbalanced ones, which are often encountered in real world scenarios. This is due to the fact that using an unbalanced dataset normally would introduce bias to a model. A paper in 2016 by Wang et al. hoped to tackle this issue and proposed two novel loss functions called **Mean False Error** and **Mean Squared False Error**. For our group project, we will do a review of their paper and reconstruct their work to confirm the effectiveness of the loss functions on image datasets.

Keywords: loss functions, deep neural network, data imbalance, deep learning, binary classification

I. INTRODUCTION

1. Context and Motivation

Rapid development of economy and technology in the recent decades has made the internet widely accessible to the public. This in turn has created an increasingly large amount of data, leading into the "Data Economy", where corporations and vendors derive monetary value from the "Big Data". However, this doesn't come without its own set of challenges. The challenge that we would like to focus on in this thesis is data imbalance, which is when certain object classes have limited representation compared to others. This phenomenon is present within all real-world data. A familiar example would be that, within the Vietnamese population, the Kinh ethnic group is vastly larger than the others. Classification problems are typically categorized binary classification and multi-class classification for binary-class datasets and multi-class datasets, respectively. This thesis will focus on the problem of binary classification, and any datasets used throughout will be binary-classed (or transformed into binary-class by binarization). Binary-class datasets are considered unbalanced if the minority class is significantly smaller than the majority class.

Data imbalance can cause unexpected mistakes and dire consequences in data analysis. This is due to the unbalanced distribution of the classes within the dataset causing bias towards the majority class in the classification algorithm, leading to the minority class not being understood properly. As a result of this, standard classifiers (classifiers that don't consider data imbalance) tend to mistake the minority class instances as belonging to the majority class when the dataset is unbalanced. In real-life application, this can cause serious sequences. For example, in a medical setting, if a sick patient were incorrectly diagnosed due to their specific condition being underrepresented in the dataset, they would receive insufficient and incorrect treatment, leading to serious harm.

The problem of data imbalance cannot be adequately dealt with by standard classifiers. Most classification algorithms are designed with the assumption that the dataset it would be fed is balanced and have even distribution between each class. Within some popular algorithms, there have been ways to counter this problem. Examples include data sampling and cost-sensitivity being applied in SVM, neural network and other classifiers. Specifically, sampling is used to transform the unbalanced data into a balanced one and standard classifiers can be made more sensitive to the minority class by adding various cost factors.

However, there have been limited studies done on this matter within deep learning, including Wang's work. Most of the current deep learning algorithms are made for balanced datasets and cannot guarantee good performance on imbalanced ones.

2. Objective

Different forms of loss functions have been proposed to address the problem of data imbalance – specifically, by making the learning algorithms more sensitive to the minority class and yield a higher classification accuracy. The purpose of our thesis is to build a proof of concept to test the performance and effectiveness of the loss functions that's been proposed by (Wang, et al., 2016)[1] and compare its performance to recent novel loss functions.

3. Related Works

Another way to deal with data imbalance is by cost sensitivity to the learning algorithm. Unlike data sampling, cost sensitive learning remedies the problem of data imbalance based on the cost of misidentifying a data sample. For example, you wouldn't hold onto a stock that is about to crash. You would lose a lot of money if you did. In binary classification, correct identifications cost zero, and misidentifying a majority class member would cost less than misidentifying a minority class member. An objective function of the costs can be formulated based on the aggregation of the cost on the entire training set. Although this method can greatly improve classification performance, it relies on the premise that the cost of misclassification is quantifiable, however, that is not always the case in real world scenarios.

a. Data Sampling

Data sampling addresses the problem from the pre-processing phase as it tries to balance the data from the beginning. For example, random oversampling or under sampling work to duplicate or remove samples to achieve a more equal representation of the majority and minority class. While easy to implement, these methods fall short in terms of effectiveness. Other similar but more powerful methods have also been proposed like synthetic minority oversampling technique (SMOTE)[2][3] (generating samples of minority class from existing data), albeit with drawbacks[4][5].

b. Cost Sensitive Learning

Another way to deal with data imbalance is by cost sensitivity to the learning algorithm. Unlike data sampling, cost sensitive learning remedies the problem of data imbalance based on the cost of misidentifying a data sample [6]. For example, you wouldn't hold onto a stock that is about to crash. You would lose a lot of money if you did. In binary classification, correct identifications cost zero, and misidentifying a majority class member would cost less than misidentifying a minority class member. An objective function of the costs can be formulated based on the aggregation of the cost on the entire training set. Although this method can greatly improve classification performance, it relies on the premise that the cost of misclassification is quantifiable, however, that is not always the case in real world scenarios.

