The Effects of Bi-Label Classification on the Learning Efficiancy of Feed-Forward Neutral Networks

Panupong Jirasetsiri

September 28, 2024

THE EFFECTS OF BI-LABEL CLASSIFICATION ON THE LEARNING EFFICIANCY OF FEED-FORWARD NEUTRAL NETWORKS

Abstract

This study examines the impact of bi-label classification on the performance of feed-forward neural networks (FFNNs) by comparing it with single-label classification models. The research focuses on key performance metrics such as convergence speed, loss reduction, and model stability across three case studies of varying complexity. The findings indicate that bi-label classification consistently achieves faster convergence, lower loss values, and greater stability compared to single-label classification, particularly in simpler datasets. As the complexity of the data increases, the performance gap between the two models narrows, though bi-label classification continues to maintain a slight advantage in terms of loss reduction, generalization, and the ability to handle diverse datasets. These results suggest that bi-label classification can significantly enhance the efficiency of deep learning models, making them more effective in solving tasks involving multiple dependent variables. The study's findings are particularly relevant to fields like predictive analytics and large-scale data classification, where processing speed and model stability are critical.

Keywords: Multi-label Classification, Bi-label classification, Single-label classification, Neural networks, Deep learning

1. Introduction

Machine learning (ML), and more recently, deep learning (DL) [1], have revolutionized data analysis by enabling the efficient handling of large-scale and complex datasets [2]. One of the key applications of DL is classification, where models assign input data to predefined categories. Classification tasks [3] are typically divided into three main types: binary classification, multi-class classification, and multi-label classification.

A study by Kiatsupaibul and Chansiripas [4] has demonstrated the potential for improving data efficiency through deep learning models, particularly multi-response neural networks. Their theoretical framework compares the performance of three types of models: the single-response model, the regular bi-response model, and the constrained bi-response model. The findings make clear that bi-response models, especially constrained ones, outperform single-response models, particularly in terms of convergence speed and loss reduction. During both training and testing, the constrained bi-response model showed faster convergence and lower loss values than the single-response model.

Building on this theoretical foundation, the current research aims to clarify these concepts through practical applications by examining bi-label classification models. Specifically, this study compares the performance of bi-label classification, analogous to a bi-response model, with single-label classification, analogous to a single-response model. The research is conducted through multiple case studies, which vary in architecture and the number of input variables. Performance is measured using key metrics such as convergence speed and loss reduction for the dependent variables y_1 in bi-label classification and y in single-label classification.

In addition to advancing theoretical understanding, this study seeks to clarify and extend the practical applicability of these models, particularly in managing complex and diverse datasets. It is expected that the results will clearly demonstrate the superior efficiency of bi-label classification in real-world applications, reducing the time and resources required for training while enabling the development of more effective models for future use.

2. Research Methodology

This section outlines the neural network architectures used in the study and describes the experimental setup, including data simulation, model structure, loss function optimization and model training.

2.1 Data Simulation

For this research, we generated input data, weights, and biases using Monte Carlo simulations [5]. The aim was to test the performance of the models under controlled conditions. The following summarizes the simulation process:

2.1.1 Input Features

Input vectors $x = [x_1, \dots, x_d]^T$ are generated from a multivariate normal distribution:

$$x(i) \sim \mathrm{N}(0,1), i = 1, \dots, N$$

Here, n is the sample size and d is the number of features.

2.1.2 Weight Initialization

The weights between layers are initialized randomly using a uniform distribution:

$$w_{k,j}^{\ell} \sim Uniform(0,1)$$

These weights represent the connection between node k in layer $\ell - 1$ and node j in layer ℓ.

2.1.3 Bias Initialization

Bias terms for each node in layer ℓ are initialized as follows:

$$b_j^{\ell} \sim Uniform(0,1)$$

The biases node j in layer ℓ account for model flexibility, shifting the activation function.

The data simulation consists of generating input features, weights, and biases for 10 rounds of simulation in each case study. The details are as follows:

Case Study 1:

Input features: $x = [x_1, x_2]^T$, with 2,000 samples.
Weight: $w_{1,1}^1, \dots, w_{4,2}^3$.
Biases: $b_1^1, \dots, b_2^3$.

