Can Local Binary Convolutions Make Neural Networks Models Smaller?

Haoyu Zhang[1], Mohamed Wahib[2], Peng Chen[1 2], Satoshi Matsuoka[1 3]
zhang.h.am@m.titech.ac.jp
[1]Tokyo Institute of Technology, Dept. of Mathematical and Computing Science, Tokyo, Japan
[2]AIST-Tokyo Tech Real World Big-Data Computation Open Innovation Laboratory
[3]RIKEN Center for Computational Science,Hyogo,Japan

1 BACKGROUND

Local Binary Convolution (LBC) [4] is a kind of spatial convolution which get inspiration from the Local Binary Pattern. CNN modules with LBC layers are called **local binary convolution neural networks (LBCNN)**.

Local Binary Convolution consists of one weight anchored convolution layer (whose filters are filled with stochastic value in range of 1, -1 and 0) and a layer of 1×1 convolution. As is shown in Figure 1.

In normal convolution, p channels of input will go through q filters and then ReLU function. At last there will be q output channels.

While in LBC, p input channels will go through m anchored-weight filters first then go through ReLU/Sigmoid function to get m intermediate output, these output will then go through q channels 1×1 convolution. At last you can also get q channels output.

The advantage of LBCNN is saving large quantity of learned parameters without losing a lot of accuracy. As is shown in Equation 1:

$$\frac{Param.\ of\ CNN}{Param.\ of\ LBCNN} = \frac{p \times h \times w \times q}{m \times q} = \frac{p \times h \times w}{m} \tag{1}$$

$$*Param.\ means\ number\ of\ parameters$$

With the assumption that m equals to p, this could save $h \times w$ times of the learned parameter in the model. In this poster we propose to extend the method to larger models to see if it can work well.

2 PROBLEM OF LBCNN

LBCNN models could save $h \times w$ times of the learned parameter in the models, **however, this is just based on the assumption that m equals to q in Equation 1**.

At some certain convolution layer, let $X \in \mathbb{R}^{(p \times H \times W) \times 1}$ be a p channels input of the layer. And let $w \in \mathbb{R}^{(p \times h \times w) \times 1}$ be a single filter in this layer. The output of this filter could be represented as Equation 2:

$$d = \sigma_{relu}(w * X) \in \mathbb{R}^{(H \times W) \times 1} \tag{2}$$

Let $B \in \mathbb{R}^{(m \times h' \times w') \times p}$ be the anchored weight filters in corresponding LBC layer. The output of the Bit Map can be expressed as Equation 3

$$\sigma_{sigmoid}(B * X) \in \mathbb{R}^{(H \times W) \times m} \tag{3}$$

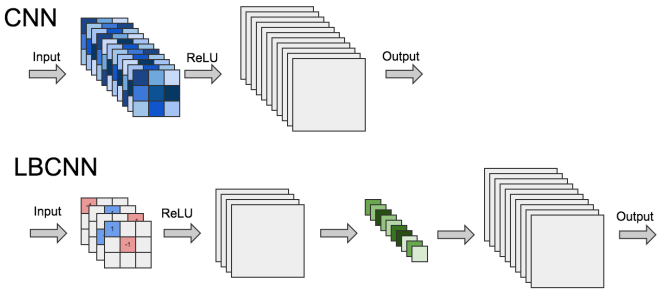

Figure 1: LBC construction.

Then let $v \in \mathbb{R}^{(1 \times 1) \times m}$ be a 1×1 filter in the latter part of the LBC layer. Since v is 1×1 filter, Equation 3 could be changed to:

$$d' = C_{Sigmoid} v \in \mathbb{R}^{(H \times W) \times 1} \tag{4}$$

Juefei-Xu et al. [4] discussed that there always exist such v that $C_{Sigmoid} v = d' = d$, **however you can not always find such v since only when $m > rank(X^{(H \times W) \times m})$ such v may exist.** If m is too small, this method will damage the accuracy of the model a lot.

Usually, we can not use such big m, especially on condition that inputs are large pictures, since **this will lead to huge amount of parameters**. So there may exist some problems when simply replace all normal convolution layers in DenseNet models by LBC.

3 LBDENSENET

3.1 Building Block

ResNet [2] are built by building block method, there are two convolution layers in each basic block, and there is a shortcut connection between the input and out put of the basic block:

$$output = ReLU(\mathcal{F}(x) + x)$$

Which adds up the input and output followed by ReLU as a new output.