c. Deep Learning on Imbalanced datasets

Most efforts aimed to address data imbalance in studies of Neural Networks fall under the umbrella of the three methods mentioned above (alterations of data sampling methods or cost sensitivity or both). Kukar[7] has outlined several methods for cost-sensitive modifications of Neural Networks, while Zhou[8] has studied empirically the effects of sampling and threshold-moving in training. In the case of imbalanced datasets, threshold-moving tries to shift the output closer to the inexpensive classes (majority class) so that the expensive ones (minority class) are harder to misclassify. While there are other works aiming to also solve the same problem, few can be seen so far.

II. METHODS AND MATERIALS

1. Methods

Our pipeline for testing the loss functions include 3 major steps.

The first is the data preparation step. Here, the CIFAR-100 dataset is downloaded and then split into 3 sub-datasets based on the existing class structure. Each dataset is also split into its own set of training dataset and test dataset.

The 2nd step is where we train our custom neural network model.

The 3rd step is where we take the trained model and test it on the 3 above created datasets. The results of the testing are taken down and compared with each other.

2. Loss Functions

In this thesis, we will be testing 2 different loss functions, proposed by Wang et al.

The 1st is the ***Mean False Error*** (MFE) loss function:

$$FPE = \frac{1}{N}\sum_{i=1}^{N}\sum_{n}\frac{1}{2}\left(d_n^{(i)} - y_n^{(i)}\right)^2$$

$$FNE = \frac{1}{P}\sum_{i=1}^{P}\sum_{n}\frac{1}{2}\left(d_n^{(i)} - y_n^{(i)}\right)^2$$

$$l' = FPE + FNE,$$

where FPE and FNE are mean false positive error and mean false negative error respectively, and they capture the error in the negative class and the positive class correspondingly. The loss l' is defined as the sum of the mean errors. N and P are the sizes of the negative class and the positive class respectively. It is recommended that the negative label be assigned to the majority class, while the positive label be assigned to the minority.

The 2nd is the ***Mean Squared False Error*** (MSFE) loss function:

$$l'' = FPE^2 + FNE^2,$$

taking FPE and FNE from the MFE function. The MSFE function was proposed as an improvement upon MFE.

With MFE, minimizing the loss only guarantees minimizing the sum of FPE and FNE. And, in the case of an unbalanced dataset, due to the smaller sample size of the minority class, its errors are less influential on the change of MFE. Therefore, minimizing MFE can't assure a good accuracy on the minority class. And on that point, MSFE can also be expressed as:

$$l'' = \frac{1}{2}\left((FPE + FNE)^2 + (FPE - FNE)^2\right),$$

which means minimizing MSFE can not only minimize the sum of FPE and FNE but also minimize the difference between them. This ensures that the errors on both classes are minimized concurrently.[1]

We will also compare the performance of the above loss functions to other loss functions.

The 3rd loss function that we used in this project is ***Mean Squared Error*** (MSE) loss function, which is an older loss function but simpler and more commonly used. It is defined to be:

$$l = \frac{1}{n}\sum_n \frac{1}{2}(d_n - y_n)^2,$$

with N being the size of the training dataset.

The 4th is the ***Weighted Cross-Entropy*** (WCE)[10] loss function, a popular loss function in imbalance data problems, where the input is the probability that the model made, and the output is an array L of probabilities of each class and is calculated by this formula:

$$L = \frac{1}{N}\sum_{i=1}^{N}(W_p y_i log(p_i) + W_n(1 - y_i)log(1 - p_i))$$

$$W_p = weight\ of\ positive\ class$$

$$W_n = weight\ of\ negative\ class$$

$$y_i = true\ label\ of\ instance\ i$$

$$p_i = prediction\ as\ probability\ of\ instance\ i$$

The final is the ***Focal Loss*** (FL) function, which is the product of the *Cross-Entropy* (CE) loss function and a modulating factor with a tunable "focusing parameter" $\gamma \geq 0$:

