Enhancement Encoding: A New Imbalanced Classification Approach via Encoding the Labels

Jia-Chen Zhao
School of Information Science and Engineering
Shandong University
jc_zhao@mail.sdu.edu.cn

Abstract

Class imbalance, which is also called long-tailed distribution, is a common problem in classification tasks based on machine learning. If it happens, the minority data will be overwhelmed by the majority, which presents quite a challenge for data science. To address the class imbalance problem, researchers have proposed lots of methods: some people make the data set balanced (SMOTE [6]), some others refine the loss function (Focal Loss [33]), and even someone has noticed the value of labels influences class-imbalanced learning (Yang and Xu. Rethinking the value of labels for improving class-imbalanced learning. In NeurIPS 2020 [48]), but no one changes the way to encode the labels of data yet. Nowadays, the most prevailing technique to encode labels is the one-hot encoding due to its nice performance in the general situation. However, it is not a good choice for imbalanced data, because the classifier will treat majority and minority samples equally. In this paper, we innovatively propose the enhancement encoding technique, which is specially designed for the imbalanced classification. The enhancement encoding combines re-weighting and cost-sensitiveness, which can reflect the difference between hard and easy (or minority and majority) classes. In order to reduce the number of validation samples and the computation cost, we also replace the confusion matrix with the novel soft-confusion matrix which works better with a small validation set. In the experiments, we evaluate the enhancement encoding with three different types of loss. And the results show that enhancement encoding is very effective to improve the performance of the network trained with imbalanced data. Particularly, the performance on minority classes is much better.

1 Introduction

1.1 Class Imbalance Problem

Today, it is widely accepted that the quantity and quality of training data are of great importance for machine learning (ML) [10, 30, 41, 42]. A balanced data set helps a lot for ML, but in quite a few cases, researchers are obliged to process class-imbalanced samples [20, 45]. Class imbalance describes the phenomenon that data of different classes is not uniformly distributed, i.e. instances in some classes are plentiful but others are lacking [5, 17]. When learning from a class-imbalanced training set, the general classifiers try to minimize loss or maximize accuracy on the imbalanced data. It may make the learner overfit the majority classes and underfit the minorities [36, 38]. At inference time, the classifiers will prefer that an unknown instance belongs to the majorities [2, 5], which declines the test accuracy of minority data.

1.2 One-Hot Encoding

One-hot encoding is one of the most prevailing techniques to encode labels or features now [1, 13, 26, 27]. Compared with directly using the index of class as the label, the one-hot encoding technique expresses the label with a vector instead of a number. Suppose there is an instance belonging to class $p$. Its one-hot label can be described as:

$$l = [l_0, l_1, l_2, \cdots, l_q, \cdots, l_{N-1}], \quad l_q = \begin{cases} 0, & q \neq p \\ 1, & q = p \end{cases}$$

(1)
where $N$ is the total number of classes. Note that the indices of classes start at 0 and end at $N-1$.

There are some significant advantages of one-hot encoding. Firstly, the label of any one class is “equally” different from all the other ones. More specifically, the Euclidean distance between any pair of different labels is always $\sqrt{2}$ and the Hemming distance is always 2. Secondly, one-hot labels work very well with some other fashionable techniques such as Softmax output activation, cross-entropy loss, etc. That’s why one-hot encoding is so popular in classification.

Nevertheless, one-hot encoding is not everything. There are still some problems:

- **Sparsity**: The one-hot labels are too sparse. They are filled with plenty of “0”s and a few “1”s. Too much redundancy of information will eat up memory of the computer.
- **Unfriendly to imbalanced data**: There are always some samples which are more difficult to be classified than others, but one-hot labels can not reflect this difference. This problem is particularly prominent when the data is class-imbalanced.

So, a new encoding technique for labels is demanded.

### 1.3 Our Work

We have studied how to make full use of one-hot labels’ redundancy to improve the performance of imbalanced classification, and in this paper, we innovatively propose the enhancement encoding technique. In contrast to some other methods for addressing the class imbalance problem (such as oversampling, re-weighting, etc.), it is not compulsory to learn about the distribution of training data for enhancement encoding. That means our method will be very effective when the data is not only imbalanced but also from a stream.

In our algorithm, calculating the confusion matrix on the validation set is a necessary step. Meanwhile, enough validation instances per class are significant for an effective confusion matrix. Unfortunately, it conflicts with the lack of minority data. So, we propose the soft-confusion matrix to replace the confusion matrix. Soft-confusion matrix can work pretty well with a small validation set.

If you want to implement the enhancement encoding in your algorithm, you don’t have to change the structure of your neural network or complicate the backpropagation (BP) too much. Re-encoding labels of the training set is all you need. Thanks to the soft-confusion matrix and fewer validation instances, a little bit of extra computation is just enough.

### 2 Proposed Approach

#### 2.1 Soft-Confusion Matrix

The confusion matrix is one of the most prevailing metrics in machine learning [23,46,47,50]. If there are $N$ classes in the classification, the confusion matrix denoted by $R$ is an $N \times N$ matrix. Its element $r_{p,q}$ denotes the number of instances belonging to class $p$ but classified into class $q$. The greater the diagonal elements, the better the classifier performs.

