Large-scale Product Recognition with Loss-guided Data Refinement

Qixin Yan , Haoyu Xu , Kuangshi Zhang [*]

Tianjin University

{qxyan,xuhaoyu,zks}@tju.edu.cn

Abstract

AliProducts Challenge is a competition which aims to address imbalanced distribution and noisy labels in the large-scale recognition. This report includes exploratory data analysis, proposed training method and different kinds of tricks tried in the experiments. We propose a loss-guided refinement strategy to filter noisy data and handle long-tailed distribution simultaneously. We first train the model on the original training set until convergence. The images with low loss are less likely to be noisy samples. Base on this intuition, we select five samples in every class which have low loss and high quality. Then we fine-tune the model on this clean and balanced dataset. By exploiting this training procedure repeatedly, our model shows significant improvement on the test set. Through the combination of the proposed training method and some tricks in the experiments, we get a 0.0901 score of mean top-1 error and rank 4th in the leaderboard finally.

1. Introduction

The widespread use of computer vision methods in the retail industry has sparked a new set of challenges, such as the difficult task of correctly distinguishing between a large number of products, especially when the products are quite similar. Tackling this fine-grained classification task may lead to new break-throughs in the field of visual recognition and representation. AliProducts Challenge is a competition proposed for studying the large-scale and fine-grained commodity image recognition problem encountered by world-leading e-commerce companies. The challenge is based on a dataset released by Alibaba, which contains near 3 million images and covers 50 thousands SKU(Stock Keeping Unit) level commodity categories. It is the first time that an industry scale and granularity dataset becomes available for the computer vision research community to understand and help tackle the challenge in real-world commodity image recognition.

Because of the brand new dataset which was just released, the main challenges in this competition are summarized as follows:

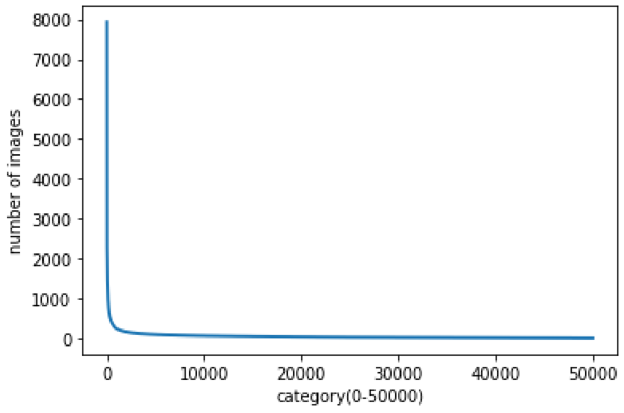

Figure 1. The frequency distribution of categories

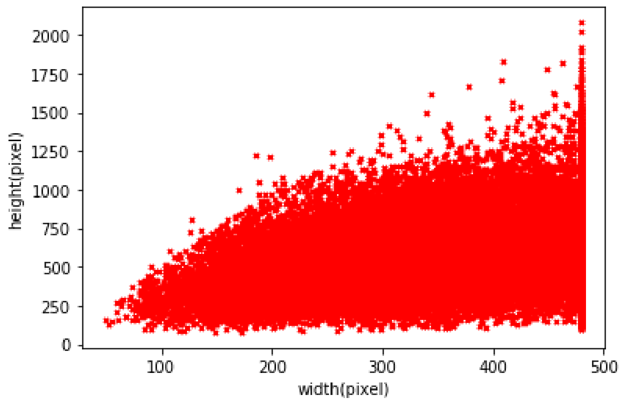

Figure 2. Different scale variance in the training set

[*]All authors contributed equally to this work.

