High Performance Binarized Neural Networks trained on the ImageNet Classification Task

Xundong Wu
wuxundong@gmail.com

April 12, 2016

Abstract

We trained Binarized Neural Networks (BNNs) on the high resolution ImageNet LSVRC-2102 dataset classification task and achieved a good performance. With a moderate size network of 10 layers, we obtained top-5 classification accuracy rate of 81 percent on validation set which is much better than previous published results. We expect training networks of a much better performance through increase network depth would be straight forward by following our current strategies. A detailed discussion on strategies used in the network training is included as well as preliminary analysis.

1 Introduction

Recent wave of deep learning research has brought the deep neural network to the frontier of AI applications. Current implementations of deep neural networks in wide variety of fields heavily rely on high computing power, energy hungry hardware such as GPUs, which limits implementation of those neural networks in embedded environment such as mobile phones or wearable hardware. Much efforts have been put to reduce computing need of deep neural networks, for example, (Han et al., 2015; Chen et al., 2015; Jaderberg et al., 2014).

Recently, (Rastegari et al., 2016; Courbariaux et al., 2016, 2015) have been trying to reduce synapse weight and/or intermediate signal representation to binary form. As been shown binarification of neural network can dramatically reduce the computing need, and also the amount of memory needed for weight and intermediate results storage. These approaches are especially useful for memory constrained environment such as the FPGA implementation.
In this work, we designed a network architecture based on Binarized Neural Networks (BNN), originally proposed by Courbariaux et al. [2016], and trained networks on the ImageNet classification task. We identified an issue that caused slow learning with original BNN when it is combined with large dimension Softmax output function. An efficient solution is proposed that greatly speed up the network training.

We also analyzed the behavior of weights adaptation during SGD training and use the observation to guide the setting of proper learning rate used in the network training. This also leads to significant gain in the learning speed of the network.

With a moderate size network an 81 percent top 5 accuracy on validation set of ImageNet classification task is achieved. We expect training of better performing networks will be quite straightforward as we observed steady performance gain with more layers added to the network.

2 Results

2.1 Softmax Input Rescaling

A pure BNN layer has an output of large range of magnitude due to the binary weights and binary inputs from previous layer. At the initial state of the network for a standard dense fully connected layer, if we assume the layer input is of 50% density, i.e. half of the inputs have activation of +1, the rest have activation of -1, and the synapse weights are also initialized to be 50% density. The mean of unit output of this layer will be 0, while the $\sigma$ will be $\sqrt{N}$, here $N$ is the number of inputs the layer receives which is typical quite large for the last layer of a network used for tasks such as ImageNet data. With a typical $N = 4096$ used in this study, which give us $\sigma$ value of 64. This won’t be a problem if the output of a BNN layer is followed by a batch normalization layer as in Courbariaux et al. [2016], since batch normalization bring the activation into right range. In Courbariaux et al. [2016] case, the network output has small dimensionality, for example in MNIST case the output dimension is 10. For ImageNet data we need to handle output of 1000 different categories thus the activation of output units are forced to be very sparse. Which means for a moderate batch size, for example 256, used in our SGD learning batch normalization will be difficult. For this reason, we cannot use batch normalization at last layer before the last layer non-linearity in this case.

Here we used log-loss with Softmax function on the last layer of the network. That is: with the last BNN layer output $X$ of $N = 1,000$ dimension, and ground truth training target $Y$, we pass $X$ through $Softmax$ function, which gave us $P_j = \frac{e^{x_j}}{\sum_k e^{x_k}}$ and from there we got loss function $l = -\frac{1}{N} \sum_i Y_i \log(P_i)$. Due to the combination of strong non-linearity with the large output variation range of BNN layer, we found that learning can be quite slow especially at the early stage of the training.
We propose an efficient solution to overcome the learning difficulty described above. Here we insert a scaling layer between the last BNN layer and the Softmax nonlinearity. In this way the input to the Softmax function will be in the proper range thus avoid the learning difficulty. In most of our experiments we used a scaling layer with an initial scale factor shared by all units to re-scale input of Softmax to standard deviation of 1 for fast learning. As shown in Figure 1, indeed such re-scaling significantly speed up the network training. We also tried a fixed-value scaling layer with $1/\sigma$ scale factor, small scale experiment shows it performs slight worse than a layer with a trainable scaling factor.