$$FL(p_t) = (1 - p_t)^{\gamma} CE(p_t) = -\alpha_t(1 - p_t)^{\gamma} log(p_t),$$

$$p_t = \begin{cases} p, & if\ y = 1 \\ 1 - p, & otherwise' \end{cases}$$

where $y \in \{\pm 1\}$ stands for the ground truth class and $p \in [0,1]$ is the estimated probability for the positive class ($y = 1$). The FL is noted to have 2 properties. (1) When a sample is misidentified and p_t is small, the modulating factor is near 1 and the loss is almost unaffected. But as $p_t \to 1$, the factor goes to 0 and the loss is down-weighted. (2) The parameter γ is for adjusting the effects of the modulating factor, and thus, the rate at which examples are down-weighted. Lin et al.[9] found that $\gamma = 2$ and $\alpha = 0.25$ worked best for them, so we used the same numbers for our tests.

III. EVALUATION

1. Dataset Preparation

We will evaluate the effectiveness of the loss functions on 3 imbalanced datasets – specifically, image sets created from the CIFAR-100 dataset.

CIFAR-100 contains 60,000 images, divided into 100 classes (600 images per class) which are also further divided into 20 superclasses.

However, our project required us to test imbalanced datasets, so we opted to create sub-datasets based on CIFAR-100 rather than use the original dataset. To simplify matters, we created binary datasets consisting of only labels [0,1]. This was achieved by a 3-step process, beginning with the gathering of all instances within a class or superclass for each label. We then determined the necessary number of instances for each label to form the final processed dataset using the formula:

Number of needed instances of label 0 = Total number of instances * (imb_level - 1)

Number of needed instances of label 1 = Total number of instances * imb_level

This ensured that the negative class constituted the majority and the positive class, the minority. The instances were then assigned their respective labels based on their classes and merged to form the final dataset.

Using this method, we created three datasets. Three of these datasets were generated from two superclasses, "`household_electrical_devices`" and "`household_furniture`", and they were combined to form the "`household`" dataset, which had a size of (50000, 32, 32, 3). The dataset size is reasonable as it consists of 50,000 instances, each of which is an image with a size of 32 x 32 pixels and 3 color channels.

Another three datasets were produced from two classes, "`maple_tree`" and "`oak_tree,`" and these were combined to form the "`tree_1`" dataset, which had a size of (500, 32, 32, 3). The final dataset, "`tree_2,`" had the same structure as "`tree_1`" but was generated from two different classes, "`oak_tree`" and "`palm_tree.`"

These datasets were further divided into nine cases, with each dataset having three scenarios controlled by the "`imb_level`" parameter, set at 20%, 10%, and 5%, respectively.

2. Experimental Setup

Our experiment is conducted on both a free Google Colab virtual machine (VM) with GPU acceleration and on a local personal computer. At the moment of result gathering, we are allocated with 1 core of an Intel Xeon CPU @ 2.20GHz, 12.7Gb of system RAM and an NVIDIA Tesla T4 GPU with 15 GiB of RAM. For our local machine, it has an 6 cores Intel® Core™ i5-12400F CPU @ 4.4GHz, 15.5 GiB of system RAM, and an GIGABYTE GeForce RTX 3060 GPU with 12 GiB of RAM.

Our software setup consists of Python 3.8.10, PyTorch & NumPy, PIL, Sklearn, tqdm, and Matplotlib libraries with the addition of the pdb module for debugging purposes. On our local machine, we run it on Visual Studio Code version 1.67.2 of Ubuntu 20.04.4 LTS and GNOME version 3.36.8.

Our training configuration for hyperparameters is defined as follows:

Number of epochs: for MSE, MSFE, and MFE, the value is set to 300, while for FL and WCE, it is 400.

Imbalance level: between 20%, 10% and 5%. A value of 0 signifies that the entire dataset consists of the negative class, while a value of 100% denotes an all-positive class dataset.

Learning rate: default value = 0.1.

Batch size of train set: takes a value between 0 and the maximum size of the training dataset. Default value = 128.

Batch size of test set: takes a value between 0 and the maximum size of the test dataset. Default value = the maximum size.

Users have the option to select the shuffling of train and test sets, choose the loss function, and determine the dataset to use. Shuffling = True by default.