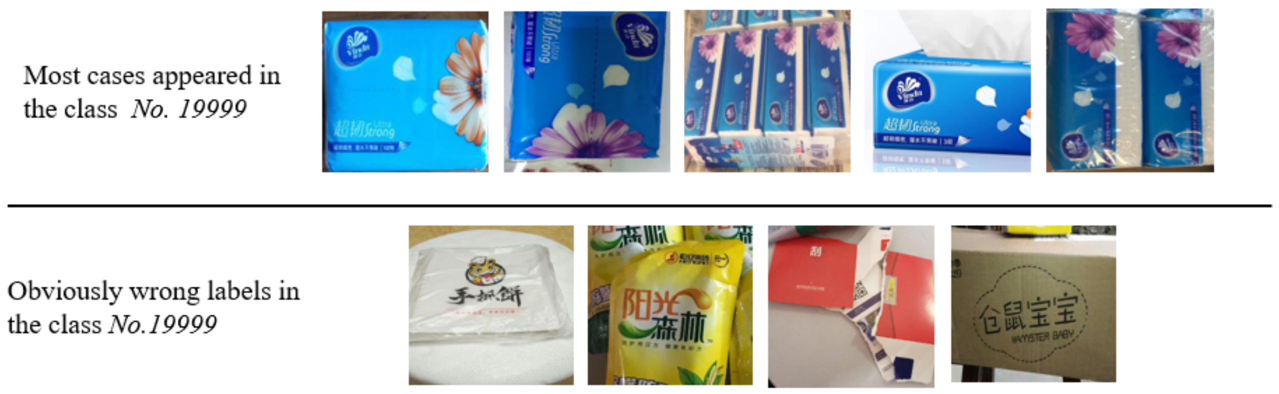
Most cases appeared in the class *No. 19999*
Obviously wrong labels in the class *No.19999*

Figure 3. An example of noisy labels in the training data

	The number of images	The number of classes
Training set	2,552,385	50,030
Validation set	148,387	50,030
Test set	250,130	50,030

Table 1. The number of classes and images in this data

- Imbalanced data. As shown in Figure 1, the frequency distribution of categories in this dataset is long-tailed, with a few common classes and many more rare classes.

- A very large number of classes and images in the data. As can be seen from Table 1,it is a very large-scale database.

- Labels are not fully reliable(take Figure3 as an examaple). The performance of deep neural network suffers from noisy labels in training data.Noisy labels refer to labels which are assigned to wrong classes in supervised learning.

- Other general problems such as different scale variance(as shown in Figure 2), scene complexity, etc.

2. Method

We propose a simple but effective approach to address the noisy data cleaning and class imbalance in the large-scale data recognition. Similarly to Huang *et al.*'s work[2], it does not require specifically designed noise-robust loss functions or networks. In our loss-guided training strategy, samples with noisy labels will be removed repeatedly and get a clean and balanced-class dataset. Then we will refine the classifier in the clean dataset to improve the performance. To train our final model, a 3-stage procedure (as shown in Figure 4) is introduced as follows:

1. **Pre-training:** Firstly, the model is loaded with ImageNet pre-trained weights and then trained on the original training set including noisy labels. At this step, data augmentation and other tricks such as label smooth are not employed. The network is trained until convergence.

2. **Data Refinement:** At this step, loss-guided data refinement is performed using the model pre-trained in step 1 to sample a balanced and relatively clean dataset from original noisy training set.

3. **Clean Data Fine-tuning:** Finally, we fine-tune the model pre-trained in step 1 on the dataset introduced in step 2 with label smooth and data augmentation. We use the validation set to monitor the performance of training. The network is trained until the accuracy in the validation set stays stable.

We first train the model on the original training set until convergence. Then we fine-tune the model with all the data in training set. During the training process, we record the average loss of each picture. After that, we sort the pictures in the same category in ascending order according to the average loss. We believe that in the same category, pictures with low loss are less likely to be noisy samples. We filter out all samples with average loss greater than 9, these samples account for 3% of the training set. These data have a high probability of being noisy data. Then we filter out the top 5 images with the smallest loss in each category. If the number of samples corresponding to this category is less than 5, it will not be filtered. These pictures are good samples that can be trusted, suitable for use in the finetune model. Considering that the samples with lowest loss may

