How Do Adam and Training Strategies Help BNNs Optimization?

Zechun Liu [*1 2] **Zhiqiang Shen** [*2] **Shichao Li** [1] **Koen Helwegen** [3] **Dong Huang** [2] **Kwang-Ting Cheng** [1]

Abstract

The best performing Binary Neural Networks (BNNs) are usually attained using Adam optimization and its multi-step training variants (Rastegari et al., 2016; Liu et al., 2020). However, to the best of our knowledge, few studies explore the fundamental reasons why Adam is superior to other optimizers like SGD for BNN optimization or provide analytical explanations that support specific training strategies. To address this, in this paper we first investigate the trajectories of gradients and weights in BNNs during the training process. We show the regularization effect of second-order momentum in Adam is crucial to revitalize the weights that are dead due to the activation saturation in BNNs. We find that Adam, through its adaptive learning rate strategy, is better equipped to handle the rugged loss surface of BNNs and reaches a better optimum with higher generalization ability. Furthermore, we inspect the intriguing role of the real-valued weights in binary networks, and reveal the effect of weight decay on the stability and sluggishness of BNN optimization. Through extensive experiments and analysis, we derive a simple training scheme, building on existing Adam-based optimization, which achieves 70.5% top-1 accuracy on the ImageNet dataset using the same architecture as the state-of-the-art ReActNet (Liu et al., 2020) while achieving 1.1% higher accuracy. Code and models are available at `https://github.com/liuzechun/AdamBNN`.

1. Introduction

Binary Neural Networks (BNNs) have gained increasing attention in recent years due to the high compression ratio (Rastegari et al., 2016) and the potential of being accelerated with logic computation on hardware (Zhang et al., 2019). Their applications range from supervised learning, e.g., classification (Courbariaux et al., 2016), segmentation (Zhuang et al., 2019), pose estimation (Bulat et al., 2019) to the self-supervised learning (Shen et al., 2021).

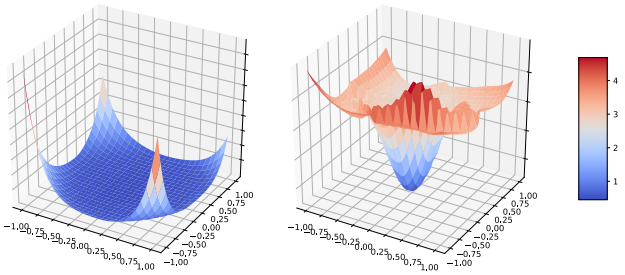


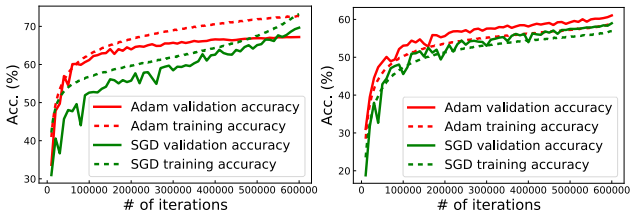
Figure 1. The actual optimization landscape from real-valued and binary networks with the same architecture (ResNet-18). We follow the method in (Li et al., 2018) to plot the landscape.

Despite the high compression ratio of BNNs, the discrete nature of the binary weights and activations poses a challenge for its optimization. It is widely known that conventional deep neural networks rely heavily on the ability to find good optima in a highly non-convex optimizing space. Different from real-valued neural networks, binary neural networks restrict the weights and activations to discrete values (-1, +1), which naturally, will limit the representational capacity of the model and further result in disparate optimization landscapes compared to real-valued ones. As illustrated in Figure 1, BNNs are more chaotic and difficult for optimization with numerous local minima compared to real-valued networks. These properties differentiate BNNs from real-valued networks and impact the optimal optimizer and training strategy design.

Since Courbariaux et al. (Courbariaux et al., 2016) adopted Adam as the optimizer for BNNs, multiple researchers independently observed that better performance could be attained by Adam optimization for BNNs (Bethge et al., 2020; Liu et al., 2020; Brais Martinez, 2020). However, few of these works have analyzed the reasons behind Adam's superior performance over other methods, especially the commonly used stochastic gradient descent (SGD) (Robbins & Monro, 1951) with first momentum.

[*]Equal contribution [1]Hong Kong University of Science and Technology [2]Carnegie Mellon University [3]Plumerai. Correspondence to: Zhiqiang Shen <zhiqians@andrew.cmu.edu>, Zechun Liu <zechun.liu@connect.ust.hk>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.



(a) real networks (b) binary networks

Figure 2. The top-1 accuracy curves of the real-valued and binary network (ResNet-18 based) trained on ImageNet. On the real-valued network, SGD achieves higher accuracy with better generalization ability in the final few iterations. Binarization has a strong regulating effect, resulting in the validation accuracy being higher than the training accuracy. Adam outperforms SGD under this circumstance.

Recent theoretical work from Wilson et al. (Wilson et al., 2017) empirically shows that adaptive learning rate methods like Adam reach fewer optimal minima than SGD with momentum, meaning that minima found by SGD generalize better than those found by Adam. This matches experience with real-valued neural networks where state-of-the-art results for many tasks in Computer Vision (Tan & Le, 2019) and Machine Translation (Wu et al., 2016) are still obtained by plain SGD with momentum. It seems counter-intuitive considering that Adam comes with better convergence guarantee and should deliver better performance over SGD. We observe the real-valued networks are fairly powerful to "overfit" on the training data as shown in Figure 2 (a), but as we will demonstrate later, this may be not true for BNNs. We observe that BNNs are usually under-fitting on the training set due to the limited model capacity (the performance on validation is higher than that on training set), even if we train BNNs thoroughly with a longer budget. From Figure 2 (b), it is evident that the validation accuracy of SGD on binary networks fluctuates more compared to Adam, which indicates that on binary training, SGD easily gets stuck in the rugged surface of the discrete weight optimization space and fails to find generalizable local optima.

Based on these observations, in this paper, we investigate the fundamental reasons why Adam is more effective for BNNs than SGD. During BNN training, a proportion of gradients tends to be zero due to the activation saturation effect. If using SGD as the optimizer, the updating step of the individual weight aligns with the corresponding gradient in the magnitude, making it hard to flip those "dead" weights out of a bad initialization or local minima. Intuitively, revitalizing the "dead" weights with appropriate gradients can significantly improve the accuracy of BNNs, which is further supported by our visualization results and the final accuracy. According to our experiments, the normalization effect from the second momentum of Adam rescales the updating value element-wisely based on the historical gradients, effectively resolving the "dead" weights issue.