Activation function: we use the logistic activation function with the formula:

$$f(x) = \frac{1}{1+e^{x}}.$$

For different dataset, our DNN's structure also differ as follows:

Dataset	Number of Hidden Layers	Number of Neurons on Hidden Layers (from bottom to up)
Household	3	1000, 300, 100
Tree 1	3	1000, 100, 10
Tree 2	3	1000, 100, 10

3. Evaluation Metrics

The use of accuracy as a metric for evaluating model effectiveness is unreliable, especially when dealing with imbalanced data. For example, if 90% of the data is labeled as 0, even a naive classifier that does not learn anything can still achieve a high accuracy score of 90% by labeling all instances as 0.

To assess the efficacy of the loss functions, we opted to employ two commonly used metrics: the F1 score and the AUC score. Although both metrics are effective, the F1 score is typically favored in cases where the dataset is imbalanced. Our process entails training the model for a suitable number of epochs and subsequently evaluating its performance on the test set.

We first determine the number of correctly and incorrectly classified data points based on their corresponding labels, namely false negative, true negative, false positive, and true positive points, where false points represent misclassified data, true points signify correctly classified data, and negative and positive refer to labels 0 and 1, respectively. For example, suppose point A is classified as false negative. In this case, it implies that its true label is 1, but it was incorrectly labeled as 0.

We then calculate two important metrics to evaluate the model's performance: precision and recall. Precision tells us how many correctly identified positive instances are out of all identified positive instances, while recall tells us how many correctly identified positive instances are out of all positive instances in the dataset. We can use the following formulas to calculate precision and recall, and an addition of False Positive Rate:

$$Precision = \frac{True\ Positive}{True\ Positive\ +\ False\ Positive}$$

$$Recall = \frac{True\ Positive}{True\ Positive\ +\ False\ Negative}$$

$$False\ Positive\ Rate = \frac{False\ Positive}{True\ Negative\ +\ False\ Positive}$$

The F1 score is a statistical measure that combines the Precision and Recall metrics using their harmonic mean. It takes values between 0 and 1, where a value of 0 indicates the worst performance, and a value of 1 represents perfect Precision and Recall. As the F1 score increases, the model's performance improves correspondingly. We can calculate the F1 score by this formula:

$$F1\ Score = 2\frac{Precision\ \times\ Recall}{Precision\ +\ Recall}$$

As for the ROC-AUC score, it means we are calculating the “Area Under the Curve” (AUC) of the “Receiver Operating Characteristic” (ROC), where the ROC curve plots the true positive rate (TPR or Precision) against the false positive rate (FPR) at different classification thresholds. A higher ROC-AUC score indicates a better-performing model, as it accurately identifies more true positives and true negatives.

4. Results

In this study, a DNN model was trained on a designated train set using five different loss functions (MSE, MFE, MSFE, FL, and WCE). After training, the model's effectiveness was evaluated on a separate test set, and the process was repeated for each loss function. The results for both evaluation metrics, F1 score and AUC score, were recorded in a table and saved for analysis and comparison. Additionally, the loss value regressed over epochs of an extremely unbalancing case was also presented for both illustrative purpose and further insights into the performance of each loss function. This allowed us to identify the most effective loss function for this specific scenario and gain a better understanding of the behavior of each loss function in cases of extreme class imbalance. This approach aimed to identify the most effective loss function for the given task and provided insights into the development of DNN models.

Table 1: Experimental Results on Three Image Datasets

Datas et	Imb. level	F-1 Score					ROC-AUC				
		MSE	MFE	MSFE	FL	WCE	MSE	MFE	MSFE	FL	WCE
House -hold	0.2	0.6698	0.6900	0.6772	0.6560	0.6612	0.7819	0.7837	0.7911	0.7658	0.7848
	0.1	0.4867	0.5133	0.5248	0.4740	0.5246	0.7680	0.7835	0.7952	0.7593	0.7949
	0.05	0.2029	0.3789	0.4096	0.3434	0.3696	0.7498	0.7481	0.7744	0.7592	0.7658
Tree 1	0.2	0.6667	0.6914	0.7191	0.6739	0.7222	0.7817	0.7996	0.8021	0.7942	0.8038
	0.1	0.6111	0.5714	0.5455	0.5365	0.5946	0.8119	0.8325	0.8188	0.8119	0.7893
	0.05	0.4000	0.4000	0.4445	0.4286	0.4444	0.8067	0.8522	0.8600	0.8344	0.8189
Tree 2	0.2	0.8864	0.8889	0.8706	0.8642	0.8974	0.9512	0.9617	0.9583	0.9588	0.9633
	0.1	0.8000	0.7143	0.7442	0.7908	0.7179	0.9463	0.9338	0.9362	0.9481	0.9450
	0.05	0.6154	0.6316	0.7368	0.6667	0.6364	0.9478	0.9289	0.9533	0.9444	0.9433

