# Revisiting One-Versus-One and One-Versus-Rest: Insights into Imbalanced Multi-class Classification

Kuan-Ting Chen
Mohamed bin Zayed University of Artificial Intelligence
andyngewcz2d@gmail.com

Chih-Jen Lin
Mohamed bin Zayed University of Artificial Intelligence
chihjen.lin@mbzuai.ac.ae
National Taiwan University
cjlin@csie.ntu.edu.tw

***Abstract*—One-versus-one (OVO) and one-versus-rest (OVR) are two widely adopted methods to decompose multi-class problems into several binary classification problems. It is well known that, in the case of kernel SVM, the two methods yield similar test accuracy. Thus, people generally assume that they differ mainly in training time and model size. However, our research reveals that if one considers an evaluation metric taking class imbalance into account, these two methods may give notable performance differences. To explore this phenomenon, we first conduct a detailed analysis of kernel SVM and then extend our study to neural networks. Additionally, we propose novel loss functions for neural networks that effectively integrate the OVO and OVR perspectives. Our experiments clearly demonstrate the robustness of OVO in handling imbalanced multi-class classification, highlighting its advantages over OVR in these challenging scenarios.**

***Index Terms*—multi-class classification, imbalance dataset, one-versus-rest, one-versus-one**

## I. Introduction

Multi-class classification is essential for solving real-world problems, such as disease diagnosis in healthcare and fault detection in industrial systems. A common approach is to decompose the multi-class task into several binary classification problems [1]–[4], which can be effectively addressed using binary classifiers like Support Vector Machine (SVM) [5], Logistic Regression [6], and Naive Bayes, etc. Among the decomposition methods for multi-class classification, one-versus-rest (OVR) [7] and one-versus-one (OVO) [8] are two widely used approaches. The OVR method trains a separate classifier for each class to distinguish it from all other classes. In contrast, the OVO method trains a classifier for each pair of classes, focusing on separating the two selected classes. After training the binary classifiers, their outputs are aggregated to produce multi-class predictions.

The differences between OVR and OVO have been widely studied, especially in terms of their performance and computational characteristics. In the context of kernel SVM, [9] conducted extensive experiments comparing these two methods. Their findings showed that OVR and OVO perform similarly in terms of accuracy.[1] Consistent observations were also reported in [10]. As a result, this view has gained wide acceptance and recognition in the community, making [9] a highly cited conclusion. Consequently, it is generally assumed that the main differences between the two methods lie in computational aspects, such as model size and training time [9].

However, our findings challenge this prevailing paradigm. We demonstrate that when evaluation metrics designed to account for class imbalance, such as Macro-F1 score [11], Balance Accuracy (BA) [12], and G-mean [13], are applied, OVR and OVO exhibit notable performance differences. To comprehensively investigate these differences between OVR and OVO, we further extend our study to neural networks, a popular choice in machine learning.

In neural networks, cross-entropy is widely used as the standard loss function for multi-class classification. From a different standpoint, we propose novel loss functions for neural networks that effectively integrate the concepts of OVR and OVO. Through theoretical analysis and experiments, we demonstrate that the OVR loss closely aligns with cross-entropy, while the OVO loss outperforms the OVR loss under imbalance, revealing aspects that were previously overlooked.

In summary, our contributions are as follows:

- Our kernel SVM experiments challenge the belief that OVR and OVO mainly differ in model size and training time, and we observe similar findings in neural networks.
- We propose novel loss functions that incorporate the concepts of OVR and OVO for our study on neural networks.

This paper is organized as follows. Section II introduces the OVR and OVO methods and their applications to kernel SVMs. We also propose new loss functions that adapt these concepts for use in neural networks. Section III describes the experimental setup, covering datasets, data split, hyperparameter search, and evaluation methods. Section IV presents the experimental results along with a detailed analysis. Finally, Section V provides the conclusions.

## II. Multi-class Decomposition Methods

In this section, we introduce two common decomposition methods, OVR and OVO, followed by their applications to kernel SVMs and neural networks. Consider a multi-class classification problem with $k$ classes, where the dataset $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ consists of $N$ instances. Each instance includes a feature vector $\boldsymbol{x}_n \in \mathbb{R}^d$ with $d$ features, and a class label $y_n \in \{1, 2, \ldots, k\}$ associated with $\boldsymbol{x}_n$.

[1] In linear SVM, [9] has shown that OVO can achieve higher accuracy.

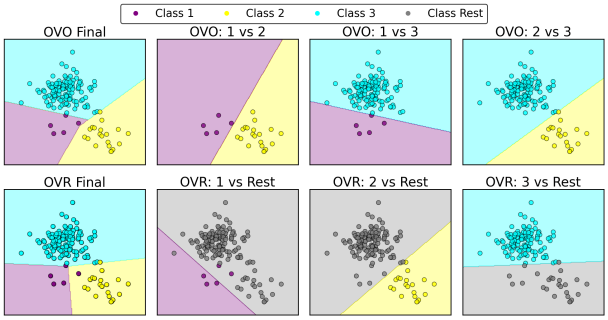


Fig. 1: Example of decision boundaries in a three-class problem using OVR and OVO. The leftmost column shows multi-class results, and the remaining columns display individual binary classifiers.

### A. One-Versus-Rest and One-Versus-One

For each class $i$, the OVR method trains a binary classifier $f_i$ using all instances, where instances in class $i$ are positives and the rest are negatives. Therefore, OVR produces $k$ binary classifiers for $k$ classes. When making predictions, a straightforward approach is to select the class with the largest decision value because this decision value reflects the strongest confidence among all classes. For an input instance $\boldsymbol{x}_n$, the predicted class is:

$$\text{class of } \boldsymbol{x}_n \equiv \arg\max_{i=1,\dots,k} f_i(\boldsymbol{x}_n), \tag{1}$$

where $f_i(\boldsymbol{x}_n)$ is the output of the binary classifier for class $i$.

In the OVO method, we train each classifier $f_{i,j}$ using only instances from classes $i$ and $j$ $(1 \le i < j \le k)$ to distinguish between these two classes, and thus obtain $k(k-1)/2$ binary classifiers for all class pairs. Unlike the OVR method, which assigns a single decision value to each class, OVO relies on pairwise comparisons and cannot directly select the largest decision value for prediction. Instead, majority voting [14], as implemented in LIBSVM [15], is commonly used. At inference time, if $f_{i,j}(\boldsymbol{x}_n) > 0$, a vote is assigned to class $i$; otherwise, class $j$ receives the vote. The final prediction is the class with the highest vote count:

$$\text{class of } \boldsymbol{x}_n \equiv \arg\max_{i=1,\dots,k} \text{votes}_i(\boldsymbol{x}_n), \tag{2}$$

where $\text{votes}_i(\boldsymbol{x}_n)$ is the total votes received by class $i$.

Fig. 1 illustrates the overall decision boundaries of OVR and OVO in a three-class problem, along with the decision boundaries for every binary problem. In this OVR setting, when class 1 is the target, classes 2 and 3 are merged into a single "rest" class. This merging process causes class imbalance and may lead to misclassifications between class 1 and the rest. In contrast, OVO trains on class pairs, and we suspect this design may offer more balanced data for better separation. Therefore, we study OVR and OVO in more detail.