Besides comparing Adam to SGD, we further explore how training strategies affect BNN optimization. Previous works proposed different training strategies: Yang et al. (Yang et al., 2019) proposed to progressively quantize the weights from 16 bits to 1 bit. Zhuang et al. (Zhuang et al., 2018) proposed binarizing weights first and binarizing activations in the second step. More recently Martinez et al. (Brais Martinez, 2020) proposed a two-step strategy to binarize activations first and then binarize weights. These works involve complex training strategy designs but seldom explain the reason behind those designs. Instead of proposing a new training strategy, in the second part of our work, we explain the mechanisms behind BNN training strategies, from an important but overlooked angle – weight decay. We quantify the effects of weight decay on the BNN optimization's stability and initialization dependency with two metrics, FF ratio and C2I ratio, respectively. Guided by these metrics, we identify a better weight decay scheme that promotes the accuracy of the state-of-the-art ReActNet from 69.4% to 70.5%, surpassing all previously published studies on BNNs.

Unlike previous studies that focus on designing network architectures for BNNs, we focus on the investigation of optimizers and training strategy, which we think is valuable for maximizing the potential in a given structure for better performance. All of our experiments are conducted on the full ImageNet[1], which is more reliable. We believe our exploratory experiments will be beneficial for the research on BNNs optimization and may inspire more interesting ideas along this direction.

Contributions. In summary, we address the following issues and our contributions are as follows:

- We provide thorough and fair comparisons among different optimizers for BNN optimization, especially between Adam and SGD, on the large-scale ImageNet dataset. We further design several metrics to analyze the patterns beneath the binary behavior and present a simple visualization method based on the alteration of gradients and weights inside training.
- We explain the difficulties that arise from a non-adaptive learning rate strategy by visualizing these trajectories and show that optimization lies in extremely rugged surface space. We conclude that gradient normalization is crucial for BNN optimization.
- We further examine the existing practice in BNN optimization strategy design and provide in-depth analysis on the weight decay effect. Based on these analyses, we propose practical suggestions for optimizing BNNs. These techniques help us to train a model with 1.1% higher accuracy than the previous state-of-the-art results.

[1]Several previous works conduct experiments on small datasets like MNIST and CIFAR-10/100, and sometimes draw conclusions that are inconsistent with experiments on large-scale/real-world.

2. Related Work

Research on binary neural network optimization can be mainly divided into several aspects:

Structure Adjustment Previous attempts to improve BNNs are mainly paid on network structure design, including adding real-valued shortcuts (Liu et al., 2018b;a; 2020), or real-valued attention blocks (Brais Martinez, 2020), expanding the channel width (Mishra et al., 2017; Zhuang et al., 2019), ensemble more binary networks (Zhu et al., 2019) or use a circulant convolution (Liu et al., 2019). These works provide advanced structures that bring breakthroughs in accuracy. In this work, we are motivated to disambiguate the binary optimization process, which is orthogonal to the structural design.

Gradient Error Reduction and Loss function Design Some studies pay attention to reduce the gradient error of the BNNs, for example, XNOR-Net (Rastegari et al., 2016) uses a real-valued scaling factor multiplying with the binary weights and activations, and ABC-Net (Lin et al., 2017) adopts more weight bases. IR-Net (Qin et al., 2020) propose Libra-PB to simultaneously minimize both quantization error and information loss. A few works adjust the loss functions. Hou et al. proposed loss-aware binarization (Hou et al., 2016) using the proximal Newton algorithm with the diagonal Hessian approximation to directly minimize the loss w.r.t. binary weights. Ding et al. proposed activation regularization loss to improve BNN training (Ding et al., 2019). These studies also aim to resolve the discreteness-brought optimization challenge in binary neural networks. Instead, we scrutinize another important yet less investigated angle, the optimizer and optimization strategy reasoning.

Optimizer Choice and Design Recently, many binary neural network choose Adam over SGD, including BNN (Courbariaux et al., 2016), XNOR-Net (Rastegari et al., 2016), Real-to-Binary Network (Brais Martinez, 2020), Structured BNN (Zhuang et al., 2019), ReActNet (Liu et al., 2020), etc. Helwegen et al. proposed a new binary optimizer design based on Adam (Helwegen et al., 2019). Empirical studies of binary neural network optimization (Alizadeh et al., 2018; Tang et al., 2017) also explicitly mention that Adam is superior to SGD and other optimization methods. However, the reason why Adam is suitable for binary network optimization is still poorly understood. In this study, we investigate the behavior of Adam, attempting to bring attention to the binary optimizer understanding and improving the binary network performance within a given network structure, which we hope is valuable for the community.

Training Strategy Multiple works proposed different multi-step training strategies to enhance the performance of BNNs. Zhuang et al. (Zhuang et al., 2018) proposed to first quantize the weights then quantize both weights and activations. Following (Zhuang et al., 2018), Yang et al. (Yang et al., 2019) proposed to progressively quantize weights and activations from higher bit-width to lower bit-width. Recent studies (Brais Martinez, 2020; Liu et al., 2020) proposed to binarize activation first, then in the second stage, further binarize the weights. Those previous work each proposed their own training techniques, but seldom generalized techniques into the reasons behind, which also brings confusion to followers in determining which technique they can use in their circumstance. In this work, we analyze the foundations of choosing optimization strategies beyond providing a possible solution, in hope of inspiring more interesting solutions in this area.

3. Methodology

This section begins by introducing several observations from real-valued networks and binary neural networks (BNNs) training. We observe the generalization ability of Adam is better than SGD on BNNs, as shown in Figure 2. This phenomenon motivates us to ask why SGD works better for a real-valued classification network, but loses its superiority in binary neural network optimization. Start by this, we visualize the activation saturation phenomenon during optimizing an actual binary neural network and deliberate its effects on gradient magnitude in Section 3.2.1. Then we observe that activation saturation will cause the unfair training problem on channel weights as described in Section 3.2.2. Further, for clear explanation we construct an imaginary two-dimensional loss landscape containing sign functions to mimic a simplified optimization process of BNNs with activation binarization in Section 3.2.3, and we analysis how Adam can help conquer the zero-gradient local minima. Moreover, we point out that the real-valued weights in BNNs can be regarded as the *confidence* score, as described in Section 3.2.4, making BNN optimization intricate. Thus, we define several metrics in Section 3.3 to depict the property of BNNs and measure the goodness of a BNN training strategy. Lastly, we provide practical suggestions for optimizing BNNs.