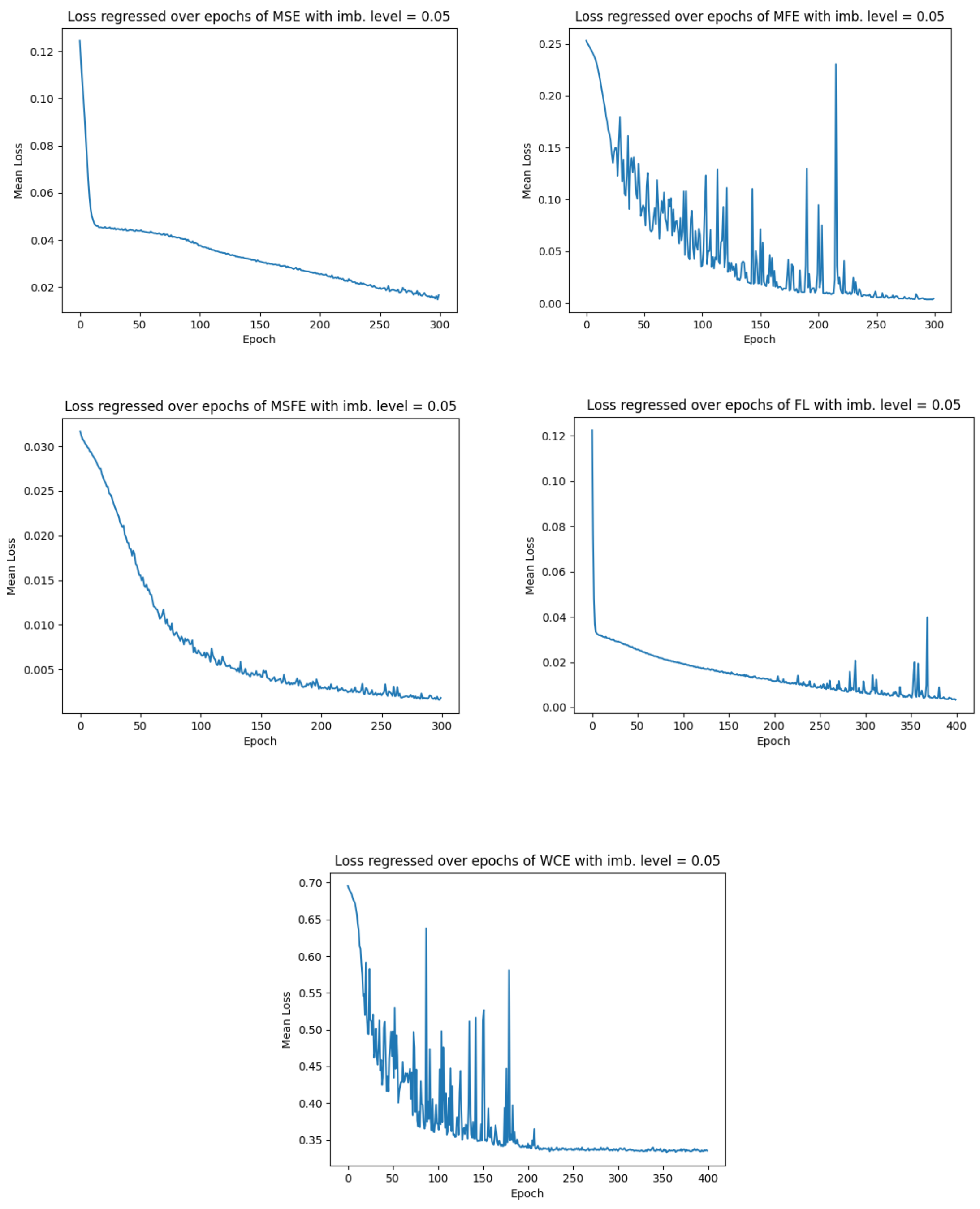

Fig. 1: Comparison of Five Loss Functions under Identical Conditions

The above results confirmed our hypothesis that MSE, although more stable and converging faster during training, performed worse than MFE and MSFE under harsher imbalance conditions. The performance difference between MSE and MFE, MSFE became more apparent with increased dataset imbalance, with MSFE exhibited better stability and produced superior results under increasingly imbalanced datasets when compared to MFE.

