Training Fully Binary Neural Networks the Easy Way - Supplementary Materials

Alasdair Paren[1]
https://alasdair-p.github.io/Alasdair-P/
Rudra P. K. Poudel[2]
https://www.rudrapoudel.com

[1] Department of Engineering Science
University of Oxford
Oxford, UK

[2] Cambridge Research Laboratory,
Toshiba Europe Ltd,
Cambridge, UK.

1 Theoretical Justification

BNEW enjoys the same theoretical convergence rate as ProxQuant. Here we restate the result presented in ProxQuant [1]. We note that the following rate assumes that f is smooth which is not the case when including ReLU or sign functions within the network. However, as suggestied in [1] it is easy to use smoothed alternatives to these functions. For example, $\tanh(k\boldsymbol{x})$ with an appropriate choice of k can be used in place of $\text{sign}(\boldsymbol{x})$ to get a desired level of smoothness.

Theorem 1 (BNEW) *We assume that f is β-smooth. Let $F_* \triangleq \min_\Omega F_\lambda(\boldsymbol{w})$. We further assume that $\eta_t = \frac{1}{2\beta}$, $\forall t$ and we have access to the batch gradient ∇f and $\lambda_t = \lambda$ then if we use BNEW with updates (5) and (12) from the main paper for T steps we have:*

$$\|\nabla F_\lambda(\boldsymbol{w}_{T_{best}})\|^2 \leq \frac{C\beta(F_\lambda(\boldsymbol{w}_0) - F_*)}{T}, \tag{1}$$

where $C > 0$ is a constant and T_{best} is defined as $T_{best} \triangleq \text{argmin}_{1 \leq t \leq T} \|\boldsymbol{w}_t - \boldsymbol{w}_{t-1}\|$.

Proof: At each time step t we solve the following proximal problem:

$$\boldsymbol{w}_{t+1} = \underset{\boldsymbol{w}\in\Omega}{\text{argmin}} \Big\{ \frac{1}{2\eta_t}\|\boldsymbol{w} - \boldsymbol{w}_t\|^2 + f(\boldsymbol{w}_t) + \nabla f(\boldsymbol{w}_t)^\top(\boldsymbol{w} - \boldsymbol{w}_t) + \lambda R(\boldsymbol{w}) \Big\}. \tag{2}$$

As $\boldsymbol{w}_{t+1}$ minimises the above objective we get:

$$F_\lambda(\boldsymbol{w}_t) \triangleq f(\boldsymbol{w}_t) + \lambda R(\boldsymbol{w}_t), \tag{3}$$

$$\geq \frac{1}{2\eta_t}\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|^2 + f(\boldsymbol{w}_{t+1}) + \nabla f(\boldsymbol{w}_t)^\top(\boldsymbol{w}_{t+1} - \boldsymbol{w}_t) + \lambda R(\boldsymbol{w}_{t+1}). \tag{4}$$

Now using smoothness of f:

$$F_\lambda(\boldsymbol{w}_t) \geq \left(\frac{1}{2\eta_t} - \frac{\beta}{2}\right) \|\boldsymbol{w} - \boldsymbol{w}_t\|^2 + f(\boldsymbol{w}_{t+1}) + \nabla f(\boldsymbol{w}_t)^\top (\boldsymbol{w}_{t+1} - \boldsymbol{w}_t) + \lambda R(\boldsymbol{w}_{t+1}). \tag{5}$$

Thus, we have the following recursive relationship:

$$F_\lambda(\boldsymbol{w}_t) \geq F_\lambda(\boldsymbol{w}_{t+1}) + \frac{\beta}{2}\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|^2. \tag{6}$$

Telescoping (7) for $t = 0, ..., T-1$ we get:

$$F_\lambda(\boldsymbol{w}_0) \geq F_\lambda(\boldsymbol{w}_T) + \frac{\beta}{2}\sum_{t=0}^{T-1}\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|^2. \tag{7}$$

Rearranging:

$$\sum_{t=0}^{T-1}\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|^2 \leq \frac{2(F_\lambda(\boldsymbol{w}_0) - F_\lambda(\boldsymbol{w}_T))}{\beta} \leq \frac{2(F_\lambda(\boldsymbol{w}_0) - F_*)}{\beta}. \tag{8}$$

Therefore, we arrive at the proximity guarantee:

$$\min_{1 \leq t \leq T}\|\boldsymbol{w}_t - \boldsymbol{w}_{t-1}\| \leq \frac{2(F_\lambda(\boldsymbol{w}_0) - F_*)}{\beta T}. \tag{9}$$

The first-order optimality condition for $\boldsymbol{w}_{t+1}$ gives:

$$\nabla f(\boldsymbol{w}_t) + \frac{1}{\eta_t}(\boldsymbol{w} - \boldsymbol{w}_t)^2 + \nabla \lambda_t R(\boldsymbol{w}_{t+1}) = 0 \tag{10}$$