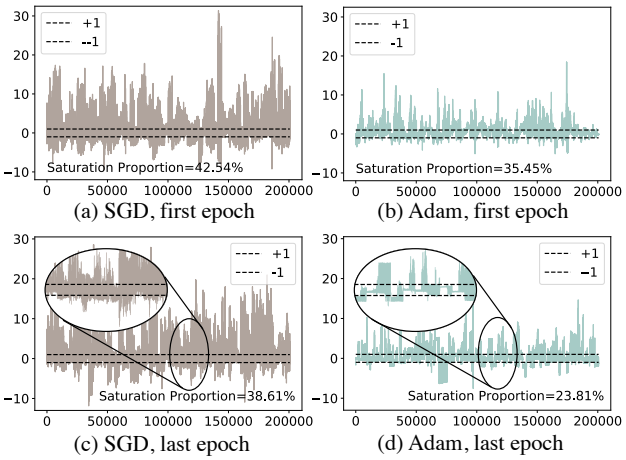
3.1. Preliminaries

Binary neural network optimization is challenging because weights and activations of BNNs are discrete values in {-1, +1}. In particular, in the forward pass, real-valued weights and activations are binarized with the sign function.

$$a_b = \mathrm{Sign}(a_r) = \begin{cases} -1 & \text{if } a_r < 0 \\ +1 & \text{otherwise} \end{cases} \tag{1}$$

$$w_b = \frac{||W_r||_{l1}}{n}\mathrm{Sign}(w_r) = \begin{cases} -\frac{||W_r||_{l1}}{n} & \text{if } w_r < 0 \\ +\frac{||W_r||_{l1}}{n} & \text{otherwise} \end{cases} \tag{2}$$

Note that, the real-valued activations a_r are the outputs of the previous layers, generated by the binary or real-valued convolution operations. The real-valued weights w_r are



(a) SGD, first epoch (b) Adam, first epoch

(c) SGD, last epoch (d) Adam, last epoch

Figure 3. Activation distributions in binary ResNet-18 structure from different optimizers on ImageNet. Dotted lines are the up (+1) and low (-1) bounds. We plot the input activation to the first binary convolution and we observe that both SGD and Adam optimized BNNs experienced activation saturation. However, Adam can alleviate activation saturation during optimization compared to SGD, as shown in the zoom-in views in (c) and (d). We further count the number of activations that are over the bounds for SGD and Adam, the percentages are 42.54% and 35.45% respectively after the first epoch, 38.61% and 23.81% after the last epoch. The activation saturation proportion from Adam optimization is significantly lower than SGD. More details please refer to Section 3.2.1.

stored as *latent* weights to accumulate the small gradients. *Latent* refers to that the weights are not used in the forward pass computation. Instead, the sign of real-valued latent weights multiplying the channel-wise absolute mean ($\frac{1}{n}||W_r||_{l1}$) is used for updating binary weights (Rastegari et al., 2016).

In the backward pass, due to the non-differentiable characteristic of the sign function, the derivative of $clip(-1, a_r, 1)$ function is always adopted as the approximation to the derivative of the sign function (Rastegari et al., 2016). It is noteworthy that, because the *sign* is a function with bounded range, the approximation to the derivative of the sign function will encounter a zero (or vanishing) gradient problem when the activation exceeds the effective gradient range ($[-1, 1]$), which leads to the optimization difficulties that will be discussed in Section 3.2.1.

3.2. Observations

3.2.1. Activation Saturation on Gradients

Activation saturation is the phenomenon that the absolute value of activations exceeds one and the corresponding gradients are suppressed to be zero, according to the definition of approximation to the derivative of the sign function (Ding et al., 2019). From our observation, activation saturation exists in every layer of a binary network and it will critically affect the magnitude of gradients in different channels. In Figure 3, we visualize the activation distributions of the first binary convolution layer. We can observe that many activations exceed the bounds of -1 and +1, making the gradient passing those nodes become zero-valued. According to the Chain Rule (Ambrosio & Dal Maso, 1990), the gradients are extremely vulnerable to the activation saturation in latter layers and thus will vibrate tempestuously in their corresponding magnitudes.

3.2.2. Fairness in Weight Training

Unfair training is the phenomenon that the weights in some channels are not optimized to learn meaningful representations. Given different batches of images, the activation saturation usually occurs on different activation channels. In these channels, the gradient will always stay small in our observation, which causes unfair training. Note that the weights refer to the real-valued latent weights in the binary neural network. The magnitude of these real-valued weights are regarded as *'inertial'* (Helwegen et al., 2019), indicating how likely the corresponding binary weights are going to change their signs.

To measure the effect of unfair training, we calculate the Channel-wise Absolute Mean (CAM) to capture the average magnitude of real-valued weights within a kernel, which is represented as *red* hyphens in Figure 4 and Figure 5. The definition of CAM is as follows:

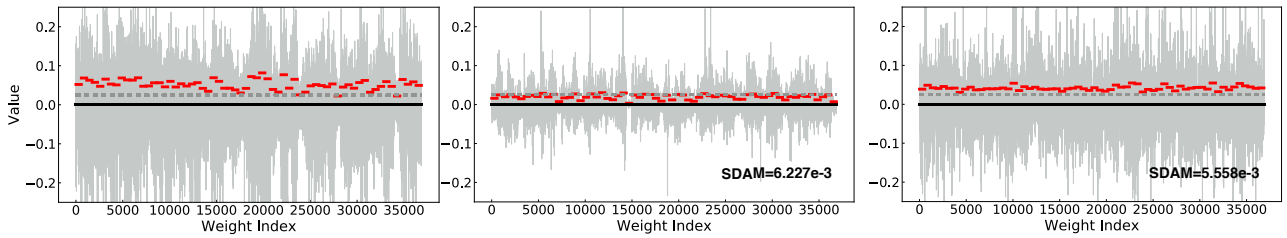
$$\text{CAM} = \frac{1}{N_{in} \cdot k \cdot k} \sum_{c=1}^{N_{in}} \sum_{i=1}^{k} \sum_{j=1}^{k} |w_{\{c,i,j\}}| \qquad (3)$$

where N_{in} is the number of input channels, w is the weights in BNNs, c is the channel index, i, j are the element position in c-channel, and k is the kernel size. We can see that when using SGD, the CAM of latent weights in a binary network are small in their values (Figure 4 (b)) compared with their real-valued counterparts (Figure 4 (a)) and is also higher in variance, which reflects the unbalanced weight training inside the SGD optimized binary network.

