Alibaba Cloud German AI Challenge 2018

Project Report

Team Name: What a Sad Life (那真是凄凉的人生)
Semi-Final Rank: 3rd
Members: Ge Liu, Sheng Yang, Luxiong Li, Ziqiang Pei, Qing Li
Huazhong University of Science and Technology

1. Development Tools

1.1 Hardware Information

- GPU: 1080Ti * 3, Alibaba Cloud PAI Platform M40 * 2
- CPU: Intel i7 7700
- RAM: 32G

1.2 System Configuration

- Ubuntu 16.04
- Python 3.6
- Anaconda 3
- Pytorch 1.0.0
- CUDA 9.2

1.3 Optimization Tool

Adam, SGD, etc.

2. Solution Description

Providing datasets consisting of radar and multispectral sensor data of 42 cities, the task of this competition is to perform LCZ classification. The aim of the contest is the transferability of algorithms to different cultural regions and SAR and optical data fusion strategies. So the key of the solution should be transferring between different datasets with variance and data process techniques.

Our team consists of two original teams from the qualification phase, so we have two sets of codes from both of the original teams. In the rest of this report, **setting 1** and **setting 2** will be used to refer the two sets of our codes. Specifically, **setting 1** is developed by Sheng Yang, Ziqiang Pei and Qing Li, and **setting 2** is developed by Ge Liu and Luxiong Li. In the semi-final phase, the results of both settings are ensembled to generate the final submission.

We can get best accuracy by ensemble for about *20 min* (on 1080Ti, test set), but can also get quite good performance using single model within seconds. But we didn't do further experiments on ensembling subset of our models to shorten inference time (due to submission limitation on Tianchi).

2.1 Setting 1

Training Set:

There are 352,366 images in official training set, and 24,119 images in official validation set. The validation set shares the same distribution with test set but the training set's distribution is quite different (showed in Figure 1). We did some ablation experiments. If training with original training set, we only receive a precision of 0.681 on the test set. However, only using the official validation set for training, our model can achieve accuracy of 0.850 without any tricks.

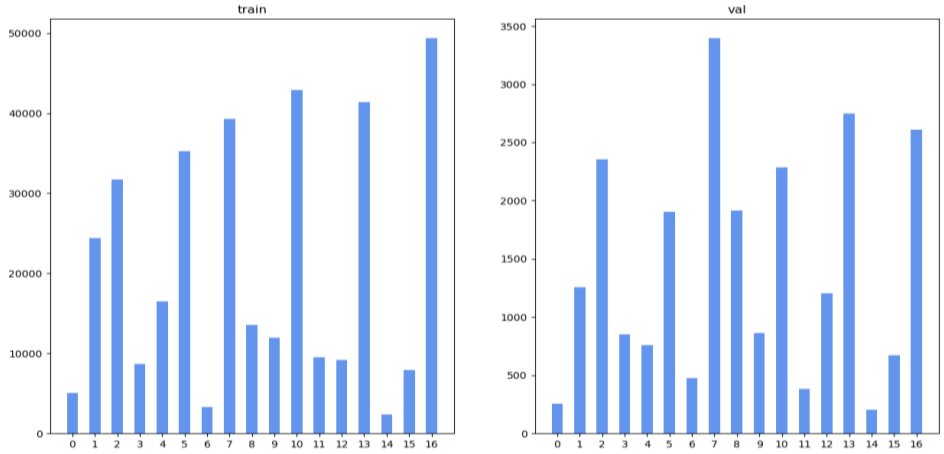

Figure 1 different class distributions between training and validation set

And we also found that the class distribution of the dataset is extremely unbalanced. In order to take advantage of both the original training and validation set, we recreate the training set called *New 50K* (3000×17≈51000) training set. We undersampled the original training set and oversampled the original validation set so that the number of samples in each class is about 3000.

Training Set	top-1
Orininal Training Set	0.681
Original Validation Set	0.850
Our *New 50K* Training Set	0.871

Table 1 performance of same model in different training set

Preprocess: Z-score, subtract per-pixel mean divided by std

Augmentation: Following augmentation techniques are performed in **setting 1** during training:

- Random H/V Flip
- Random Rotate
- Random Crop

- Mixup training[3][9]

Refinement	Top-1
standard	0.858
+H/V Flip and + Rotate	0.865
+Random Crop	0.868
+Mixup training	0.873
+Knowledge Distillation	0.884

Table 2 the validation accuracies for staking refinements one by one.

