Reproducibility Challenge
Latent Weights Do Not Exist: Rethinking Binarized Neural Network Optimization

Bharathwaj Krishnaswami Sreedhar[1] and Vinit Jain[2]

[1,2]KTH Royal Institute of Technology
[1]bks@kth.se
[2]vinitj@kth.se

Abstract

Binary neural networks, until now have been optimized using the existing optimizers such as Adam or SGD. However, the existing optimizers are built for latent weights and do not perform well when used on Binarized Neural Networks. The paper proposes a novel "Binary Optimizer" (BoP) specifically developed for binary neural networks. A VGG like network is trained on the CIFAR-10 dataset, and finally compared against the baseline model. To conclude, rigorous experiments are performed by varying the hyper-parameters, and evaluating on different datasets and architectures, thereby evaluating the overall training of the Binary Neural Networks against the conventional CNNs. Code is available at this https URL

1 Introduction

Binary Neural Networks (BNN) [2] were introduced to make deep learning less computationally intensive. This was achieved by constraining the weights of the network to {-1,+1} [2, 3]. Since the introduction of BNN, many conventional deep learning architectures have been adapted and have produced state-of-art results at reduced computation and energy costs [14, 7, 15].

While the weights and activations in a BNN are constrained, this constraint is merely implemented as a clipping function when used with conventional optimizers such as Stochastic Gradient Descent [11] or Adam [4]. Since these optimizers were developed for "latent weights" i.e real valued weights, the authors of this paper show that when a customized optimizer (Binary Optimizer) is used for training, the results produced are improved.

The paper was implemented in Keras using a open source library [6] that is developed and maintained by Plumerai. Larq was used to create the binary layers. The binary model used in this paper is derived from [3]. The model was run on NVIDIA Tesla V100 GPUs. The performance of the binary optimizer was compared against the Adam optimizer for two datasets, CIFAR-10[5] and ImageNet. The authors code can be found at [10].

2 Background

2.1 Training with Latent Weights

The conventional method of training for a binary network $y = f(x, w)$, where the weights $w \in R^n$. Here the objective is to find w^*_{bin} such that:

$$w^*_{bin} = \text{argmin}_{w_{bin} \in \{-1,+1\}^n} E_{x,y}[L(f(x, w_{bin}), y_{label})] \tag{1}$$

Equation 1 provides the constraint that the weights must be binary. In this situation for the forward pass, the weights are binarized as:

$$w_{bin} = sign(w) \tag{2}$$

and the backward pass is simplified as a straight through estimator [1]:

$$\Phi(L, w) = \frac{\partial L}{\partial w_{bin}} \approx \frac{\partial L}{\partial w} \tag{3}$$

2.2 Binary Optimizer

Instead only considering the sign as in equation 2, the authors also take the magnitude m, which provides $inertia$ to the network. This is called so because as the weight grows, it requires a stronger gradient signal to flip it. So the latent weights can be considered as variable that contain both binary weights and inertia m, which is similar to momentum.

The function of the optimizer is reduced to only flipping the weights. It also takes the past gradients into account to check for consistency. Finally the strength of the gradient (absolute value) is taken into consideration to reduce noise during training. Consistent signals are found by :

$$m_t = (1 - \gamma)m_{t-1} + \gamma g_t \tag{4}$$

In equation 4, g_t is the gradient at time t, m_t is the exponential moving average and γ is the adaptivity rate. The moving average is then compared with threshold (τ) to determine whether the weight flip is necessary.

As a result it can be seen that BoP effectively reduces the number of variables to be optimized to two.

3 Methods

For the reproducibility challenge, we use Python 3.6 with Tensorflow 2.0. The model was run on NVIDIA P100 GPU.

3.1 Reproducibility of Original Results

As part of the reproducibility of the original results, we implemented the model architecture and pipeline without referring the authors base code. The model was written with the help of the architecture from [2] and example codes of Larq API. As per the paper, the input weights are quantized by a straight through estimator [1], while no constraints are applied to the activations.