To measure the uniformness of the trained latent real-valued weight magnitude, we propose the Standard Deviation of the Absolute Mean (SDAM) of the real-valued weight magnitude on each output channel. The statistics of SDAM for SGD and Adam are shown in Figure 4. It is evident that the SDAM of Adam is lower than that of SGD, revealing higher fairness and stability in the Adam training than SGD.

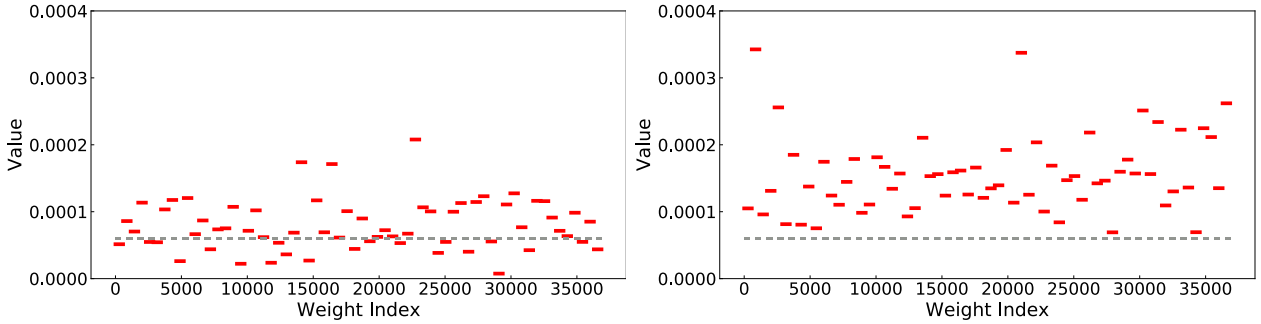
3.2.3. Why is Adam Better than SGD?

For better illustration, we plot a two-dimensional loss surface of a network with two nodes where each node contains a sign function binarizing its input. As shown in Figure 6 (a), the sign functions result in a discretized loss landscape



(a) Real-valued network with SGD (b) Binary network with SGD (c) Binary network with Adam

Figure 4. The weight value distribution in the first binary convolutional layer after training one epoch. For clarity, we use *red* hyphens to mark the Channel-wise Absolute Mean (CAM) of real-valued weights in each kernel. The *grey* dotted line denotes the minimum CAM value (0.0306) of weights in the Adam optimized binary network. Compared to Adam, SGD optimization leads to much lower CAM value, and higher Standard Deviation (SDAM), which indicates that the weights optimized with SGD are not as fair (well-trained) as those with Adam. More detailed analysis can be found in Section 3.2.3.



(a) Binary network with SGD (b) Binary network with Adam

Figure 5. The update value distribution of weights in the first binary convolutional layer after trained with one epoch. For clarity, we omit the original update value distribution and use *red* hyphens to mark the Channel-wise Absolute Mean (CAM) of the weights' update values in each kernel. In this layer, 34.3% of the kernels in SGD have a lower CAM than the minimum CAM in Adam. See also Section 3.2.3.

with zero gradients at almost all input intervals, making the landscape infeasible to be optimized with gradient descent.

In literature, the derivative of $clip(-1, a_r, 1)$ function is always adopted as the approximation to the derivative of the sign function. Thus, the actual landscape where gradients are computed is constructed with clip nodes. In Figure 6 (b), the approximated gradients of binary activations retain their values in both direction only when both inputs land in the interval of [-1, 1], denoted as the slashed area in Figure 6 (b). Outside this region, the gradient vector either has value in only one direction or contains zero value in both directions, which is the so-called flattened region.

During the actual BNN optimization, the activation value depends on the input images and will vary from batch to batch, which is likely to exceed [-1, 1]. This activation saturation effect in turn results in the gradient vanishing problem. For illustration, on this 2D-loss surface, we denote the starting point of optimization in grey circles. Started with the same sequence of gradients, the SGD optimizer computes the update value with the first momentum by definition: $v_t = \gamma v_{t-1} + g_t$, where g_t denotes the gradient and v_t denotes the first momentum for weight update. While the update value in Adam is defined as: $u_t = \frac{\hat{v}_t}{\sqrt{\hat{m}_t}+\epsilon}$, $\hat{v}_t$ and $\hat{m}_t$ denote exponential moving averages of the gradient and the squared gradient, respectively. At the flattened region, with $\hat{m}_t$ tracing the variance of gradients, the update value u_t is normalized to overcome the difference in the gradient value. In contrast to SGD that only accumulates the first momentum, the adaptive optimizer, Adam, naturally leverages the accumulation in the second momentum to amplify the learning rate regarding the gradients with small historical values. As shown in Figure 6 (c) and (d), Adam contains higher proportion in update value of x direction compared to SGD when the gradient in x direction vanishes. In our experiments, we found this property crucial for optimizing BNNs with more rugged surfaces and local flatten regions due to binarization. Figure 5 also shows the update values of each iteration with CAM form in training an actual BNN. It confirms that with Adam training, the update values are usually larger than a threshold but with SGD, the values are very close to zero. As a result, "dead" weights from saturation are easier to be re-activated by Adam than SGD.

