Research on unbalanced training samples based on SMOTE algorithm

Kai Li$^{1,2}$ and Yueli Hu$^{1,2,3}$

$^1$Shanghai Key Laboratory of Power Station Automation Technology, Shanghai 200072, China
$^2$School of Mechatronic Engineering and Automation, Shanghai University, Shanghai 200072, China
$^3$E-mail: huyueli@shu.edu.cn

Abstract. In the classification tasks of deep learning, unbalanced data distribution of training samples is a serious problem. In this case, the deep neural networks will bias to the majority-class samples and can not learn data characteristics of the minority-class samples well, making it difficult to identify the minority-class samples. To improve the performance of deep neural networks on minority-class samples, this paper proposes a novel scheme based on SMOTE algorithm, which generates new minority-class samples to balance the training dataset. Compared with the traditional over-sampling operations commonly used in image classification tasks such as copying images simply, image flipping, color adjustment etc, our scheme generates new minority-class samples, which improves the features of minority-class samples and avoids the over-fitting problem in classification. The workflow of image preprocessing and SMOTE over-sampling operation are introduced in this paper. The selection basis of the network model and the comprehensive model indicators are also presented. Comparative experiments are performed by testing models trained respectively on unbalanced dataset and balanced dataset after SMOTE over-sampling operation. Results shows that the proposed scheme is feasible and effective to improve the neural network’s ability of identifying minority-class samples in classification tasks.

1. Introduction

At present, image classification technology based on deep learning has been widely used in smart cameras, driver-less technology and face recognition. The two key aspects of deep learning are training data and deep neural network. The networks update their weights by training on the dataset which enables the network to learn the feature representation of different categories of objects and have the ability to identify them [1]. When training deep neural networks, we often use supervised learning methods which rely on labeled dataset. Larger networks with more parameters need more data for training. Therefore, selecting the appropriate network based on the amount of training data is very important in classification tasks. In addition, labeled dataset is often difficult to collect, when lacking images for some specific categories, the training dataset will be unbalanced [2]. In general, when the ratio of the minority-class samples to the majority-class samples is greater than 1:3, we classify it as an unbalanced dataset. In unbalanced training samples, the contribution of the minority-class samples to the final network model is certainly much smaller than that of the majority-class samples, as a result, the network model is heavily biased towards majority-class samples. Although the accuracy of the model on the testing dataset may not be very low, we will find that the images
correctly recognized are almost majority-class samples [3]. There is no doubt that such a network model is not practical in engineering applications [4]. Current solutions to the problem of unbalanced training samples are divided in two aspects.

The first idea is to balance the training samples and it can be achieved by over-sampling minority-class samples or under-sampling the majority-class samples. A common method for under-sampling majority-class samples is to randomly discard the majority-class samples until the number of it is roughly equal to the number of minority-class samples. This approach is often applied when the training dataset is large enough, otherwise, the classifier can not learn enough data features due to lack of training data and have a bad performance in applications. There are two methods for over-sampling minority-class samples. One is to violently copy minority samples to force a balanced training dataset, and then scram it for training. The other is to split the majority-class samples into several training subsets, the number of each subset is roughly equal to the minority-class samples. Then each subset is added with minority-class samples to balance the training dataset and is trained in turn during network training process. Both of these two over-sampling methods just simply copy the minority-class samples for training, although the accuracy of model is higher than before, the network is easily over-fitting minority-class samples and behave badly when testing [5].

The second idea is to use the loss function with unbalanced cross entropy which gives minority-class samples with higher training weights and directly promotes the contribution of minority-class samples to the final model. So that the network pays more attention to them when training [6]. An important point in the design of deep neural networks is the choice of loss function. In general, the final output layer of the deep neural network will be connected by a softmax layer to transform the output into a vector that satisfies the probability distribution, which means that we would use cross entropy between model prediction and real tags as the loss function. Sometimes, in order to prevent the network from over-fitting the training samples, regularization terms are added to the loss function [7]. In the paper “Focal Loss for Dense Object Detection”, the author proposed the unbalanced cross entropy as a loss function [8]. By observing the loss in training process, the author gave higher weights to the hard-to-train samples whose training loss is higher than others, which makes the network more biased toward such samples in the later stages of training, and improves the performance of the final network. Experiments proved that this method is practical and effective for the unbalanced sample problem in deep learning.