The confusion matrix is pretty handy in classification, but there are still some shortcomings, such as working poorly when the data is insufficient. In order to improve the effect and make the most of the data, we propose the soft-confusion matrix $S$. $S$ is the same size as $R$, and its elements can be described as:

$$s_{p,q} = \sum_{i \in D_p} P(q|x_i)$$  \hspace{1cm} (2)

where $P(q|x_i)$ denotes the prediction probability that instance $i$ belongs to class $q$. $D_p$ is a subset of the data set, which contains all instances belonging to class $p$. That is “$i \in D_p$” means instance $i$ belongs to class $p$. An essential difference between the confusion matrix and
the soft-confusion matrix is that the former counts the number of instances and the latter calculates the sum of probability. And obviously, the confusion matrix (without normalization) only contains integers, and the elements of the soft-confusion matrix can be non-integers.

Certainly, the soft-confusion matrix can be calculated by nested loops, but there is a more efficient method via matrix multiplication:

\[ S = L^T Y \]  

where \( Y \) contains outputs of the classifier and \( L \) contains one-hot encoded labels. More specifically, the element \( y_{i,q} \) of \( Y \) is equal to \( P(q|x_i) \) and the element \( l_{i,p} \) of \( L \) can be “0” (when instance \( i \) does not belong to class \( p \)) or “1” (when instance \( i \) belongs to class \( p \)). That is:

\[ y_{i,q} = P(q|x_i), \quad l_{i,p} = \begin{cases} 0, & i \notin D_p \\ 1, & i \in D_p \end{cases}. \]  

The equivalence of Equation (2) and Equation (3) can be proved very easily:

\[ S = L^T Y \iff s_{p,q} = \sum_i l_{i,p} y_{i,q}, \]  

\[ s_{p,q} = \sum_i l_{i,p} y_{i,q} = \sum_i l_{i,p} P(q|x_i) = \sum_{i \in D_p} P(q|x_i). \]

By the way, the method of Equation (3) can also be used to calculate the general confusion matrix:

\[ y'_{i,q} = \begin{cases} 0, & q \neq \text{arg max}_j y_{i,j} \\ 1, & q = \text{arg max}_j y_{i,j} \end{cases}, \quad R = L^T Y'. \]

2.2 The Basic Idea of Enhancement Encoding

As discussed in Section 1.2, one-hot encoding is not a good choice for imbalanced learning. In an imbalanced classification task, identifying a minority instance is much more difficult than identifying a majority one, so it should be better to let the classifier pay more attention to learning from the minority data.

The enhancement encoding is an improved method adapted from the one-hot encoding. The graphical explanation for its basic idea is shown in Figure 1. Because of circumscribed by the output layer’s activation function (such as Softmax and Sigmoid), the elements of the output vector are all less than “1”, greater than “0” and their sum is “1”:

\[ 0 < y_{i,q} < 1, \quad \sum_{q=0}^{N-1} y_{i,q} = 1. \]

That means points representing outputs are always on the solid line in Figure 1. Meanwhile, enhancement encoding moves the label of minority class from \((1, 0)\) to the location on the dashed line. The process of training can be regarded as making the output and the label closer and closer together. The distance between them is related to the loss. So the new encoding method enhances the part of loss caused by minority samples while training.

2.3 The Implementation of Enhancement Encoding

In linear block code theory, the generator matrix \( G \) is usually utilized to encode the information:

\[ c = mG \]  

where \( m \) is the message, and \( c \) is the channel code. Note that \( m \) and \( c \) in Equation (9) are row vectors. In this way, the labels can be encoded again to implement the enhancement encoding:

\[ l_i^e = l_i G \]

where \( l_i \) denotes the one-hot label of instance \( i \) and \( l_i^e \) denotes the enhancement label. It should be noted that the matrix multiplication in Equation (9) is defined over GF(2), but multiplication in Equation (10) is defined over the real number.
We add some extra steps before a training where the updating rate
where

\[ C \]

Then, calculate the cost matrix
different from the general cost matrix:

into the interval
malization is in order to rescale the value
ning to class
p
in the validation set. The nor-
ization can be
design an algorithm to calculate and update
design an algorithm to calculate and update
the generator matrix dynamically based on
the soft-confusion matrix.

In consideration of these conditions, we
draw a line between the general cost matrix:

We add some extra steps before a training epoch. Firstly, compute the soft-confusion
matrix of the network on the validation set and normalize it. The normalization can be
described as follows:

\[ s'_{p, q} = \frac{s_{p, q}}{n_p} \]  \hspace{1cm} (15)

where \( n_p \) is the number of instances belonging to class \( p \) in the validation set. The normali-
zation is in order to rescale the value into the interval [0, 1] and avoid the trouble of class imbalance in the validation set.

Then, calculate the cost matrix \( C \):

\[ C = S' - I \]  \hspace{1cm} (16)

where \( I \) denotes the identity matrix. The diagonal of \( C \) can be negative, which is different from the general cost matrix:

\[ c_{p, p} = s'_{p, p} - 1 < 0, \]  \hspace{1cm} (17)

\[ c_{p, q} = s'_{p, q} > 0 \quad (p \neq q). \]  \hspace{1cm} (18)

Next, update the generator matrix of labels:

\[ G = (1 - \mu)G^* + \mu(I - \epsilon C) \]  \hspace{1cm} (19)

where the updating rate \( \mu \) and the enhancement rate \( \epsilon \) are two hyper-parameters, \( G^* \)

However, how to get the generator matrix is still a question. There are some properties we wish the generator matrix to possess:

1. To re-weight the classes, the diagonal elements should be greater than “1”.
2. To make the labels cost-sensitive, the non-diagonal elements should be less than “0”.
3. To constrain the labels and the classifier outputs in the same hyperplane, as shown in Figure [1] the sum of elements in the same row should be “1”.

