A Technical Report for 2021 VIPriors Image Classification Challenge

Jiahao Wang, Hao Wang, Yifei Chen, Yanbiao Ma, Fang Liu, Licheng Jiao
School of Artificial Intelligence
Xidian University, Xi'an, China
Email: 18391763127@163.com

***Abstract*—Training a model from scratch in a data-deficient environment is a challenging task.In this challenge, we use multiple differentiated backbones and multiple losses for training, and use a number of tricks to assist in model training, such as initializing weights, warm up, mixup[1], and cutmix[2]. finally, we propose a two-stage model fusion and a helpful strategy (result coverage) to improve our accuracy. Our final accuracy of Top-1 on the test set is 75.178%.**

***Index Terms*—multiple model,model fusion,cutmix**

I. Introduction

The VIPriors Image Classification Challenge is one of "Visual Inductive Priors for Data-Efficient Computer Vision" challenges. The main goal of the challenge is to obtain the highest Top-1 accuracy on a small sample of the classification dataset.The training data and validation data are two subsets of the training portion of Imagenet 2012. The test set is taken from the validation set of the Imagenet 2012 dataset. The final data obtained consists of 50,000 training sets, 50,000 validation sets and 50,000 test sets with a total of one thousand categories, each containing 50 images.

In recent years, a lot of work has achieved excellent results on the Imagenet dataset. The seminal ResNet[3] models introduced in 2016, revolutionized the world of deep learning. ResNeXt[4] adopts group convolution in the ResNet bottleblock, which converts the multi-path structure into a unified operation. SE-Net[5] introduces a channel-attention mechanism by adaptively recalibrating the channel feature responses. ResNeSt[6] introduces a Split-Attention block that enables attention across feature-map groups. TResNet[7] introduces a series of architecture modifications that aim to boost neural networks' accuracy while retaining their GPU training and inference efficiency. The above work has inspired us a lot in this Challenge.

Compared to the Imagenet dataset, which has over 14 million images, the number of images in this challenge is much smaller. In our approach, based on multipe differentiated backbones network, we use multiple loss functions, data enhancement strategies and model integration to improve the classification performance.

II. Method

In this section, the method used to find an answer to the research questions should be presented.

A. Model Architecture

We have selected a few models that we used in this competition to introduce.

1) ResNest: ResNest[6] demonstrates a simple module: Split-Attention, a block that enables attention across feature maps. By stacking these Split-Attention blocks in ResNet style, a new ResNet variant called ResNest is obtained. ResNest retains the full ResNet structure and can be used directly for downstream tasks (e.g., target detection and semantic segmentation) without incurring additional computational costs.

2) TResNet: TResNet[7] is designed based on the ResNet50 architecture with special modifications and optimizations, and contains three variants, TResNet-M, TResNet-L and TResNet-XL. Only the model depth and number of channels differ between variants.Figure 1. shows TResNet BasicBlock and Bottleneck design.

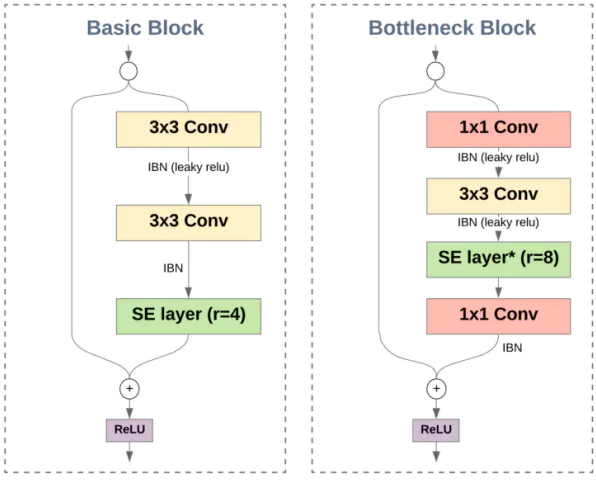

Figure 1. TResNet BasicBlock and Bottleneck design.

3) ReXNet: The ReXNet[8] network architecture improves on the expression bottleneck: expanding the number of input channels of the convolutional layers; replacing the ReLU6 activation function; and designing more expansion layers, i.e., reducing the rank before and after each expansion. The model gets a significant performance improvement.

4) Inception-ResNet: The Inception-ResNet[9] network is the residual structure of ResNet introduced in the Inception

module, which has two versions, where Inception-ResNet-v1 is aligned to Inception-v3, while Inception-ResNet-v2 is aligned to Inception-v4, both of which have similar computational complexity respectively.The Inception-ResNet network structure is shown in Figure 2, and the two figures on the right are the stem module structures of the Inception-ResNet-v1 and Inception-ResNet-v2 networks, respectively.