Both under-sampling majority-class samples and simply replicating minority-class samples just improve the training tricks and can not basically enhance the characteristics of the minority-class samples. Thus, this paper proposes a novel scheme based on SMOTE algorithm for the training of deep neural networks with unbalanced training samples. We first preprocess the unbalanced dataset, convert the images into arrays of float values with the interval (0,1) and label them. The network model is selected according to the scale of our dataset size. Comprehensive indicators that measure network performance are proposed, including accuracy, recall, precision, and F1 score. After testing LeNet, AlexNet, and VGGNet, we choose VGGNet as the classification network for it’s better performance than others. Then we introduce the steps of the SMOTE algorithm of generating new minority-class samples. VGGNet-16 is trained respectively on the unbalanced dataset and the balanced dataset after SMOTE over-sampling. Finally, we compare the performance of the two network models on the testing dataset. Results shows that the model’s performance on minority-class samples has a large development after SMOTE over-sampling. Therefore, our proposed method is feasible and effective for improving the networks’ ability to identify minority-class samples in classification tasks of deep learning.

2. Data preprocessing

2.1. One-hot encoding
One-hot encoding encodes $S$ states using $S$ bits with each state bit controlling a state, and in any case only one state bit is valid. Assuming that the number of categories for image classification is $S$, the
one-hot code corresponding to each image can be represented by an S-dimension vector recorded as \( v = (v_0, v_1, \ldots, v_S) \) which satisfies the probability distribution. It should be noted that the one-hot code starts from the 0th bit as the status bit, so the labels of the images in the data preprocessing section start from number 0. For example, when the image label is 0 and the total category number is S, the code corresponding to the label is an S-dimension vector \((1,0,\ldots,0)\) with corresponding status bit set to 1, and others set to 0 [9]. Cross entropy which measures the distance between two probability distributions is often used as an important component of the model loss function during back propagation. Therefore, the cross entropy between the neural network forward propagation prediction and the corresponding label can be conveniently calculated by encoding the image with one-hot encoding.

\[
\begin{align*}
v &= (v_0, v_1, \ldots, v_S) \\
v &= (1,0,\ldots,0) \\
\end{align*}
\]

2.2. Image preprocessing
We divide the 1500 images of daisy and dandelion collected from the Internet into two categories, including 300 daisy images for class A and 1200 dandelion images for class B. In our experiment, class A is set as positive class. All images are saved in jpg format or jpeg format. The ratio of the number of Class A to Class B is 1:4, which can be considered as an unbalanced distribution. Then the dataset is preprocessed with steps shown in Figure 1. The two categories of images are read in turn first, and then decoded into Numpy arrays under the Tensorflow framework. The value of the image array is integer with the interval \((0,255)\). We need to convert the data type of the image array into

\[
\begin{align*}
\text{Start} \\
\text{Initialize num}=0 \\
\text{Read one image} \\
\text{Decode the image into an integer array} \\
\text{Convert the image array to float32} \\
\text{Resize the image array to 224*224*3} \\
\text{Generate a random number rnd in the interval (0,1)} \\
\text{num=num+1} \\
\end{align*}
\]

**Figure 1.** Data preprocessing steps.
float32 with interval (0,1) to reduce the precision loss during image preprocessing process. In order to unite the size of input images of the network, we resize the images to 224*224*3 and label them with number 0 and 1 for convenient use of one-hot coding. The images are divided into a testing set and a training set in a ratio of approximately 1:4.

Finally, the training set is scrambled so that the network model can better learn the characteristics of different types of data. Then save the processed dataset in the format of Numpy array(.npy) for convenient loading when training neural networks [10].

3. Network model

3.1. Selection of Convolutional Neural Networks

Convolutional neural networks (CNN) have their unique advantages in image classification tasks. From LeNet [11], AlexNet [12], VGGNet [13], Google InceptionNet [14], ResNet [15] to the present, the depth and complexity of networks have been increasing rapidly. The latest network depth has reached 200 layers with more powerful character expression ability than before. These networks have been enthusiastically sought after by deep learning researchers and learners after been open sourced on Github, and soon been employed in a variety of research projects and products. Larger networks tend to handle more complex image classification tasks. Meanwhile, they require more training data for convergence. Therefore, choosing CNN with appropriate scale according to the complexity of the task and the size of the dataset is very important. The dataset used in this paper has only 1500 images, after division, the training set has only about 1200 images, so it is suitable to choose a medium-sized CNN.

