Implement the train-test resolution discrepency

2020310102 KunWoo Park
2019310762 Yoonkyung Jang
2020310107 Hoyoung Jeon

Abstract— **Inconsistency occurs between the distribution of the training data and the distribution of the testing data. As a way to solve this problem, there is a method to solve the problem of inconsistency between the distribution of training data and the distribution of test data because the size of the object viewed by the neural network during training and the size of the object viewed during testing are different. In this paper, we implement the train-test resolution discrepancy.**

I. INTRODUCTION

Convolutional Neural Networks (CNNs) are artificial neural networks that analyze visual images using linear operations in deep learning.[1] It is used to image classification [2], object detection [3], inpainting [4], and even image compression[5].

Data Augmentation is used for model generalization and training to reduce overfitting. Data Augmentation is essential part in training neural networks for image classification. To obtain the best performance, training data distribution and testing data distribution must match well.

However, inconsistency occurs between the distribution of the training data and the distribution of the testing data. There are many factors that cause this inconsistency. One of them is that the size of the object viewed by the neural network during training and the size of the object viewed during testing are different.

A. Fixing the train-test resolution discrepancy

As a way to solve this problem, there is a method of training a neural network. It is not a new architecture of a neural network, but a method to solve a problem simply by learning a neural network. This is a method to solve the problem of inconsistency between the distribution of training data and the distribution of test data because the size of the object viewed by the neural network during training and the size of the object viewed during testing are different.

Often, for image classification, a rectangular area is randomly selected from the image during the training time. Then this rectangular area is cropped and grew to a size of 224x224 to train this data. At the time of the test, a rectangular area with a size of 224x224 is selected based on the center point of the input image. After that, it is cropped and passed through the neural network. [6]

Because of this phenomenon, a discrepancy occurs between the distribution of the training data and the distribution of the testing data. This is called 'train-test resolution discrepancy'.

To solve this problem, there are two improvements to the standard setting.

1) Calibrating the object sizes by adjusting the crop size

First, by increasing the image resolution during testing, it is possible to significantly reduce the difference in object size of images used for training and testing.

2) Adjusting the statistics before spatial pooling

Second, to reduce the sparse problem after global pooling, slightly change the network before global pooling.

If the increased resolution of cropped image is used for testing, it is effective for domain-shift. One of the ways to compensate for this change is the fine-tune method. The fine-tune method is to change the resolution to the resolution corresponding to the test, and then fine-tune with the same training dataset .

At the first step, to execute the fine-tune method, initialize the network using the ImageNet dataset. After that, some of the whole is trained at a specific resolution. Then, fine-tune the last batch norm and fully connected layer at high resolution.

It is limited to the last layer of the network based on the experimental results. Because it deviates from the distribution of the data, the sparsity needs to be adjusted. For this, at least before global pooling, batch sophistication is included in fine tuning. Also, to prevent additional domain-shift from occurring, the test time extension method is used during fine-tuning. At this time, batch normalization must be included in the network before global pooling.

II. EXPERIMENTAL RESULT

TABLE I
RESULTS

Ver	Network	Early stop patience	Batch size	Epoch	Accuracy (%)
1.0	Custom VGG	5	256	200	44.0
1.1	Custom RESNET	5	256	200	41.6
1.2	Custom RESNET	5	128	200	61.0
1.3	Custom RESNET	5	64	200	65.3
1.4	Custom RESNET	5	64	200	66.2
1.5	Custom RESNET	5	64	200	50.8
1.6	Wide RESNET	10	64	500	68.0
1.7	Wide RESNET	10	64	500	73.0
1.8	Wide RESNET	10	64	500	81.8

All version, Weight decay = 1e-4 and other parameters are same as baseline`s parameter.
From ver.1.4 we utilize Data augmentation (50k)
From ver.1.5 we utilize LR scheduling
From ver.1.6 we change from RESNET to wide RESNET (101*32*16)
From ver.1.9 we utilize fixing the train-test resolution discrepancy