### B. Kernel SVM for Multi-class Classification

For a direct comparison with [9], we employ kernel SVM to construct binary classifiers for OVR and OVO methods. In OVR, the binary classifier $f_i$ assigns each instance $\boldsymbol{x}_n$ a label $y_i(\boldsymbol{x}_n) = 1$ if $y_n = i$, and $y_i(\boldsymbol{x}_n) = -1$ otherwise. The output of $f_i$ is defined as $f_i(\boldsymbol{x}_n) = \boldsymbol{w}^T\phi(\boldsymbol{x}_n) + b$, where $\boldsymbol{w}$ and $b$ are the weight vector and bias term, and the function $\phi$ maps $\boldsymbol{x}_n$ to a higher-dimensional space through a nonlinear transformation. The optimization problem for $f_i$ in kernel SVM is:

$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{n=1}^{N} \xi(\boldsymbol{w}, b; y_i(\boldsymbol{x}_n), \boldsymbol{x}_n), \tag{3}$$

where $\xi(\boldsymbol{w}, b; y_i(\boldsymbol{x}_n), \boldsymbol{x}_n)$ is the loss function for each instance $\boldsymbol{x}_n$ with label $y_i(\boldsymbol{x}_n)$, and $C$ is the regularization parameter.

In OVO, each binary classifier $f_{i,j}$ labels instances from class $i$ as $y_{i,j}(\boldsymbol{x}_n) = 1$ and instances from class $j$ as $y_{i,j}(\boldsymbol{x}_n) = -1$, with output $f_{i,j}(\boldsymbol{x}_n) = \boldsymbol{w}^T\phi(\boldsymbol{x}_n) + b$. The corresponding kernel SVM optimization problem for $f_{i,j}$ is:

$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{n\in O_i\cup O_j} \xi(\boldsymbol{w}, b; y_{i,j}(\boldsymbol{x}_n), \boldsymbol{x}_n), \tag{4}$$

where

$$O_i = \{n \mid y_n = i\} \tag{5}$$

represents instances from class $i$.

After training $k$ models for OVR and $k(k-1)/2$ for OVO, their binary predictions are respectively combined using (1) and (2) to obtain the final multi-class decision.

Although OVO trains more models than OVR, each model uses data from only two classes. This makes each problem smaller and faster to train. As a result, the total training time can be less than that of OVR (see [9] for more details).

### C. Neural Network for Multi-class Classification

To comprehensively compare OVR and OVO, we extend our study to neural networks due to their widespread use in modern machine learning. Typically, neural networks for multi-class classification employ a shared-weight architecture, where the last layer produces outputs $f_1, f_2, \dots, f_k$ for each class.

To apply OVR and OVO on neural networks, although we could train $k$ independent binary networks for OVR and $k(k-1)/2$ for OVO, as done in kernel SVM, this setup differs from common neural network settings and requires high computational resources. Instead, we maintain a single network that produces $f_1, f_2, \dots, f_k$ in the last layer and design novel loss functions that seamlessly integrate OVR and OVO concepts. This design enables direct comparison between OVR and OVO without the computational burden of training multiple networks.

A commonly used loss function in neural networks for multi-class classification is the cross-entropy (CE) loss:

$$-\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{k} \mathbf{1}_{\{y_n=i\}} \log p_i(\boldsymbol{x}_n), \tag{6}$$

where

$$p_i(\boldsymbol{x}_n) = \frac{e^{f_i(\boldsymbol{x}_n)}}{\sum_{j=1}^{k} e^{f_j(\boldsymbol{x}_n)}} \tag{7}$$

is the probability of class $i$ versus the rest and the indicator function $\mathbf{1}_{\{y_n=i\}} = 1$ if $y_n = i$, and 0 otherwise. The idea behind CE loss is to consider all classes at once through maximum log-likelihood optimization, but OVR and OVO rely on binary comparisons. To reflect the designs of OVR and OVO, we propose novel loss functions that maximize the log-likelihood of the predicted probabilities for instances in their corresponding binary settings.

For OVR, the loss in the binary problem for each class $i$ aims to maximize $p_i(\boldsymbol{x}_n)$ for $O_i$ and $1-p_i(\boldsymbol{x}_n)$ for the others. We define the total OVR loss across all $k$ classes as the average negative log-likelihood over all instances:

$$-\sum_{i=1}^{k}\frac{1}{N}\left(\sum_{n\in O_i}\log p_i(\boldsymbol{x}_n)+\sum_{n\notin O_i}\log(1-p_i(\boldsymbol{x}_n))\right). \quad (8)$$

Instead of comparing class $i$ versus the rest from the class perspective, we prove in Appendix A-A that (8) is equivalent to the following form from the instance perspective:

$$-\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{k}\Big(\mathbf{1}_{\{y_n=i\}}\log p_i(\boldsymbol{x}_n)+ (1-\mathbf{1}_{\{y_n=i\}})\log(1-p_i(\boldsymbol{x}_n))\Big). \quad (9)$$

Interestingly, this OVR setting is similar to CE because the first term of (9) is identical to (6), with only a difference in the second term, as each binary comparison introduces the negative instances.

In OVO, for each class pair $(i, j)$, we can simply consider classes $i$ and $j$ in the denominator of (7) to define the probability of class $i$ versus class $j$ as:

$$p_{i,j}(\boldsymbol{x}_n) = \frac{e^{f_i(\boldsymbol{x}_n)}}{e^{f_i(\boldsymbol{x}_n)}+e^{f_j(\boldsymbol{x}_n)}}. \quad (10)$$

The loss of each binary problem aims to maximize $p_{i,j}(\boldsymbol{x}_n)$ for $O_i$ and $p_{j,i}(\boldsymbol{x}_n)$ for $O_j$. The total OVO loss is calculated as the negative log-likelihood averaged across all class pairs:

$$-\sum_{i,j:i\neq j}\frac{1}{|O_i|+|O_j|}\left(\sum_{n\in O_i}\log p_{i,j}(\boldsymbol{x}_n)+\sum_{n\in O_j}\log p_{j,i}(\boldsymbol{x}_n)\right). \quad (11)$$

Similar to the relation between (8) and (9), we derive in Appendix A-B that (11) is:

$$-2\sum_{n=1}^{N}\sum_{i=1}^{k}\mathbf{1}_{\{y_n=i\}}\sum_{j:j\neq i}\frac{1}{|O_i|+|O_j|}\log p_{i,j}(\boldsymbol{x}_n). \quad (12)$$

From (9) and (12), we can see the OVR and OVO concepts at the instance level. OVR focuses on whether each instance belongs to class $i$, while OVO performs $k-1$ comparisons per instance by comparing class $i$ to each other class $j$. Moreover, we note a critical difference. In (8) and (11), the loss is averaged over $N$ instances in OVR and $|O_i|+|O_j|$ instances in OVO for each binary problem. When class $i$ is a minority class, the fixed OVR scaling term $1/N$ averages the loss over $N-|O_i| \approx N$ negatives, so the contribution of the $|O_i|$ instances in class $i$ is limited. In contrast, the OVO scaling term

$$\frac{1}{|O_i|+|O_j|} \quad (13)$$

adapts to the number of instances in each pair. Since the OVO loss includes only $|O_j|$ negatives, class $i$ becomes more prominent in its associated binary task. Therefore, (13) naturally acts as a weight in (12) and helps account for class imbalance.

Our proposed loss functions provide a useful way to compare OVR and OVO. The reason is that we fix everything such as the network architecture and the setting in (1) for prediction, except the loss function, in order to reflect the OVR and OVO concepts. We can ensure that any observed performance differences are solely due to the choice of the loss function.

## III. Experimental Setup

This section describes the evaluation metrics, datasets, model settings, test set construction for evaluation, and hyperparameter search for assessing OVR and OVO under class imbalance.