3.2.4. Physical Meaning of Real-Valued Weight

The superiority of Adam for BNNs is also fortified in the final accuracy. As shown in Figure 7 (a), Adam achieves

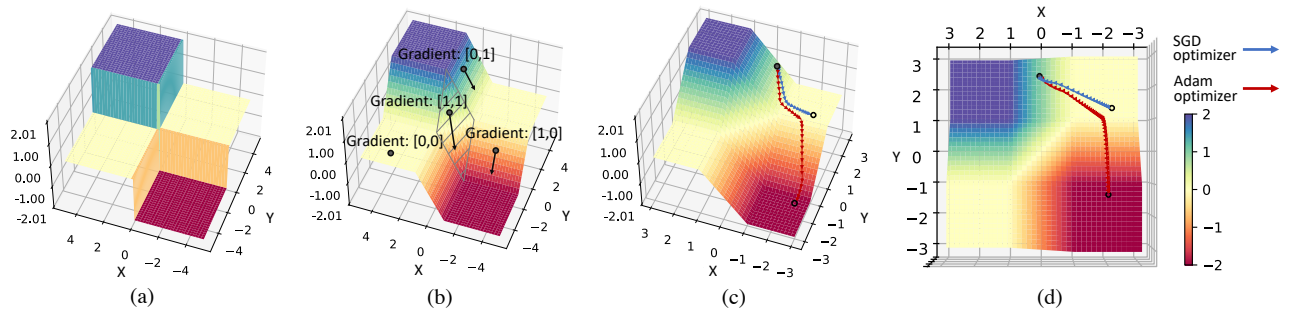


Figure 6. The loss landscape visualization of a network constructed with the summation of two binary nodes. (a) the loss surface of the binary network in the forward pass, binarization functions $sign(x)$ discretized the landscape, (b) the loss surface for actual optimization after using the derivative of $clip(-1, x, 1)$ in approximating the derivative of $sign(x)$, (c) the comparison between using SGD optimizer and Adam optimizer in conquering the zero gradient local minima, (d) the top view of the actual optimization trajectory.

61.49% top-1 accuracy, comparing to 58.98% of SGD in Figure 7 (b) with a consistent setting imposed on both experiments in terms of hyper-parameters and network structures. Furthermore, we investigate the weight distribution in Figure 7 of final models and obtain some interesting discoveries. We find that the real-valued latent weights of better-performing models usually emerge to three peaks, one is around zero and the other two are beyond -1 and 1. For those poorly optimized models with SGD, the distributions of real-valued weights only contain one peak centering around zero. The physical significance of real-valued weights indicates the degree of how easy or difficult the corresponding binary weights can switch their signs (-1 or +1) to the opposite direction. If the real-valued weights are close to the central boundary (0), it will be simple for them to fall or bias to -1 or +1 through a few steps of gradient updating, making the whole network unstable. Thus, it is not far-fetched that real-valued weights can be regarded as the *confidence* of a binary value to be -1 or +1, as also being mentioned in (Helwegen et al., 2019). From this perspective, the weights learned by Adam are definitely more confident than those learned by SGD, which consistently verifies the conclusion that Adam is a better optimizer to use for binary neural networks.

3.3. Metrics for Understanding BNN Optimization

Given the superiority of Adam over SGD, we take this finding further and investigate the training strategy for BNNs. Based on the intriguing fact that the BNN optimization relies on real-value weights for gradient accumulation and their signs for loss computation, BNN optimization is intractable compared to real-valued networks. Thus for better revealing the mechanism of the perplexing BNN training, we propose two metrics to depict the training process and further find that the weight decay added on the real-valued latent weight plays a non-negligible role in controlling the binary weights evolving.

3.3.1. Weight Decay in BNN Optimization

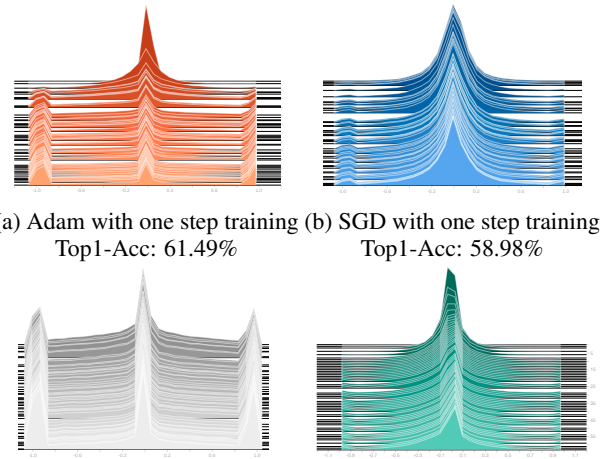
In a real-valued neural network, weight decay is usually used to regularize the real-valued weights from growing too large, which prevents over-fitting and helps to improve the generalization ability (Krogh & Hertz, 1992).

However, for a binary neural network, the effect of weight decay is less straightforward. As the absolute values of weights in BNNs are restricted to -1 and +1, the weight decay is no longer effective to prevent the binary weights from being extremely large. Moreover, in a binary neural network, the weight decay is applied to the real-valued latent weights. Recall that in Section 3.2.4, the magnitude of real-valued weights in BNNs can be viewed as the *confidence* of corresponding binary weights to their current values. Adding weight decay on these real-valued weights is actually attempting to decay the *confidence* score.

From this perspective, the weight decay will lead to a dilemma in binary network optimization between the stability and the dependency of weight initialization. With high weight decay, the magnitude of the *latent* weights is regularized to be small, making the corresponding binary weights "less confident" in their signs, and further prone to switch their signs frequently, i.e., reducing the stability in optimization. With smaller or even zero weight decay, the *latent* weights tend to move towards -1 and +1, the corresponding binary weights will be more stable to stay in the current status. However, this is a trade-off since larger gradients are required to promote the weights to switch their signs in order to overcome the "dead" parameters issue. That is to say, with small or zero weight decay, the performance of a network will be influenced by initialization critically.

