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.

B. *Labsel smoothing*

A mislabeling occurs when there are incorrectly labeled images. The mislabeling occurs especially when the source of the image is from the Internet.

For example, when training a deep learning model that classifies dogs and cats, a cat image but labeled as a dog is trained. If the number of data is small, a person can visually check and correct it, but if the number of data is large, it is not possible.

When calculating the loss while training a deep learning model, the loss value is largely calculated for incorrectly labeled samples. This can cause problems during training.

One possible way to solve this problem is label smoothing. Label smoothing gives the label smoothly rather than 0 or 1 in order to reduce the effect of possible erroneous loss.

C. Learning rate scheduling

When training a deep learning model, a method of reducing the learning rate is often used. This is to converge to a lower loss. However, in order to obtain optimal results, it is impossible to simply reduce the learning rate unconditionally. This is because the reduction in the learning rate depends on the learning model.

Learning rate scheduling is used to solve this problem. Learning rate scheduling includes 'Step-wise Decay' and 'Reduce on Loss Plateau Decay'.

‘Step-wise Decay’ calculates the next learning rate by doing 'previous learning rate * gamma'.

‘Reduce on Loss Plateau Decay' is a method of reducing the learning rate when the error is no longer reduced and stays at the same value. A Patience is to decide how many epochs to reduce the learning rate after the error starts not to decrease. If the epoch is large, the patience should be set larger.

D. L2 Regularization

L1 regularization and L2 regularization are used to do regulation. This is a technique used to prevent overfitting.

The reason for dividing L1 and L2 is because of the norm used when applying regulation. L2 Regularization adds a regularization term to the Loss function to be minimized.

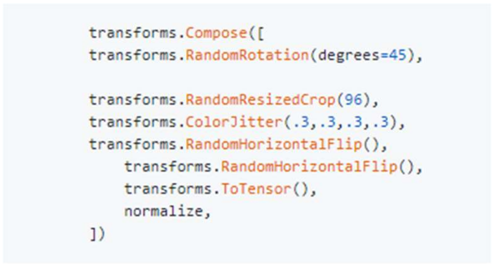
II. EXPERIMENTAL RESULT

TABLE I
RESULTS

Ver	Network	Lr scheduling step size	Batch size	Epoch	Accuracy (%)
2.0	LE-Net	100	256	500	70.0
2.1	Custom-Net	5	256	500	68.0
2.2	Custom-Net with Leaky ReLU	5	128	500	70.0
2.3	Custom-Net with scheduling epoch 50	5	64	500	72.3
2.4	?	5	64	300	77.2
2.5	RESNET	5	64	700	79.8

All version, weight decay = 1e-4 and learning rate = 1e-3

We trained neural networks on data sets by different hyperparameters and networks. First, we trained neural networks by using LE-net and weight decay. And The number of augmented data is 50,000 and transforms are as follows:



```
transforms.Compose([
transforms.RandomRotation(degrees=45),

transforms.RandomResizedCrop(96),
transforms.ColorJitter(.3,.3,.3,.3),
transforms.RandomHorizontalFlip(),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    normalize,
])
```

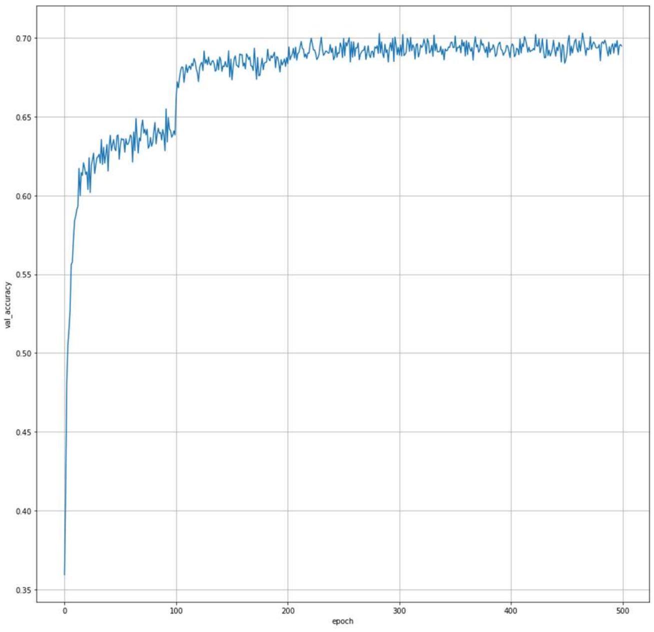


Fig. 1. Ver-2.0 LE-Net

and Network was changed from LE-Net to Custom-Net and increase augmented data to 100,000.(fig.2)