### A. Evaluation Metrics

Although accuracy is a common metric, it reflects only overall correctness and is often dominated by the majority class. Therefore, accuracy may not be suitable in imbalanced scenarios. Since [9] and [10] evaluate only accuracy, we study additional metrics, including Macro-F1, G-mean, and Balance Accuracy (BA). These metrics are computed based on Precision and Recall, which are defined for each class $i$ as:

$$\text{Precision}_i = \frac{\text{TP}_i}{\text{TP}_i+\text{FP}_i}, \quad (14)$$

$$\text{Recall}_i = \frac{\text{TP}_i}{\text{TP}_i+\text{FN}_i}, \quad (15)$$

where $\text{TP}_i$, $\text{FP}_i$, $\text{FN}_i$, and $\text{TN}_i$ denote the True Positives, False Positives, False Negatives, and True Negatives for class $i$, respectively.

The F1 score, defined as the harmonic mean of Precision and Recall for each class, is given by:

$$\text{F1}_i = 2\cdot\frac{\text{Precision}_i\cdot\text{Recall}_i}{\text{Precision}_i+\text{Recall}_i}, \quad (16)$$

and the Macro-F1 score[2] is defined as:

$$\text{Macro-F1} = \frac{1}{k}\sum_{i=1}^{k}\text{F1}_i. \quad (17)$$

Both G-mean and BA use Recall and are defined as:

$$\text{G-mean} = \left(\prod_{i=1}^{k}\text{Recall}_i\right)^{\frac{1}{k}}, \quad (18)$$

[2]A related metric, Micro-F1, is the harmonic mean of Precision and Recall calculated without class distinction. In multi-class settings, it is equivalent to accuracy.

TABLE I: Data statistics. A higher IR (Imbalance Ratio) indicates that a dataset is more imbalanced.

| ID | Dataset | #Instances | #Features | #Classes | IR |
|---|---|---|---|---|---|
| AUT | Automobile | 159 | 25 | 6 | 16.00 |
| BAL | Balance | 625 | 4 | 3 | 5.88 |
| CAR | Car | 1,728 | 6 | 4 | 18.62 |
| CLE | Cleveland | 297 | 13 | 5 | 12.31 |
| DER | Dermatology | 358 | 34 | 6 | 5.55 |
| ECO | Ecoli | 336 | 7 | 8 | 71.50 |
| FLA | Flare | 1,066 | 11 | 6 | 7.70 |
| GLA | Glass | 214 | 9 | 6 | 8.44 |
| HAY | Hayes-roth | 160 | 4 | 3 | 2.10 |
| HCV | Hcv | 589 | 12 | 5 | 75.14 |
| LYM | Lymphography | 148 | 18 | 4 | 40.50 |
| NEW | New-thyroid | 215 | 5 | 3 | 5.00 |
| SEG | Segment | 2,310 | 19 | 7 | 1.00 |
| SHU | Shuttle | 2,175 | 9 | 5 | 853.00 |
| THY | Thyroid | 720 | 21 | 3 | 39.18 |
| VEH | Vehicle | 846 | 18 | 4 | 1.10 |
| ZOO | Zoo | 101 | 16 | 7 | 10.25 |

$$\text{Balance Accuracy} = \frac{1}{k}\sum_{i=1}^{k}\text{Recall}_i. \tag{19}$$

From (17)–(19), Macro-F1, G-mean, and BA evaluate each class individually before producing their respective metrics. This design may reduce biases from majority classes, so these metrics are widely used for imbalanced datasets.

### B. Datasets

We selected 17 datasets from the UCI Machine Learning Repository [16], the LIBSVM repository,[3] and the KEEL Dataset repository [17]. We summarize the data statistics in Table I. In addition to basic dataset properties, we report the imbalance ratio (IR), defined as

$$\text{IR} = \frac{\max_{i=1,\dots,k}|O_i|}{\min_{i=1,\dots,k}|O_i|}, \tag{20}$$

which is the number of instances in the largest class divided by the number in the smallest. Notably, most of the selected datasets exhibit a high IR, indicating a significant class imbalance.

Since KEEL provides stratified 5-fold splits, we directly adopt them for all selected KEEL datasets. For the UCI dataset HCV and the LIBSVM datasets SEG and VEH, where predefined splits are unavailable, we applied the same stratification method to generate 5-fold splits. This stratification setting ensures that each fold maintains the original class distribution by evenly distributing the instances of each class. We linearly scaled each numerical feature to [0,1] and applied one-hot encoding for categorical features.

### C. Model Settings

For kernel SVM, we implemented OVR and OVO using LIBSVM [15]. In the OVR setting, the model is optimized with the Hinge loss defined as

$$\xi(\boldsymbol{w}, b; y_i(\boldsymbol{x}_n), \boldsymbol{x}_n) = \max(0, 1 - y_i(\boldsymbol{x}_n)(\boldsymbol{w}^T\phi(\boldsymbol{x}_n) + b)), \tag{21}$$

[3]https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets

where the nonlinear mapping function $\phi$ is defined by the RBF kernel:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j) = e^{-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}, \tag{22}$$

where $\gamma$ is the kernel parameter. In the OVO setting, we replace $y_i(\boldsymbol{x}_n)$ in (21) with $y_{i,j}(\boldsymbol{x}_n)$ to represent the binary relationship between classes $i$ and $j$.

For the neural network, we constructed a model with three linear transformation layers. The first maps the input features to 128 hidden units, and the second maps 128 hidden units to 64. Both layers are sequentially followed by LayerNorm, ReLU activation, and Dropout (rate = 0.1). The last layer outputs a $k$-dimensional vector, where each element $f_1, f_2, \dots, f_k$ corresponds to the output of a class. In all experiments, we use a batch size of 8 and optimize the loss with Adam [18] optimizer. Early stopping is applied if validation performance does not improve for 10 consecutive epochs, with training limited to a maximum of 40 epochs.

### D. Test Set Construction for Evaluation

For a reliable evaluation, we ensure that the test set remains independent of hyperparameter search and model training, and we report only the test results. For each dataset, we sequentially hold out one fold from the 5-fold splits as the test set, while the remaining four folds are used for hyperparameter search and model training. The trained model then makes predictions on the test set. After obtaining test predictions for each fold, we combine all test predictions and evaluate the performance using Accuracy, Macro-F1, G-Mean, and BA.

### E. Hyperparameters Search

To search for the optimal hyperparameter using four folds, we further divide these four folds into five subsets. In each evaluation for a given hyperparameter setting, we train the model on four subsets and evaluate the validation performance on the remaining subset.

For kernel SVM, we perform this evaluation on each of the five subsets used for hyperparameter search and compute the overall validation performance. We search over the kernel parameter $\gamma = [2^4, 2^3, \dots, 2^{-10}]$ and the regularization parameter $C = [2^{12}, 2^{11}, \dots, 2^{-2}]$ to identify the best hyperparameter setting. The setting that achieves the best validation performance is then selected to retrain the model.

For neural networks, we conduct hyperparameter search over optimizer learning rates $\alpha = [10^{-5}, 5\times10^{-5}, 10^{-4}, 5\times10^{-4}, 10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}]$. Unlike kernel SVM, we evaluate only a single subset instead of all five to reduce the computational cost. The learning rate that achieves the best validation performance is then selected to retrain the model.

Since optimizing a different criterion in hyperparameter search can affect the test performance, we conduct two separate searches: one optimized for validation accuracy and the other for validation Macro-F1. The former does not account for class imbalance, while the latter does.