Building block method is also applied to **Densely connected convolutional Network (DenseNet)** [3] design. We propose to make new models based on DenseNet models.

3.2 LBDenseNet Family

First of all, we directly replace all normal convolution layers by LBC layers. As is shown in Figure 2 (all LBC layers are under the condition that $m = p$):

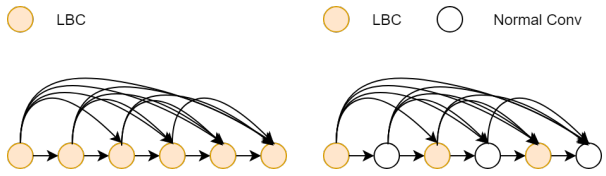

Figure 2: Basic block of DenseNet full (left) and fused (right).

LBDenseNet-Full, since the Denseblock have "bottle neck" architecture, all 3×3 convolution layers will be followed by Relu-BN-1×1 convolution, we can easily apply our Full model by replacing the 3×3 convolution layers with LBC layer.

LBDenseNet-Fuse, we call the model with half normal convolution layer half LBC layer fused model. We apply LBDenseNet-Fuse by replacing half of the 3×3 layers in each basic block. (as is shown in Figure2).

4 EXPERIMENTS

We train models on **ImageNet**[5], and we use the same parameters as Yu et al. at [6] for training. We use Top-1 and Top-5 accuracy as our acurracy measurements.

We apply our methods to DenseNets-121 and train it on ImageNet classification task. Also we add an hyper parameter m which described the amount of filters in LBC layers. Noted that the parameter m will affect the numbers of input channels for 1×1 convolution layers. Hence higher m will leads to more learned parameter.

After we add m to test full and fused models. We can test the results under different m. Figure 3 shows the relationship of parameters and accuracy.

Figure 3: Results of models under different m.

As is shown in this figure, we can adjust m to make full and fused models have the same numbers of learned parameter, and the results are very closed to each other. More learned parameter leads to higher accuracy. This means LBC together with 1×1 convolution can somehow replace 3×3 convolution but do not help saving parameters.

Furthermore, we tested the time of each models on ImageNet, we define the reciprocal of total training time as x-axis named speed. And let the accuracy be the y-axis, we can get Figure 4.

It can be seen that, full version have a very low efficiency since higher m obviously increase the computational amount of the model, which makes the model become much slower. The figure indicates that although it may be possible for using LBCNN to get a accuracy close to the original version of model, it will be far too slow to train.

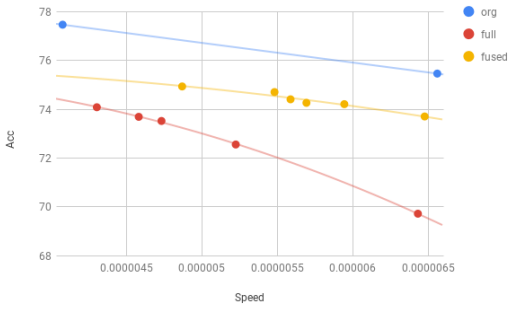

Figure 4: Accuracy and Performance of different models.

5 CONCLUSION

We extend the method of LBCNN to larger model: DenseNet on ImageNet to see if it helps. In addition, during our extension we also proposed a trade off way that fuse normal convolution and LBCNN. It is hard for the method of LBCNN get better results in more complex models and this method seems to effect the training speed a lot. After analyzing the results of out tests, we find that, with LBCNN method, it is possible for 1 convolution after LBC filters to get the similar results with 3×3 convolution, but this method have a very low time efficiency.

6 FUTURE WORK

We are going to Test LBCNN on more datasets and more models (Resnet[2] for example). And extend LBCNN on more tasks such as semetic segmentation on Dataset like Cityscapes[1]. Since we finally find out that LBC is not such a good method to approximate normal convolution we need to find a better choice.

REFERENCES

[1] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3213–3223.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[3] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. 2017. Densely Connected Convolutional Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2261–2269.

[4] Felix Juefei-Xu, Vishnu Naresh Boddeti, and Marios Savvides. 2017. Local Binary Convolutional Neural Networks. In *IEEE Computer Vision and Pattern Recognition (CVPR)*.

[5] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115, 3 (2015), 211–252.

[6] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. 2017. Dilated Residual Networks. In *Computer Vision and Pattern Recognition (CVPR)*.