Combining with (10) and the smoothness of ℓ_z:

$$\begin{aligned} \|\nabla F_\lambda(\boldsymbol{w}_{t+1})\| &= \|\nabla f(\boldsymbol{w}_{t+1}) + \lambda R(\boldsymbol{w}_{t+1})\| && (11)\\ &= \|\nabla f(\boldsymbol{w}_{t+1}) - \nabla f(\boldsymbol{w}_t) - \frac{1}{\eta_t}(\boldsymbol{w} - \boldsymbol{w}_t)^2\| && (12)\\ &\leq \left(\frac{1}{\eta} + \beta\right)\|\boldsymbol{w} - \boldsymbol{w}_t\| = 3\beta\|\boldsymbol{w} - \boldsymbol{w}_t\| && (13) \end{aligned}$$

Inserting $t = T_{best} - 1$ and applying (9), we obtain the desired result.

$$\begin{aligned} &\|\nabla F_\lambda(\boldsymbol{w}_{T_{best}})\|^2 \leq 9\beta^2\|\boldsymbol{w}_{T_{best}} - \boldsymbol{w}_{T_{best}-1}\|^2, && (14)\\ &\|\nabla F_\lambda(\boldsymbol{w}_{T_{best}})\|^2 \leq 9\beta^2 \operatorname*{argmin}_{1 \leq t \leq T}\|\boldsymbol{w}_t - \boldsymbol{w}_{t-1}\|^2 \leq \frac{18\beta(F_\lambda(\boldsymbol{w}_0) - F_*)}{T}. && (15) \end{aligned}$$

□

Table 1: Accuracies on CIFAR-100 data set with shorter epoch budget.

EPOCHS	200	1000	200	1000
DISTILLATION	NO		YES	
STE	53.6σ0.9	55.0σ0.5	56.1σ0.4	56.8σ0.3
BMD	53.3σ0.4	54.8σ0.5	55.7σ0.6	56.8σ0.5
BOP	54.3σ0.7	55.4σ0.4	56.5σ0.2	57.7σ0.4
BNEW	52.7σ0.3	55.0σ0.5	55.8σ0.5	57.5σ0.3

2 Shorter Training Budget

Setting and Method. In this section we investigate the performance of STE, BMD, BOP, and BNEW when using a shorter training time. Here we repeat the CIFAR-100 experiments from Section 5.1 with the modification that the epoch budget is reduced to 200.

Results. The results of this investigation are shown in Table 1, along side the results produced using the 1000 Epoch training budget for ease of comparison. For all methods the shorter training duration decreases performance. BOP performs the best for the shorter budget, dropping in generalisation accuracy by at most 1.2%. In contrast BNEW, experiences a significant drop in accuracy with the shorter training duration. Dropping 2.3% and 1.7% for the CIFAR-10 and CIFAR-100 data sets, respectively. From these results we would recommend i) use of the BOP optimiser if training duration is a limiting factor and; ii) a long training budget when using BNEW.

3 Ablation Study

Setting. In this section we investigate the performance when removing various aspects of the ReActNet architectures [7]. ReActNet is a bespoke fully binary neural network architecture, which achieves state-of-the-art performance on ImageNet. ReActNet is based on MobileNet [4] with a number of modifications making it better suited for use with binary weights and activations. In Section 5.1 we provided results of training a ResNet20 [3] with these modifications using STE, BMD and BNEW. Here, using the same model we investigate the effect of removing these modifications one at a time on the performance of BNEW. We also provide results produced using the STE method for a comparison. We do this to help disentangle what architectural choices are useful irrespective of optimisation method.

Method. In order to save computation similar to Section 2 we use a 200 epoch budget and $\eta_0 = 0.01$ in combination with a linearly decaying step size schedule. We use $\eta_0 = 0.01$ with a linear decay for the pre-training and quantisation phases. We use $\lambda_{rate} = 0.01$. We again report test error and standard deviation values calculated over five independent runs with different random seeds.

Baseline. The baseline models that we perform the Ablation study are trained using distillation as detailed in Section 2 above.

Learnable Bias Layers. [7] increase the expressive power of ReActNet by using RSign and RPReLU activation functions rather than the non-parametric versions; Sign and PReLU. RSign and RPReLU are generalised activation functions, which effectively add additional real valued parameters to each channel in the form of a bias, see [7] for more details. This modification can be viewed as adding several learnable bias layers to the model. These extra layers are located before each sign activation and before and after each PReLU. However, as these biases are per channel, in practice they only increase the size and computational cost of the network marginally.

PReLU vs ReLU. To quantify the benefit of the PReLU non-linearities we train an additional model containing the learnable bias layers in combination with ReLU activations instead of the PReLU activations.