TABLE II: Average test results of kernel SVM optimized with the chosen criterion (Accuracy or Macro-F1) in hyperparameter search. OVO columns report the growth over OVR, defined as (OVO/OVR- 1) $\times$ 100%.

| Hyperparameter Search Criterion | Test Performance | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Accuracy | | Macro-F1 | | G-mean | | BA | |
| | OVR | OVO | OVR | OVO | OVR | OVO | OVR | OVO |
| Accuracy | 87.36 | +0.7% | 76.13 | +2.8% | 60.74 | +9.4% | 74.87 | +3.9% |
| Macro-F1 | 87.28 | +0.4% | 77.35 | +3.0% | 65.95 | +10.5% | 76.20 | +3.6% |

TABLE III: Confusion matrix of the SHU dataset using kernel SVM, with hyperparameters optimized for Macro-F1. We also report F1 scores for each class.

| | | Target | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | Class | 1 | 2 | 3 | 4 | 5 | F1 |
| Predict | OVO | 1 | 1,704 | 0 | 1 | 0 | 2 | 99.85 |
| | | 2 | 0 | 2 | 0 | 0 | 0 | 100.0 |
| | | 3 | 1 | 0 | 4 | 0 | 0 | 72.73 |
| | | 4 | 0 | 0 | 1 | 338 | 0 | 99.85 |
| | | 5 | 1 | 0 | 0 | 0 | 121 | 98.78 |
| | OVR | 1 | 1,701 | 0 | 1 | 0 | 4 | 99.71 |
| | | 2 | 0 | 0 | 0 | 0 | 0 | 0.0 |
| | | 3 | 0 | 0 | 2 | 0 | 0 | 50.0 |
| | | 4 | 5 | 1 | 2 | 338 | 0 | 98.83 |
| | | 5 | 0 | 1 | 1 | 0 | 119 | 97.54 |
| | | Total | 1,706 | 2 | 6 | 338 | 123 | |

TABLE IV: Average test results of neural networks optimized with the chosen criterion (Accuracy or Macro-F1) in hyperparameter search. OVO columns report the growth over OVR, defined as (OVO/OVR- 1) $\times$ 100%.

| Hyperparameter Search Criterion | Test Performance | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Accuracy | | Macro-F1 | | G-mean | | BA | |
| | OVR | OVO | OVR | OVO | OVR | OVO | OVR | OVO |
| Accuracy | 85.16 | -0.04% | 71.74 | +2.8% | 53.16 | +11.6% | 72.13 | +1.4% |
| Macro-F1 | 85.34 | +0.0% | 73.41 | +1.7% | 53.78 | +13.4% | 72.34 | +2.6% |

TABLE V: Confusion matrix of the SHU dataset using neural networks, with hyperparameters optimized for Macro-F1. We also report F1 scores for each class.

| | | Target | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | Class | 1 | 2 | 3 | 4 | 5 | F1 |
| Predict | OVO | 1 | 1,688 | 0 | 1 | 5 | 1 | 99.26 |
| | | 2 | 0 | 0 | 0 | 0 | 0 | 0.0 |
| | | 3 | 0 | 0 | 2 | 0 | 0 | 50.0 |
| | | 4 | 18 | 0 | 2 | 333 | 0 | 96.38 |
| | | 5 | 0 | 2 | 1 | 0 | 122 | 98.39 |
| | OVR | 1 | 1,682 | 0 | 2 | 17 | 1 | 98.71 |
| | | 2 | 0 | 0 | 0 | 0 | 0 | 0.0 |
| | | 3 | 0 | 0 | 1 | 0 | 0 | 28.57 |
| | | 4 | 23 | 0 | 2 | 321 | 0 | 93.86 |
| | | 5 | 1 | 2 | 1 | 0 | 122 | 97.99 |
| | | Total | 1,706 | 2 | 6 | 338 | 123 | |

## IV. Experiments and Analysis

Building on the experimental setup described above, this section presents a comprehensive comparison of OVR and OVO across kernel SVM and neural networks, with a particular focus on how class imbalance affects their relative performance.

### A. Results of Kernel SVM

Table II summarizes the average test performance of OVR and OVO, with results for each dataset provided in Appendix B-A. As shown in Table II, both methods achieve comparable accuracy. This observation aligns with the findings of [9], a highly cited study that compares OVR and OVO. However, our results further indicate that OVO consistently outperforms OVR in Macro-F1, G-mean, and BA.

The SHU dataset, which has the highest imbalance ratio among the studied datasets, vividly illustrates this disparity. The confusion matrix in Table III reveals that for the minority classes 2 and 3, OVO achieves higher F1-scores, while OVR misclassifies most of them as other classes.

### B. Results of Neural Networks

We report the test results using OVR and OVO losses in Table IV. For the CE loss, the results are similar to OVR, and we provide them in Appendix B-B. While both methods achieve comparable accuracy, OVO continues to outperform OVR in Macro-F1, G-mean, and BA, as observed in kernel SVM. Note that we use the same network architecture, so the only difference lies in the loss functions designed to reflect OVR and OVO. Therefore, our results clearly show that OVO handles class imbalance more effectively.

Similar to kernel SVM in Table III, we also check the confusion matrix in Table V. While neither OVR nor OVO successfully classifies any instance in minority class 2, OVO achieves a higher F1-score for the minority class 3. Moreover, OVR tends to favor the majority class by misclassifying more instances from other classes as the most frequent class 1.

### C. Results Across Different Imbalance Levels

In Tables II and IV, we compared OVR and OVO based on their average performance across all datasets and observed notable differences in metrics that account for class imbalance. Since the datasets have varying levels of imbalance, as indicated by their imbalance ratio (IR), we further examine whether the performance gap between OVR and OVO becomes more pronounced as IR increases.

We follow [19] to categorize our collected datasets into three groups based on their IRs. Specifically, datasets with IR $< 3$ are classified as low imbalance, those with $3 \leq$ IR $< 9$ as medium imbalance, and those with IR $\geq 9$ as high imbalance. Accordingly, we group the datasets as follows:

- low imbalance: HAY, SEG, VEH
- medium imbalance: BAL, DER, FLA, GLA, NEW
- high imbalance: AUT, CAR, CLE, ECO, HCV, LYM, SHU, THY, ZOO

For kernel SVM, Table VI reports the test results for each group, using hyperparameters optimized for Accuracy and Macro-F1. Interestingly, accuracy happens to be similar

TABLE VI: Average test results of kernel SVM under different imbalance groups. OVO columns report the growth over OVR, defined as (OVO/OVR- 1) $\times$ 100%

(a) Results using hyperparameters optimized for Accuracy.

| Imbalance Group | Test Performance | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Accuracy | | Macro-F1 | | G-mean | | BA | |
| | OVR | OVO | OVR | OVO | OVR | OVO | OVR | OVO |
| Low | 87.74 | -0.13% | 88.56 | -0.12% | 87.97 | -0.15% | 88.59 | -0.16% |
| Medium | 87.45 | +0.54% | 82.23 | +2.02% | 68.38 | +15.87% | 81.14 | +3.11% |
| High | 87.17 | +1.17% | 68.6 | +4.66% | 47.41 | +10.19% | 66.82 | +6.23% |

(b) Results using hyperparameters optimized for Macro-F1.

| Imbalance Group | Test Performance | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Accuracy | | Macro-F1 | | G-mean | | BA | |
| | OVR | OVO | OVR | OVO | OVR | OVO | OVR | OVO |
| Low | 88.16 | -0.91% | 88.78 | -0.68% | 88.11 | -0.66% | 88.75 | -0.65% |
| Medium | 87.55 | +0.57% | 83.39 | +1.89% | 79.53 | +3.43% | 82.22 | +2.85% |
| High | 86.83 | +0.81% | 70.18 | +5.29% | 51.02 | +23.03% | 68.67 | +5.96% |