We trained unbalanced dataset respectively on LeNet, AlexNet, and VGGNet for 10000 steps with batch size as 64 and tested them. The prediction accuracy of LeNet, Alexnet, and VGGNet were 73.3%, 75.2%, and 80.0%. We found that VGGNet performed better than other networks. What’s more, a large number of open sourced projects have proved that VGGNet is well generalized on multiple classification tasks. Therefore, we selected VGGNet-16 as the classifier of this paper and used TensorFlow to establish the network model, because it can easily realize network construction and GPU computing deployment [16].

This paper adopts VGGNet-16 which has 5 parts of convolution operation. Its structure is shown in Figure 2. The entire network has a total of 13 convolutional layers and three fully connected layers, after each convolutional layer there will be a max-pooling layer. The first segment of the convolutional layer use 64 convolution kernels, the second segment of the convolutional layer use 128 convolution kernels, the third segment of the convolutional layer use 256 convolution kernels, and the last two segments of convolutional layer use 512 convolutional kernels [17].

![Figure 2. Structure of VGGNet-16.](image)
The original VGGNet was designed for the ILSVRC image classification competition which needs to distinguish 1000 types of objects, so the fully connection layers responsible for feature integration were relatively large. The dimensions of the three fully-connected layers are 4096, 4096 and 1000. We know that the quantity of the parameters of convolutional neural networks mainly comes from its fully connected layers, because the parameter quantity of the convolutional layer is only related to the size of the convolution kernel. Since the dataset size of this paper is much smaller than that of ImageNet, the use of such a large scale of fully connected layers tends to over-fitting the training data. So, after multiple experimental tests we determined the dimensions of the fully-connected layers as 1024, 1024, and 2. The dimension of the last layer were determined based on the number of categories of the classification task.

3.2. Model indicators

It is very important to select appropriate evaluation indicators when evaluating the performance of the network models and the evaluation indicators of the network model are different with different tasks. This paper use accuracy, recall, precision and F1 score as the evaluation indicators of the network model [18]. Here are several basic concepts in the classification task, the true positive rate TPR, the true negative rate TNR, the false positive rate FPR, and the false negative rate FNR. Supposing that the number of positive samples which belong to class A is P, and the number of negative samples which belong to class B is N, TP refers to the number of samples that are recognized as positive with positive labels, TN refers to the number of samples that are identified as negative with negative labels, FP refers to the number of samples that are recognized as positive with negative labels, FN refers to the number of samples that are recognized as negative with positive labels.

Accuracy refers to the ratio of the number of correctly identified samples in the classification project to the total samples, as defined in Equation (1):

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}
\]

Recall refers to the ratio of positive samples that are correctly identified to the number of all positive samples, as defined in Equation (2):

\[
\text{Recall} = \frac{TP}{TP + FN}
\]

Precision refers to the ratio of positive samples that are correctly identified to the number of all samples which are recognized as positive samples, as defined by Equation (3):

\[
\text{Precision} = \frac{TP}{TP + FP}
\]

To some extent, recall and precision are contradictory. Assuming that in extreme cases, the number of positive samples is P and the classifier only determines one sample with positive label to be positive. So the precision of the classifier is 100%, but the recall is 1/P. Conversely, if we recall all positive samples in the sample space, it is almost inevitable that some of the negative samples will be misclassified as positive samples, and the precision of the classifier will decrease. In practice, it’s useless to simply bias to the precision or recall, it’s important to balance the two points. In this case, researchers proposed the concept of F1 score, which is defined by the the harmonic average value of precision and recall [19]. The definition of F1 score is as shown in Equation (4):

\[
F_1\text{score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]

This paper use accuracy, recall, precision and F1 score to evaluate the performance of VGGNet-16. We compare the performance of VGGNets trained respectively on the unbalanced dataset and the balanced dataset after SMOTE over-sampling to verify the validity and practicability of our proposed scheme for solving the problem of unbalanced training samples.
3.3. Test on unbalanced dataset

The original VGGNet-16 network was trained on unbalanced dataset for 10000 steps. Since each training step with a large batch size consumes too much memory resources of the computer and is prone to cause resource overflow errors. The selection of batch size needs to be determined based on the computer's memory. The batch size selected in this paper is 64. The training device was a GTX1080TI GPU with memory of 12GB. The whole training process took about 90 minutes. Tensorboard was used to monitor the loss value and the prediction accuracy of each batch during training process. The loss value of each training step is shown in Figure 3, it can be seen that the loss value decreased continuously during training process, and tended to be 0 after 10,000 iterations. The prediction accuracy on each batch during training process is shown in Figure 4, it can be seen that the accuracy of each batch on the training dataset after the training of 10000 steps was stable at around 98%, which indicates that the network fitted the training data well.