That is:

\[ g_{p, p} > 1, \]  \hspace{1cm} (12)

\[ g_{p, q} < 0 \quad (p \neq q), \]  \hspace{1cm} (13)

\[ \sum_{q=0}^{N-1} g_{p, q} = 1. \]  \hspace{1cm} (14)

\[ e \]: Enhancement rate.

\[ \mu \]: Updating rate for the generator matrix \( G \).

\[ v \]: Batch size.

\[ E \]: Number of training epochs.

\[ \eta \]: Learning rate for the network’s weights \( \theta \).

\[ \theta^* \]: Weights of the network which have been trained.

\[ \eta \]: Learning rate for the network’s weights \( \theta \).

\[ \mu \]: Updating rate for the generator matrix \( G \).

\[ v \]: Batch size.

\[ E \]: Number of training epochs.

\[ b \]: Batch size.

Algorithm 1

The BP algorithm with enhancement encoding

\textbf{Input:}

\( X_{tr} \): Input values of the training set.

\( L_{tr} \): One-hot encoded labels of the training set.

\( X_v \): Input values of the validation set.

\( L_v \): One-hot encoded labels of the validation set.

\( \eta \): Learning rate for the network’s weights \( \theta \).

\( \mu \): Updating rate for the generator matrix \( G \).

\[ b \]: Batch size.

\[ E \]: Number of training epochs.

\textbf{Output:}

\( \theta^* \): Weights of the network which have been trained.

1: Construct the neural network.
2: Initialize the network’s weights \( \theta \).
3: \( G \leftarrow I \) \hspace{1cm} \triangleright \text{Initialize } G \text{ as identity matrix}
4: for \( e_{epoch} = 1 \) to \( E \) do
5: \( Y_{v} \leftarrow \text{forward_pass}(X_v, \theta) \) \hspace{1cm} \triangleright \text{Eq. (3)}
6: \( S \leftarrow \text{soft_confusion_matrix}(L_v, Y_v) \) \hspace{1cm} \triangleright \text{Eq. (4)}
7: \( S' \leftarrow \text{normalize_SCM}(S) \) \hspace{1cm} \triangleright \text{Eq. (15)}
8: \( C \leftarrow \text{cost_matrix}(S') \) \hspace{1cm} \triangleright \text{Eq. (16)}
9: \( G \leftarrow \text{update_generator_matrix}(G, C, \mu, \epsilon) \) \hspace{1cm} \triangleright \text{Eq. (19)}
10: \( L_{tr}^e \leftarrow \text{encode_training_labels}(L_{tr}, G) \) \hspace{1cm} \triangleright \text{Eq. (20)}
11: \( (X_{trb}, L_{trb}) \leftarrow \text{batch_the_training_set}(X_{tr}, L_{tr}^e, b) \)
12: for \( (X_{trb}, L_{trb}) \) in \( (X_{trb}, L_{trb}) \) do
13: \( Y_{trb} \leftarrow \text{forward_pass}(X_{trb}, \theta) \)
14: \( J_{trb}^e \leftarrow \text{loss}(L_{trb}, Y_{trb}) \)
15: \( \nabla \theta J_{trb}^e \leftarrow \text{backward_pass}(J_{trb}^e, \theta, X_{trb}) \) \hspace{1cm} \triangleright \text{Gradient}
16: \( \theta \leftarrow \text{update_the_weights}(\nabla \theta J_{trb}^e, \theta, \eta) \)
17: end for
18: end for
19: \( \theta^* \leftarrow \theta \)
20: return \( \theta^* \)
is the old generator matrix before updating. When updating $G$, the residual value (i.e. $(1 - \mu)G^*$) is to make the training process more stable. It can be proved easily that the properties described by Expression (12), (13), and (14) are attached to the generator matrix $G$ in Equation (19). Lastly, encode the labels of instances in the training set:

$$L^e_{it} = L_{it}G$$

(20)

where $L_{it}$ contains the one-hot labels of training instances. The whole BP algorithm can be briefly described as Algorithm 1.

Note that some details are omitted in Algorithm 1 such as adjusting the learning rate dynamically, L1 or L2 regularization, etc. You can add them by yourself if you need to.

The mathematical derivation of enhancement encoding, which proves its effectiveness, is shown in the Appendices.

3 Experiments

In this section, we do some experiments to evaluate the performance of enhancement encoding and report the results in the end.

3.1 Experimental Setup

In every experiment, we add a Batch Normalization layer [24] before the output activation function (i.e. Softmax), as shown in Figure 2. The Batch Normalization layer is very helpful to improve the performance of the enhancement encoding. When it is applied, the enhancement rate can be set as a higher value.

**MNIST**: MNIST [32] is a fairly basic standard data set, which is usually chosen to be the start for learning computer vision. It contains 7,000 handwriting digit images, 6,000 of which are for training and the rest are for tests. The original data set is balanced. To evaluate our algorithm, we manually unbalance the data in two ways: “10% of even classes” and “long-tailed distribution” (see the Appendices). For the first way (10% of even classes), we refer to the data distribution created by Khan et al. [28]. We choose LeNet-5 [32] as the backbone net and make some alterations. We utilize ReLU [15] as the hidden layers’ activation function, utilize Softmax as the output activation function, and replace the original Gaussian connection layer with a fully connected layer.

**CIFAR-10-LT**: The standard CIFAR-10 [29] is a balanced data set with 10 classes in total. There are 5,000 images for training and 1,000 images for tests per class. We choose its long-tailed version (CIFAR-10-LT) [9] with the imbalance factor of 100 to evaluate our approach. The imbalance factor [9] describes the imbalance degree of the data set. It can be computed by $\max_p n_p / \min_p n_p$, where $n_p$ denotes the number of instances belonging to class $p$. We randomly select 5 instances per class as the validation set. We choose the ResNet-32 [21] as the backbone net.

