Abstract

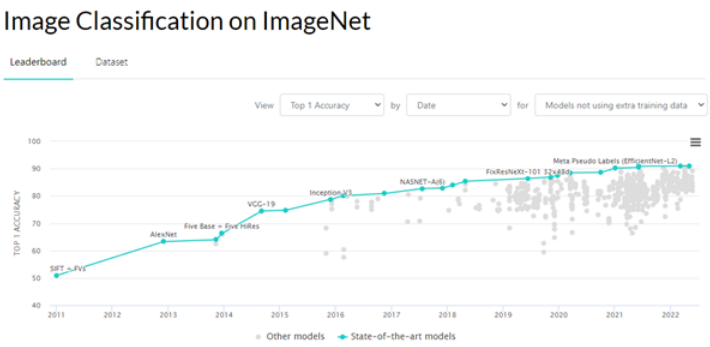

Figure 1: SOTA of image classification task on ImageNet

In Figure 1, The name shown in graph is the model that achieved State-of-the-art in Image classification on ImageNet each year. In 2021, the Top 1 accuracy of the model[4] exceeded 90%, and the accuracy of the model[7] announced in 2022 achieved sota with 91%. However, in 2018, there was an incident in which the evaluation scores of female applicant were deducted in Amazon's AI recruitment system. The reason is that AI judged that men are more suitable for hiring differently from the criteria for hiring. At that time, the Top 1 Accuracy of the Classification task on ImageNet was about 85%. What causes these problems to occur?

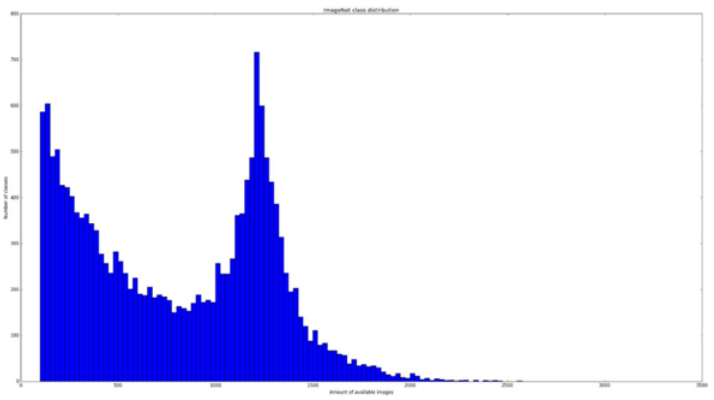

Figure 2: ImageNet class distribution

In figure2, we can see the distribution of imagenet training set (ILSVRC2012). Extremely, data in the real open world is not uniform as its distribution is like Exponential distribution. That is, the number of data belonging to some class is very large and the data if other classes is very small. This problem is called the long-tailed dataset or class imbalance problem. In order to solve this problem, it is more time-consuming and expensive to collect data of classes with fewer samples than dominent samples. Therefore, research is in progress by intentionally creating a long-tailed dataset like ImageNet-LT, CIFAR100-LT.

Figure 3 shows the Top 1 Accuracy of the model trained on the Long-tailed ImageNet dataset. The first published model has very low ability to perform classification on unbalanced datasets. Even it is very surprising

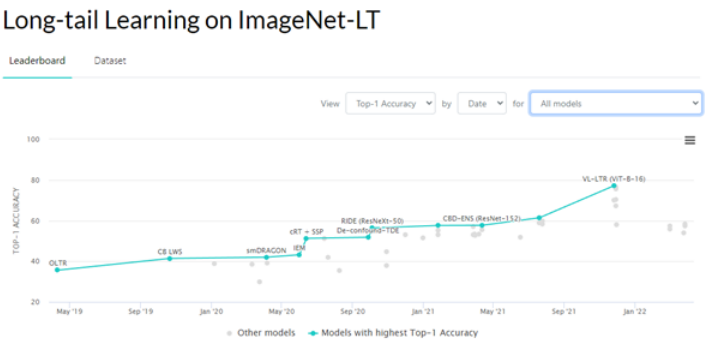

Figure 3: SOTA of image classification task on ImageNet-LT

that the performance of the model is lower than that of Alexnet[5] using a typical dataset. Researched from 2019, the most recent model[6] achieved an accuracy of 77.2%. Compared to the performance of the classification model on ImageNet dataset, it shows poor performance.