TABLE VII: Average test results of neural networks under different imbalance groups. OVO columns report the growth over OVR, defined as (OVO/OVR- 1) $\times$ 100%

(a) Results using hyperparameters optimized for Accuracy.

| Imbalance Group | Test Performance | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Accuracy | | Macro-F1 | | G-mean | | BA | |
| | OVR | OVO | OVR | OVO | OVR | OVO | OVR | OVO |
| Low | 84.53 | +0.14% | 85.13 | +0.79% | 84.86 | +0.13% | 85.64 | -0.07% |
| Medium | 86.11 | +0.00% | 80.15 | +0.62% | 66.69 | -2.10% | 80.42 | +0.29% |
| High | 84.85 | -0.13% | 62.6 | +5.19% | 35.08 | +35.26% | 63.02 | +2.82% |

(b) Results using hyperparameters optimized for Macro-F1.

| Imbalance Group | Test Performance | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Accuracy | | Macro-F1 | | G-mean | | BA | |
| | OVR | OVO | OVR | OVO | OVR | OVO | OVR | OVO |
| Low | 84.25 | +0.89% | 84.98 | +0.91% | 84.6 | +0.50% | 85.14 | +0.61% |
| Medium | 85.46 | -1.16% | 78.95 | +0.46% | 63.77 | +3.37% | 78.3 | +0.83% |
| High | 85.63 | +0.35% | 66.48 | +2.95% | 37.96 | +32.38% | 64.76 | +4.59% |

across different imbalance groups, but Macro-F1, G-Mean, and BA decrease as imbalance increases. Moreover, for these metrics, the performance gap between OVR and OVO in the medium and high imbalance groups is much more pronounced than in the low imbalance group. When hyperparameters are optimized for Macro-F1, this gap widens even further in the high imbalance group compared to the medium imbalance group.

For neural networks, we provide the test results for each group in Table VII under two different hyperparameter search criteria. Similar to the trends in Table VI, Macro-F1, G-Mean, and BA decline more sharply than accuracy as the group becomes more imbalanced. Although OVR and OVO perform similarly in the low and medium imbalance groups, the performance gap between OVR and OVO is substantially larger in the high imbalance group.

These findings across both kernel SVM and neural networks further underscore the influence of imbalance severity on the differences between OVR and OVO.

## V. Conclusions

This study re-examines OVR and OVO, which have traditionally been considered comparable in accuracy within kernel SVM, with differences primarily attributed to model size and training time. However, our findings in both kernel SVM and neural networks reveal that when metrics accounting for class imbalance are considered, OVO surpasses OVR in preserving the performance of minority classes. This advantage demonstrates OVO's greater robustness to class imbalance. For neural networks, by novelly devising the loss functions to reflect OVR and OVO, we facilitate rigorous comparisons while keeping other components fixed. These contributions offer insights for selecting OVR and OVO decomposition methods in multi-class classification problems.

## VI. Acknowledgments

This work was supported in part by National Science and Technology Council of Taiwan grant NSTC-113-2222-E-002-005-MY3, and in part by the Featured Area Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (114L900901). The authors thank the reviewers for their insightful and constructive comments.

## Appendix A
## Proofs

### A. Neural Networks Loss Function for OVR

For OVR, we recall that $p_i(\boldsymbol{x}_n)$ is defined in (7). The loss in the binary problem for each class $i$ aims to maximize $p_i(\boldsymbol{x}_n)$ for $O_i$ and $1 - p_i(\boldsymbol{x}_n)$ for the others as:

$$\xi_i = -\frac{1}{|O_i| + (N - |O_i|)} \cdot \left( \sum_{n \in O_i} \log p_i(\boldsymbol{x}_n) + \sum_{n \notin O_i} \log(1 - p_i(\boldsymbol{x}_n)) \right), \tag{23}$$

where

$$|O_i| + (N - |O_i|) = N \tag{24}$$

represents the total number of instances, including those in $O_i$ (positive class) and those outside $O_i$ (negative class). This normalization ensures that the loss is averaged over all instances.

The total OVR loss across all $k$ classes is:

$$L_{\text{OVR}} = \sum_{i=1}^{k} \xi_i. \tag{25}$$

By pluging (23) into (25), we have:

$$
\begin{aligned}
&L_{\text{OVR}} \\
&= -\sum_{i=1}^{k} \frac{1}{N} \left( \sum_{n \in O_i} \log p_i(\boldsymbol{x}_n) + \sum_{n \notin O_i} \log(1 - p_i(\boldsymbol{x}_n)) \right) \\
&= -\sum_{i=1}^{k} \frac{1}{N} \left( \sum_{n=1}^{N} \mathbf{1}_{\{n \in O_i\}} \log p_i(\boldsymbol{x}_n) \right. \\
&\left. + \sum_{n=1}^{N} (1 - \mathbf{1}_{\{n \in O_i\}}) \log(1 - p_i(\boldsymbol{x}_n)) \right) \\
&= -\frac{1}{N} \sum_{i=1}^{k} \left( \sum_{n=1}^{N} \mathbf{1}_{\{n \in O_i\}} \log p_i(\boldsymbol{x}_n) \right. \\
&\left. + \sum_{n=1}^{N} (1 - \mathbf{1}_{\{n \in O_i\}}) \log(1 - p_i(\boldsymbol{x}_n)) \right) \\
&= -\frac{1}{N} \sum_{n=1}^{N} \left( \sum_{i=1}^{k} \mathbf{1}_{\{n \in O_i\}} \log p_i(\boldsymbol{x}_n) \right. \\
&\left. + \sum_{i=1}^{k} (1 - \mathbf{1}_{\{n \in O_i\}}) \log(1 - p_i(\boldsymbol{x}_n)) \right) \\
&= -\frac{1}{N} \sum_{n=1}^{N} \left( \sum_{i=1}^{k} \mathbf{1}_{\{y_n = i\}} \log p_i(\boldsymbol{x}_n) \right. \\
&\left. + \sum_{i=1}^{k} (1 - \mathbf{1}_{\{y_n = i\}}) \log(1 - p_i(\boldsymbol{x}_n)) \right) \\
&= -\frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{k} \left( \mathbf{1}_{\{y_n = i\}} \log p_i(\boldsymbol{x}_n) \right. \\
&\left. + (1 - \mathbf{1}_{\{y_n = i\}}) \log(1 - p_i(\boldsymbol{x}_n)) \right). \qquad (26)
\end{aligned}
$$

When comparing (6) with (26), we can notice that the first term of (26) is identical to (6), with only a difference in the second term, as each binary comparison introduces the negative instances.