**Caltech 101**: Caltech 101 [14] is an originally imbalanced data set. It contains 8,677 images belonging to 101 classes (exclude the class named “BACKGROUND_Google”). There are about 30~800 images per class and the imbalance factor of Caltech 101 is 25.81 ($IF = 800/31 = 25.81$). We randomly split the data set into three parts: a validation set containing 5 images per class, a test set containing 20 images per class, and a training set containing the rest of the data. The resolution of these images is roughly 300×200 and we resize them to 224×224. The backbone net is ResNet-34.

We compare the performances of enhancement encoding and one-hot encoding in three different situations (cross-entropy loss, mean square error, and focal loss [33]). Focal loss of instance $i$ can be expressed as:

$$J_i = - \sum_{q=0}^{N-1} \alpha_q l_i,q(1 - y_{i,q})^\gamma \log y_{i,q}$$

(21)

where $\alpha_q$ denotes the class weight and $\gamma$ denotes the focusing parameter. Because the enhancement labels re-weight the instances of different classes, we set all class weights ($\alpha_q$) as “1.0”.

Figure 2: We add a Batch Normalization layer before the Softmax. And then, the enhancement rate can be set as a higher value.
As mentioned in (12) and (13), the enhancement encoding technique attaches both re-weighting and cost-sensitiveness to the training process. So we also evaluate the incomplete enhancement labels by setting the non-diagonal elements of generation matrix $G$ (see Section 2.3) as “0” (marked with “re-weighting”) or setting the diagonal elements as “1” (marked with “cost-sensitiveness”).

### Table 1: Test Top-1 Accuracy on Imbalanced MNIST Data Set (%)

| Imbalance Settings | Losses         | Baseline (One-Hot Encoding) | Re-Weighting | Cost-Sensitiveness | Enhancement Encoding |
|--------------------|----------------|-----------------------------|--------------|--------------------|----------------------|
| 10% of Even Classes| MSE            | 97.82±0.03                  | 97.92        | 97.89              | 97.99±0.06           |
|                    | CE             | 97.75±0.04                  | 97.85        | 97.91              | 98.00±0.03           |
|                    | FL ($\gamma = 2.0, \alpha_q = 1.0$) | 98.06±0.04                  | 98.21        | 98.32              | 98.54±0.13           |
| Long-Tail Distribution | MSE            | 92.12±0.56                  | 92.56        | 92.61              | 92.78±0.26           |
|                    | CE             | 89.14±0.13                  | 90.95        | 90.65              | 91.43±0.27           |
|                    | FL ($\gamma = 2.0, \alpha_q = 1.0$) | 91.56±0.32                  | 91.80        | 91.87              | 93.34±0.61           |

### 3.2 Results

We define the test **minority accuracy** as the top-1 accuracy evaluated on test instances belonging to minority classes. We regard the minority classes as the $N_{\text{min}}$ classes with the fewest instances in the training set. For MNIST, we report the top-1 accuracy in Table 1. For CIFAR-10-LT, we report the top-1 accuracy and the minority accuracy ($N_{\text{min}} = 5$) in Table 2. And for Caltech 101, top-1 accuracy, top-5 accuracy, and minority accuracy ($N_{\text{min}} = 50$) are shown in Table 3. Obviously, with any one of the three losses, the enhancement encoding always performs better than the one-hot encoding.

Particularly, the minority accuracy of enhancement encoding is much higher in all experiments. As shown by the soft-confusion matrices in Figure 3 our method may decline the performance on some majority classes (class 0 and class 1) a little bit, but it improves the performance on minority classes (class 5-9) a lot. In Figure 4 we show the t-SNE visualization of the network’s outputs. It can be found that enhancement encoding makes samples with the same prediction result more concentrated. We run the programs of baseline and enhancement encoding 5 times per situation and report the error bars (Mean±SD) in Table 1, Table 2, and Table 3.

### Ablation Studies:

We evaluate the two incomplete enhancement encoding methods (re-weighting and cost-sensitive) and show the results in Table 1, Table 2, and Table 3. It can be observed that the two types of encoding also perform better than the baseline in most situations. Note the four elements located at the bottom left corner of the normalized soft-confusion matrices ($s'_{8,0}$, $s'_{9,1}$, $s'_{8,0}$, and $s'_{9,1}$) in Figure 3 the cost-sensitivity is lower, overall. It illustrates that the cost-sensitivity part of our method is very helpful to mitigate the adverse impact that the majority data overwhelms the minority.
4 Related Work

Researches on imbalanced learning can be divided into two categories: data-level methods and algorithm-level methods [3]. The former is to change the distribution of the training data and the latter is to change the strategy of training.
4.1 Data-Level Methods

Data-level methods include oversampling and undersampling.

**Oversampling:** This kind of method refers to increasing the number of minority instances to balance the data set. The simplest approach is to repeat the minority data during training, but in this way, the decision region of the minority class may be shrunk, which goes against the expectation [6]. Another idea for oversampling is to artificially produce new instances. Ha and Bunke [18] adjusted the original data slightly to get new training data. A more representative method is SMOTE, which is proposed by Chawla et al. [6]. SMOTE is to linearly synthesize instances between a sample and its $k$ neighbors with a little random perturbation. Based on the research of [6], some verities of SMOTE have been proposed, such as SMOTEBoost (Chawla et al. [7]), borderline-SMOTE (Han et al. [19]), k-means SMOTE (Last [31]), etc.