1 How to solve long-tailed dataset

1.1 The simplest way to solve imbalance problem

The data rebalancing method is the most easily approached method to solve the class imbalance problem. This is to allow existing datasets to have different distributions. Ultimately, the purpose of this resampling is to make the distribution of the data evenly. The data rebalancing method is to remove some data of head class(major class) by under-sampling and use the data of the tail(minor) class as duplicate by over-sampling. (The dominant classes that occupy most of the data are called the head class, and the classes with a very small number of data are called the tail class) Over-sampling has a problem that overfitting can occur by using duplicate data. On the other hand, downsampling is a waste of time and money to obtain data. It is necessary to check whether this method has a good effect on the performance of the model, since there are problems.

1.2 Joint learning vs. Decoupling representation and classifier

1.2.1 Data Re-balancing

The paper[3] separates representation learning and classification in the learning process and shows the difference in how each learning affects long-tailed recognition. First, to go throught the effect of representation learning, the long-tailed dataset is readjusted with four sampling strategies.

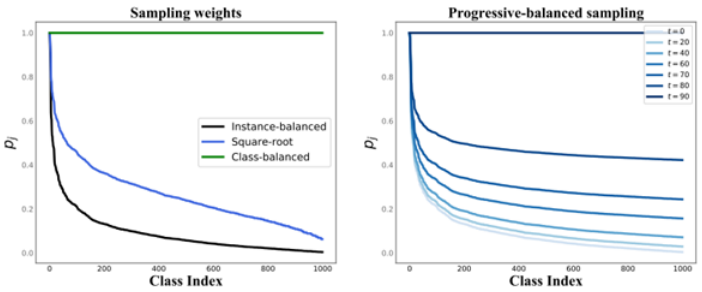

Figure 4: Data sampling strategies

Figure 4 shows the weights for each class when sampling data.

Instance-balanced sampling: all samples are extracted with the same probability. The number of samples belonging to the label affects the probability that the class is extracted. That is, it is a strategy that has the same effect as not performing data re-balancing. The sampling weight is equal to long-tail distribution.

Class-balanced sampling: Each class is selected with the same probability, and then samples are uniformly extracted from data of selected class. We can guess that this method causes the most re-balancing among the four methods.

Square-root sampling: When the number of samples belonging to a class is n, the probability of data extraction from the class is expressed as

$$p_j = \frac{n_j^q}{\sum_{i=1} Cn_i w}, \tag{1}$$

where $q = \frac{1}{2}$ and C is the number of training classes.

Progressively-balanced sampling: It is a method that starts with instance-balanced sampling at first and gradually changes to class-balanced sampling as the train progresses.

1.2.2 Classifier adjustment

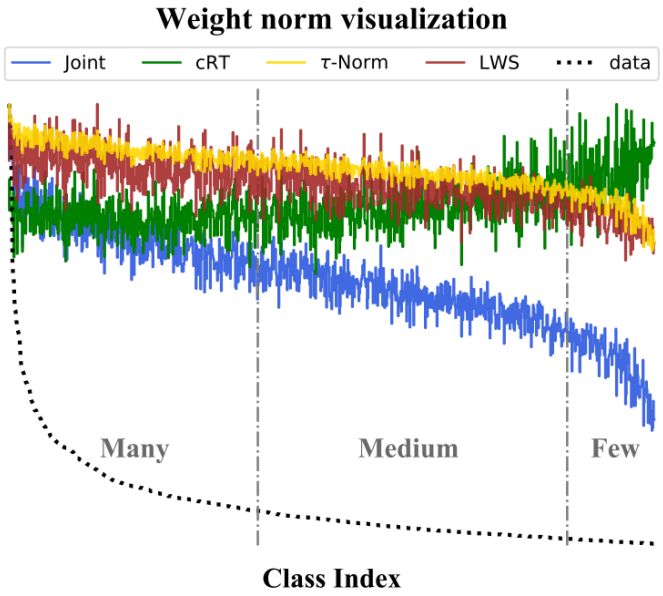

Figure 5: Weight Balancing strategies

Classifiers are learned jointly with the representations end-to-end. This scheme is called joint learning. But we have to confirm the performance of representation learning. So, Compared to the commonly used joint learning, we introduce methods to replace classifiers or readjust weights. Methods such as Classifier Re-training (cRT), Nearest Class Mean classifier (NCM) and Γ-normalized classifier (Γ-normalized), and LWS are used for long-tailed recognition. Figure 5 is the result of adjusting the weights through each method.

cRT The main idea of the cRT method is to re-train the classifier using class-balanced sampling. To be more specific, randomly re-initialize and optimize the classifier weights for a small number of epochs using class-balanced sampling, keeping the representations fixed.