Case Study 2:

Input features: $x = [x_1, x_2]^T$, with 10,000 samples.
Weight: $w_{1,1}^1, \dots, w_{4,2}^5$
Biases: $b_1^1, \dots, b_2^5$

Case Study 3:

Input features: $x = [x_1, \ldots, x_{10}]^T$, with 20,000 samples.

Weight: $w_{1,1}^1, \ldots, w_{4,2}^3$.

Biases: $b_1^1, \ldots, b_2^3$.

2.2 Model Architecture

The architecture of deep learning models [6], especially in neural networks, typically consists of multiple layers that work together to learn from data. This structure incorporates state-of-the-art techniques and can be described as follows:

2.2.1 Input Layer

The input layer is the first layer of the neural network denoted by $\ell = 0$, where the number of nodes corresponds to the number of input variables. The number of nodes in the input layer is denoted as $m_0 = d$, where d is the number of input features. For each input feature j, it is defined as:

$$x_j = a_j^0 \ , j = 1, \ldots, m_0$$

2.2.2 Hidden Layers

The neural network may contain several hidden layers, denoted by $\ell = 1, \ldots, L - 1$. Each hidden layer applies an activation function to transform the input it receives. Common activation functions include the sigmoid function. The output for each hidden layer ℓ is computed as:

$$a^\ell = f^\ell\left(a^{\ell-1} * w^\ell + b^\ell\right)$$

Where

$a^{\ell-1}$: The output from the previous layer.

w^ℓ: The weight matrix for layer ℓ.

b^ℓ : The bias vector for layer ℓ.

f^ℓ : The activation function (e.g., sigmoid).

2.2.3 Output Layer

The final layer of the neural network is the output layer $\ell = L$, which predicts the target variable. The $y(x)$is calculated using the results from the last hidden layer:

$$y(x) = f^L\left(a_j^{L-1} * w_{k,j}^L + b_j^L\right)$$

Where

f^L : The activation function of the output layer.

In the single-label model, the output $y(x) = [y]^T$consists of one node corresponding to a binary classification problem. For the bi-label model, the output $y(x) = [y_1, y_2]^T$consists of two nodes representing the two labels to each instance.

2.2.4 Activation Function

A common activation function used in both hidden and output layers is the sigmoid function [7], which is defined as:

$$f^{\ell}(x) = \frac{\exp(x)}{1 + \exp(x)}$$

Where

f^{ℓ}: The activation function in $\ell = 1, \dots, L$ layer, specifically the sigmoid function in this instance. This function maps input values to the range (0, 1), which is particularly useful in classification problems, where the output can represent a probability.

The neural network model architecture is divided into layers: an input layer, hidden layers, and an output layer. The number of nodes in each layer and the number of layers vary based on the structure of each case study.

2.3 Case Studies

Three case studies were designed with different neural network architectures:

Case Study 1:

1. Two input features, $x = [x_1, x_2]^T$, with 2,000 samples.
2. The neural network has 2 hidden layers, each with 4 nodes.
3. The output is either 1 node for single-label classification or 2 nodes for bi-label classification.