Due to memory and time constraints, we tested the performance only on CIFAR-10 dataset. As per CIFAR-10 standards, 50,000 examples were used for training and 10,000 examples were used for testing. The augmentations performed were padding with 4 pixels on each side and then taking random crops of $32x32$ and random horizontal flips. The accuracy was measured using Cross Entropy. The model arguments are as follows:

Table 1: Training Parameters

Parameters	Value
Initial LR	10^{-2}
Epochs	500
Batch size	50
τ	10^{-8}
γ	10^{-4}
γ decay	0.1 for 100 epochs

The model was run thrice and the average accuracy of the baseline was found to be 91.02% while BoP achieved 91.6%. This result is similar to the one obtained in the paper.

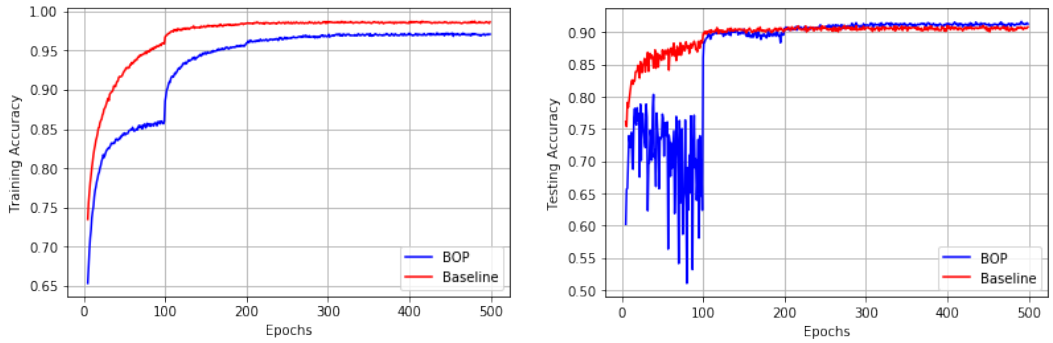

Figure 1: Training and Test accuracy comparison between BoP and Adam Baseline

4 Experiments

Different hyper-parameters tuning are performed in this phase. The hyper-parameters tested are discussed in the following subsections. Hyper-parameter tuning was performed by changing few of the training parameters. This was performed to check if the results obtained using the BoP approach are conclusive, and work in most cases.

4.1 Learning-Rate

One of the hyper-parameters tuned is the learning rate. The learning rate is kept constant with a value of 0.01 and trained for 500 epochs on a CIFAR-10 dataset. The model achieves a training accuracy of 96%, and the validation accuracy of 91%. From the graphs shown in the figure 2 and 3, the training and validation results are stable during the entire course of the training, except during the first 100 epochs of the validation phase. Although the results are not really conclusive, the understanding here is that a lot of learning occurs in the first 100 epochs of the training phase which causes constant flipping of the gradients$\{+1, -1\}$. These repetitive switching then induce substantial changes in the model's output. Although there is a difference of 6% in accuracy in terms of training and validation phase, there is no degradation of training performance when compared against the baseline model.

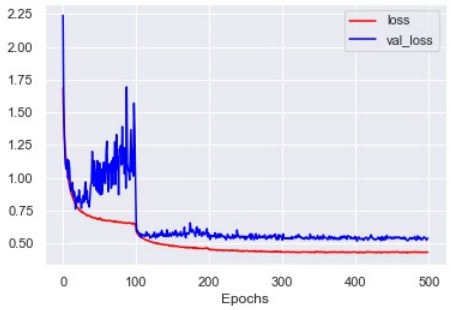

Figure 2: Training and Validation Loss

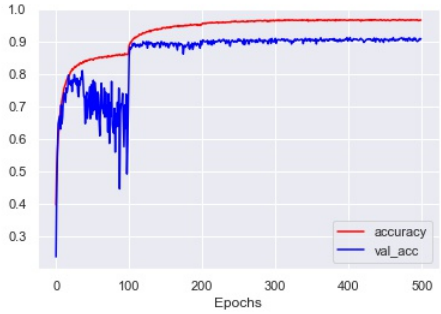

Figure 3: Training and Validation Accuracy

4.2 Dropout

Although Binary Neural networks are themselves good regularizers and hence, we try to understand the effect of adding multiple regularizers. Dropout[13] is added after every layer of the network with certain randomness. This means that the dropped neurons are not considered for the forward and backward pass, thereby giving some form of generalization to every layer, and in turn to the overall network. The Binary Optimizer Network is trained for 500 epochs, with a dropout probability of 0.25 applied to every convolution and dense layer. From the two figures shown below, it is evident that the overall training phase is quite smooth. However, there is a sudden change in loss/accuracy after the