NCM: first calculate the mean feature representation for each class on the training dataset and then perform nearest neighbor search using cosine similarity or Euclidean distance.

τ-normalized: After joint training using instance-balanced sampling, it was found that the norm$||w_J||$ of the classifier's weight was related to the number of samples n_j belonging to the class j. And, when the cRT method using class-balanced sampling was used, it was found that the norms of each class were similar(we can see the wave of green color in Figure 5). Therefore, this method solves the imbalance by directly adjusting the norm value using a hyperparameter τ named "temperature". Formally, let w_i is the classifier weights corresponding to class i. we formulate scaled weight $\widetilde{w}_i$ as:

$$\widetilde{w}_i = \frac{w_i}{||w_i||^\tau} \tag{2}$$

If τ is 1, $\widetilde{w}_i$ is the same as L2-normalization, and if τ is 0, it is the same as not scaling. For example, When the weights from 0.1 to 1 are τ-*normalized in units of* 0.1 *and* $\tau = 0.7$:
In Figure??, we can see that the smaller the weight will be more rescaled, thereby control

1.2.3 Compare all strategies and select one method

A surprising finding discovered through this experiment is that instance-balanced sampling that does not use a special sampling strategy can learn the most generalizable representation. In other words, it shows that long-tailed recognition is well achieved just by adjusting the classifier when learning by sampling data as it is (instance-balanced). We can see the result intuitively in Figure 7. When training jointly(joint learning) is performed, it is better to apply a complex(or good) sampling strategy in the order of Red(Progressively-balanced) - Yellow(Square-root) - Green(Class-balanced) - Blue(Instance-balanced). However, it can be seen that the decoupled method shows the best performance, especially when the representation is learned by performing instance-balanced sampling. And, For "Medium" and "Few" data, the $\tau - norm$ method shows the best results.

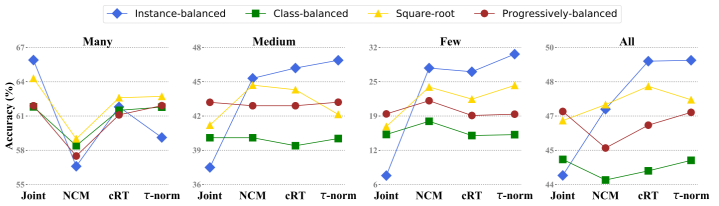

Figure 7: The performance of different classifiers for each split on ImageNet-LT with ResNeXt-50. Colored markers denote the sampling strategies used to learn the representations

1.3 Additional method : optimal weight decay

In order to solve the long-tailed recognition problem, a hyperparameter that plays an important role in the learning process of the model is "weight decay". In aforementioned description, the norm of the weight of the classifier is proportional to the number of data to which each class belongs. In other words, unlike rare classes, common classes with a lot of data have larger weights.
One way to solve this is L2-normalization. L2-normalization balances the weights of all classes with the unit norm. However, this hard constraint prevents the classifier from learning better. As a result, flexible learning is difficult, and the accuracy of tail data may increase, but overall performance may decrease.
On the other hand, the weight decay method penalizes weights with relatively large values. The loss is constructed by giving the square of the weight of the model as a penalty term. It can be expressed in the following formulation.

$$Loss(w,x) = DataLoss(w,x) + \frac{1}{2}\lambda||w||^2 \tag{3}$$

By using weight decay, we allow the model to focus more on weights with small values in training. Accordingly, the generalization of the model can be expected to improve, and we need to select the weight decay value suitably.