**Undersampling:** In contrast to oversampling, undersampling means deleting some majority instances to keep the training set balanced. It is generally believed that undersampling is not a good option for imbalanced learning because it strips out some information from the training set, but Drummond and Holte [12] found that undersampling may perform better than oversampling in some cases. There are some undersampling methods: Tomek [43] removed all the Tomek Links during training, Zhang and Mani [49] designed the NearMiss algorithm, and Liu et al. [34] proposed EasyEnsemble.

4.2 Algorithm-Level Methods

There are three kinds of algorithm-level methods for imbalanced classification. Note that these methods are not in conflict and sometimes they can work together.

**Thresholding:** In most classification tasks, the decision result is the class whose prediction probability is the highest (multi-classification) or higher than the threshold (binary classification). Thresholding is to adjust the decision threshold or rescale the prediction probability. Hernández-Orallo et al. [22] systematically compared several threshold choice methods with different metrics and described how to pick one. Sheng and Ling [40] implemented cost-sensitiveness via thresholding. Collell et al. [8] combined thresholding and the bagging algorithm (PT-bagging).

**Cost-Sensitive Learning:** Unlike general machine learning algorithms aiming to minimize classification error, cost-sensitive learning minimizes the cost of misclassification. With the larger cost of classifying a minority instance into a majority class, the classifier can reduce the misclassification of the minority data. Masnadi-Shirazi and Vasconcelos [35] combined the SVM and cost-sensitive learning. Khan et al. [28] altered the loss functions to make neural network cost-sensitive. Menon et al. [37] proposed a very similar approach. The main difference is that the cost in the former is learned from the data and the latter is based on the prior probability.

**Re-weighting:** This kind of method can be considered as setting different weights of loss function for the majority or minority data. Lin et al. [33] designed Focal Loss (see Equation (21)), which calculates the classification weights with outputs of the network. Cui et al. [9] re-weighted instances belonging to different classes based on the effective number. Cao et al. [4] proposed the Label-Distribution-Aware Margin Loss. Ren et al. [39] and Jamal et al. [25] separately designed two meta-learning algorithms to get the class weights.

5 Conclusion and Future Work

In this paper, we introduce the novel enhancement encoding, which combines re-weighting and cost-sensitivity. It provides an entirely new idea to address the class imbalance problem by changing the way to encode training labels. We also present the soft-confusion matrix, which reduces the validation data and extra computation for enhancement encoding. We evaluate the performances of enhancement encoding and separately study the effects of re-weighting and cost-sensitive. The results show that the enhancement encoding of labels effectively improves the performances of the neural network on the imbalanced classification, especially for the minority data.

At present, there are still some limitations of our approach and we hope to settle them in the future work. Firstly, We evaluate the enhancement encoding with some classic networks (LeNet-5 [32] and ResNet [21]). Experiments on emerging models (ViT [11], External Attention Transformer [16], etc.) are lacking. How much the enhancement encoding improves the performance of these new methods is still not clear. Secondly, we at present find the enhancement rate $\epsilon$ and the updating rate $\mu$ (see Equation (19)) by comparing the validation accuracy, which takes too much time and energy. Next, we are going to find a more efficient way to fine-tune the two hyper-parameters and explore if the hyper-parameters can be dynamic during training.
References

[1] Will C. Anderson, Kevin Carey, Eric M. Sturzinger, and Christopher J. Lowrance. Autonomous navigation via a deep q network with one-hot image encoding. In 2019 IEEE International Symposium on Measurement and Control in Robotics (ISMCR), pages A2–2–1–A2–2–6, 2019.

[2] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria C. Monard. Balancing strategies and class overlapping. In Advances in Intelligent Data Analysis VI, pages 24–35, 2005.

[3] Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. Neural Networks, 106:249–259, 2018.

[4] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In Advances in Neural Information Processing Systems, volume 32, pages 1565–1576, 2019.

[5] Nitesh V. Chawla. C4.5 and imbalanced data sets: investigating the effect of sampling method, probabilistic estimate, and decision tree structure. In ICML 2003 Workshop on Learning from Imbalanced Datasets II, 2003.

[6] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16:321–357, 2002.

[7] Nitesh V. Chawla, Aleksandar Lazarevic, Lawrence O. Hall, and Kevin W. Bowyer. SMOTEBoost: Improving prediction of the minority class in boosting. In Knowledge Discovery in Databases: PKDD 2003, pages 107–119, 2003.

[8] Guillem Collell, Drazen Prelec, and Kaustubh R. Patil. A simple plug-in bagging ensemble based on threshold-moving for classifying binary and multiclass imbalanced data. Neurocomputing, 275:330–340, 2018.

[9] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9260–9269, 2019.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In IEEE Conference on Computer Vision and Pattern Recognition, pages 248–255, 2009.

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In International Conference on Learning Representations, 2021.

[12] Chris Drummond and Robert C. Holte. C4.5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling. In ICML 2003 Workshop on Learning from Imbalanced Datasets II, 2003.

[13] Mohammed A. El Affendi and K. H. S. Al Rajhi. Text encoding for deep learning neural networks: A reversible base 64 (tetrasexagesimal) integer transformation (rit64) alternative to one hot encoding with applications to arabic morphology. In The Sixth International Conference on Digital Information, Networking, and Wireless Communications (DINWC2018), pages 70–74, 2018.

[14] Li Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. In 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW’04), volume 12, 2004.

[15] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, volume 15 of Proceedings of Machine Learning Research, pages 315–323, 2011.

[16] Meng-Hao Guo, Zheng-Ning Liu, Tai-Jiang Mu, and Shi-Min Hu. Beyond self-attention: External attention using two linear layers for visual tasks. arXiv preprint arXiv:2105.02358, 2021.