After training, we tested the model to observe the performance of the network trained on unbalanced dataset. As a result, the prediction accuracy was 0.80, the recall was 0.755, the precision was 0.783, and the F1 score was 0.768. It can be seen that the model does not perform well on the testing samples, mainly because the network cannot learn the characteristics of minority-class samples well. Therefore, we used SMOTE algorithm to oversampling minority-class samples to balance the training dataset, and then trained a new VGGNet-16 with the processed dataset and compare its testing results with the model trained on unbalanced dataset.

4. SMOTE algorithm

4.1. K-neighbors

The k-neighbors of a sample refers to the k samples closest to the sample in the sample space. This paper uses Euclidean distance to calculate the distance between samples. If the coordinates of two points with N-dimension are \( X = (x_1, x_2, ..., x_N) \), \( Y = (y_1, y_2, ..., y_N) \), then the Euclidean distance between them is defined as Equation (5) [20]:

\[
d(X,Y) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}
\] (5)

In this paper, the RGB image with three color channels is converted into a Numpy array with size of 224*224*3 according to data preprocessing section. Therefore, when calculating the k-neighbors of images, it can be regarded as a 3-dimension feature space. The Euclidean distance between each pixel of the image and the corresponding pixel of other images in the sample space can be calculated by Equation (5), then add the Euclidean distance between all the pixels of two images to obtain the distance. The calculation can be carried out very quickly using the scientific computing library of Python.
4.2. Generating minority-class samples by SMOTE algorithm

The procedure for generating new minority-class samples using SMOTE method is shown in Figure 5. Assuming that the number of minority-class samples in training dataset is $m$, the number of majority-class samples in training dataset is $n$, the sampling magnification $N$ is defined by $N = \frac{n - m}{m}$. That is to say, after oversampling, the number of minority-samples is basically equal to the number of majority-class samples. We first select a sample $x$ from minority-class samples and calculate its k-neighbors. It needs to be noted that there is no absolute standard for the selection of k-value. It is necessary to perform several experiments of k-value within a reasonable range according to its own sampling magnification. The k-value is definitely greater than or equal to the over-sampling magnification, because we have to select at least N different samples from the k-neighbors to generate new minority-samples. Using k-value which is greater than N helps to improve the data characteristics of the randomly generated sample, but it also brings a lot of extra calculation. The sampling magnification in this paper is 3, after trying several values such as 5, 7, 9, we choose 7 as the last k-value to improve the data characteristics of the randomly generated sample and avoid too much calculation which can not bring certain improvement for the algorithm.

Randomly picking a neighbor sample $x_n$ in the k-neighbors of $x$ and generate a new sample according to Equation (6):

$$x_{new} = x + rand(0,1) \left| x - x_n \right|$$ (6)

By repeating the above process of generating new samples $N$ times, we can obtain $N$ synthetic minority-class samples. In this way, $N$ new samples are synthesized for each sample in minority-class samples, and a total of $m*N$ new samples can be generated, so that the number of minority-class samples and majority classes in sample space will be balanced.

![Figure 5. SMOTE over-sampling steps.](image)
5. Experiment

After oversampling the training dataset with the SMOTE algorithm, we unite the data format following the data preprocessing section above, and then use the over-sampled dataset to train VGGNet-16. As with the previous training, a total of 10000 steps with batch size 64 were trained. The whole training process will be longer than before, for about two hours. As shown in Figure 6 and Figure 7, the loss of the network decreased with the increase of the number of iterations and finally be close to 0, and the accuracy of the prediction continued to rise and finally tended to be 1.

![Figure 6. Loss value of balanced dataset.](image1)

![Figure 7. Accuracy of balanced dataset.](image2)

Finally, we use the trained network to predict the image categories in the testing dataset and compare the result with that of the unbalanced dataset. As shown in Table 1, the accuracy after SMOTE over-sampling was 0.852, the recall was 0.836, the precision was 0.829, and the F1 score was 0.833. Compared with the performance of the network trained on the unbalanced dataset before, we can see that the accuracy of the network has increased by approximately 5%, and other indicators also have a large improvement. Because after balancing the training dataset, the model learns data features without bias towards majority-class samples, it can better learn the data characteristics of minority-class samples. Results shows that the deep neural network can identify minority-class samples better after SMOTE over-sampling.

| Model Indicators | Unbalanced dataset | Balanced dataset |
|------------------|--------------------|------------------|
| Accuracy         | 0.80               | 0.852            |
| Recall           | 0.755              | 0.836            |
| Precision        | 0.783              | 0.829            |
| F1 score         | 0.768              | 0.833            |