100th epoch which is quite surprising. The magnitude of this change decreases after the 100th epoch, but still persists on the 200th epoch, and fades away during the later stages of the training phase.

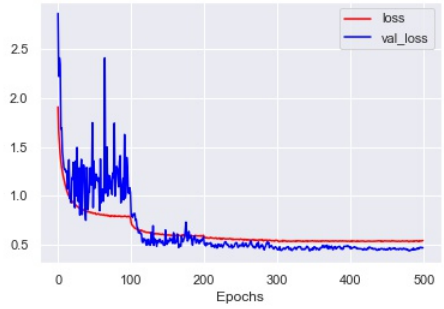

Figure 4: Training and Validation Loss

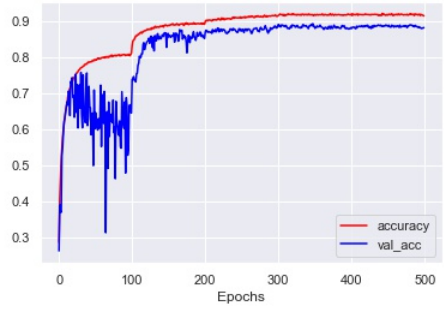

Figure 5: Training and Validation Accuracy

On the other hand, the validation results are stochastic during the first 100 epochs of the training phase. As dropout randomly drops neurons during training, this is problematic in-case of Binary Neural Networks. The overall accuracy of the network decreases to 92% and 88% during the training and validation phase. This is due to the fact that, dropout randomly introduces zeros into the network which violates the basic principle of BNNs.

4.3 Weight Decay

Compared to the original BoP approach described in the paper where the γ is decayed by 0.1 after every 100 epochs, the γ in this case is kept constant during the entire training period. The results obtained in the training and validation performance drops as mentioned in the original paper. In terms of the training and validation, the accuracy flattens at 35% and 10% simultaneously. This ascertains that the γ plays a major role in proper functioning of the Binary Networks with BoP optimizer.

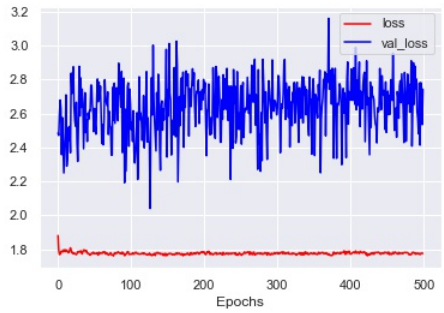

Figure 6: Training and Validation Loss

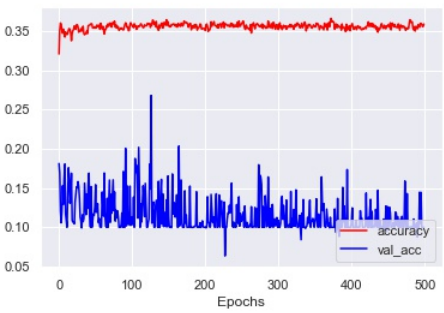

Figure 7: Training and Validation Accuracy

4.4 SVHN Dataset

The binary model with bop was also tested on SVHN dataset [8] by Stanford University. Given that this dataset is similar to CIFAR10 in terms of image size and number of classes, we would like to test the performance of bop in a large dataset. As per the SVHN standards, 73257 images are used for training, 26032 images for testing and 30000 images (from extra) was used as validation. The performance of bop against Adam was tested. The hyper-parameters used are similar to those used in Table 1 except the model was only run for 300 epochs.