Mixup is a simple and data-agnostic data augmentation routine

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j, \quad \text{where } x_i, x_j \text{ are raw input vectors}$$

$$\tilde{y} = \lambda y_i + (1-\lambda)y_j, \quad \text{where } y_i, y_j \text{ are onehot label encodings}$$

$(x_i, y_i)$and $(x_j, y_j)$are two examples drawn at random from our training data, and $\lambda \in [0,1]$. where $\lambda \in [0,1]$ is a random number drawn from the Beta (α,α) distribution. Mixup can be implemented in a few lines of code, and introduces minimal computation overhead. In mixup training, we only use the new example $(\tilde{x}, \tilde{y})$.

Networks: In this setting, following CNN networks are mainly used:

- SENet-50[8]
- Densenet-101 (k=12)[6]
- Xception[7]
- CBAM-50[2]
- ResNext-29, 32×4d

The network inputs are 32×32 images, with the per-pixed mean subtracted. The first layer is 3×3 convolutions. Then we use a stack of 6n layers with 3×3 convolutions on the feature maps of sizes {32, 16, 8} respectively, with 2n layers for each feature map size. The numbers of filters are {64, 128, 256} respectively. The subsampling is performed by convolution with a stride of 2. The network ends with a global average, a 17-way fully connected layer, and softmax. There are totally 6n+2 stacked weighted layers. The following table summarizes the architecture.

Output map size	32×32	16×16	8×8
#layers	1+2n	2n	2n
#filters	64	128	256

Table 3 model architecture summarization

We didn't resize our images to 128×128 or 256×256 like the ImageNet dataset. Although enlarging the image is usually good for CNN models, it brings the cost of the memory and the inference time. It multiplies the FLOPS and inference time with N square times. And in the same time, we didn't choose the Inception Network and its variants. Despite good accuracy in the ImageNet dataset, the realization of Inception models has been accompanied with a series of complicating factors. Although careful combinations of these components yield excellent neural network recipes, it is in general unclear how to adapt the Inception architectures to our new

datasets, especially when there are many factors and hyper-parameters to be designed.

We mainly choose the variants of ResNet, such as ResNeXt, SENet and etc. We redesigned these classical networks to fit the input of the satellite images with 32×32 pixels.

Model	Params	inference time	top-1
ResNeXt-29, 32×4d	18M	2.19s (0.45ms/image)	0.867
Xception	41M	5.62s (1.16ms/image)	0.869
SENet-50	98M	9.21s (1.90ms/image)	0.869
Densenet-101 （k=12）	3M	1.94s (0.40ms/image)	0.871
CBAM-50	72M	5.33s (1.10ms/image)	0.873

Table 4 model performance (note that the inference time in table is time for prediction all 4835 images in test set)

Implementation Details:

- Optimization configuration
  - Optimizer: SGD
  - Momentum: 0.9
  - Initial learning rate: 0.025
  - Batch Size: 32
  - Weight decay:0.0005
  - Learning rate decay: Multistep decay:

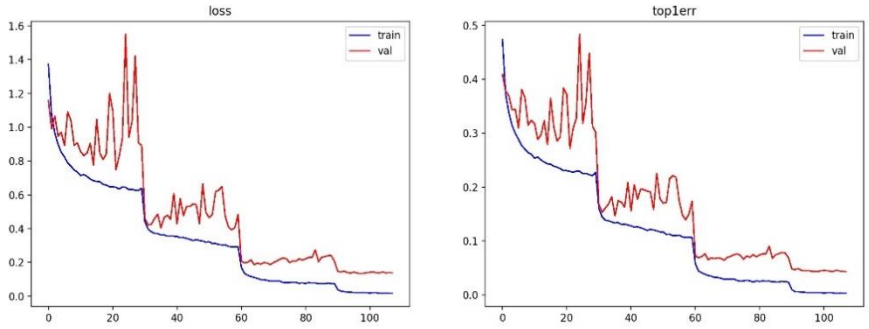

Figure 2 multistep decay

We train the models on our 50k training set. The input image is 32×32 randomly cropped from a zero-padded 40×40 image.