3.3.2. Quantification Metrics

To quantify these two effects (network stability and initialization dependency), we introduce two metrics: the flip-flop (FF) ratio for measuring the optimization stability, and the



(a) Adam with one step training Top1-Acc: 61.49% (b) SGD with one step training Top1-Acc: 58.98%

(c) Adam with two step training Top1-Acc: 63.23% (d) SGD with two step training Top1-Acc: 61.25%

Figure 7. The final weight distribution. We found that Adam has more latent real-valued weights with larger absolute values compared to SGD. Since the real-valued weights can be viewed as the *confidence score* of the corresponding binary weights in their current sign, Adam-optimized binary networks are more confident in values than that of SGD, and the final accuracy is also higher.

correlation-to-initialization (C2I) ratio for measuring the dependency on initialization. The FF ratio is defined as:

$$\mathbf{I_{FF}} = \frac{|\text{Sign}(w_{t+1}) - \text{Sign}(w_t)|_{abs}}{2}, \tag{4}$$

$$\mathbf{FF_{ratio}} = \frac{\sum_{l=1}^{L}\sum_{w\in W_l}\mathbf{I_{FF}}}{N_{\text{total}}}, \tag{5}$$

where $\mathbf{I_{FF}}$ is the indicator of whether a weight changes its sign after the updating at iteration t. N_{total} is the total number of weights in a network with L convolutional layers. $\mathbf{FF_{ratio}}$ denotes the ratio of flip-flops, *i.e.,* percentage of weights that change their signs.

Then we define C2I ratio as:

$$\mathbf{I_{C2I}} = \frac{|\text{Sign}(w_{\text{final}}) - \text{Sign}(w_{\text{init}})|_{abs}}{2}, \tag{6}$$

$$\mathbf{C2I_{ratio}} = 1 - \frac{1}{2}\frac{\sum_{l=1}^{L}\sum_{w\in W_l}\mathbf{I_{C2I}}}{N_{\text{total}}}, \tag{7}$$

where $\mathbf{I_{C2I}}$ is the indicator of whether a weight has different sign to its initial sign and $\mathbf{C2I_{ratio}}$ denotes the correlation between the signs of final weights and the initial values.

Here we study the FF ratio and C2I ratio for different weight decay values. From Table 1, it is easy to find that the FF ratio is in negative correlation with C2I ratio. With the increase of weight decay, the FF ratio increases exponentially while the C2I ratio decreases linearly. This indicates that some flip-flops do not contribute to the final weights, but just harm the training stability.

Table 1. The FF ratio, C2I ratio and Top-1 accuracy by Adam optimization with different weight decay. Note that the FF ratios in this table are averaged over the total training iterations.

	Weight decay	FF ratio	C2I ratio	Top1-acc
One-step	1e-5	2.33×1e-3	0.4810	61.73
	5e-6	1.62×1e-3	0.4960	**61.89**
	0	2.86 ×1e-4	0.5243	61.49
	-1e-4	1.07 ×1e-7	0.9740	26.21
Two-step	*Step1:* 1e-5 *Step2:* 0	4.89×1e-4	0.6315	62.63
	Step1: 5e-6 *Step2:* 0	4.50×1e-4	0.6636	**63.23**

In this experiment, we found using the weight decay of 5e-6 produces the highest accuracy. Further, we discover that a particular two-step training scheme (Brais Martinez, 2020; Liu et al., 2020) can disentangle the negative correlation between FF ratio and C2I ratio.

3.3.3. Practical Training Suggestion

Intrinsically, the dilemma of whether adding weight decay on real-valued latent weights originates from the fact that the binary weights are discrete in value. For real-valued latent weights around zero, a slight change in value could result in a significant change in the corresponding binary weights, thus making it fairly tricky to encourage real-valued latent weights to gather around zero.

Interestingly, we found that a good weight decay scheme for the recent two-step training algorithm (Brais Martinez, 2020; Liu et al., 2020) can disentangle this dilemma. In *Step1*, only activations are binarized and the real-valued weights with weight decay are used to accumulate small update values. Since real-valued networks have no worries about the FF ratio, we can simply add weight decay to harvest the benefit of low initialization dependency. Then, in *Step2*, we initialize latent real weights in the binary networks with weights from *Step1*, and enforce a weight decay of 0 on them. With this operation, we can reduce the FF ratio to improve stability and utilize the good initialization from *Step1* (similar to pre-training) rather than the random parameters. In this stage, a high C2I ratio will not harm the optimization. From this perspective, we found that 5e-6 as weight decay performs best for balancing the weight magnitude for a good initialization in *Step2*.

As shown in Figure 7 (c), more real-valued weights in two-step training tend to gather around -1 and +1, indicating that this strategy is more confident than one-step. By simply eliminating the undesirable weight decay value just by looking at the FF ratio in the early epochs we can find a good weight decay with fewer trials and errors. We will see in Section 4 that our training strategy outperforms the state-of-the-art ReActNet by 1.1% with identical architectures.

Table 2. Comparison with state-of-the-art methods that binarize both weights and activations.

Networks	Top1 Acc %	Top5 Acc %
BNNs (Courbariaux et al., 2016)	42.2	67.1
ABC-Net (Lin et al., 2017)	42.7	67.6
DoReFa-Net (Zhou et al., 2016)	43.6	-
XNOR-ResNet-18 (Rastegari et al., 2016)	51.2	69.3
Bi-RealNet-18 (Liu et al., 2018b)	56.4	79.5
CI-BCNN-18 (Wang et al., 2019)	59.9	84.2
MoBiNet (Phan et al., 2020a)	54.4	77.5
BinarizeMobileNet (Phan et al., 2020b)	51.1	74.2
PCNN (Gu et al., 2019)	57.3	80.0
StrongBaseline (Brais Martinez, 2020)	60.9	83.0
Real-to-Binary Net (Brais Martinez, 2020)	65.4	86.2
MeliusNet29 (Bethge et al., 2020)	65.8	–
ReActNet ResNet-based (Liu et al., 2020)	65.5	86.1
ReActNet-A (Liu et al., 2020)	69.4	88.6
StrongBaseline + Our training strategy	63.2	84.0
ReActNet-A + Our training strategy	**70.5**	**89.1**

Table 3. Comparison of computational cost between the state-of-the-art methods and our method.