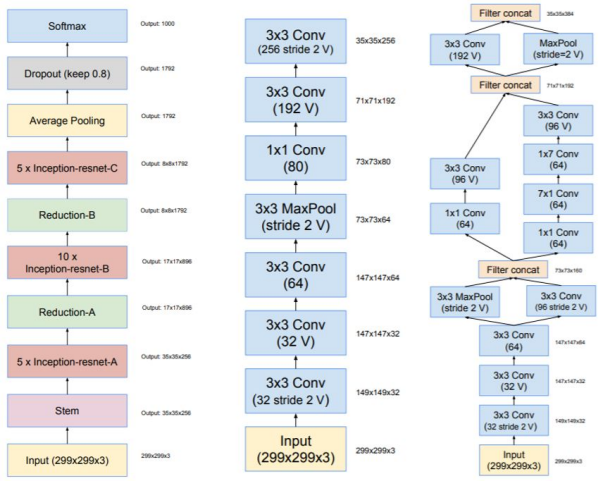

Figure 2. Inception-ResNet network structure and stem module.

5) RegNet: RegNet networks combine the advantages of manual design networks and neural network search (NAS). In this network, the concept of constructing network design space is used. Design Space The design process is to design a progressively simplified version of the original design space that is not restricted. The initial design space for RegNet was AnyNet.

B. Loss functions

1) Label Smoothing: The idea of label smoothing was first proposed to train Inception-v2.Label smoothing is a mechanism of regularize the classifier layer by estimating the marginalized effect of label-dropout during training. It changes 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 ε is a small constant. Now the optimal solution becomes

$$z_i^* = \begin{cases} \log((K-1)(1-\varepsilon)/\varepsilon) + \alpha & \text{if } i = y \\ \alpha & \text{otherwise} \end{cases} \tag{2}$$

where α can be an arbitrary real number. This encourages a finite output from the fully-connected layer and can generalize better.

2) Taylor Cross Entropy Loss: In order to to improve the robustness of the model to label noise, we use a general robust learning framework to train deep models in the presence of label noise. The loss are calculated as follows.

$$\mathcal{L}_{t-\mathrm{CE}}(f(\boldsymbol{x}), y) = \sum_{i=1}^{t} \frac{\left(1 - f_y(\boldsymbol{x})\right)^i}{i} \tag{3}$$

3) Bi-Tempered Logistic Loss : Bi-Tempered Logistic Loss loss is proposed to reduce the sensitivity of Logistic Loss to outliers, increase the discrimination and contribution of correct samples, and mitigate the impact of mislabeling. This loss is calculated as follows.

$$\log_t(x) := \frac{1}{1-t}\left(x^{1-t} - 1\right) \tag{4}$$

$$\exp_t(x) := [1 + (1-t)x]_+^{1/(1-t)} \tag{5}$$

C. Data Augmentation

Data augmentation is a commonly used method to improve model performance in deep learning, mainly used to increase the training data set and improve the generalization ability of the model, the data augmentation methods we used is described as follows.

1) Auto Augmentation: Auto-Augmen[10] is a strategy that augments the training data with transformed images, where the transformations are learned adaptively. A search which tries various candidate augmentation policies returns the best 25 best combinations. One of these 25 policies is then randomly chosen and applied to each sample image during training.

2) mixup: Here is another augmentation method called mixup[1]. In mixup, each time we randomly sample two examples (xi, yi) and (xj, yj). Then we form a new example by a weighted linear interpolation of these two examples:

$$\hat{x} = \lambda x_i + (1-\lambda)x_j \hat{y} = \lambda y_i + (1-\lambda)y_j \tag{6}$$

where $\alpha \in [0, 1]$is a random number. In mixup training, we only use the new example $(\hat{x}, \hat{y})$.

3) cutmix: Cutmix[2] is a data enhancement strategy that randomly fills a portion of the image with regional pixel values of other data in the training set, and the results of the classification are assigned in proportion to the area of the patch. The original image and the image after Cutmix processing are shown as Figure 3.

Figure 3. The Original images and the image after cutmix.

D. Proposed method

We propose dynamic semantic scale balance loss (DSB loss) and have submitted it to AAAI2022.Unlike sample size imbalance, we focus on semantic scale imbalance between categories, and define and quantify it, achieving significant performance gains on datasets such as mini-imagenet, CIFAR100, CUB2011, and Cars196.In this paper, we propose a loss function called $DSB_{Focalloss}$ for training models by improving Focal Loss[11].

$Donatep_i^t = sigmoid(z_i^t) = 1/(1 + exp(-z_i^t))$, Focal loss is defined as:

$$FL(\mathbf{z}, y) = -\sum_{i=1}^{C} (1 - p_i^t)^{\gamma} log(p_i^t) \quad (7)$$

$DSB_{Focalloss}$ is defined as:

$$\text{DSB}_{\text{ Focal loss }}(Z, y_i) = -\frac{1}{S_i} \sum_{i=1}^{C} \left(1 - p_i^t\right)^{\gamma} \log\left(p_i^t\right). \quad (8)$$

where S_i represents the semantic scale of category i. We define the feature volume for measuring the semantic scale of each category, and due to the principle of double-blind review, we cannot release the specific details and code of g the method for the time being.