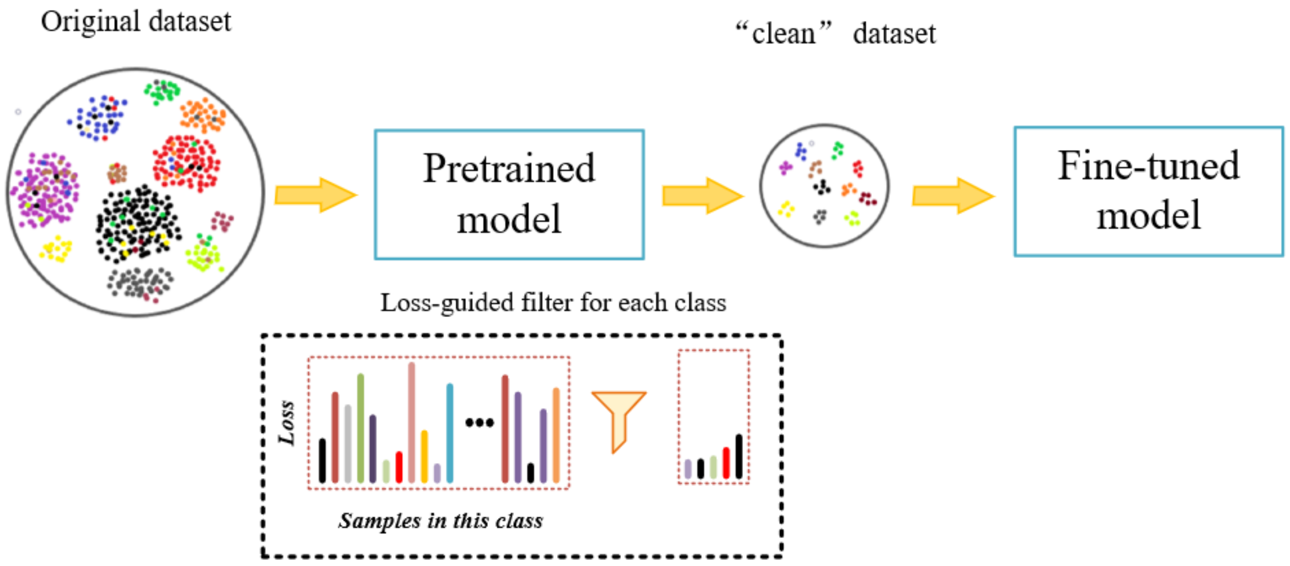

Figure 4. loss-guided filter to clean and balance data.

be too simple to improve the model's performance on photos with poor angle lighting, we produced filtered training data with the samples loss ranked 6 to 10 and 11 to 15. We also use 5 samples which loss ranked in the middle of the class to make a filtered training data. In the process of making the train list ranked 6 to 10 and ranked 11 to 15, our strategy is that if the losses of the 5 images is greater than the losses of median 5 images of the category, we will instead to choose the median 5 images.Through this strategy, we can ensure that the quality of the pictures we filtered is better than the average level of the class.

3. Experiments

We developed our models based on Pytorch[5] library. We use Ranger(RAdam[3] + Lookahead[7]) optimizer to train out network with batchsize of 24. Initial learning rate is set to 1e-4 which is gradually decreased to 1e-5 following the cosine scheduler. A number of adjustments and tricks are conducted in our experiments to improve the accuracy.

3.1. Image Preprocessing Pipeline

During training we roughly follow the strategy in [1]. The input pipeline performs the following steps on-by-one:

1. Randomly sample an image from training set.
2. Resize the image to 256×256.
3. Flip the image with 0.5 probability.
4. Normalize RGB channels to mean value of 123.68, 116.779, 103.939 and standard deviation of 58.393, 57.12, 57.375, respectively.

3.2. Backbone Selection

We have tried ResNet101, ResNeXt101, ResNeSt101, Res2Net101, iResNet101 and EfficientNet-b7 as our backbone respectively, finding that ResNeSt[6] outperforms other networks by a large margin both in training and validation dataset.

3.3. Label Smoothing

Label smoothing can greatly increase performance in this task for the fact that many of the images in training set are mislabelled. Traditional one-hot encoding encourages the output scores dramatically distinctive which potentially leads to overfitting. This problem can be even worse when training set contains a lot of noise. Attempting to fit random wrong labels in different mini-batches may result in non-convergence of the network.

In contrast, label smoothing alleviates the problem by changing the construction of the true probability to

$$q_i = \begin{cases} 1-\varepsilon & \text{if } i = y \\ \varepsilon/(K-1) & \text{otherwise} \end{cases} \tag{1}$$

where y is the true label, K is the number of classes, and ε is a small constant. We set $\varepsilon = 0.2$ empirically which can achieve best convergence in our experiments.