Networks	BOPs $\times 10^9$	FLOPs $\times 10^8$	OPs $\times 10^8$
XNOR-ResNet-18 (Rastegari et al., 2016)	1.70	1.41	1.67
Bi-RealNet-18 (Liu et al., 2018b)	1.68	1.39	1.63
CI-BCNN-18 (Wang et al., 2019)	–	–	1.63
MeliusNet29 (Bethge et al., 2020)	5.47	1.29	2.14
StrongBaseline (Brais Martinez, 2020)	1.68	1.54	1.63
Real-to-Binary (Brais Martinez, 2020)	1.68	1.56	1.83
ReActNet-A (Liu et al., 2020)	4.82	0.12	0.87
StrongBaseline + Our training strategy	1.68	1.54	1.80
ReActNet-A + Our training strategy	4.82	0.12	0.87

4. Experiments

4.1. Dataset and Implementation Details

All the analytical experiments are conducted on the ImageNet 2012 classification dataset (Russakovsky et al., 2015). We train the network for 600K iterations with batch size set to 512. The initial learning rate is set to 0.1 for SGD and 0.0025 for Adam, with linear learning rate decay. We also adopt the same data augmentation in (Brais Martinez, 2020) and the same knowledge distillation scheme as (Liu et al., 2020) for training ReActNet structures. For a fair comparison of optimization effects, we use the same network structures as StrongBaseline in (Brais Martinez, 2020) for all the illustrative experiments and compared our training strategy on two state-of-the-art network structures including StrongBaseline, and ReActNet (Liu et al., 2020).

4.2. Comparison with State-of-the-Arts

Our training strategies bring constant improvements to both structures. As shown in Table 2. With the same network architecture, we achieve 2.3% higher accuracy than the StrongBaseline (Brais Martinez, 2020). When applying our training strategy to the state-of-the-art ReActNet (Liu et al., 2020), it further brings 1.1% enhancement and achieves 70.5% top-1 accuracy, surpassing all previous BNN models.

Our training strategy will not increase the OPs as we use identical structures as the baselines: StrongBaseline (Brais Martinez, 2020) and ReActNet (Liu et al., 2020). Table 3 shows the computational costs of the networks we utilized in experiments. StrongBaseline is a ResNet-18 based binary neural network, and it has similar OPs as Bi-RealNet-18 (Liu et al., 2018b) and Real-to-Binary Network (Brais Martinez, 2020). ReActNet is a MobileNet-based BNN, and it contains small overall OPs than other binary networks.

Table 4. Comparison of different binarization orders in two-step training on the StrongBaseline (Brais Martinez, 2020) structure.

	Top1 Acc	Top5 Acc
first binarize weight then binarize activation (*BWBA*)	60.17	82.05
first binarize activation then binarize weight (*BABW*)	**63.23**	**84.02**

Table 5. Comparison between Adam and other adaptive methods.

	Adam	RMS-prop	Ada-Grad	Ada-Delta	AMS-Grad	Ada-Bound
Top1-acc	**61.49**	57.90	50.74	56.90	60.71	58.13
Top5-acc	**83.09**	79.93	74.62	79.47	82.44	80.58

4.3. Ablation Study

4.3.1. Comparison of Adam and SGD under Different Learning Rates

In Figure 8, we illustrate the Top-1 accuracy curves with different learning rates. To control variables, experiments are done with one-step training strategy on the ImageNet dataset with the StrongBaseline (Brais Martinez, 2020) structure. In general, Adam can achieve higher accuracy across a variety of learning rate values and is also more robust than SGD. Besides, we observe that Adam enjoys small learning rates. The reason is that Adam adopts the adaptive method to update the gradients, which will amplify the actual learning rate values during training, so it requires a smaller initial learning rate to avoid update values being too large.

4.3.2. Two-step Training

To reassure the credibility of choosing the suggested two-step training algorithm, we make a controlled comparison between different training schemes. In Table 4, our suggested order which first binarizes activations then weights (*BABW*) obtained a 2.93% better accuracy over the reversed

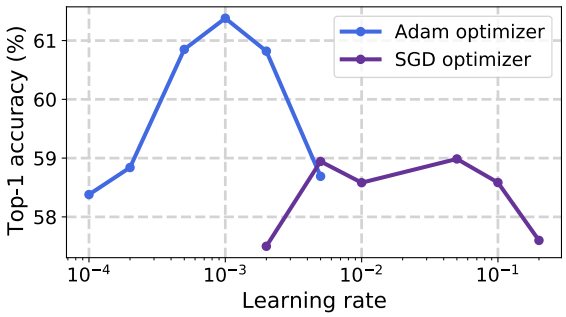


Figure 8. Accuracy *vs.* initial learning rate on Adam and SGD.

order (*BWBA*). In *BWBA*, binary weights are adopted in both steps, which are confined to be discrete. So compared to the real-valued weights in *Step1* of *BABW*, it is harder for binary weights in *Step1* of *BWBA* to be well optimized for delivering a good initialization for *Step2*. Thus *BWBA* can not achieve the effect of breaking the negative correlation between FF ratio and C2I ratio.

4.3.3. Comparison with Other Adaptive Methods

In this experiment, the initial learning rates for different optimizers are set to the PyTorch (Paszke et al., 2019) default values (0.001 for Adam, 0.01 for RMSprop, 0.01 for AdaGrad, 1.0 for AdaDelta, and 0.001 for AMSGrad). For Adabound, we adopt the default learning rate schedule in (Luo et al., 2019) by setting the initial learning rate to 0.001 and transiting to 0.1. The weight decay is set to 0. For fair comparison, these experiments are carried out with one-step training on the ImageNet dataset with the StrongBaseline (Brais Martinez, 2020) structure.

In Table 5, Adam (Kingma & Ba, 2014) achieves similar accuracy with its variant AMSGrad (Reddi et al., 2019), and better results than other adaptive methods. RMSprop (Tieleman & Hinton, 2012) and Adadelta (Zeiler, 2012) are adaptive methods without using the first momentum of the gradients. In binary neural networks, since the gradients with respect to the discrete weights are noisy, the first momentum is also crucial for averaging out the noise and improve the accuracy. AdaGrad (Duchi et al., 2011) is known that its accumulation of the squared gradients in the denominator will keep growing during training, causing the learning rate to shrink and eventually become infinitesimally small, and preventing the algorithm to acquire additional knowledge. Thus the performance of AdaGrad is modest. As a variant of Adam, AMSGrad uses "long-term memory" of the past gradients to avoid extreme adaptive learning rate, which achieves comparable accuracy as Adam on a binary classification network, while AdaBound (Luo et al., 2019) is proposed to smoothly transit from Adam to SGD in order to harvest the good generalization ability of SGD at the end of training. However, in binary neural network optimization, SGD does not show its superiority in improving the generalization as in real-valued networks. But instead, in BNNs optimization, transiting to SGD leads to unstableness in training and failure in dealing with extremely small gradients, which leads to a worse accuracy.