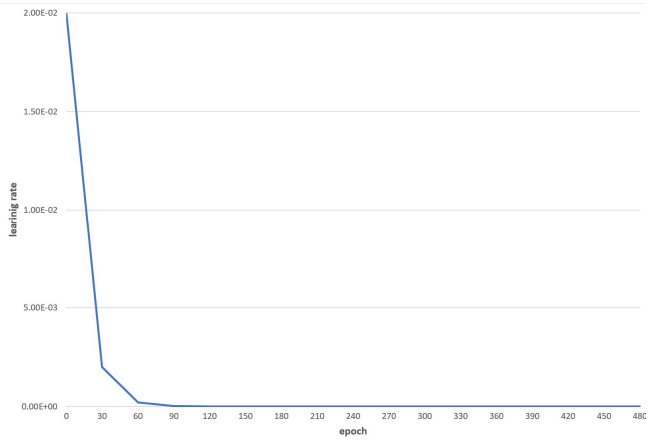

Fig. 1 Learning Rate

We trained neural networks on data sets by different hyperparameters and networks. First, we just trained neural networks by using Custom VGG and Custom RESNET.to avoid overfitting and ensure convergence we set Early stop patience 5.and we get very low accuracy and iteration stops by Early stop patience(Fig.2, Fig.3).

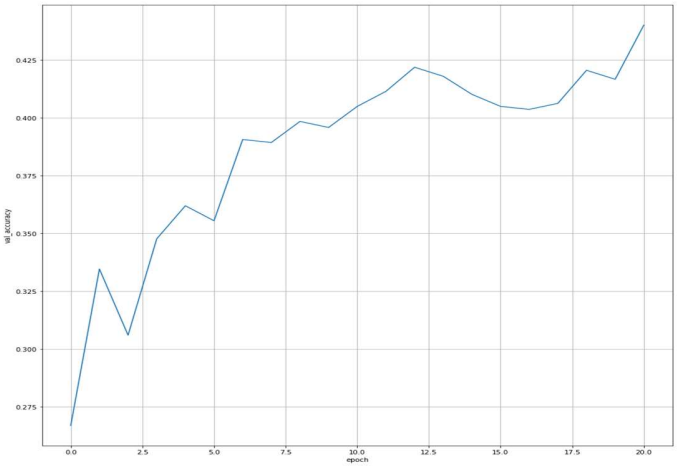

Fig. 2 Ver-1.0 Custom VGG

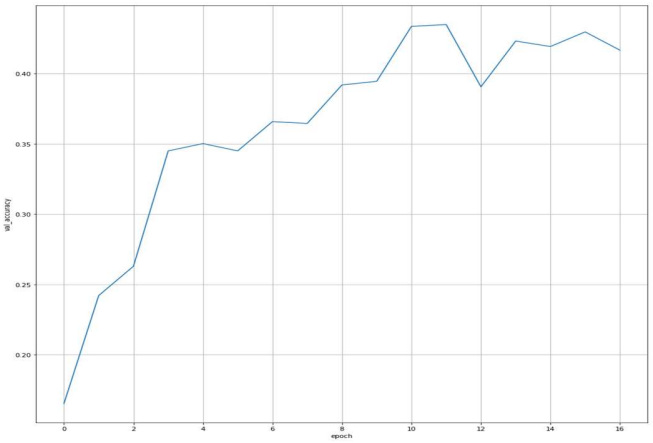

Fig. 3 Ver-1.1 Custom RESNET

Second, to avoid local value we decrease batch size(128,64). Fig.4 and Fig.6 show accuracy increase by 65%. Third, we add Data augmentation(Fig.6) and add LR scheduling(Fig.7). Fig.7 shows iteration stop early so we increase Early stop patience by 10.

