LAB: Learnable Activation Binarizer for Binary Neural Networks

Sieger Falkena¹,², Hadi Jamali-Rad¹,², Jan van Gemert¹
¹ TU Delft, Delft, The Netherlands
² Shell Global Solutions International B.V., Amsterdam, The Netherlands
sieger.falkena@shell.com, h.jamalirad@tudelft.nl, j.c.vangemert@tudelft.nl

Abstract

Binary Neural Networks (BNNs) are receiving an upsurge of attention for bringing power-hungry deep learning towards edge devices. The traditional wisdom in this space is to employ sign(.) for binarizing feature maps. We argue and illustrate that sign(.) is a uniqueness bottleneck, limiting information propagation throughout the network. To alleviate this, we propose to dispense sign(.), replacing it with a learnable activation binarizer (LAB), allowing the network to learn a fine-grained binarization kernel per layer - as opposed to global thresholding. LAB is a novel universal module that can seamlessly be integrated into existing architectures. To confirm this, we plug it into four seminal BNNs and show a considerable accuracy boost at the cost of tolerable increase in delay and complexity. Finally, we build an end-to-end BNN (coined as LAB-BNN) around LAB, and demonstrate that it achieves competitive performance on par with the state-of-the-art on ImageNet. Our code can be found in our repository: https://github.com/sfalkena/LAB.

1. Introduction

Convolutional Neural Networks (CNNs) dominate the current state-of-the-art computer vision tasks. With evolving research, models gained increasingly higher accuracy, but in parallel they have grown in size and complexity. This imposes a significant burden for deploying deep learning models on resource-constrained edge devices. Recent studies explore model compression techniques to reduce model size and latency, such as pruning [25], quantization [36], knowledge distillation [13], neural architecture search [10] and low rank approximation [38]. The most extreme form of quantization is realized by binarization, resulting in binary weights and activations {−1, +1}. Networks utilizing this are known as Binary Neural Networks (BNNs) and promise a bright future for energy-efficient deep learning. By quantizing weights and activations aggressively, one can theoretically achieve a memory reduction of 32× and a computational speedup of 58× on typical CPUs [31].

The current consensus in literature is to use sign(.) as a mapping from the full-precision to binary values. However, this imposes three widely-known issues: (i) the representational power of sign(.) with respect to the floating point counterpart decreases from $2^{32}$ to only 2 information levels [26]; (ii) the derivative of sign(.) is a Dirac Delta returning a zero gradient almost everywhere [7]; (iii) sign(.) spatially uses the same threshold everywhere, which we further refer to as the global threshold. In this work, we identify and debate about a fourth problem (iv) which we refer to as uniqueness bottleneck. Approaching the problem from both qualitative and quantitative angles, we demonstrate that using sign(.) further limits the representational capacity of the network. Interestingly, several studies state that sign(.) is a sub-optimal binarization operation and that it is not straightforward to find a new binarization function [6, 34]. This is exactly why we embark on this challenge in this paper.

Multiple remedies have been proposed to cope with the aforementioned issues of sign(.), including the introduction of scaling factors [31], gradient approximation [29] and pre-binarization distribution shaping [18]. Amongst these directions, we believe that the pre-binarization reshaping shows the most potential to alleviate the information bottleneck of BNNs. However, in contrast to the existing studies, we argue that shaping the pre-binarization distribution is a means an the end, and not the end in itself. To address the issues enumerated, we design a learnable activation binarization function (LAB) to automate the mapping from the full-precision feature maps to the binary counterparts, so that the representational capacity of the network (compared to full-precision) is least impacted. This is shown schematically in Fig. 1 (and elaborated in Section 3), where application of a global sign(.) threshold on the diverse spectrum of values in the discrete feature maps results in similar looking outputs (acting like a diversity bottleneck), whereas LAB can potentially avoid such loss of information and reveal important features for later layers.
Our contributions can be summarized as follows:

- To the best of our knowledge, for the first time, we identify the *uniqueness bottleneck* imposed by the \( \text{sign}(.) \). We demonstrate that \( \text{sign}(.) \) limits the representational capacity of binary feature maps.