5. Conclusion and Future Work

Many state-of-the-art BNNs are optimized with Adam, but the essential relations between BNNs and Adam are still not well-understood. In this work, we made fair comparisons between Adam and SGD for optimizing BNNs. We explain how Adam helps to re-activate those "dead" weights for better generalization. All our explanations are reflected in the visualization results. Furthermore, we elucidate why weight decay and initialization are critical for Adam to train BNNs and how to set their values. As we have shown, with the appropriate scheme of two-step training, our method achieved a competitive result of 70.5% on ImageNet. We hope these findings and understandings can inspire more studies in BNN optimization. Our future work will focus on designing new optimizers specifically for binary networks.

References

Alizadeh, M., Fernández-Marqués, J., Lane, N. D., and Gal, Y. An empirical study of binary neural networks' optimisation. 2018.

Ambrosio, L. and Dal Maso, G. A general chain rule for distributional derivatives. *Proceedings of the American Mathematical Society*, 108(3):691–702, 1990.

Bethge, J., Bartz, C., Yang, H., Chen, Y., and Meinel, C. Meliusnet: Can binary neural networks achieve mobilenet-level accuracy? *arXiv preprint arXiv:2001.05936*, 2020.

Brais Martinez, Jing Yang, A. B. G. T. Training binary neural networks with real-to-binary convolutions. *International Conference on Learning Representations*, 2020.

Bulat, A., Tzimiropoulos, G., Kossaifi, J., and Pantic, M. Improved training of binary networks for human pose estimation and image recognition. *arXiv preprint arXiv:1904.05868*, 2019.

Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

Ding, R., Chin, T.-W., Liu, Z., and Marculescu, D. Regularizing activation distribution for training binarized deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11408–11417, 2019.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.

Gu, J., Li, C., Zhang, B., Han, J., Cao, X., Liu, J., and Doermann, D. Projection convolutional neural networks for 1-bit cnns via discrete back propagation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 8344–8351, 2019.

Helwegen, K., Widdicombe, J., Geiger, L., Liu, Z., Cheng, K.-T., and Nusselder, R. Latent weights do not exist: Rethinking binarized neural network optimization. In *Advances in neural information processing systems*, pp. 7531–7542, 2019.

Hou, L., Yao, Q., and Kwok, J. T. Loss-aware binarization of deep networks. *arXiv preprint arXiv:1611.01600*, 2016.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krogh, A. and Hertz, J. A. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pp. 950–957, 1992.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pp. 6389–6399, 2018.

Lin, X., Zhao, C., and Pan, W. Towards accurate binary convolutional neural network. In *Advances in Neural Information Processing Systems*, pp. 345–353, 2017.

Liu, C., Ding, W., Xia, X., Zhang, B., Gu, J., Liu, J., Ji, R., and Doermann, D. Circulant binary convolutional networks: Enhancing the performance of 1-bit dcnns with circulant back propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2691–2699, 2019.

Liu, Z., Luo, W., Wu, B., Yang, X., Liu, W., and Cheng, K.-T. Bi-real net: Binarizing deep network towards real-network performance. *International Journal of Computer Vision*, pp. 1–18, 2018a.

Liu, Z., Wu, B., Luo, W., Yang, X., Liu, W., and Cheng, K.-T. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 722–737, 2018b.

Liu, Z., Shen, Z., Savvides, M., and Cheng, K.-T. Reactnet: Towards precise binary neural network with generalized activation functions. *ECCV*, 2020.

Luo, L., Xiong, Y., Liu, Y., and Sun, X. Adaptive gradient methods with dynamic bound of learning rate. *arXiv preprint arXiv:1902.09843*, 2019.

Mishra, A., Nurvitadhi, E., Cook, J. J., and Marr, D. Wrpn: wide reduced-precision networks. *arXiv preprint arXiv:1709.01134*, 2017.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

Phan, H., Huynh, D., He, Y., Savvides, M., and Shen, Z. Mobinet: A mobile binary network for image classification. In *The IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2020a.

Phan, H., Liu, Z., Huynh, D., Savvides, M., Cheng, K.-T., and Shen, Z. Binarizing mobilenet via evolution-based searching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13420–13429, 2020b.

Qin, H., Gong, R., Liu, X., Shen, M., Wei, Z., Yu, F., and Song, J. Forward and backward information retention for accurate binary neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2250–2259, 2020.

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pp. 525–542. Springer, 2016.

Reddi, S. J., Kale, S., and Kumar, S. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

Robbins, H. and Monro, S. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3): 211–252, 2015.

Shen, Z., Liu, Z., Qin, J., Huang, L., Cheng, K.-T., and Savvides, M. S2-bnn: Bridging the gap between self-supervised real and 1-bit neural networks via guided distribution calibration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

Tang, W., Hua, G., and Wang, L. How to train a compact binary neural network with high accuracy? In *Thirty-First AAAI conference on artificial intelligence*, 2017.

Tieleman, T. and Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4 (2):26–31, 2012.

Wang, Z., Lu, J., Tao, C., Zhou, J., and Tian, Q. Learning channel-wise interactions for binary convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., and Recht, B. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pp. 4148–4158, 2017.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Yang, Y., Huang, Q., Wu, B., Zhang, T., Ma, L., Gambardella, G., Blott, M., Lavagno, L., Vissers, K., Wawrzynek, J., et al. Synetgy: Algorithm-hardware co-design for convnet accelerators on embedded fpgas. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 23–32, 2019.

Zeiler, M. D. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Zhang, J., Pan, Y., Yao, T., Zhao, H., and Mei, T. dabnn: A super fast inference framework for binary neural networks on arm devices. In *Proceedings of the 27th ACM International Conference on Multimedia*, pp. 2272–2275, 2019.

Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., and Zou, Y. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

Zhu, S., Dong, X., and Su, H. Binary ensemble neural network: More bits per network or more networks per bit? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4923–4932, 2019.

Zhuang, B., Shen, C., Tan, M., Liu, L., and Reid, I. Towards effective low-bitwidth convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7920–7928, 2018.

Zhuang, B., Shen, C., Tan, M., Liu, L., and Reid, I. Structured binary neural networks for accurate image classification and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 413–422, 2019.