3.4. Cosine Learning Rate Decay

Cosine learning rate decay[4] decreases the learning rate from the initial value to the minimum value by following the cosine function. Denote the total number of batches as T, then at batch t, the learning rate η_t is computed as:

$$\eta_t = \frac{1}{2}\left(1 + \cos\left(\frac{t\pi}{T}\right)\right)(\eta_0 - \eta_{min}) + \eta_{min} \quad (2)$$

where η_0 is the initial learning rate and η_{min} is the minimum learning rate. We set $\eta_{min} = \frac{1}{10}\eta_0$ in our experiments.

The cosine decay decreases the learning rate slowly at the beginning, and then nearly linear in the middle, and slows down again at the end. Compared to the traditional step decay method, the cosine scheduler decays the learning rate more smoothly, which can potentially improve the training progress.

3.5. Model Ensemble

We trained 3 base models with mean top-1 error around 0.11 on validation set . Due to the huge number of classes, model ensemble by saving output(50030 dims) is unrealistic. The output of all samples in Test dataset take up 100GB space(250130*50030 float). Instead of saving the output, we choose to save the feature before the final fc (fully-connected) layer. Since the dimension of this feature is only 2048, we save nearly 50 times space compared to saving the output directly. In the merging process, we load the fc layer for each model, and only use the feature and fc layer to calculate the output. This method can merge 16 models on one GTX1080ti GPU with 11GB memory.

3.6. Test Time Augmentation

We also tried the Test Time Augmentation(TTA) trick. Our model ensemble is based on saving features. So we should calculate the features with different transforms of TTA. Because we don't set the activation function on the final output, the part from the feature to the final output is a linear model. So we only need to accumulate the features of TTA by adding directly. In that case, the inference of fc layer will only need to be run once for each image. This is another benefit of model ensemble based on saving features. In the experiment, we analyze the direction of the picture will also provide useful information. So we did not use upside-down-like transform in the data augmentation. In the end, we found that the results of only TTA by horizontally fliping is better than using all eight transformations TTA, which is as expected.

3.7. Submission Entries

Our submission entries which improve the online score are summarized in Table 2. As we can see, changing a stronger backbone boost the performance from 0.37 to 0.33, Ranger optimizer brings 0.04 points gain as well. Loss-guided data refinement, which is our main contribution, greatly improves the performance by ∼0.08 on the test set. Training tricks such as label smoothing and data augmentation also work in our experiments. At our last entry, 3 independent models with different random seeds are trained and then model ensemble is employed, as our final submission.

Entry	Top1-Err
ResNet101+Adam	∼0.37
ResNeSt101+Adam	∼0.33
ResNet101+Ranger	∼0.29
+ Loss-guided data refinement	∼0.21
+ Label smoothing / data augmentation	∼0.14
+ Loss-guided data refinement again	∼0.11
+ 3 models ensemble	∼0.09

Table 2. Our submission entries.

4. Conclusion

In this report, our team's solution to the task of CVPR2020 AliProducts Challenge is described. Experimental results have validated the effectiveness of the proposed data refinement method. With the help of the cleaned data, the accuracy in test set is greatly increased. After model ensemble, our team finally obtains an average top-1 error of 0.0901 on the AliProducts test set and rank *4th* in the leaderboard.

References

[1] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[2] Jinchi Huang, Lie Qu, Rongfei Jia, and Binqiang Zhao. O2u-net: A simple noisy label detection approach for deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3326–3334, 2019.

[3] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *ArXiv*, preprint arXiv/1908.03265, 2020.

[4] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

[5] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, and Luca and Antiga. Pytorch: An imperative style, high-performance deep learning library. 2019.

[6] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Mueller, R. Manmatha, Mu Li, and Alexander Smola. Resnest: Split-attention networks. *ArXiv*, preprint arXiv/2004.08955, 2020.

[7] Michael Ruogu Zhang, James Lucas, Geoffrey E. Hinton, and Jimmy Ba. Lookahead optimizer: k steps forward, 1 step back. In *NeurIPS*, 2019.