5. Discussion

During the project, we encountered two main issues.

Firstly, we noticed that the MFE and MSFE loss converged differently than we had expected. This raised questions as to why this was happening and whether there were any underlying issues that needed to be addressed. We hypothesized that the random initialization and batch size may have been the contributing factors. Further investigation would be required to fully understand the behavior of the MFE and MSFE loss functions and their impact on the overall performance of the model. This affected the accuracy of the model and hindered our ability to achieve the desired results.

Secondly, the lack of availability of computing resources, as students, we were only able to utilize limited resources such as free Colab accounts and a time-limited hired PC to train and evaluate the model under various scenarios. Due to these limitations, we were unable to test many different settings and configurations, which led to results that did not fully align with the proposed approach.

Despite these challenges, the project provided valuable insights into the use of different loss functions and highlighted the need for continued research in this area.

IV. CONCLUSION AND FUTURE WORKS

1. Conclusion

Despite the many advances of research in Deep Neural Network, few have been focused on the issue of imbalanced datasets. The two loss functions proposed by Wang[1] offered us a new approach with higher efficacy than the conventional MSE loss function in a binary classification problem. Through our work, we found that while the claim is true, the differences are marginal and inconsistent at different levels of imbalance. In conclusion, MFE and MSFE serve as new additional tools for problems involving imbalanced datasets, albeit not definitively superior to existing ones.

2. Future Works

To extend our findings, we intend to conduct additional testing of the loss functions in various scenarios, utilizing FCN and CNN architectures. We also plan to investigate the possibility of including additional terms in MFE and MSFE, similar to the weighted loss functions used in previous studies. Furthermore, we will explore the application of these loss functions with the prototype learning method to gain deeper insights into their performance. These approaches could provide valuable insights into the effectiveness of different loss functions under varying conditions.

REFERENCES

[1]. Wang, S., Liu, W., Wu, J., Cao, L., Meng, Q., & Kennedy, P. J, “Training Deep Neural Networks on Imbalanced Data Sets”. *2016 International Joint Conference on Neural Networks.* Sydney: Institute of Electrical and Electronics Engineers, 2016.

[2]. N.V. Chawla, K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer, “SMOTE: Synthetic Minority Over-Sampling Technique,” J. Artificial Intelligence Research, vol. 16, pp.321-357, 2002.

[3]. J. Mathew,M. Luo, C. K. Pang and H. L.Chan , ”Kernel-based smote for SVM classification of imbalanced datasets,” IECON2015, pp.1127-1132.

[4]. B. X. Wang and N. Japkowicz, “Imbalanced Data Set Learning with Synthetic Samples,” Proc. IRIS Machine Learning Workshop, 2004.

[5]. H. B. He and A. G. Edwardo, “Learning from imbalanced data,” IEEE Transactions On Knowledge And Data Engineering, vol.21(9), pp.1263–1284, 2009.

[6]. N. Thai-Nghe, Z. Gatner, L. Schmidt-Thieme, “Cost-sensitive learning methods for imbalanced data,” IJCNN2010, pp.1–8.

[7]. M. Z. Kukar and I. Kononenko, “Cost-Sensitive Learning with Neural Networks,”ECAI 1998, pp.445-449.

[8]. Z. H. Zhou and X.Y. Liu, “Training Cost-Sensitive Neural Networks with Methods Addressing the Class Imbalance Problem,”IEEE Trans. Knowledge and Data Eng, vol. 18 (1), pp.63-77, 2006.

[9]. Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollár, “Focal Loss for Dense Object Detection”, 2017 IEEE International Conference on Computer Vision (ICCV), 2017.

[10]. Bingting Guo, Shuixuan Chen, Zhaobin Hong and Guoqi Xu,“Pattern Recognition and Analysis: Neural Network using Weighted Cross Entropy”, Journal of Physics: Conference Series, 2022.