### 2.2 First layer architecture

In the original BNN layer designed for CIFAR-10 dataset as in Courbariaux et al. (2016), all layers of network are binary. The filters used in the first layer of that network have small kernel size of 3 and small stride step of 1. Such design is suitable for CIFAR data since it comes with image size of 32x32, which is very small when compared with the typical input image size used in ImageNet data of 224x224. To reduce the computing need for the network, typically a large convolutional kernel size and big stride steps are used for the first layer of in most of state of art networks trained on ImageNet data. For this reason, most networks implemented in this work also start with large kernel sizes (7 or 11) and big stride steps (2 or 4) on first layer and directly followed by a max-pooling layer to reduce first convolutional layer output size. Show in Figure 2 when a real value weight layer is used in the first layer the network performance is slightly better than pure binary network. Therefore for the rest of this study all networks trained in this study we use real-value weight in the first layer of network.
Figure 2: Comparison of network performance with real-valued first layer or BNN layer. Networks are trained 20 full epochs with batch size of 256.

Since the first layer input only have 3 input channels, this won’t significantly increase computing load for the whole network. Due to the bottle-neck effect of a binary representation, a BNN might benefit from extra communication channels, it is possible with smaller convolutional kernel size and stride steps on the first layer we might achieve similar or even better performance with binary weight first layer. However, limited by the available computing resource, currently we are not able to test this hypothesis.

2.3 Weight adaptation behavior analysis and learning rate selection

In a BNN layer, at training stage real-valued version of synaptic weights are used during SGD training. This extra resolution is removed at the stage of computing layer output. It is believed that real-valued weights are needed for smooth out the noise in the SGD learning (Courbariaux et al., 2016). Here we show indeed this is the case. When a high learning rate is used for training as suggested in (Rastegari et al., 2016), after one epoch of training, we observed most weight value stay around -1 and +1 (Figure 3a) because of weight clipping. Under such condition weights quickly jump between edge values of -1 and +1 with little accumulation process. Correspondingly a rather slow loss decline process is observed (Figure 3c). When we lower learning rate to 0.001, we observed a weight distribution more toward uniform or concentrated around 0, indicating weights are taking accumulated steps to travel between -1 and +1 (Figure 3b). Correspondingly, a much faster loss decline was observed during the training process (Figure 3c). We also show the result of 20 full epoch training with initial learning rate of 0.1 and 0.001 decay at fix rate in Figure 3d.
Figure 3: Weight update behavior under different learning rate
(a) Weight distribution of layer 2 and 6 after one epoch of training under learning rate of 0.1. (b) Weight distribution of layer 2 and 6 after one epoch of training under learning rate of 0.001. (c) Networks trained with first 500 batches of data each epoch for 20 epochs. (d) Network trained with full 20 epochs
3 Discussion

Our results show that an ImageNet scale of BNN can be trained to a good performance with a good network architecture and training strategies. At moment the best network we trained is a 10 layers network with 81 percent top 5 accuracy rate, which is much better than the result shown in [Rastegari et al., 2016].

Limited by the available computing resource, we mostly tested networks with real value weight first layer of big filters and big stride. Judge from some of our small scale experiments and also results shown by [Courbariaux et al., 2016], it is very possible a purely binary network can achieve state of art level of performance with small filter size and smaller stride step. A pure binary network is also desirable since it further reduce and simplify the network design in constrained environment.

References

Chen, Wenlin, Wilson, James T, Tyree, Stephen, Weinberger, Kilian Q, and Chen, Yixin. Compressing neural networks with the hashing trick. arXiv preprint arXiv:1504.04788, 2015.

Courbariaux, Matthieu, Bengio, Yoshua, and David, Jean-Pierre. Binaryconnect: Training deep neural networks with binary weights during propagations. In Advances in Neural Information Processing Systems, pp. 3105–3113, 2015.

Courbariaux, Matthieu, Hubara, Itay, Soudry, Daniel, El-Yanivand, Ran, and Bengio, Yoshua. Binarynet: Training deep neural networks with weights and activations constrained to+ 1 or-1. arXiv preprint arXiv:1602.02830, 2016.

Han, Song, Pool, Jeff, Tran, John, and Dally, William. Learning both weights and connections for efficient neural network. In Advances in Neural Information Processing Systems, pp. 1135–1143, 2015.

Jaderberg, Max, Vedaldi, Andrea, and Zisserman, Andrew. Speeding up convolutional neural networks with low rank expansions. arXiv preprint arXiv:1405.3866, 2014.

Rastegari, Mohammad, Ordonez, Vicente, Redmon, Joseph, and Farhadi, Ali. Xnor-net: Imagennet classification using binary convolutional neural networks. arXiv preprint arXiv:1603.05279, 2016.