- To address this bottleneck, we introduce a novel learnable activation binarization: \textsc{lab}. We show that \textsc{lab} is a universal module that can readily be plugged into any existing BNN architecture, and improve its performance. Our experimentation on four seminal BNN baselines corroborates this claim.

- We build an end-to-end network around \textsc{lab} (coined as \textsc{lab–bnn}) and demonstrate that it offers competitive accuracy (64.2% Top-1 validation accuracy) on par with the state-of-the-art in this space on ImageNet.

2. Related Work

Current BNNs binarize the full precision weights and activations by applying \( \text{sign}(.) \) on them:

\[
x_b = \text{sign}(x_r) = \begin{cases} 
+1, & \text{if } x_r > 0 \\
-1, & \text{if } x_r \leq 0 
\end{cases}
\]

where \( x_b \) and \( x_r \) denote the binary and real (full precision) values, respectively. Naively applying these quantizations to a CNN yields low accuracy and to close the gap between BNN implementations and their real-valued counterparts, several research directions have arisen: minimization of quantization error [7, 31], loss function improvement [9, 16, 24], gradient approximation [21, 24, 29], different network architecture designs [5, 8, 17, 24, 26, 41], training strategies [1, 23, 27, 33] and binary inference engines [3, 12, 37, 39]. Apart from these main directions, a few studies investigate new methodologies for binarizing weights and activations, which will be elucidated next.

**Weight binarization.** A novel approach for weight binarization is presented in [14] where both full precision and binary weights are employed as noisy supervisors for learning a mapping towards the final binary weights. As this mapping is learnable, it can exploit the relationships between weights. SiMaN [20] and RBNN [21] both propose a new binarization method based on so-called angle alignment between the full-precision and binary weights.

**Activation binarization.** One way to approach activation binarization is through classic computer vision techniques, such as dithering. This technique can binarize an image in a way that shifts quantization error towards higher frequencies. As the human visual system is more receptive to lower frequencies, the binarized image is perceived as having a low quantization error, and thus, carrying more information. A realization of this idea is called DitherNN but it only reports mild improvement [2]. Most activation binarization approaches focus on shaping the pre-binarization distribution, from which a higher entropy can be achieved after binarization [29]. For instance, an extra regularization term is proposed in [9] to explicitly shape the pre-binarization distribution so that it counteracts degeneration, saturation, and gradient mismatch problems. It is argued in [18] that BNNs benefit from an unbalanced pre-binarization distribution. ReActNet [24] argues that BNNs benefit from learning a similar activation distribution as their full-precision counterparts. On a related note, Si-BNN [34] approaches the activation binarization problem from a somewhat different angle and introduces sparsity in the activation binarization process. Even though these studies show promising performance results for BNNs, we argue that changing the pre-binarization distribution is still a form of adaptive global thresholding, and thus a suboptimal approach. Therefore, the output feature map does not fully reflect on, and adapt to the local information of the input feature map.
3. Problem Formulation

In this section, we reflect on the limitation a sign(.) binarization function inflicts on a BNN. We examine the capability of the network to cope with the single threshold value of sign(.), which we refer to as global thresholding - given that its value is the same for all spatial dimensions. In practice, as the input feature map to sign(.) is the output of a previous (convolutional) layer, the kernel of that convolution will learn to push or pull parts of the output feature map above or beneath the global threshold value, resulting in the fact that the output feature map will be closely centered around the threshold value. The batch normalization layer can further guide this process by effectively shifting the threshold value. Although this combination is essential for the current learning process of BNNs, we argue that there still is a limitation in its efficacy.