Specifically, the learning rate is set to 0.025 with a minibatch size of 32, and it is then divided by 10 after 30, 60 and 80 epochs when training for 90 epochs; or after 60, 120 and 180 epochs when training for 200 epochs.

In the mixup training, we choose $\alpha = 1$ in the Beta distribution and increase the number of epochs from 90 to 200 because the mixed examples ask for a longer training progress to converge better. In the semi-final, all models are trained on a single Nvidia GTX 1080Ti GPU using PyTorch for 200 epochs on the training set with 32 examples per minibatch, and evaluated on the test set. Learning rates start at 0.025 and are divided by 10 after 60 ,120 and 180 epochs for all models

In test time, data augmentations are performed the same way as training time, the evaluation is performed several times to get average prediction results.

2.2 Setting 2

Training Set: Because of the variance between training set and validation/test set, there should be a balance between generalization and transferability. The validation set are repeatedly sampled during training to have *# training : # validation = 1 ： 1* in every batch, which helps models to fit well on both training and validation set without overfitting any of it.

Training Set	top-1(on test1a)
Orininal Training Set	0.713
Original Validation Set	0.833
Batch Sampled Training Set	0.850

Table 5 top-1 accuracies of same model on test1a using different training sets

Preprocess: Original images of 18 channels are processed to have 26 channels.

- For channels of sentinel-1, absolute value of amplitude of VH, VV, lee filtered VH, lee filtered VV signals are used, and then normalized to lie between 0 and 1 using the formula:

$$\mathbf{A_{norm}} = \mathbf{1} - \boldsymbol{e}^{-|\boldsymbol{A}|}$$

- For channels of sentinel-2, except for original bands, following hand-featured spectral indices[1] calculated from original sentinel-2 bands are added as new channels:

  SAVI (Soil Adjusted Vegetation Index)
  PSSR (Pigment Specific Simple Ratio)
  NDVI (Normalized Difference Vegetation Index)
  UCNDVI (Uncertainty of NDVI)
  NDWI (Normalized Difference Water Index)
  MSI (Moisture Stress Index)
  NBR (Normalized Burn Ratio)
  MCARI (Modified Chlorophyll Absorption in Reflectance Index)
  GNDVI (Green Normalized Difference Vegetation Index)
  CHL-RED (Chlorophyll Red-Edge)

Note: The input image is **not resized** to larger sizes, resulting in **faster training and predicting**.

Augmentation: Following augmentation techniques are performed in **setting 2** during training:

- Random H/V Flip
- Random Rotate

Networks: In this setting, following CNN networks are mainly used:

[1] https://www.sentinel-hub.com/develop/documentation/eo_products/Sentinel2EOproducts

- GACNet (a Grouped Attention CNN designed by ourselves)
- ResNet-10 (with CBAM[2])
- ResNet-18 (with CBAM[2])
- ResNext-29
- SE-ResNet (not used in final submission)
- DenseNet (not used in final submission)
- Xception (not used in final submission)

Note: All the input channel numbers and input sizes of those networks are modified to fit the 32*32*26 input data, so pre-trained models are not used. And some networks are not used in final submission because of training time limitation.

Details: Some details of training/testing tricks in **setting 2** are as follows:

- **Snapshot Ensembles: Train 1, Get M for Free[1]**

  An aggressive annealing schedule is combined with periodic "restarts" to the original starting learning rate.

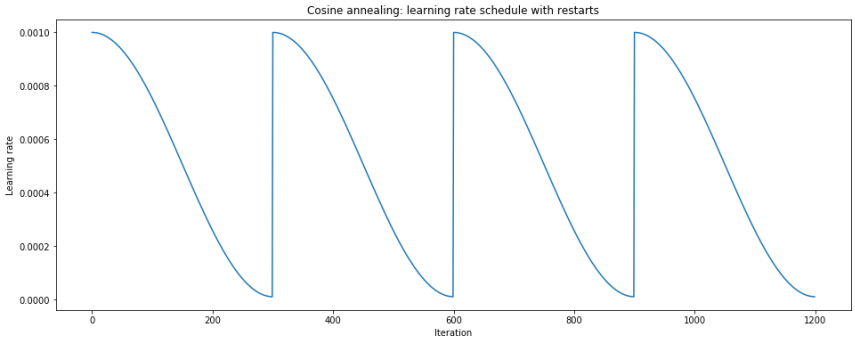

Figure 3: cyclic lr schedule

This schedule can be written as:

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})\left(1 + \cos\left(\frac{T_{current}}{T}\pi\right)\right)$$

Where η_t is the learning rate at step t (incremented each mini batch), η_{min} and η_{max} define the range of desired learning rates, $T_{current}$ represents the number of epochs since the last restart (this value is calculated at every iteration and thus can take on fractional values), and T defines the number of epochs in a cycle.

By drastically increasing the learning rate at each restart, the network can essentially exit a local minima and continue exploring the loss landscape. And by snapshotting the weights at the end of each cycle, ensemble of models can be built at the cost of training a single model. This is because the network "settles" on various local optima from cycle to cycle, as shown in the figure below.

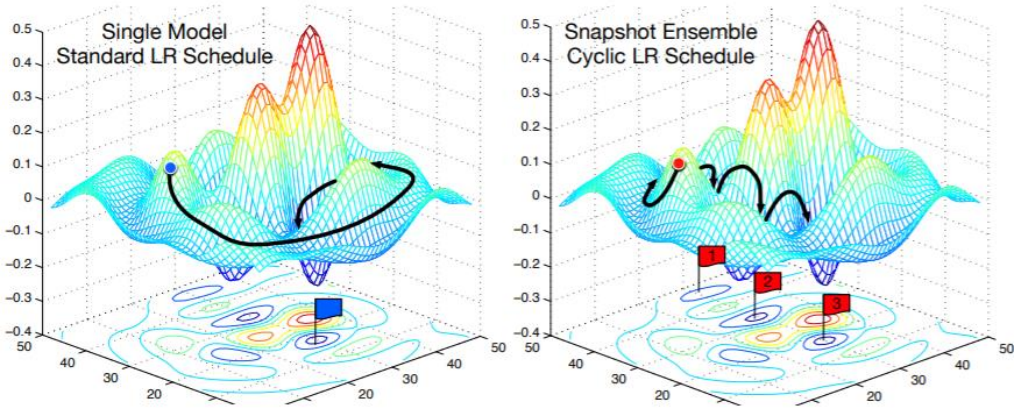

Figure 4: **Left**: Standard lr schedule. **Right**: Snapshots generated from cyclic lr schedule

- **Classification Focal Loss**

  Focal loss gives the harder examples bigger weights to make the network focus more on hard examples, less on easy ones. The loss can be described as:

  $$\mathbf{L}(\mathbf{y}, \mathbf{p}) = |\boldsymbol{y} - \boldsymbol{p}|^{\boldsymbol{\lambda}}(-\boldsymbol{y}\boldsymbol{log}(\boldsymbol{p}))$$

  Where $\mathbf{y}$ is the ground truth label and $\mathbf{p}$ is probability predicted by network, $\boldsymbol{\lambda}$ is a tunable parameter, in this setting, $\boldsymbol{\lambda} = \mathbf{2}$.

- **Semi-supervised Knowledge Distillation on Test Sets** 错误!未找到引用源。

  Knowing that the test sets and validation sets are more similar in distribution, and have variance with training set, the test sets in each competition phase become precious resources.

  To make full use of provided test sets, a semi-supervised knowledge distillation method is performed during semi-finals. Ensemble probability predictions on the 4 test sets (*test1a*, *test1b*, *test2a* and *test2b*) of our **setting 1** and **setting 2** models are used as fake labels of test data to train new single models.

  Specifically, when training the final model, the real training set is the combination of original training set, validation set, and fake labeled test sets. In **setting 2**, the real training set are sampled to have *# training : # validation : # test1a : # test1b : # test 2a : # test2b = 4 : 4 : 1 : 1 : 1 : 1* in every mini batch.

  This method provided huge improvements on single model performance (from 0.859 to 0.879 **single model**), thus can be used for speeding up model prediction when deployed in practice.