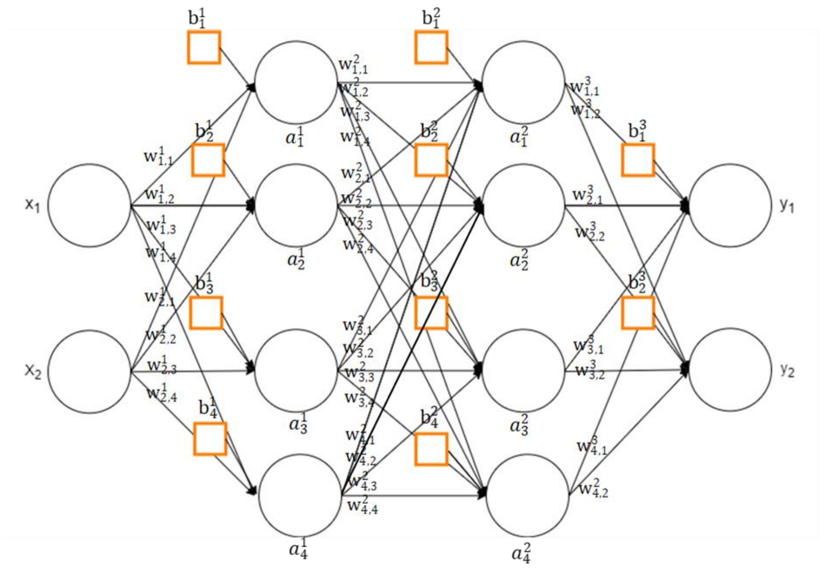

Figure 1: The architecture structure for Case Study 1

Case Study 2

1. Two input features $x = [x_1, x_2]^T$, with 10,000 samples.
2. The neural network has 4 hidden layers, each with 4 nodes.
3. The output is either 1 node for single-label classification or 2 nodes for bi-label classification

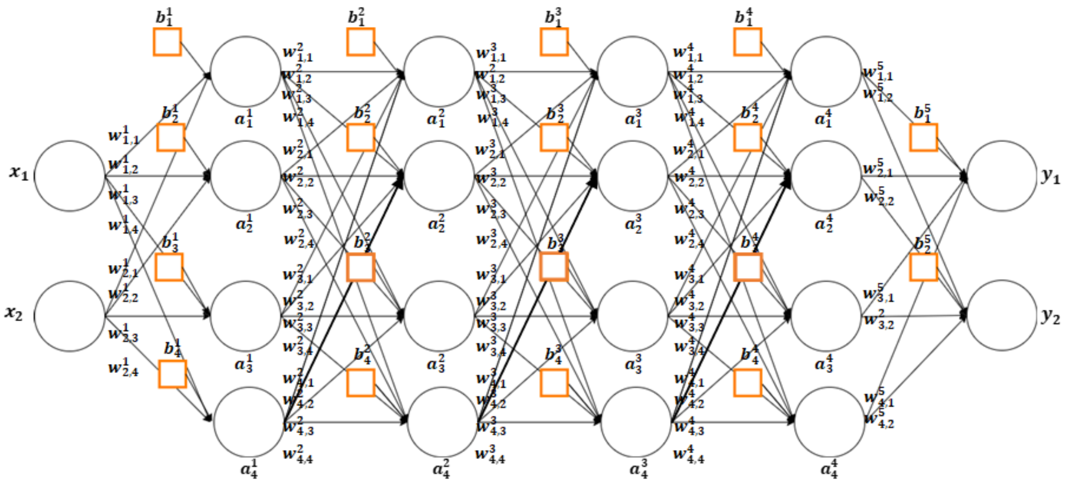

Figure 2: The architecture structure for Case Study 2

Case Study 3

1. Ten input features $x = [x_1, \ldots, x_{10}]^T$, with 20,000 samples.
2. The neural network has 2 hidden layers, each with 20 nodes.
3. The output is either 1 node for single-label classification or 2 nodes for bi-label classification.

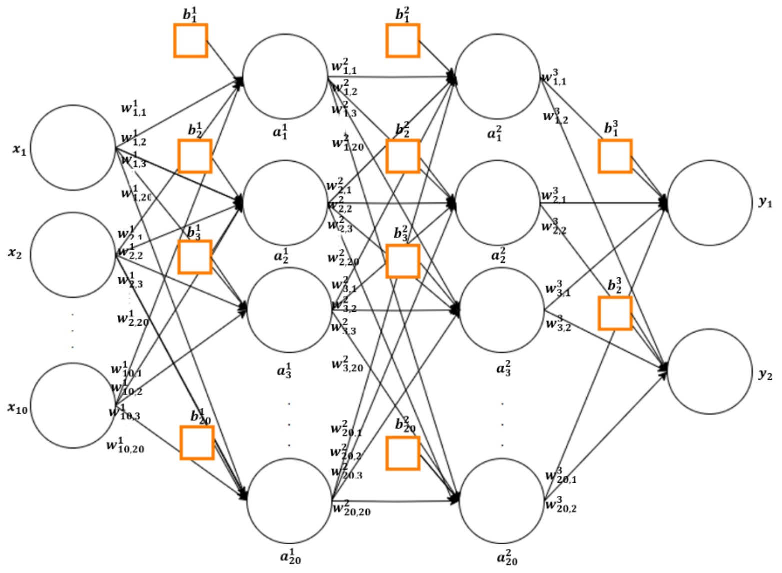

Figure 3: The architecture structure for Case Study 3

In all cases, the sigmoid activation function is applied, and the models are simulated to generate the following results:

1. Single-label classification: $y(x) = [y_1]^T$
2. Bi-label classification: $y(x) = [y_1, y_2]^T$

2.4 Loss Function

The binary cross-entropy (BCE) [8] loss function is used to optimize both the single-label and bi-label models. The BCE function evaluates the loss for a single label as:

$$L(\mathrm{y}, p) \quad = \quad -(y \log(\mathrm{p}) + (1 - y) \log(1 - p))$$

For bi-label classification, the loss is calculated for each label separately, and the total loss is the sum of the individual losses:

$$L(\mathrm{y}_1, \mathrm{y}_2, p_1, p_2) \quad = \quad -(\mathrm{y}_1 \log(p_1) + (1 - \mathrm{y}_1) \log(1 - p_1)) - (\mathrm{y}_2 \log(p_2) + (1 - \mathrm{y}_2) \log(1 - p_2))$$

The goal of training [9] is to minimize this loss function using stochastic gradient descent (SGD).

2.5 Model Training

For each study case, the model will be trained for 10 rounds to evaluate the model's performance and stability. Model training involves various steps such as compiling the model using libraries like Keras [10] and TensorFlow, setting batch size[11], and determining the number of epochs. The training process follows these configurations:

Shared Settings for All Case Studies:
- Optimizer: Stochastic Gradient Descent (SGD)
- Epochs: 50
- Batch Size: 100
- Training/Test Split: 0.5 (50% training, 50% test)
- Activation Function: Sigmoid
- Loss Function: Binary Cross-Entropy

Differentiated Learning Rates:
- Case Study 1: Learning rate = 0.1
- Case Study 2: Learning rate = 0.01
- Case Study 3: Learning rate = 0.001

3. Research Results and Discussion

The results of the study are discussed across three case studies, each varying in network complexity and input data characteristics.

3.1 Case Study 1

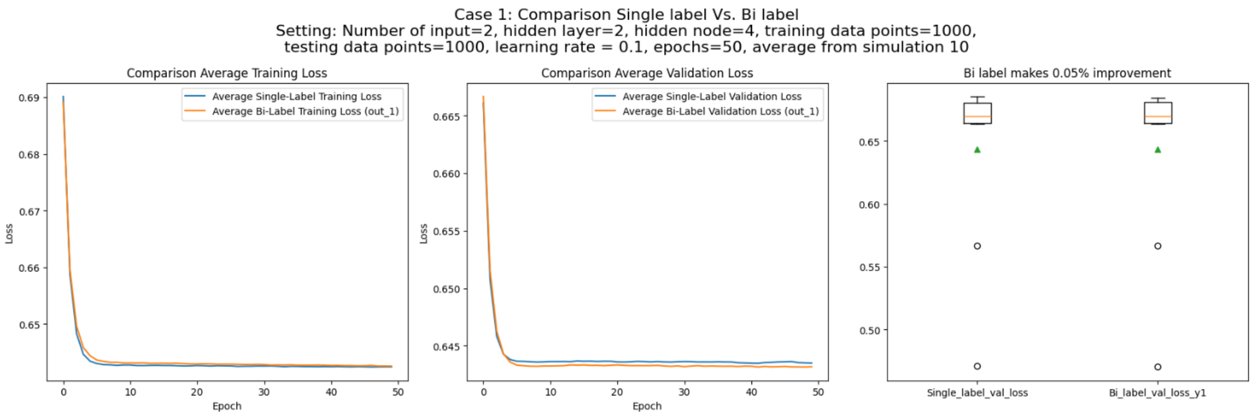

Figure 4: Results from Case Study 1

Results:

Training Loss: The bi-label model showed faster convergence, reducing the loss more rapidly than the single-label model. By epoch 25, the bi-label model achieved a stable minimum loss, whereas the single-label model continued to oscillate slightly.