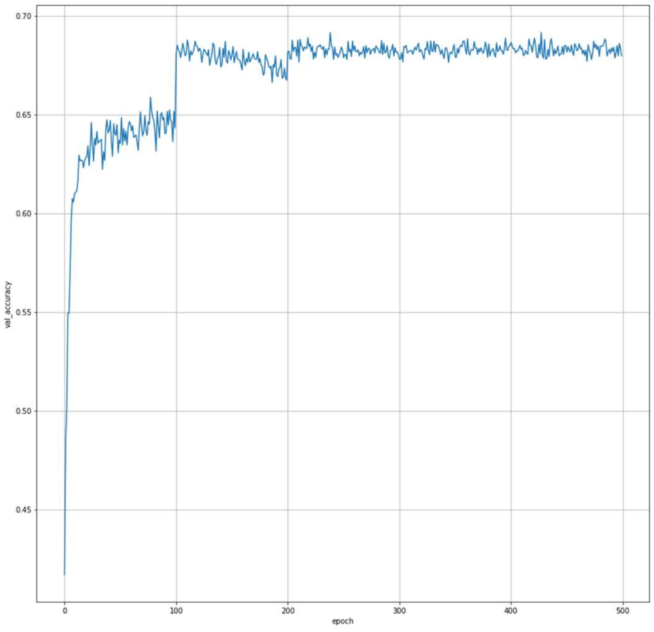


Fig. 2. Ver-2.1 Custom-Net

Next, activation function was changed form ReLU to Leaky ReLU (fig.3)

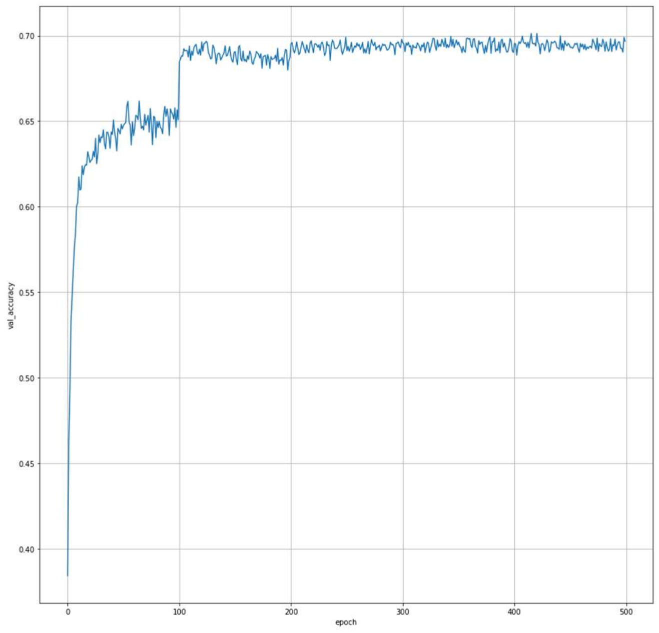


Fig. 3. Ver-2.2 Custom-Net with Leaky ReLU

and lr scheduling epoch decrease to 50 (fig.4)