1.3.1 Search Algorithm for hyperparameter optimization

Hyperparameters must be set manually when constructing a model, and it takes a lot of time to find an optimized value. Therefore, it is necessary to know how to optimize hyperparameters. The most used algorithms are Grid Search and Random Search. Grid Search selects candidate hyperparameters' value within a specific section to be searched at the same interval and measures the performance of each of them. Random search selects candidate hyperparameters' value in the search target section through random sampling. In both methods, the best parameter is selected from the measurement results. The two methods only test each parameter, but do not find the optimal parameter.
On the other hand, Bayesian optimization[2] is a method that can systematically perform the entire search process while sufficiently reflecting 'prior knowledge' when conducting investigations on new hyperparameter values every time. it can find hyperparameters of the model with high performance as a result because Because Bayesian optimization proceeds with the search based on the information obtained from the previous search. The method used to select hyperparameters is the TPE sampler[1], which uses this Bayesian approach. At every step, we build a probabilistic model of the function and select the most promising parameters in the next step.

2 Implementation

2.1 Base line

We will compare the performance of the base model and the upgraded models. First, we look at the results of the model with the weight decay set to 0.

DATASET(epochs)	Acc	Head Acc	Tail Acc
10-exp-0.1(90)	71.72	76.40	67.04
10-exp-0.01(90)	50.84	72.88	28.80
10-step-0.1(90)	68.02	85.38	50.66
10-step-0.01(200)	49.11	87.66	10.56
100-exp-0.1(200)	38.57	48.10	29.04
100-exp-0.01(200)	26.90	41.80	12.00
100-step-0.1(200)	37.98	56.44	19.52
100-step-0.01(200)	29.49	57.10	1.88

Table 1: Baseline

2.2 Optimized weight decay

"Optuna: hyperparameter optimization framework" was used to find weight decay. The training dataset of CIFAR10 and CIFAR100 was divided into a training set and a validation set, and the ratios were 80% and 20%, respectively. At this time, the validation set has the same transform as the test set. In the process of finding hyperparameters, the number of epochs was given as 90 or 200, and the value of weight decay was found between $1e^{-1}$ and $1e^{-5}$. The number of attempts to find the optimized weight decay was 10 to 12 times. The optimal WD is shown in Table 2.
Table 2 show that the accuracy decreased because only 80% of the data in the train dataset was used. This is a result that occurs because the number of tail classes is too small. Therefore, it may not be the optimal weight decay. However, the experiment is conducted using the obtained WD value. Additionally, an important conclusion that can be drawn from the difference between 'min Acc' and 'max Acc' is that if the weight decay is set incorrectly, the performance will drop sharply.

Table 3 show that the performance of the model trained by applying optimal weight decay. In all cases, the performance of the model was improved than base line. It can be seen that not only achieved the first goal of improving the performance of the tail class by using appropriate weight decay, but it also prevents overfitting of the head class and learns better.

DATASET	min Acc	max Acc	WD(epochs)
10-exp-0.1	31.9	67.6	0.000023(90)
10-exp-0.01	29.0	55.6	0.000047(90)
10-step-0.1	33.1	65.0	0.000052(90)
10-step-0.01	24.7	44.6	0.00288(200)
100-exp-0.1	1.1	10.6	0.001405(200)
100-exp-0.01	1.0	6.1	0.001043(200)
100-step-0.1	1.0	10.2	0.000479(200)
100-step-0.01	1.0	10.5	0.000670(200)

Table 2: DATASET representation: CIFAR10 = 10, CIFAR100 = 100, Data imbalance type and imbalance factor sequentially. min Acc, Max Acc: showed the lowest accuracy and the highest accuracy obtained by trials. WD: This is the weight decay of the model with the highest accuracy. If the integer values of accuracy are the same, the larger tail accuracy is selected.

DATASET(epochs)	Acc	Head Acc	Tail Acc
10-exp-0.1(90)	73.42	79.90	66.94
10-exp-0.01(90)	52.05	73.62	30.48
10-step-0.1(90)	69.88	85.74	54.02
10-step-0.01(200)	54.88	90.90	18.86
100-exp-0.1(200)	41.86	51.52	32.20
100-exp-0.01(200)	32.09	50.62	13.56
100-step-0.1(200)	43.56	63.20	23.92
100-step-0.01(200)	32.42	63.80	1.04

Table 3: Applying Weight decay

2.3 τ-normalization

We proceed with the task of finding an appropriate tau value for τ-normalization. Similar to finding weight decay, the training set and validation set are configured in a ratio of 8:2 using only the training dataset. When training the model, optimal WD is applied as a hyperparameter. After learning is finished, the performance of the model is checked for each class. The τ value is obtained through the average of the accuracy of the model classifying each class.