Validation Loss: The validation loss for the bi-label model was consistently lower, with a final improvement of 0.05% over the single-label model. This indicates that the bi-label model generalizes better to unseen data.

Conclusion: The bi-label model demonstrates superior learning speed and loss reduction for small datasets and simpler architectures.

3.2 Case Study 2

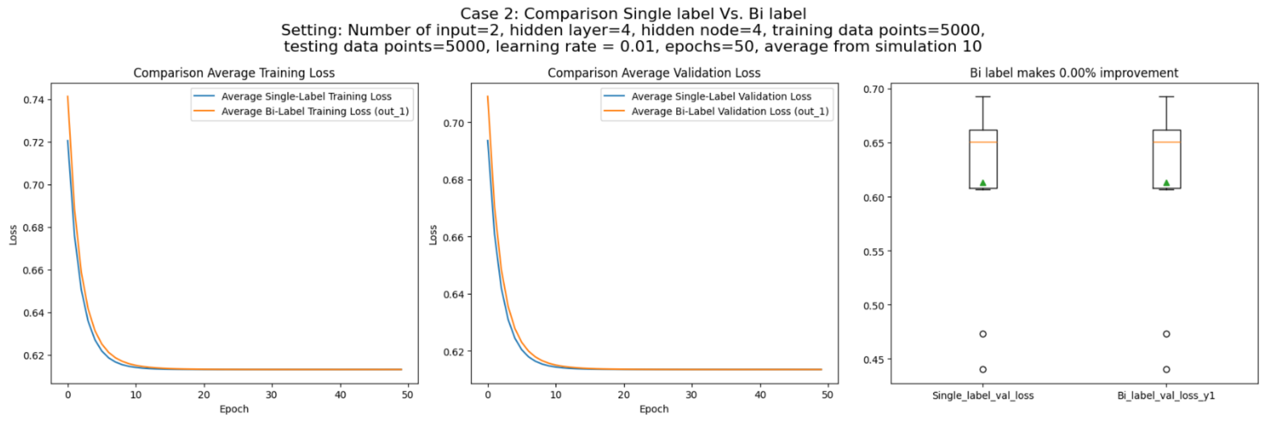

Figure 5: Results from Case Study 2

Results:

Training Loss: Both models showed similar performance during training. The loss values converged at nearly the same rate, with the bi-label model exhibiting a slight edge in early epochs.

Validation Loss: Although the bi-label model outperformed the single-label model, the margin of improvement was smaller (approximately 0.00% difference). The models exhibited comparable generalization performance as the data complexity increased.

Conclusion: For larger datasets, the difference in performance between single-label and bi-label models diminishes. However, the bi-label model still retains a slight advantage in loss reduction.