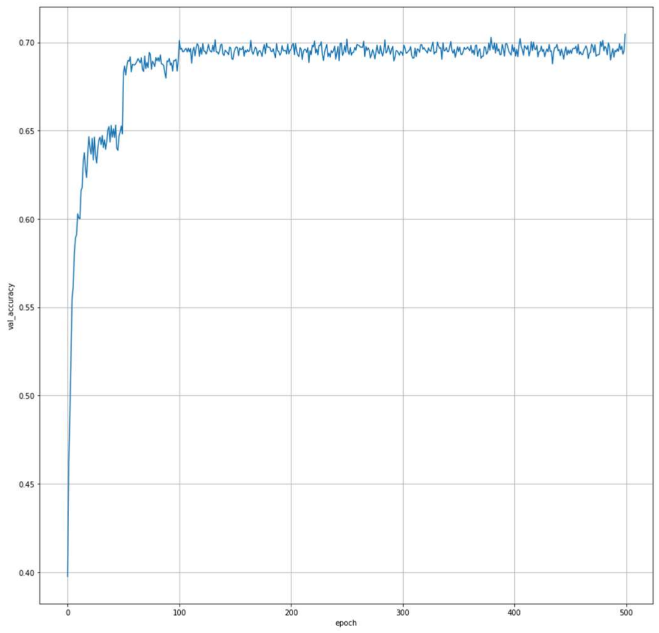


Fig. 4. Ver-2.3 Custom-Net with scheduling epoch 50

And we use Technique in fixing the tarin-test resolution discrepancy. Random resized crop is 80*80 resolution and additionally, fine tune by 120*120 resolution.(fig.5)

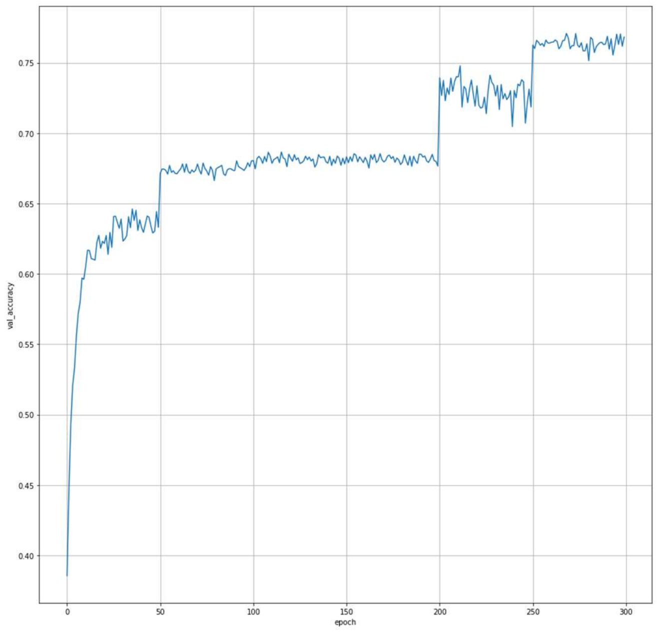


Fig. 5. Ver-2.3 LE-Net

Lastly, we change model to RESNET (fig.6)

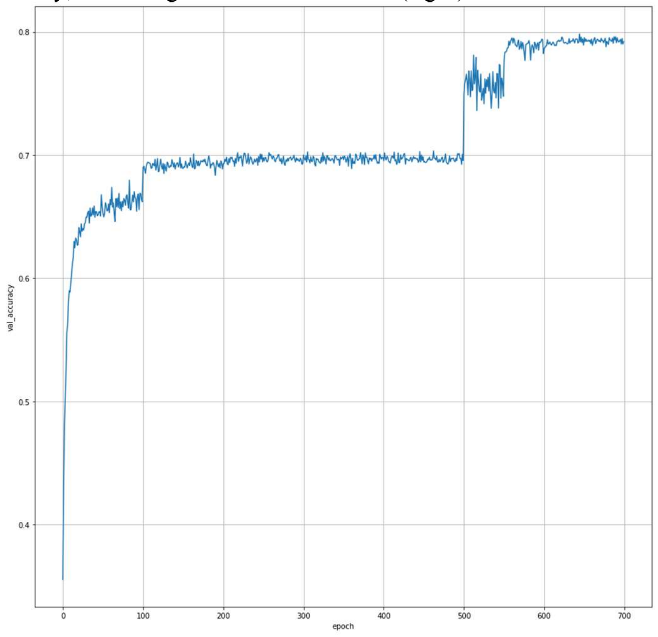


Fig. 6. Ver-2.3 RESNET

III. Conclusion

We trained neural networks on data sets by different hyperparameters and networks. At our code, there are various techniques. First, learning rate scheduling. Second, we add data augmentation and LR scheduling. Last, we utilize fixing the train-test resolution discrepancy and get 79% 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).