[17] Xinjian Guo, Yilong Yin, Cailing Dong, Gongping Yang, and Guangtong Zhou. On the class imbalance problem. In Fourth International Conference on Natural Computation, volume 4, pages 192–201, 2008.

[18] Thien M. Ha and Horst Bunke. Off-line, handwritten numeral recognition by perturbation method. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(5):535–539, 1997.

[19] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. In Advances in Intelligent Computing, pages 878–887, 2005.

[20] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering, 21(9):1263–1284, 2009.
[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016.

[22] José Hernández-Orallo, Peter Flach, and César Ferri. A unified view of performance metrics: Translating threshold choice into expected classification loss. Journal of Machine Learning Research, 13(91):2813–2869, 2012.

[23] Loretta Ichim and Dan Popescu. Retinal image segmentation based on weighted local detectors and confusion matrix. In 2017 40th International Conference on Telecommunications and Signal Processing (TSP), pages 654–658, 2017.

[24] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on Machine Learning, volume 37 of Proceedings of Machine Learning Research, pages 448–456, 2015.

[25] Muhammad Abdullah Jamal, Matthew Brown, Ming-Hsuan Yang, Liqiang Wang, and Boqing Gong. Rethinking class-balanced methods for long-tailed visual recognition from a domain adaptation perspective. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7607–7616, 2020.

[26] Justin M. Johnson and Taghi M. Khoshgoftaar. Encoding techniques for high-cardinality features and ensemble learners. In 2021 IEEE 22nd International Conference on Information Reuse and Integration for Data Science, pages 355–361, 2021.

[27] R. Karthiga, G. Usha, N. Raju, and K. Narasimhan. Transfer learning based breast cancer classification using one-hot encoding technique. In 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), pages 115–120, 2021.

[28] Salman H. Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A. Sohel, and Roberto Togneri. Cost-sensitive learning of deep feature representations from imbalanced data. IEEE Transactions on Neural Networks and Learning Systems, 29(8):3573–3587, 2018.

[29] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems, volume 25, pages 1097–1105, 2012.

[31] Felix Last. Oversampling for imbalanced learning based on K-Means and SMOTE. Master’s thesis, Universidade NOVA de Lisboa, 2018.

[32] Yan LeCun, Léon. Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[33] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(2):318–327, 2020.

[34] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39(2):539–550, 2009.

[35] Hamed Masnadi-Shirazi and Nuno Vasconcelos. Risk minimization, probability elicitation, and cost-sensitive SVMs. In Proceedings of the 27th International Conference on Machine Learning, 2010.

[36] Fadel M. Megahed, Ying-Ju Chen, Aly Megahed, Yuya Ong, Naomi Altman, and Martin Krzywinski. The class imbalance problem. Nature Methods, 18(11):1270–1272, November 2021.

[37] Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. Long-tail learning via logit adjustment. In International Conference on Learning Representations, 2021.

[38] Albert Orriols and Ester Bernadó-Mansilla. The class imbalance problem in learning classifier systems: A preliminary study. In Proceedings of the 7th Annual Workshop on Genetic and Evolutionary Computation, pages 74–78, 2005.

[39] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 4334–4343, 2018.

[40] Victor Sheng and Charles Ling. Thresholding for making classifiers cost sensitive. In Proceedings of the National Conference on Artificial Intelligence, volume 6, pages 476–481, 2006.

[41] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In International Conference on Learning Representations, 2015.
[42] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–9, 2015.

[43] Ivan Tomek. Two modifications of CNN. IEEE Transactions on Systems, Man, and Cybernetics, SMC-6(11):769–772, 1976.

[44] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. Journal of Machine Learning Research, 9(86):2579–2605, 2008.

[45] Sofia Visa and Anca Ralescu. Learning imbalanced and overlapping classes using fuzzy sets. In ICML 2003 Workshop on Learning from Imbalanced Datasets II, 2003.

[46] Dirk B. Walther. Using confusion matrices to estimate mutual information between two categorical measurements. In 2013 3rd International Workshop on Pattern Recognition in Neuroimaging, pages 220–224, 2013.

[47] Yunbo Xiong. Building text hierarchical structure by using confusion matrix. In 2012 5th International Conference on Biomedical Engineering and Informatics, pages 1250–1254, 2012.

[48] Yuzhe Yang and Zhi Xu. Rethinking the value of labels for improving class-imbalanced learning. In Advances in Neural Information Processing Systems, volume 33, pages 19290–19301, 2020.

[49] Jian-Ping Zhang and Inderjeet Mani. kNN approach to unbalanced data distributions: a case study involving information extraction. In ICML 2003 Workshop on Learning from Imbalanced Datasets II, 2003.

[50] Xianzhe Zhou and Arturo Del Valle. Range based confusion matrix for imbalanced time series classification. In 2020 6th Conference on Data Science and Machine Learning Applications, pages 1–6, 2020.
Appendices

A Mathematical Derivation on BP with Enhancement Encoding

In this appendix, we are going to talk about how the enhancement encoding influences the BP algorithm and show how it works effectively for imbalanced learning with cross-entropy loss and mean square error loss.

A.1 Cost-sensitiveness and Re-weighting

According to Equation (4) and Equation (10), it can be inferred that if the instance \( i \) belongs to class \( p \), the enhancement label of instance \( i \) is equal to the \( p \)-th row of generator matrix \( G \): \[
l_e^i = g_p \quad (i \in D_p). \tag{22}
\]
Then, some generator matrix elements’ properties described by Expression (12), (13), and (14) are also attached to the enhancement labels.