Refinement	Top-1(on test2a)
standard	0.850
+Focal Loss	0.855
+Snapshot Ensemble	0.859
+Knowledge Distillation	0.879

Table 6 the test2a accuracies for staking refinements one by one

2.3 Ensemble

We perform ensemble on both **setting 1** and **setting 2** output probabilities. Average ensemble all model results in **setting 1** and **setting 2** separately to get **setting 1 ensemble** and **setting 2 ensemble**, and then we average them to get **ensemble 1**. And we get another ensemble output, **ensemble 2**, by directly ensemble all the output probabilities of **setting 1** and **setting 2.** The final submission is the average of **ensemble 1** and **ensemble 2**.

ensemble	Top-1(on test2a)
Setting 1 ensemble	0.887
Setting 2 ensemble	0.886
Ensemble 1	0.888
Ensemble 2	0.888
Final ensemble	0.890

Table 7 different ensemble method accuracies on test2a

2.4 Inference time

By using knowledge distillation, our single model can get 0.884 top-1 accuracy on test2a and 0.888 top-1 accuracy on test2b **in 2 mins** with a single Nvidia GTX 1080Ti GPU. Meanwhile, our ensemble model can get 0.892 top-1 accuracy on test2b **in 20 mins**.

Model	Top-1	Inference Time
Single Model(without distillation)	0.873	<2mins
Single Model(with distillation)	0.888	<2mins
Ensemble Model	0.892	<20mins

Table 8 inference time in different models

1. Team Info

Ge Liu

M.S. School of Computer Science & technology, Huazhong University of Science and Technology

B.S. College of Computer Science, Chonqing University

E-mail: liuge1229@foxmail.com

Phone: (+86) 13212761229

Sheng Yang

M.S. School of Artificial Intelligence and Automation, Huazhong University of Science and Technology

B.S. School of Aeronautics, Northwestern Polytechnical University

E-mail: yangsheng_95@foxmail.com

Phone: (+86) 15927303165

Luxiong Li

M.S. School of Computer Science & technology, Huazhong University of Science and Technology

B.S. College of Computer Science, Chonqing University

E-mail: m13368289607_1@163.com

Phone: (+86) 13368289607

Ziqiang Pei

M.S. School of Artificial Intelligence and Automation, Huazhong University of Science and Technology

B.S. College of Physics, South West University

E-mail: dfzsqzq@163.com

Phone: (+86) 15927185273

Qing Li

M.S. School of Aeronautics, Northwestern Polytechnical University

B.S. School of Aeronautics, Northwestern Polytechnical University

Phone: (+86) 15927185

2. Conclusions

After three months of hard work, we finally got the chance to compete in the final. Although the task of this competition is the basic image classification, the data situation is closer to the real scene: the distribution of training set and the test/validation set are different, and the category ratio is not balanced, which tests our ability to analyze and solve problems. And the other teams involved in the competition are also very good, which makes us continue to improve. During the preliminary team formation, we also met teammates from different colleges in the same university. The whole game was very rewarding and we are very grateful to the organizers for this great opportunity.

Reference

[1] Huang G, Li Y, Pleiss G, et al. Snapshot Ensembles: Train 1, get M for free[J]. 2017.

[2] Woo S, Park J, Lee J Y, et al. CBAM: Convolutional Block Attention Module[J]. 2018.

[3] He T , Zhang Z , Zhang H , et al. Bag of Tricks for Image Classification with Convolutional Neural Networks[J]. 2018.

[4] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. of Computer Vision and Pattern Recognition (CVPR). (2016)

[5] Xie,S.,Girshick,R.,Dollár,P.,Tu,Z.,He,K.: Aggregated residual transformations for deep neural networks. arXiv preprint arXiv:1611.05431 (2016)

[6] Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L.: Densely connected convolutional networks. arXiv preprint

arXiv:1608.06993 (2016)

[7] Chollet, F.: Xception: Deep learning with depthwise separable convolutions. arXiv preprint arXiv:1610.02357 (2016)

[8] Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. arXiv preprint arXiv:1709.01507 (2017)

[9] H. Zhang, M. Ciss´e, Y. N. Dauphin, and D. LopezPaz. mixup: Beyond empirical risk minimization. CoRR, abs/1710.09412, 2017.