III. Experiment

A. Data Preprocessing

Data preprocessing can be very effective in reducing the risk of model overfitting and can help a lot in improving the robustness of the model. We use the following strategy on the training set in addition to the previously introduced Data Augmentation:

1) : Randomly crop a rectangular region whose aspect ratio is randomly sampled in [3/4,4/3] and area randomly sampled in [8%,100%], then resize the cropped region into a 224-by-224 square image.

2) : The training set images are flipped horizontally and vertically based on a probability of 0.5.

3) : Adjust brightness, contrast, saturation and hue with parameters [0.5, 1.5], [0.5, 1.5], [0.5, 1.5], [-0.1, 0.1] respectively.

4) : Add Gaussian noise with parameter settings of 0, 1, 20.

5) : The RGB channels are normalized by subtracting the mean and dividing by the standard deviation.

For the validation and testing phase, we use the TTA strategy to pre-process the data using four methods respectively,includes random horizontal flipping, color dithering, adding Gaussian noise and random rotation.

B. Training Details

We trained a total of 22 models to pave the way for the later model fusion,namely resnest101 wtih ce, taylor_ce and bi_tempered_logistic loss, TresnetXL with ce, taylor_ce and bi_tempered_logistic loss, mixnetXXL[12]with ce, taylor_ce and bi_tempered_logistic loss, resnest200 wtih ce, taylor_ce and bi_tempered_logistic loss, densenet161[13]with ce, taylor_ce and bi_tempered_logistic loss, skresnext50_32*4d[14]with ce loss, seresnet152 with ce loss, seresnet101 with ce loss, seresnext101 with ce loss, rexnet200 with ce loss, regnetx120 with ce loss and inceptionresnetV2 with ce loss.

During the experiments, we use Xavier algorithm to initialize the convolution and fully connected layers. The optimizer we choose to use SGD and set the momentum to 0.9. The initialization learning rate we set to 0.1 and use warm up to linearly increase to the preset learning rate in the first 5 epochs and then decay according to the cos function value and the weight decay is set to 0.0001. After comparison, we find that the small batch size To make the training faster, we call the apex library and use mixed precision for training. For all tasks we use 4 Nvidia 2080tiGPUs and 4 Nvidia V100GPUs on the pytorch framework.

C. Single Model Results

We fused the training and validation sets to train a single model, and at the same time extracted a new validation set for verification, without using pre-training weights during the training process. The results of all single models are shown below:

Table I
Results in the cross-entropy loss single model.

Method	test size	Add TTA	top1 accuracy
resnest101	224	yes	0.676
	320		0.6802
	380		0.6697
TresnetXL	224	yes	0.66892
	320		0.68356
	380		0.66562
mixnetXXL	224	yes	0.6612
	320		0.65822
	380		0.63734
resnest200	224	yes	0.6844
	320		0.6852
	380		0.67138
densenet161	224	yes	0.64218
	320		0.65146
	380		0.64066
skresnext50	224	yes	0.631
	320		0.63562
	380		0.62774
seresnet152	224	yes	0.66174
	320		0.66536
	380		0.65788
seresnet101	224	yes	0.65938
	320		0.6733
	380		0.64066
seresnext101	224	yes	0.66626
	320		0.67304
	380		0.65112
rexnet200	224	yes	0.65992
	320		0.66596
	380		0.64692
regnetx120	224	yes	0.66532
	320		0.66292
	380		0.64044
inceptionresnetV2	224	yes	0.6555
	320		0.66664
	380		0.63898

Table II
RESULTS IN THE TAYLOR CROSS-ENTROPY LOSS SINGLE MODEL.

Method	test size	Add TTA	top1 accuracy
resnest101	224	yes	0.6811
	320		0.6733
	380		0.6675
TresnetXL	224	yes	0.66968
	320		0.68813
	380		0.65561
mixnetXXL	224	yes	0.6682
	320		0.6633
	380		0.64712
resnest200	224	yes	0.6861
	320		0.6903
	380		0.6665
densenet161	224	yes	0.65172
	320		0.65536
	380		0.65017

Table III
RESULTS IN THE BI-TEMPERED LOGISTIC LOSS SINGLE MODEL.