At the end of 300 epochs, the baseline model obtained an accuracy of 92% while the Bop obtained 93.2%. Although the hyper-parameters used were determined for CIFAR-10, the model still performs exceptionally well on SVHN. It is interesting to note that the baseline model has over-fitted on the

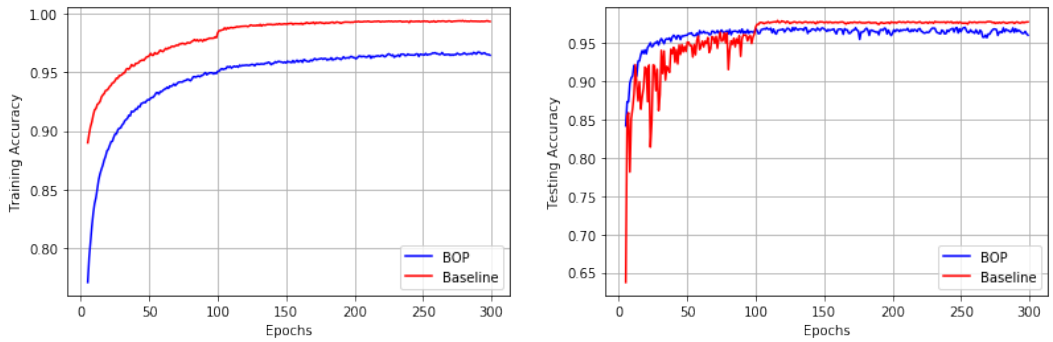

Figure 8: Comparison of Training and validation accuracy of BoP with baseline on SVHN

training data while the Bop has not. This can also be a new direction to explore as it appears that the *Bop* has regularized the model to an even greater extent.

4.5 Image Segmentation

Image segmentation architectures such as Unet [12] are computationally intensive. We wanted to find out what kind of result would a binarized Unet produce and how would *Bop* be able to increase its performance. Such an architecture would increase portability and could be deployed in mobile devices. We compared the results of Unet (pure convolution), Binarized Unet and Binarized Unet with Bop on the Oxford-IIIT Pet dataset [9]. All three versions were run for 300 epochs. For the pure unet model, Adam optimizer with initial learning rate of 0.1 was used. In the binarized unet, the convolutional layers were replaced with quantized layers from the larq library. As per the previous experiments, the learning rate was dropped by 0.01 every 100 epochs. For the binarized unet with Bop, the gamma was decayed by 0.1 every 100 epochs.

As seen in Figure 9, during training and testing the accuracy of the model is erratic. While the training accuracy of pure unet is more stable, the testing accuracy for all the three cases has high variance. There could be lots of reasons for this. Model Architecture, optimizer selection, initial learning rate, etc. Also for in the case of Bop, the erratic behaviour could be attributed to the continuous switching of weights without any buildup of inertia. From this experiment, it can be said that although binarized unet does not provide better performance than pure unet, this statement cannot be generalized as it depends on model selection, kernel initialization and stopping criteria, i.e saving the best weights of the model based on validation accuracy.

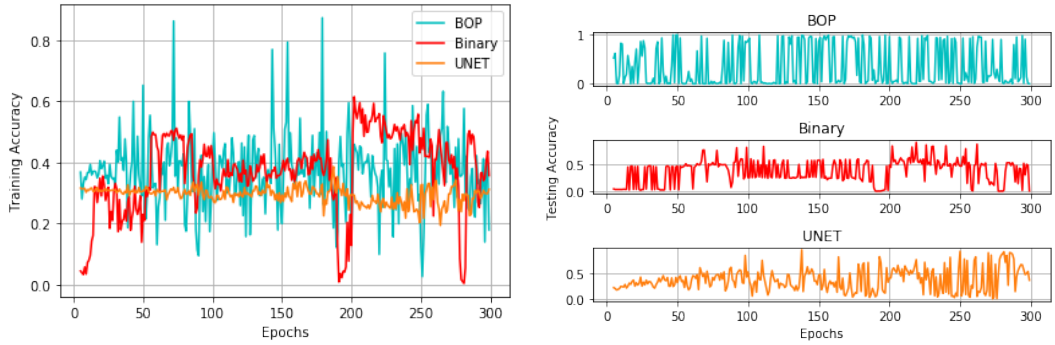

Figure 9: Comparison of Training and Testing accuracy of Binary Unet vs Binary Unet with Bop vs Conv Unet on Oxford IIIT