That is if the instance \( i \) belongs to class \( p \) (\( i \in D_p \)), then:

\[
\begin{align*}
l_{e,p}^i &> 1, \tag{23} \\
l_{e,q}^i &< 0 \quad (q \neq p), \tag{24} \\
\frac{N-1}{\sum_{q=0}^{N-1} l_{e,q}^i} & = 1. \tag{25}
\end{align*}
\]

To simplify the analysis, let \( \mu \) in Equation (19) be “1”:

\[
G = I - \epsilon C. \tag{26}
\]

Note the elements of one-hot labels in Equation (4), then:

\[
\begin{align*}
l_{e,p}^i &= g_{p,p} \\
&= 1 - \epsilon c_{p,p} \\
&= l_{i,p} - \epsilon c_{p,p} \quad (i \in D_p), \tag{27}
\end{align*}
\]

\[
\begin{align*}
l_{e,q}^i &= g_{p,q} \\
&= -\epsilon c_{p,q} \\
&= l_{i,q} - \epsilon c_{p,q} \quad (i \in D_p, p \neq q). \tag{28}
\end{align*}
\]

So:

\[
l_{e,q}^i - l_{i,q} = -\epsilon c_{p,q} \tag{29}
\]

Equation (29) shows how the enhancement labels implement cost-sensitiveness and re-weighting.

A.2 Situation with Cross-Entropy Loss

When dealing with classification, cross-entropy is usually utilized as the network’s loss function. It can be described as follows:

\[
J_i = -\sum_{q=0}^{N-1} l_{i,q} \log y_{i,q}. \tag{30}
\]

Let the base number of logarithms in Equation (30) be the natural constant “e”, then the partial derivative of cross-entropy with respect to \( y_{i,q} \) is:

\[
\frac{\partial J_i}{\partial y_{i,q}} = \frac{l_{i,q}}{y_{i,q}}. \tag{31}
\]
The Softmax activation of the output layer is another critical function for analysis, whose expression is:

$$ y_{i,q} = \frac{e^{z_{i,q}}}{\sum_{j=0}^{N-1} e^{z_{i,j}}} $$

where $z_{i,k}$ denotes the weighted sum of the output layer’s inputs. Softmax activation can rescale the network’s outputs into the interval $(0, 1)$ and let their sum be “1”. This attribute of Softmax can make the network predict probability very well. The partial derivative is:

$$ \frac{\partial y_{i,q}}{\partial z_{i,k}} = \frac{\partial e^{z_{i,q}}}{\partial z_{i,k}} \left[ \sum_{j=0}^{N-1} e^{z_{i,j}} \right] - \frac{e^{z_{i,q}} e^{z_{i,k}}}{\left[ \sum_{j=0}^{N-1} e^{z_{i,j}} \right]^2} $$

where

$$ \frac{\partial e^{z_{i,q}}}{\partial z_{i,k}} = \sum_{j=0}^{N-1} e^{z_{i,j}} - y_{i,q} y_{i,k} $$

$$ = \begin{cases} y_{i,k} - y_{i,k}^2, & q = k \\ -y_{i,q} y_{i,k}, & q \neq k. \end{cases} $$

Go further:

$$ \frac{\partial J_i}{\partial z_{i,k}} = \sum_{q=0}^{N-1} \frac{\partial J_i}{\partial y_{i,q}} \frac{\partial y_{i,q}}{\partial z_{i,k}} $$

$$ = \frac{\partial J_i}{\partial y_{i,k}} (y_{i,k} - y_{i,k}^2) - \sum_{q \neq k} \frac{\partial J_i}{\partial y_{i,q}} y_{i,q} y_{i,k} $$

$$ = \frac{\partial J_i}{\partial y_{i,k}} y_{i,k} - \sum_{q=0}^{N-1} \frac{\partial J_i}{\partial y_{i,q}} y_{i,q} y_{i,k} $$

$$ = -\frac{l_{i,k}}{y_{i,k}} \sum_{q=0}^{N-1} l_{i,q} y_{i,q} y_{i,k} $$

$$ = -l_{i,k} + \sum_{q=0}^{N-1} l_{i,q} y_{i,k} $$

$$ = y_{i,k} - l_{i,k} \quad \triangleright \text{Eq. (4)} $$

Obviously, enhancement labels also satisfy Equation (30) and (31), so $l_{i,k}$ in Equation (34) can be replaced with $l_{i,k}^e$:

$$ \frac{\partial J_i^e}{\partial z_{i,k}} = -l_{i,k}^e + \sum_{q=0}^{N-1} l_{i,q} y_{i,k} $$

$$ = y_{i,k} - l_{i,k}^e \quad \triangleright \text{Eq. (25)} $$

When updating the network’s parameters with gradient descent, the direction of updating vector $\Delta \theta$ on the loss surface is opposite to the gradient of $J$:

$$ \Delta \theta = -\eta \nabla \theta J. $$

Suppose instance $i$ belongs to class $p$ ($i \in D_p$), the difference between BPs with and without enhancement labels can be described by:

$$ \text{DBP}_{i,k} = \left( -\frac{\partial J_i^e}{\partial z_{i,k}} \right) - \left( -\frac{\partial J_i}{\partial z_{i,k}} \right) $$

$$ = (l_{i,k}^e - y_{i,k}) - (l_{i,k} - y_{i,k}) $$

$$ = l_{i,k}^e - l_{i,k} $$

$$ = -\epsilon c_{p,k} \quad \triangleright \text{Eq. (29)} $$

$$ = \begin{cases} -\epsilon c_{p,p} = \epsilon (1 - s_{p,p}') > 0, & p = k \\ -\epsilon c_{p,k} = -\epsilon s_{p,k}' < 0, & p \neq k. \end{cases} $$
Equation (37) means that compared with one-hot labels, the BP with enhancement labels makes $z_{i,p}$ increase more and $z_{i,k}$ ($k \neq p$) decrease more.