### B. Neural Networks Loss Function for OVO

For OVO, we recall that $p_{i,j}(\boldsymbol{x}_n)$ is defined in (10). For each class pair $(i, j)$, we aim to maximize $p_{i,j}(\boldsymbol{x}_n)$ for $O_i$ and $p_{j,i}(\boldsymbol{x}_n)$ for $O_j$ as:

$$
\xi_{i,j} = -\frac{1}{|O_i| + |O_j|} \cdot \left( \sum_{n \in O_i} \log p_{i,j}(\boldsymbol{x}_n) + \sum_{n \in O_j} \log p_{j,i}(\boldsymbol{x}_n) \right). \qquad (27)
$$

The total OVO loss over all class pairs is:

$$
L_{\text{OVO}} = \sum_{i,j:i \neq j} \xi_{i,j}. \qquad (28)
$$

By pluging (27) into (28), we have:

$$
\begin{aligned}
L_{\text{OVO}} &= -\sum_{i,j:i \neq j} \frac{1}{|O_i| + |O_j|} \left( \sum_{n \in O_i} \log p_{i,j}(\boldsymbol{x}_n) \right. \\
&\left. + \sum_{n \in O_j} \log p_{j,i}(\boldsymbol{x}_n) \right) \\
&= -2 \sum_{i,j:i \neq j} \frac{1}{|O_i| + |O_j|} \sum_{n \in O_i} \log p_{i,j}(\boldsymbol{x}_n) \\
&= -2 \sum_{i=1}^{k} \sum_{j=1}^{k} \mathbf{1}_{\{i \neq j\}} \frac{1}{|O_i| + |O_j|} \sum_{n \in O_i} \log p_{i,j}(\boldsymbol{x}_n) \\
&= -2 \sum_{i=1}^{k} \sum_{j=1}^{k} \sum_{n \in O_i} \mathbf{1}_{\{i \neq j\}} \frac{1}{|O_i| + |O_j|} \log p_{i,j}(\boldsymbol{x}_n) \\
&= -2 \sum_{i=1}^{k} \sum_{n \in O_i} \sum_{j=1}^{k} \mathbf{1}_{\{i \neq j\}} \frac{1}{|O_i| + |O_j|} \log p_{i,j}(\boldsymbol{x}_n) \\
&= -2 \sum_{i=1}^{k} \sum_{n \in O_i} \sum_{j:j \neq i} \frac{1}{|O_i| + |O_j|} \log p_{i,j}(\boldsymbol{x}_n) \\
&= -2 \sum_{i=1}^{k} \sum_{n=1}^{N} \mathbf{1}_{\{n \in O_i\}} \sum_{j:j \neq i} \frac{1}{|O_i| + |O_j|} \log p_{i,j}(\boldsymbol{x}_n) \\
&= -2 \sum_{n=1}^{N} \sum_{i=1}^{k} \mathbf{1}_{\{n \in O_i\}} \sum_{j:j \neq i} \frac{1}{|O_i| + |O_j|} \log p_{i,j}(\boldsymbol{x}_n) \\
&= -2 \sum_{n=1}^{N} \sum_{i=1}^{k} \mathbf{1}_{\{y_n = i\}} \sum_{j:j \neq i} \frac{1}{|O_i| + |O_j|} \log p_{i,j}(\boldsymbol{x}_n).
\end{aligned}
$$

## APPENDIX B
## FULL EXPERIMENTAL RESULTS

### A. Results of Kernel SVM

For kernel SVM, we present the test results for each dataset in Tables VIII and IX, where the former optimizes Accuracy and the latter optimizes Macro-F1 in hyperparameter search.

### B. Results of Neural Networks

For neural networks, we present the test results of OVR and OVO losses in Tables X and XI under two different hyperparameter search criteria. Additionally, we report the test results of CE and OVR losses in Tables XII and XIII. While the formulations of CE and OVR shown in (6) and (9) are similar, their performance depends on the hyperparameter optimization criterion. When hyperparameters are optimized for Accuracy, CE and OVR perform comparably. However, when optimized for Macro-F1, OVR outperforms CE in terms of Macro-F1, G-Mean, and BA. This performance gap may reflect the difference between (6) and (9). Since each binary comparison in OVR considers both positive and negative instances, positive instances from a given class can serve as negative instances for all other classes. As a result, when OVR is used, the minority class still contributes to model updates via negative instances. In contrast, CE considers only positive instances during training. This may result in worse predictive

TABLE VIII: Test results of kernel SVM optimized with Accuracy in hyperparameter search.

| ID | Accuracy | | Macro-F1 | | G-mean | | BA | |
|---|---|---|---|---|---|---|---|---|
| | OVR | OVO | OVR | OVO | OVR | OVO | OVR | OVO |
| AUT | 74.21 | 79.87 | 70.29 | 70.23 | 70.20 | 65.65 | 70.52 | 68.91 |
| BAL | 98.08 | 99.84 | 94.77 | 99.60 | 91.77 | 99.32 | 92.40 | 99.32 |
| CAR | 98.84 | 99.94 | 96.70 | 99.96 | 97.12 | 99.98 | 97.14 | 99.98 |
| CLE | 58.92 | 59.60 | 27.69 | 28.39 | 0.00 | 0.00 | 29.80 | 30.02 |
| DER | 97.77 | 97.21 | 97.57 | 96.93 | 97.25 | 96.58 | 97.36 | 96.74 |
| ECO | 84.52 | 83.33 | 63.15 | 62.20 | 0.00 | 0.00 | 62.31 | 61.61 |
| FLA | 75.52 | 74.30 | 59.39 | 58.49 | 0.00 | 38.32 | 60.35 | 58.91 |
| GLA | 68.69 | 71.03 | 63.32 | 68.21 | 57.70 | 65.77 | 60.32 | 67.15 |
| HAY | 80.00 | 80.62 | 82.41 | 82.98 | 81.35 | 81.99 | 82.41 | 82.92 |
| HCV | 93.89 | 91.68 | 59.11 | 60.51 | 43.56 | 54.11 | 57.53 | 59.67 |
| LYM | 83.78 | 87.16 | 58.85 | 65.15 | 0.00 | 0.00 | 55.11 | 62.70 |
| NEW | 97.21 | 97.21 | 96.09 | 96.23 | 95.19 | 96.14 | 95.27 | 96.16 |
| SEG | 96.93 | 97.06 | 96.93 | 97.06 | 96.87 | 97.00 | 96.93 | 97.06 |
| SHU | 99.63 | 99.72 | 86.17 | 94.24 | 75.29 | 91.89 | 79.35 | 92.98 |
| THY | 94.72 | 95.42 | 67.07 | 73.11 | 56.34 | 66.96 | 61.76 | 70.09 |
| VEH | 86.29 | 85.22 | 86.35 | 85.30 | 85.68 | 84.53 | 86.43 | 85.37 |
| ZOO | 96.04 | 97.03 | 88.41 | 92.37 | 84.20 | 91.57 | 87.86 | 92.86 |
| **Avg.** | **87.36** | **88.01** | **76.13** | **78.29** | **60.74** | **66.46** | **74.87** | **77.79** |

TABLE IX: Test results of kernel SVM optimized with Macro-F1 in hyperparameter search.