6. Conclusions

In this paper, we propose a novel scheme based on SMOTE algorithm to solve the problem of unbalanced training samples in deep learning which results in the network’s bad performance in recognizing minority-class samples. After SMOTE over-sampling operation, the neural networks can better learn the data characteristics of minority-class samples. VGGNet-16 was trained respectively on the unbalanced dataset and the balanced dataset after SMOTE over-sampling. Experiments showed that the accuracy, recall, precision and F1 score of the network had a large improvement after SMOTE over-sampling operation. The proposed scheme also has a shortcoming point. When performing SMOTE over-sampling operation, the choice of k-value must ensure the requirements of SMOTE over-sampling magnification and avoid too much extra calculation which will affect the efficiency of the algorithm. We need to perform several tests of k-value based on our own tasks to select the suitable value. In conclusion, results proved that our proposed scheme can effectively improve the
neural network’s ability to identify minority-class samples in image classification tasks of deep learning.

References
[1] Joe Minichino and Joseph Howse 2016 *Learning OpenCV3 Computer Vision with Python* (Beijing) p 1-3
[2] Arshad A, Riaz S and Jiao LC 2019 Semi-Supervised Deep Fuzzy C-Mean Clustering for Imbalanced Multi-Class Classification *IEEE ACCESS* 7 28100-28112
[3] Zhang C, Tan KC, Li HZ and Hong GS 2019 A Cost-Sensitive Deep Belief Network for Imbalanced Classification *IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS* 30 109-122
[4] Xiao Z, Wang L and Du JY 2019 Improving the Performance of Sentiment Classification on Imbalanced Datasets With Transfer Learning *IEEE ACCESS* 7 28281-28290
[5] Zhang HY, Zhao Z, An T, Lao BQ and Chen X 2019 Pulsar candidate recognition with deep learning *COMPUTERS & ELECTRICAL ENGINEERING* 73 1-8
[6] Hashemi SR and Salehi SSM 2019 Asymmetric Loss Functions and Deep Densely-Connected Networks for Highly-Imbalanced Medical Image Segmentation: Application to Multiple Sclerosis Lesion Detection *IEEE ACCESS* 7 1721-1735
[7] Khan SH, Hayat M and Porikli F 2019 Regularization of deep neural networks with spectral dropout *NEURAL NETWORKS* 110 82-90
[8] Goyal Priyal and He Kaiming 2018 Focal loss for dense object detection *IEEE transactions on pattern analysis and machine intelligence* 39 2999-3007
[9] Gullace NF 2019 One Hot Summer: Dickens, Darwin, Disraeli, and the Great Stink of 1858 *HISTORIAN* 81 163-164
[10] Eric Matthes 2016 *Python programming from entry to practice* (Beijing) pp 180-186
[11] Lecun Y L, Bottou L and Bengio Y 1998 Gradient-Based Learning Applied to Document Recognition *Proceedings of the IEEE* 86 2278-2324
[12] Krizhevsky A, Sutskever I and Hinton G 2012 ImageNet Classification with Deep Convolutional Neural Networks *Advances in neural information processing systems* 25 1097-1105
[13] Simonyan K and Zisserman A 2015 Very Deep Convolutional Networks for Large-Scale Image Recognition *International Conference of Learning Representation (ICLR)* 1-14
[14] Szegedy C, Vanhoucke V and Ioffe S 2016 Rethinking the Inception Architecture for Computer Vision *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 2818-2826
[15] He K, Zhang X and Ren S 2016 Deep Residual Learning for Image Recognition *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 770-778
[16] Chang Yan, Wei Daxiu and Jia Huihui 2019 Spin-Scenario: A flexible scripting environment for realistic MR simulations *Journal of magnetic resonance* 301 1-9
[17] Huang WJ and Tang Y 2017 *Practice of Tensorflow* (Beijing) pp 108-119
[18] Afzali MH and Sunderland M 2019 Machine-learning prediction of adolescent alcohol use: a cross-study, cross-cultural validation *ADDICTION* 114 662-671
[19] Perez F, Font J, Arcega L and Cetina C 2019 Collaborative feature location in models through automatic query expansion *AUTOMATED SOFTWARE ENGINEERING* 26 161-202
[20] Lan Goodfellow, Yoshua Bengio and Aaron Courville 2017 *DEEP LEARNING* (Beijing) pp 92-94