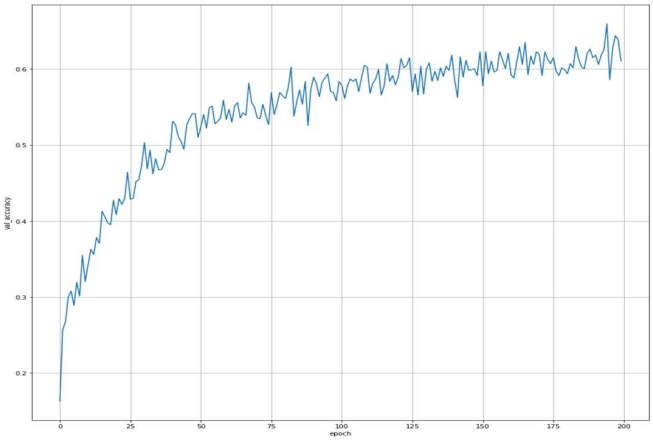

Fig. 4 Ver-1.2 Custom RESNET

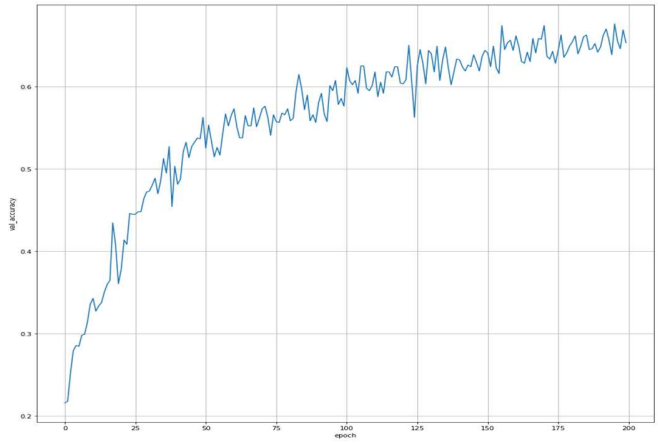

Fig. 5 Ver-1.3 Custom RESNET

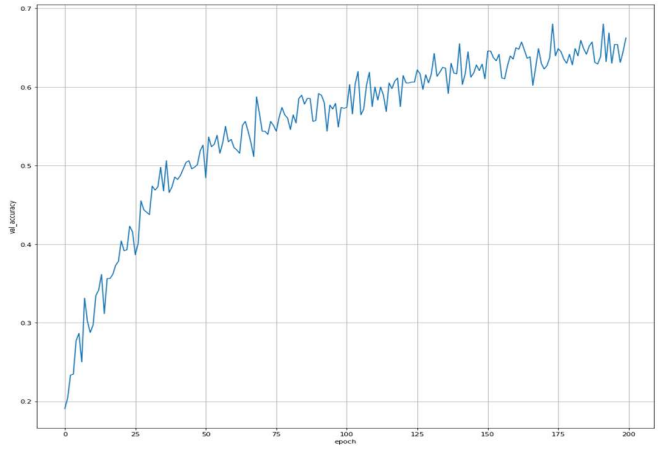

Fig. 6 Ver-1.4 Custom RESNET

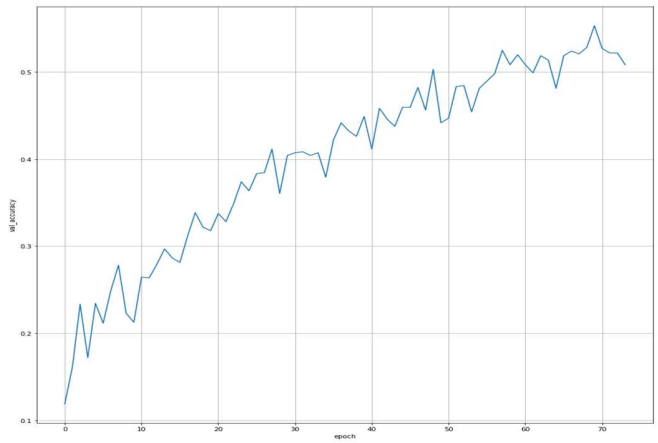

Fig. 7 Ver-1.5 Custom RESNET

Third, we change from RESNET to wide RESNET. Last, we Add fixing the train-test resolution discrepancy. Fig 10. Shows high accuracy.