| ID | Accuracy | | Macro-F1 | | G-mean | | BA | |
|---|---|---|---|---|---|---|---|---|
| | OVR | OVO | OVR | OVO | OVR | OVO | OVR | OVO |
| AUT | 74.84 | 80.50 | 76.93 | 79.07 | 74.36 | 78.54 | 75.64 | 79.67 |
| BAL | 98.40 | 99.84 | 95.72 | 99.60 | 93.35 | 99.32 | 93.76 | 99.32 |
| CAR | 98.67 | 99.71 | 96.46 | 99.33 | 96.06 | 99.81 | 96.12 | 99.81 |
| CLE | 56.23 | 54.55 | 29.35 | 28.14 | 19.13 | 17.38 | 29.85 | 28.42 |
| DER | 97.77 | 97.21 | 97.57 | 96.93 | 97.25 | 96.58 | 97.36 | 96.74 |
| ECO | 83.93 | 83.63 | 62.78 | 61.58 | 0.00 | 0.00 | 61.83 | 62.17 |
| FLA | 73.83 | 74.02 | 61.34 | 62.35 | 49.34 | 51.39 | 61.09 | 61.65 |
| GLA | 70.56 | 71.96 | 66.56 | 69.76 | 62.50 | 67.87 | 63.64 | 68.94 |
| HAY | 81.25 | 80.62 | 83.07 | 82.98 | 81.79 | 81.99 | 82.90 | 82.92 |
| HCV | 92.70 | 91.68 | 57.39 | 58.94 | 46.19 | 49.91 | 55.38 | 58.04 |
| LYM | 84.46 | 86.49 | 79.63 | 81.19 | 77.14 | 72.54 | 79.60 | 74.38 |
| NEW | 97.21 | 97.21 | 95.74 | 96.23 | 95.19 | 96.14 | 95.27 | 96.16 |
| SEG | 96.93 | 97.06 | 96.93 | 97.06 | 96.87 | 97.00 | 96.93 | 97.06 |
| SHU | 99.31 | 99.72 | 69.22 | 94.24 | 0.00 | 91.89 | 65.96 | 92.98 |
| THY | 95.28 | 95.42 | 71.43 | 73.11 | 62.07 | 66.96 | 65.79 | 70.09 |
| VEH | 86.29 | 84.40 | 86.35 | 84.50 | 85.68 | 83.60 | 86.43 | 84.54 |
| ZOO | 96.04 | 96.04 | 88.41 | 89.43 | 84.20 | 87.89 | 87.86 | 89.29 |
| **Avg.** | **87.28** | **87.65** | **77.35** | **79.67** | **65.95** | **72.87** | **76.20** | **78.95** |

TABLE X: Test results of neural networks optimized with Accuracy in hyperparameter search.

| ID | Accuracy | | Macro-F1 | | G-mean | | BA | |
|---|---|---|---|---|---|---|---|---|
| | OVR | OVO | OVR | OVO | OVR | OVO | OVR | OVO |
| AUT | 68.55 | 64.78 | 56.85 | 61.23 | 0.00 | 60.44 | 56.65 | 62.87 |
| BAL | 95.20 | 95.84 | 89.16 | 90.22 | 89.84 | 89.58 | 90.32 | 90.22 |
| CAR | 99.48 | 99.54 | 98.61 | 99.14 | 98.60 | 99.01 | 98.61 | 99.02 |
| CLE | 58.25 | 58.59 | 31.19 | 36.33 | 18.78 | 27.73 | 32.90 | 37.41 |
| DER | 97.49 | 97.49 | 97.18 | 97.30 | 97.06 | 97.29 | 97.15 | 97.35 |
| ECO | 77.98 | 80.36 | 54.38 | 57.41 | 0.00 | 0.00 | 54.40 | 57.82 |
| FLA | 75.23 | 74.58 | 62.40 | 60.49 | 50.64 | 45.35 | 62.05 | 60.30 |
| GLA | 65.42 | 65.42 | 55.92 | 59.24 | 0.00 | 0.00 | 56.60 | 61.00 |
| HAY | 75.62 | 75.62 | 77.54 | 79.14 | 77.96 | 77.69 | 78.75 | 78.18 |
| HCV | 92.87 | 89.30 | 56.50 | 49.20 | 49.93 | 36.36 | 57.98 | 46.83 |
| LYM | 78.38 | 83.78 | 47.82 | 79.78 | 0.00 | 71.43 | 52.44 | 73.04 |
| NEW | 97.21 | 97.21 | 96.11 | 96.01 | 95.90 | 94.21 | 96.00 | 94.38 |
| SEG | 95.58 | 96.54 | 95.61 | 96.51 | 95.47 | 96.43 | 95.58 | 96.54 |
| SHU | 97.79 | 98.48 | 58.15 | 64.47 | 0.00 | 0.00 | 58.53 | 62.59 |
| THY | 94.31 | 91.81 | 70.05 | 55.62 | 62.01 | 44.20 | 65.65 | 54.32 |
| VEH | 82.39 | 81.80 | 82.25 | 81.76 | 81.14 | 80.80 | 82.59 | 82.01 |
| ZOO | 96.04 | 96.04 | 89.82 | 89.43 | 86.42 | 87.89 | 90.00 | 89.29 |
| **Avg.** | **85.16** | **85.13** | **71.74** | **73.72** | **53.16** | **59.32** | **72.13** | **73.13** |

TABLE XI: Test results neural networks optimized with Macro-F1 in hyperparameter search.

| ID | Accuracy | | Macro-F1 | | G-mean | | BA | |
|---|---|---|---|---|---|---|---|---|
| | OVR | OVO | OVR | OVO | OVR | OVO | OVR | OVO |
| AUT | 71.70 | 69.81 | 59.39 | 63.77 | 0.00 | 60.64 | 59.63 | 62.83 |
| BAL | 94.40 | 95.52 | 85.61 | 88.65 | 80.74 | 86.43 | 83.53 | 87.73 |
| CAR | 98.50 | 99.19 | 95.93 | 98.74 | 93.75 | 99.53 | 93.89 | 99.53 |
| CLE | 55.89 | 57.58 | 29.67 | 31.65 | 0.00 | 21.78 | 30.55 | 32.59 |
| DER | 96.93 | 95.25 | 96.65 | 94.62 | 96.47 | 94.18 | 96.60 | 94.58 |
| ECO | 80.95 | 80.65 | 59.53 | 57.14 | 0.00 | 0.00 | 59.30 | 58.04 |
| FLA | 73.36 | 73.17 | 60.63 | 62.42 | 48.76 | 55.00 | 60.15 | 62.44 |
| GLA | 66.82 | 61.68 | 57.57 | 55.50 | 0.00 | 0.00 | 58.26 | 55.83 |
| HAY | 76.25 | 76.88 | 78.14 | 79.19 | 78.26 | 78.32 | 78.68 | 78.67 |
| HCV | 93.72 | 95.25 | 59.25 | 70.80 | 44.59 | 64.40 | 59.15 | 68.13 |
| LYM | 83.11 | 83.78 | 79.55 | 76.61 | 71.53 | 76.76 | 73.04 | 79.19 |
| NEW | 95.81 | 96.74 | 94.28 | 95.35 | 92.86 | 94.00 | 92.98 | 94.16 |
| SEG | 96.36 | 96.67 | 96.36 | 96.65 | 96.27 | 96.57 | 96.36 | 96.67 |
| SHU | 97.75 | 98.62 | 63.83 | 68.81 | 0.00 | 0.00 | 61.88 | 66.00 |
| THY | 95.00 | 93.47 | 69.80 | 61.08 | 59.81 | 46.23 | 63.98 | 56.84 |
| VEH | 80.14 | 81.44 | 80.44 | 81.40 | 79.27 | 80.17 | 80.38 | 81.64 |
| ZOO | 94.06 | 95.05 | 81.35 | 87.32 | 71.97 | 82.94 | 81.43 | 86.43 |
| **Avg.** | **85.34** | **85.34** | **73.41** | **74.69** | **53.78** | **61.00** | **72.34** | **74.19** |

performance compared to OVR for the minority class under class imbalance.

## REFERENCES


[1] T.-K. Huang, R. C. Weng, and C.-J. Lin, "Generalized Bradley-Terry models and multi-class probability estimates," *Journal of Machine Learning Research*, vol. 7, pp. 85–115, 2006. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/papers/generalBT.pdf

[2] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *Journal of Artificial Intelligence Research*, vol. 2, pp. 263–286, 1995.

[3] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: a unifying approach for margin classifiers," *Journal of Machine Learning Research*, vol. 1, pp. 113–141, 2001.

[4] J. C. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin DAGs for multiclass classification," in *Advances in Neural Information Processing Systems*, vol. 12. MIT Press, 2000, pp. 547–553.