DATASET(epochs)	Acc	Head Acc	Tail Acc
10-exp-0.1(90)	73.47	74.76	72.18
10-exp-0.01(90)	55.03	71.54	38.52
10-step-0.1(90)	72.51	82.66	62.36
10-step-0.01(200)	54.42	87.66	21.82
100-exp-0.1(200)	40.04	43.90	36.18
100-exp-0.01(200)	27.34	31.62	23.06
100-step-0.1(200)	40.92	48.54	33.30
100-step-0.01(200)	22.18	19.58	24.78

Table 4: Baseline + τ-normalization

Table 3 shows the results of applying τ-normalization to the base model using the obtained tau value. It is not suitable τ value because the base model was trained with WD as 0. However, it shows a quite significant performance increase. Now, let's check the performance of the model with WD and τ-normalization applied at the same time.

2.4 τ-normalization + Optimized weight decay

Finally, the model was trained by applying weight decay, and the results of applying tau-normalization after learning were completed are shown in Table 5.

First, in the case of the model trained with DATASET "100-step-0.01", the accuracy fell when τ-normalization was applied. This is because the bias of the data is too large. In case of head class data, the number of trained input images is 500, and in case of tail class data, there are 5. Therefore, Head Acc and Tail Acc differ the most from the results of other DATASETs. Then, when τ-normalization is applied, the head acc is reduced too much, resulting in poor overall performance. However, in the model trained on other datasets, the overall performance was improved along with tail acc when τ normalization was applied.

DATASET(epochs)	Acc	Head Acc	Tail Acc
10-exp-0.1(90)	74.95	75.54	74.36
10-exp-0.01(90)	55.48	71.36	39.60
10-step-0.1(90)	73.89	82.48	65.30
10-step-0.01(200)	60.12	90.12	30.12
100-exp-0.1(200)	43.86	48.78	38.94
100-exp-0.01(200)	35.85	48.30	23.40
100-step-0.1(200)	47.81	57.86	37.76
100-step-0.01(200)	28.45	33.68	23.22

Table 5: Weight decay + τ-normalization

3 Conclusion

Of the many hyperparameters, only weight decay, which is likely to play an important role in the LTR problem, was adjusted. Looking at Table 3, we should realize that finding the appropriate hyperparameter is not a trivial task. Finding hyperparameters can be time consuming, but it's basically what we should do.

It is very important information that the norm of the classifier's weight is related to the number of samples belonging to the class. By properly flattening the weights following the long-tail distribution after training, the performance of the model can be significantly improved without additional training. In the experiment, τ values that were not properly obtained for some DATASETs may not have been used. However, seeing that the performance of the model increases, good performance can be found even by normalizing it to a value obtained by a method other than the τ-normalization.

AI is being used in various fields in real world. In addition to the recruitment system mentioned at the beginning, there is an AI that can physically harm people like an autonomous car. Various road features such as trees, crosswalks, trailers, people, cars, etc. are well set as commons class. However, there is a lack of learning about rare classes such as wild animals, cracks on the road, and trash cans, which can lead to serious accidents. It will be solved if we keep collecting data, but it is very unfortunate that we cannot utilize the already advanced technology. Therefore, Long Tail Recognition is one of the fields that should be well studied. We need to optimize not only weight decay but also all hyperparameters like learning rate, the number of epochs, momentum.

[1] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

[2] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[3] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. *arXiv preprint arXiv:1910.09217*, 2019.

[4] Hieu Pham, Zihang Dai, Qizhe Xie, and Quoc V Le. Meta pseudo labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11557–11568, 2021.

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[6] Changyao Tian, Wenhai Wang, Xizhou Zhu, Xiaogang Wang, Jifeng Dai, and Yu Qiao. Vl-ltr: Learning class-wise visual-linguistic representation for long-tailed visual recognition. *arXiv preprint arXiv:2111.13579*, 2021.

[7] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917*, 2022.