5 Conclusion

The paper talks about Binary Neural Network using the BoP optimizer which is developed specifically for these kind of networks. The entire training is performed and compared with the baseline model, and also extended by tuning the hyper-parameters. We were able to verify the baseline results of the paper. From the results, it is evident that there is a performance improvement with BoP approach albeit a small one. The work is further extended by varying the learning rate, dropout, and training on a different dataset. Under different hyper-parameters, the model still yields desirable results which are comparable to the baseline model.

Finally, our goal is to train the Binary network on a segmentation task to see how this works out. The segmentation task was trained using different architectures like the Binary UNet, Binary UNet with BoP, and a convolutional UNet. To conclude, the end goal of using Binary Neural Networks is to make them perform faster at the inference level when the model is deployed on self-driving cars, or edge-computing devices.

Acknowledgments

The authors of this report would like to acknowledge the faculty and TAs of the Robotics and Perception Laboratory Group at the KTH Royal Institute of Technology, Sweden. Not only did they provide the ideas that made this research possible, but their instruction and guidance during the course were critical in achieving the best possible results.

Additional Remarks

Since one of our teammate Andre Carvalho unfortunately left the course, we were not able to complete many of the experiments that we had initially planned.

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation". In: *arXiv e-prints*, arXiv:1308.3432 (Aug. 2013), arXiv:1308.3432. arXiv: 1308.3432 [cs.LG].

[2] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. "BinaryConnect: Training Deep Neural Networks with binary weights during propagations". In: *arXiv e-prints*, arXiv:1511.00363 (Nov. 2015), arXiv:1511.00363. arXiv: 1511.00363 [cs.LG].

[3] Matthieu Courbariaux et al. "Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1". In: *arXiv e-prints*, arXiv:1602.02830 (Feb. 2016), arXiv:1602.02830. arXiv: 1602.02830 [cs.LG].

[4] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv e-prints*, arXiv:1412.6980 (Dec. 2014), arXiv:1412.6980. arXiv: 1412.6980 [cs.LG].

[5] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Tech. rep. 2009.

[6] *Larq*. https://larq.dev.

[7] Zechun Liu et al. "Bi-Real Net: Enhancing the Performance of 1-bit CNNs With Improved Representational Capability and Advanced Training Algorithm". In: *arXiv e-prints*, arXiv:1808.00278 (Aug. 2018), arXiv:1808.00278. arXiv: 1808.00278 [cs.CV].

[8] Yuval Netzer et al. "Reading Digits in Natural Images with Unsupervised Feature Learning". In: (2011).

[9] O. M. Parkhi et al. "Cats and Dogs". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2012.

[10] *Rethinking Binarized Neural Network Optimization*. https://github.com/plumerai/rethinking-bnn-optimization.

[11] Herbert Robbins and Sutton Monro. "A Stochastic Approximation Method". In: *Ann. Math. Statist.* 22.3 (Sept. 1951), pp. 400–407. DOI: 10.1214/aoms/1177729586. URL: https://doi.org/10.1214/aoms/1177729586.

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *arXiv e-prints*, arXiv:1505.04597 (May 2015), arXiv:1505.04597. arXiv: 1505.04597 [cs.CV].

[13] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[14] Shilin Zhu, Xin Dong, and Hao Su. “Binary Ensemble Neural Network: More Bits per Network or More Networks per Bit?” In: *arXiv e-prints*, arXiv:1806.07550 (June 2018), arXiv:1806.07550. arXiv: 1806.07550 [cs.LG].

[15] Bohan Zhuang et al. “Structured Binary Neural Networks for Accurate Image Classification and Semantic Segmentation”. In: *arXiv e-prints*, arXiv:1811.10413 (Nov. 2018), arXiv:1811.10413. arXiv: 1811.10413 [cs.CV].