[5] C. Cortes and V. Vapnik, "Support-vector network," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[6] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–232, 1958.

[7] L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L. Jackel, Y. LeCun, U. A. Müller, E. Säckinger, P. Simard, and V. Vapnik, "Comparison of classifier methods: a case study in handwriting digit recognition," in *International Conference on Pattern Recognition*, 1994, pp. 77–87.

[8] S. Knerr, L. Personnaz, and G. Dreyfus, "Single-layer learning revisited: a stepwise procedure for building and training a neural network," in *Neurocomputing: Algorithms, Architectures and Applications*, J. Fogelman, Ed. Springer-Verlag, 1990.

[9] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multi-class support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[10] R. Rifkin and A. Klautau, "In defense of one-vs-all classification," *Journal of Machine Learning Research*, vol. 5, pp. 101–141, 2004.

[11] J. M. Tague, "Information retrieval experiment," in *The pragmatics of information retrieval experimentation*, K. S. Jones, Ed. London: Butterworths, 1981, ch. 5, pp. 59–102.

TABLE XII: Test results using OVR and CE losses in neural networks optimized with Accuracy in hyperparameter search.

| | Accuracy | | Macro-F1 | | G-mean | | BA | |
|---|---|---|---|---|---|---|---|---|
| **ID** | CE | OVR | CE | OVR | CE | OVR | CE | OVR |
| AUT | 69.81 | 68.55 | 57.27 | 56.85 | 0.00 | 0.00 | 55.88 | 56.65 |
| BAL | 95.84 | 95.20 | 89.53 | 89.16 | 85.85 | 89.84 | 87.39 | 90.32 |
| CAR | 99.19 | 99.48 | 98.18 | 98.61 | 97.56 | 98.60 | 97.60 | 98.61 |
| CLE | 55.89 | 58.25 | 28.08 | 31.19 | 0.00 | 18.78 | 29.16 | 32.90 |
| DER | 96.09 | 97.49 | 95.31 | 97.18 | 95.17 | 97.06 | 95.27 | 97.15 |
| ECO | 82.14 | 77.98 | 58.42 | 54.38 | 0.00 | 0.00 | 59.76 | 54.40 |
| FLA | 74.48 | 75.23 | 61.35 | 62.40 | 45.62 | 50.64 | 60.85 | 62.05 |
| GLA | 65.89 | 65.42 | 57.01 | 55.92 | 0.00 | 0.00 | 56.31 | 56.60 |
| HAY | 82.50 | 75.62 | 84.70 | 77.54 | 84.11 | 77.96 | 84.44 | 78.75 |
| HCV | 94.23 | 92.87 | 63.03 | 56.50 | 54.76 | 49.93 | 60.91 | 57.98 |
| LYM | 83.78 | 78.38 | 58.99 | 47.82 | 0.00 | 0.00 | 61.05 | 52.44 |
| NEW | 94.42 | 97.21 | 92.11 | 96.11 | 89.77 | 95.90 | 89.97 | 96.00 |
| SEG | 96.49 | 95.58 | 96.48 | 95.61 | 96.40 | 95.47 | 96.49 | 95.58 |
| SHU | 98.39 | 97.79 | 64.38 | 58.15 | 0.00 | 0.00 | 62.62 | 58.53 |
| THY | 93.75 | 94.31 | 64.91 | 70.05 | 55.58 | 62.01 | 63.31 | 65.65 |
| VEH | 80.14 | 82.39 | 79.86 | 82.25 | 78.44 | 81.14 | 80.36 | 82.59 |
| ZOO | 96.04 | 96.04 | 89.82 | 89.82 | 86.42 | 86.42 | 90.00 | 90.00 |
| **Avg.** | **85.83** | **85.16** | **72.91** | **71.74** | **51.16** | **53.16** | **72.43** | **72.13** |

TABLE XIII: Test results using OVR and CE losses in neural networks optimized with Macro-F1 in hyperparameter search.

| | Accuracy | | Macro-F1 | | G-mean | | BA | |
|---|---|---|---|---|---|---|---|---|
| **ID** | CE | OVR | CE | OVR | CE | OVR | CE | OVR |
| AUT | 72.33 | 71.70 | 59.43 | 59.39 | 0.00 | 0.00 | 59.51 | 59.63 |
| BAL | 96.16 | 94.40 | 91.57 | 85.61 | 93.11 | 80.74 | 93.27 | 83.53 |
| CAR | 98.96 | 98.50 | 97.56 | 95.93 | 97.97 | 93.75 | 97.99 | 93.89 |
| CLE | 57.24 | 55.89 | 28.21 | 29.67 | 0.00 | 0.00 | 31.56 | 30.55 |
| DER | 97.49 | 96.93 | 97.28 | 96.65 | 97.21 | 96.47 | 97.29 | 96.60 |
| ECO | 80.36 | 80.95 | 58.67 | 59.53 | 0.00 | 0.00 | 59.13 | 59.30 |
| FLA | 73.17 | 73.36 | 60.70 | 60.63 | 47.14 | 48.76 | 60.50 | 60.15 |
| GLA | 64.95 | 66.82 | 55.43 | 57.57 | 0.00 | 0.00 | 55.85 | 58.26 |
| HAY | 78.12 | 76.25 | 79.70 | 78.14 | 79.91 | 78.26 | 80.23 | 78.68 |
| HCV | 92.70 | 93.72 | 56.03 | 59.25 | 42.72 | 44.59 | 57.10 | 59.15 |
| LYM | 81.08 | 83.11 | 55.21 | 79.55 | 0.00 | 71.53 | 53.88 | 73.04 |
| NEW | 93.95 | 95.81 | 91.45 | 94.28 | 88.11 | 92.86 | 88.44 | 92.98 |
| SEG | 96.97 | 96.36 | 96.99 | 96.36 | 96.92 | 96.27 | 96.97 | 96.36 |
| SHU | 98.76 | 97.75 | 58.89 | 63.83 | 0.00 | 0.00 | 59.33 | 61.88 |
| THY | 94.44 | 95.00 | 65.44 | 69.80 | 52.34 | 59.81 | 58.47 | 63.98 |
| VEH | 82.98 | 80.14 | 82.79 | 80.44 | 81.63 | 79.27 | 83.17 | 80.38 |
| ZOO | 95.05 | 94.06 | 87.04 | 81.35 | 82.94 | 71.97 | 86.43 | 81.43 |
| **Avg.** | **85.57** | **85.34** | **71.91** | **73.41** | **50.59** | **53.78** | **71.71** | **72.34** |


[12] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, "The balanced accuracy and its posterior distribution," in *Proceedings of the 20th International Conference on Pattern Recognition*, 2010, pp. 3121–3124.

[13] E. R. Fernandes and A. C. de Carvalho, "Evolutionary inversion of class distribution in overlapping areas for multi-class imbalanced learning," *Information Sciences*, vol. 494, pp. 141–154, 2019.

[14] J. H. Friedman, "Another approach to polychotomous classification," Department of Statistics, Stanford University, Tech. Rep., 1996. [Online]. Available: http://www-stat.stanford.edu/~jhf/ftp/poly.pdf

[15] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[16] A. Asuncion and D. J. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[17] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, and S. García, "KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 17, no. 2-3, pp. 255–287, 2011.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of International Conference on Learning Representations (ICLR)*, 2015.

[19] A. Fernández, S. García, M. J. del Jesus, and F. Herrera, "A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets," *Fuzzy Sets and Systems*, vol. 159, no. 18, pp. 2378–2398, 2008.