Parameter Scaling. ReActNet uses a binary quantisation scheme where the binary parameters are scaled per channel, specifically with $w \in \{-\alpha_c, \alpha_c\}$ where c indexes over the output channels of a given layer. The scalars α_c are calculated to be the mean of the absolute values of the parameters in the c^{th} output layer. Note, once a model is trained, that parameters $\boldsymbol{w}^b$ can then be converted to $\{-1,1\}^p$ by multiplying the relevant batch-norm parameters by α_c. We employ a similar scaling, but we use a learnable scale per channel, as this leads to an easier comparison. However, we note in both cases, due to the presence of batch norm, the inclusion of the scalars should have little to no effect.

Distillation. Similar to Section 5.1 we detail the performance without distillation.

Choice of Approximation for Sign Function's Gradient. In equations (1) and (2) we detail two choices to approximate the gradient of the sign function. In Section 5.1 we made use of the more complex version (2), here we investigate the effect of instead using the original Straight Through Estimator (equation (1)) as suggested in [5]. We label the model with this modification "Classic STE".

Binary First and Last Layers. It is standard to retain floating point parameters within the first and last layer of an FBNN. We investigate the effect of making these layers binary as well.

Binary Bottlenecks. A number of recent works [2, 7] recommend against binary 1x1 convolutional layers in bottlenecks. We investigate the effect on performance of ignoring this advice, and quantising these layers as well.

No Pretraining. In order to determine how important the pretraining phase is to the accuracy we try skipping this phase. We instead run the quantisation phase directly on a random initialisation.

Different Regularisation Functions. Finally, we include results for BNEW using the regularisation functions detailed in equation (8).

3.1 Results.

The results of the ablation study are shown in Table 2. Out of all the modifications considered here, binarising the first and last layer caused the largest accuracy degradation of over 10%. Binarising the bottleneck layers resulted in the second largest drop of 4%, reaffirming the suggestion of [2] that binary bottleneck layers should be avoided. Removing distillation resulted in the third largest drop in accuracy at a more modest 3%. Skipping the per-training phase and training the model from scratch resulted in a performance loss of roughly 2%. Using ReLU activations but still including the bias layers resulted in a 1.3% drop. Not using the learnable bias layers only resulted in a 0.2% drop in accuracy suggesting that in this setting these floating point weights could be excluded with minor cost, resulting in even faster inference. We found in this study that the classic STE performed better than the more complex approximation of the sign function, equation (2), introduced by [6]. However, the difference here is not statistically significant, and we would suggest trying both versions as there does not seem to be a consensus in the literature on which works better [2, 6].

The results of this ablation study suggest that the performance of an FBNN architecture is insensitive to the training method used. The differences in the changes in performance between the two methods were relatively consistent, with STE slightly outperforming BNEW due to the small epoch budget used.

As a result of the ablation study we train a additional model that uses the Classic STE to approximate the gradient of the sign function and does not include weight scaling parameters. We present the results of this experiment in the bottom row of Table 2. However combining these changes does not seem to boost performance, but does reinforce the idea that these aspects of the ReActNet architecture are not always necessary.

Table 2: *Ablation study test accuracies.*

Real Values	$67.1\sigma0.7$	
Optimiser	STE	BNEW
Baseline	$56.1\sigma0.4$	$55.8\sigma0.5$
No Learnable Bias Layers	$55.6\sigma0.5$	$55.6\sigma0.6$
No PReLU	$54.3\sigma0.1$	$54.5\sigma0.3$
No Scale	$56.2\sigma0.3$	$56.0\sigma0.5$
No Distillation	$53.6\sigma0.9$	$52.7\sigma0.5$
Classic STE	$56.5\sigma0.3$	$56.0\sigma0.4$
Binary First and Last	$43.5\sigma1.2$	$42.5\sigma0.5$
Binary Bottlenecks	$52.9\sigma0.3$	$51.8\sigma0.3$
No Pretrain	$55.0\sigma0.2$	$54.0\sigma0.5$
R_{ℓ_1} - Regulariser	NA	$54.5\sigma0.5$
R_{ℓ_2} - Regulariser	NA	$54.9\sigma0.4$
No Scale & Classic STE	$56.2\sigma0.3$	$56.0\sigma0.4$

References

[1] Yu Bai, Yu-Xiang Wang, and Edo Liberty. Proxquant: Quantized neural networks via proximal operators. *International Conference on Learning Representations*, 2019.

[2] Joseph Bethge, Haojin Yang, Marvin Bornstein, and Christoph Meinel. Back to simplicity: How to train accurate bnns from scratch? *arXiv preprint arXiv:1906.08637*, 2019.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Conference on Computer Vision and Pattern Recognition*, 2016.

[4] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. 2019.

[5] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. *Neural Information Processing Systems*, 2016.

[6] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. *European Conference on Computer Vision*, 2018.

[7] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. *European Conference on Computer Vision*, 2020.