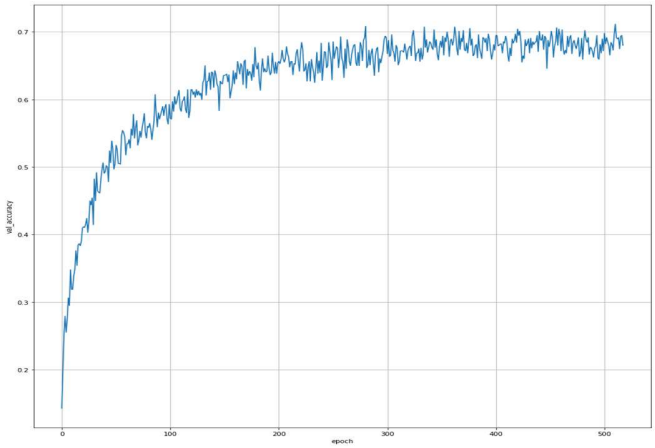

Fig. 8 Ver-1.6 Custom RESNET

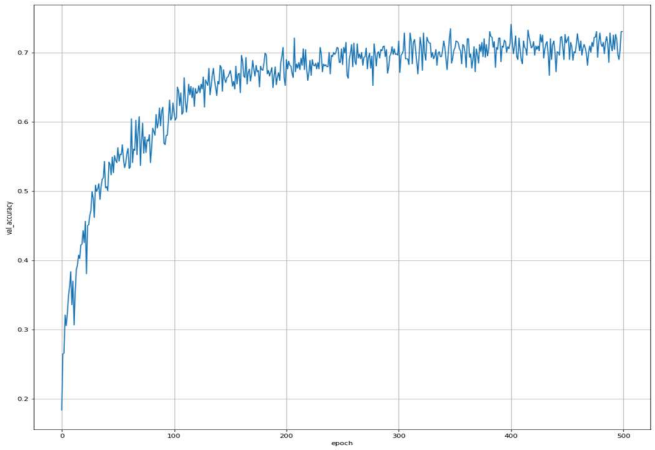

Fig. 9 Ver-1.7 Wide RESNET

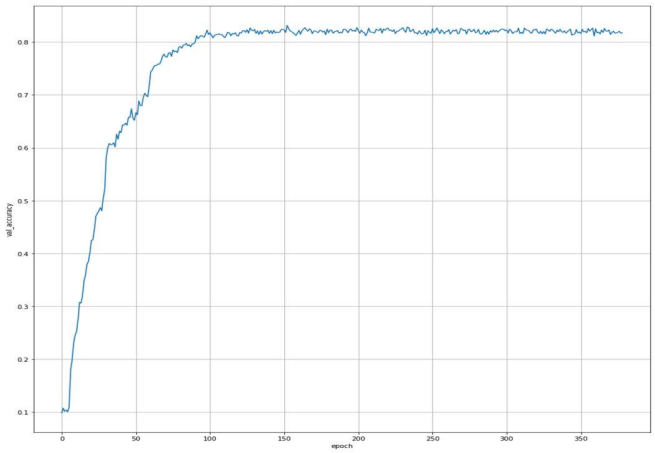

Fig. 10 Ver-1.8 Wide RESNET

III. CONCLUSION

We trained neural networks on data sets by different hyperparameters and networks. At our code, there are various techniques. First, to avoid local minima we add Early stop patience. Second, we add data augmentation and LR scheduling. Last, we utilize fixing the train-test resolution discrepancy and get 81.8% accuracy

REFERENCES

[1] Britz, Denny. "Understanding convolutional neural networks for NLP." URL: http://www. wildml. com/2015/11/understanding-convolutional-neuralnetworks-for-nlp/(visited on 11/07/2015) (2015).
[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems, 2012.
[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in Neural Information Processing Systems, 2015.
[4] Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. In Advances in Neural Information Processing Systems, pages 341–349, 2012.
[5] Oren Rippel and Lubomir Bourdev. Real-time adaptive image compression. In International Conference on Machine Learning, 2017.
[6] Touvron, H., Vedaldi, A., Douze, M., & Jégou, H. (2019). Fixing the train-test resolution discrepancy. In Advances in Neural Information Processing Systems (pp. 8252-8262).