3.3 Case Study 3

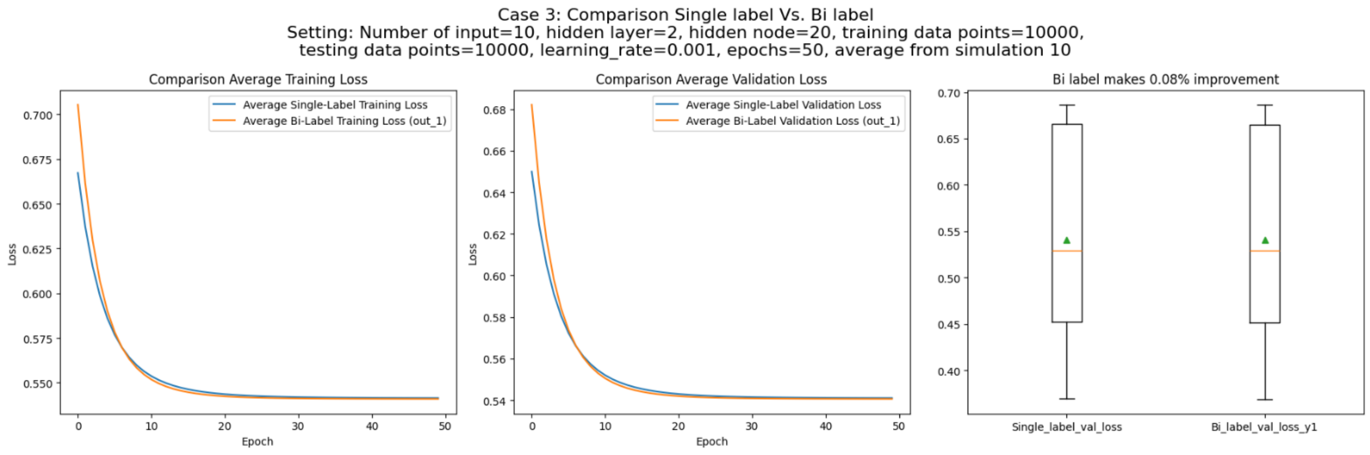

Figure 6: Results from Case Study 3

Results:

Training Loss: The bi-label model showed significant improvement in loss reduction, converging faster than the single-label model. The bi-label model reached a stable loss after 30 epochs, while the single-label model required additional epochs to stabilize.

Validation Loss: In this case, the bi-label model reduced the validation loss by 0.08%, showing a clear improvement in generalization for complex data.

Conclusion: With increasing complexity, the bi-label model significantly outperforms the single-label model in terms of both training efficiency and validation performance.

4. Conclusions and Recommendations

4.1 Conclusions

This study compared the performance of single-label and bi-label classification models across three case studies. The results indicate that bi-label classification provides faster convergence and reduced loss, particularly in scenarios with lower data complexity. As data complexity increases, the performance difference between the two models diminishes, although the bi-label model continues to demonstrate slight advantages.

4.2 Advantages

Bi-label classification consistently achieves lower training and validation loss. This reduction is evident particularly in simpler datasets, where faster convergence rates are observed. Moreover, the bi-label model demonstrates superior stability during the validation phase when compared to the single-label model, suggesting improved generalization performance.

4.3 Limitations

As the complexity of data increases, the performance gap between single-label and bi-label models diminishes. Additionally, bi-label models are sensitive to the learning rate, requiring precise tuning for optimal performance based on the dataset.

4.4 Applications

Bi-label classification is well-suited for fields requiring multiple outputs, such as medical diagnosis or multimedia tagging. Its ability to handle concurrent conditions or categories makes it highly effective in such domains. This approach also proves beneficial in environments where faster training times and improved model stability are essential.

5. Acknowledgements

6. References

1. Moroney L. AI and Machine Learning for Coders: O'Reilly Media; 2020.
2. Sabharwal R, Miah SJ. A new theoretical understanding of big data analytics capabilities in organizations: a thematic analysis. Journal of Big Data. 2021;8(1).
3. Venkatesan R, Er MJ, editors. Multi-label classification method based on extreme learning machines. 2014 13th International Conference on Control Automation Robotics & Vision (ICARCV); 2014 10-12 Dec. 2014.
4. Kiatsupaibul S, Chansiripas P. Data Efficiency in Multi-response Neural Networks. Unpublished work. 2024.
5. Thaler D, Elezaj L, Bamer F, Markert B. Training Data Selection for Machine Learning-Enhanced Monte Carlo Simulations in Structural Dynamics. Applied Sciences [Internet]. 2022; 12(2).
6. Géron A. Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems: O'Reilly Media, Incorporated; 2019.
7. Dubey SR, Singh SK, Chaudhuri BB. Activation functions in deep learning: A comprehensive survey and benchmark. Neurocomputing. 2022;503:92-108.
8. Wu T, Huang Q, Liu Z, Wang Y, Lin D, editors. Distribution-Balanced Loss for Multi-label Classification in Long-Tailed Datasets. Computer Vision – ECCV 2020; 2020 2020//; Cham: Springer International Publishing.
9. Alagözlü M. Stochastic Gradient Descent Variants and Applications2022.
10. Gulli A, Pal S. Deep Learning with Keras: Packt Publishing; 2017.
11. Ioffe S, Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. 2015.