Suppose there is an instance $a$ belonging to a minority class $u$ ($a \in D_u$) and an instance $b$ belonging to a majority class $v$ ($b \in D_v$). In general:

\[
\begin{align*}
{s'_{u,u}} & < s'_{o,v}, \\
{s'_{u,v}} & > s'_{o,u}.
\end{align*}
\]  

Then:

\[
\begin{align*}
DBP_{a,u} & > DBP_{b,v} > 0, \\
DBP_{a,v} & < DBP_{b,u} < 0.
\end{align*}
\]  

Expression (40) means $z_{a,u}$ increases more than $z_{b,v}$ and Expression (41) means $z_{a,v}$ decreases more than $z_{b,u}$. In other words, the minority classes are attached with heavier class weights and higher misclassification costs than the majority classes.

### A.3 Situation with Mean Square Error Loss

Mean square error (MSE) loss is another useful loss function in machine learning. It can be described as follows:

\[
J_i = \frac{1}{N} \sum_{q=0}^{N-1} (y_{i,q} - l_{i,q})^2.
\]  

(42)

The partial derivative of MSE with respect to $y_{i,q}$ is:

\[
\frac{\partial J_i}{\partial y_{i,q}} = \frac{2}{N}(y_{i,q} - l_{i,q}).
\]  

(43)

And the partial derivative of MSE with enhancement labels is:

\[
\frac{\partial J_e}{\partial y_{i,q}} = \frac{2}{N}(y_{i,q} - l_{i,q}).
\]  

(44)

When analyzing the network with MSE loss, Softmax activation can complicate the derivation. So, we define $DBP_{i,k}$ ($i \in D_p$) with MSE based on partial derivative with respect to $y_{i,q}$:

\[
DBP_{i,k} = \left( -\frac{\partial J_e}{\partial y_{i,k}} \right) - \left( -\frac{\partial J_i}{\partial y_{i,k}} \right)
\]

\[
= \frac{2}{N}(l_e_{i,k} - y_{i,k}) - \frac{2}{N}(l_{i,k} - y_{i,k})
\]

\[
= \frac{2}{N}(l_e_{i,k} - l_{i,k})
\]

\[
= -\frac{2\epsilon c_{p,k}}{N} \quad \text{Eq. (29)}
\]

\[
\begin{cases}
-\frac{2\epsilon c_{p,p}}{N} = \frac{2\epsilon}{N}(1 - s'_{p,p}) > 0, & p = k \\
-\frac{2\epsilon c_{p,k}}{N} = -\frac{2\epsilon}{N}s'_{p,k} < 0, & p \neq k.
\end{cases}
\]

(45)

So, there are two conclusions which are similar to Section A.2:

1. Compared with one-hot label, enhancement label makes $y_{i,p}$ increase more and $y_{i,k}$ ($k \neq p$) decrease more.
2. Suppose there is an instance $a$ belonging to a minority class $u$ ($a \in D_u$) and an instance $b$ belonging to a majority class $v$ ($b \in D_v$), then:

\[
\begin{align*}
{s'_{u,u}} & < s'_{o,v}, \\
{s'_{u,v}} & > s'_{o,u}.
\end{align*}
\]  

(46)

And:

\[
\begin{align*}
DBP_{a,u} & > DBP_{b,v} > 0, \\
DBP_{a,v} & < DBP_{b,u} < 0.
\end{align*}
\]  

(47)

(48)

(49)

That means $y_{a,u}$ increases more than $y_{b,v}$ and $y_{a,v}$ decreases more than $y_{b,u}$.
B The Imbalanced Setting of MNIST

Figure 5: The training sets of imbalanced MNISTs. The original data set is balanced. To evaluate our algorithm, we manually unbalance the data.

The distributions of MNIST can be described as:

\[ n_{tr}^{*} = \left\lfloor \frac{n_{tr}^{p}}{f_{p}} \right\rfloor - n_{v}^{p} \]  \hspace{1cm} (50)

where \( n_{tr}^{p} \) denotes the number of training images belonging to class \( p \) before adjustment and \( n_{tr}^{*} \) denotes the number after adjustment, \( n_{v}^{p} \) denotes the number of validation samples per class. We adjust the data set in two ways:

\[ f_{p} = \begin{cases} 9[(p + 1) \mod 2] + 1, & \text{10\% of even classes} \\ 2^{p}, & \text{long-tailed distribution} \end{cases} \]  \hspace{1cm} (51)

\[ n_{v}^{p} = \begin{cases} 10, & \text{10\% of even classes} \\ 5, & \text{long-tailed distribution} \end{cases} \]  \hspace{1cm} (52)

where \( p \) denotes the index of class and note that \( 0 \leq p \leq 9 \). In the first one (10\% of even classes), we refer to the data distribution created by Khan et al. [28]. Because Khan’s beginning index of class and ours are different, there are some statement differences. The distributions are shown in Figure 5.

C Potential Negative Societal Impacts

The imbalanced learning will help the minority groups get enough social attention, but it may also be used to identify them and cause discrimination (such as HIV patients). It’s necessary to make sure imbalanced classification methods (including the enhancement encoding) will not harm the legal rights of minority groups.

D Computational Resources

We train all models in this paper with the internal cluster and the GPU is NVIDIA Tesla V100S.