Method	test size	Add TTA	top1 accuracy
resnest101	224	yes	0.66744
	320		0.67242
	380		0.65664
TresnetXL	224	yes	0.65711
	320		0.67826
	380		0.65698
mixnetXXL	224	yes	0.6519
	320		0.6661
	380		0.64925
resnest200	224	yes	0.6766
	320		0.6709
	380		0.6651
densenet161	224	yes	0.64891
	320		0.6555
	380		0.64201

During the experiment, we tested and verified on 3 scales, 224, 320 and 380 respectively.We found that the highest accuracy is usually achieved when the test size is 320. We also used TTA to further improve the performance of the single model.

D. Model Fusion

Model fusion has very much improved our results, and we use a two-stage model fusion strategy.In the first stage we fuse the results of different models at the same scale with the same loss using soft voting. It is worth noting that in the fusion process, we use the scores of each single model as weights, and then perform softmax calculations on all the weights to obtain the final weights, and finally weight the model results for fusion.In the second stage, we use the nine results generated in the first stage and conduct a second vote, this time using hard voting, with a final top1 accuracy of 72.4% of the results.

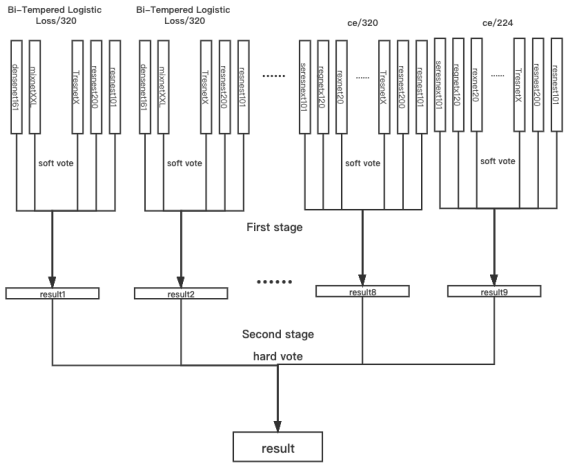

Figure 4. Specific structure diagram of model fusion.

E. Result Coverage

For the above fusion results, we analyzed the accuracy output of all categories and found that there were 136 categories with accuracy less than 0.5, so we trained separately for these 136 categories. Specifically, we resampled the data of these 136 categories, and each category was resampled to 500 data. For the other categories, we randomly selected 5 samples for each category, and then fused all the newly selected samples to form a new category. We trained these 137 categories using resent101, rexnet200, inveptionv4, efficientnetb4, resnest200 and resnext101, respectively, and we used the $DSB_{focalloss}$ proposed in this paper to participate in the training, and finally the result labels were transformed and coverage.

We analyzed the results of these targeted trainings and found that while the accuracy of these 136 classes did improve significantly, it would have an impact on the other classes, causing the model to misclassify the other 864 classes as these 136 classes. To address this situation, we covered it by taking out the labels of the other 864 classes from the previous fusion results and covering them with the newly trained results. After validation, we found that the top1 accuracy of the coverage results on the newly extracted validation set was about 74.5% at the highest. Finally, we fused the 6 coverage results by hard voting and achieved 75.178% accuracy.

IV. DISCUSSION

We also tried to train many transformer[15] models and mlp[16] models, but the amount of data was too small, so it was very difficult to train from scratch, and the results were not good. For model fusion we tried to use stacking strategy for score improvement, but the results were generally good.

V. CONCLUSION

In this paper, we use multiple differentiated trunks and multiple losses to increase the variability of the results. Various techniques are used to improve the robustness of the model as

well as the speed of convergence. Finally, our proposed two-stage model combination strategy and targeted training result coverage strategy effectively improve our scores on the test set.

REFERENCES

[1] L. Huang, C. Zhang, and H. Zhang, "Self-adaptive training: beyond empirical risk minimization," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 19 365–19 376. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/e0ab531ec312161511493b002f9be2ee-Paper.pdf

[2] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," 2019.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[4] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," 2016.

[5] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 8, pp. 2011–2023, 2020.

[6] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, Z.-L. Zhang, H. Lin, Y. Sun, T. He, J. Mueller, R. Manmatha, M. Li, and A. Smola, "Resnest: Split-attention networks," *ArXiv*, vol. abs/2004.08955, 2020.

[7] T. Ridnik, H. Lawen, A. Noy, and I. Friedman, "Tresnet: High performance gpu-dedicated architecture," *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1399–1408, 2021.

[8] D. Han, S. Yun, B. Heo, and Y. Yoo, "Rexnet: Diminishing representational bottleneck on convolutional neural network," *ArXiv*, vol. abs/2007.00992, 2020.

[9] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," 2016.

[10] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 113–123.

[11] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2017.

[12] M. Tan and Q. V. Le, "Mixconv: Mixed depthwise convolutional kernels," 2019.

[13] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," 2016.

[14] X. Li, W. Wang, X. Hu, and J. Yang, "Selective kernel networks," 2019.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[16] I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, M. Lucic, and A. Dosovitskiy, "Mlp-mixer: An all-mlp architecture for vision," 2021.