Fig. 2 explains what we perceive as the bottleneck in information propagation of BNNs. Here, A denotes the discrete input feature map to the binary convolution. Assume W’s represent the set of all unique weight kernels one can imagine. Given a kernel size $k$, the number of input channels $C$, and a single output channel per kernel, the total number of unique $W_i$’s, $\forall i \in [n]$ will then be $n = 2^{k^2 \times C}$. The output feature maps $D_1$ to $D_n$ are discrete finite-alphabet tensors. Note that in this case is smaller than the theoretical maximum number of unique activations $N = (k^2 \times C)^{H \times W}$, given a specific input A. Impacted by the activation design of the previous layers, proper design of A could potentially minimize the gap between $n$ and $N$. Applying sign(.) on $D_1$ to $D_n$ maps them to their binary counterparts $B_1$ to $B_n$. In theory, it is possible to have $n$ unique binary feature maps $B_i$’s, $\forall i \in [n]$, even though in practice, we show the fact that if we give W the freedom to take any value, the set of possible tokens of the output feature map is limited for the sign(.). Low diversity means that multiple distinct values of W will lead to an identical binary output, which will hinder the optimization of the model. We dub the aforementioned issue as the uniqueness bottleneck. We argue that due to this bottleneck, the network does not utilize its full potential and the representational capacity of the network is going to be impacted.

To demonstrate that this hypothesis is valid, we design a toy experiment that uses single-layer binary input feature maps A (in this case with $C = 1$) extracted from the binarized feature maps (using sign(.)) in a trained Bi-RealNet-18 [26]. Kernel size $k$ is set to 3, which makes up for a total of $2^{6427} = 512$ unique kernels. Following the steps sketched in Fig. 2, we take A as the starting point, convolve it with every possible kernel $W_i$ and binarize the output activations $D_i$’s with the sign(.) function. We then count the number of unique binary feature maps $B_i$’s, and average over all the channels (per different layers) of 20 different input images. We denote the ratio of counted unique feature maps ($n_u$), and theoretical total number of unique feature maps ($n_t$) as the uniqueness ratio $\eta = n_u/n_t$. The results are shown in Table 1. We can see that going deeper with convolutions, leading to smaller feature maps for layers 9 to 16, the uniqueness ratio decreases and the bottleneck becomes more evident. In the next section, we propose a learnable activation binarizer (LAB) as a remedy for this bottleneck.

4. The Proposed Method: LAB

One possible approach towards addressing the bottleneck of sign(.) binarization is to find a mapping from full-precision activation values to corresponding binary values, in such a way that the embedded spatial information from the input feature map is preserved. To do so, we propose to forge a different path in contrast to the current wisdom of activation distribution shaping [9,18,24]. More concretely, we propose a novel learnable activation binarizer (LAB) to learn a binarization kernel per layer, as shown in Fig. 3. The figure demonstrates LAB as a building block of a standard BNN. Zooming into the LAB unit, as we need to apply channel-wise binarization like sign(.), the input is first reshaped for per-channel operations. To capture local spatial information per channel, a $3 \times 3$ depth-wise convolution with a channel multiplier of 2 is applied. The core idea behind this channel doubling is to construct a miniature segmentation layer within the LAB unit to classify the input as $-1$ or $+1$. This classification is done through an ArgMax(.) across both channels, reducing the feature map back to a single output channel which is finally reshaped back to its original size. As the ArgMax(.) is non-differentiable, we apply the Soft-ArgMax(.) for the backward pass [11]. Given that we are dealing with only two classes, the Soft-ArgMax(.) in (2) simplifies to a single entry of the SoftMax(.) with an extra temperature controlling parameter $\beta$ which controls the “hardness” of the ArgMax(.) approximation:

$$\text{Soft-ArgMax}(x) = \frac{1}{\sum_j e^{\beta x_j}} \sum_i e^{\beta x_i} = \frac{e^{\beta x_i}}{\sum_j e^{\beta x_j}}. \quad (2)$$

The new binarization approach is only used for binarization of the feature maps and not for the binary convolution kernels, because the kernels do not contain enough spatial information for LAB to capture.

<table>
<thead>
<tr>
<th>Layer</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\eta$</td>
<td>0.964</td>
<td>0.994</td>
<td>0.996</td>
<td>0.998</td>
<td>0.998</td>
<td>0.986</td>
<td>0.991</td>
<td>0.994</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Layer</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\eta$</td>
<td>0.994</td>
<td>0.927</td>
<td>0.943</td>
<td>0.951</td>
<td>0.959</td>
<td>0.747</td>
<td>0.781</td>
<td>0.803</td>
</tr>
</tbody>
</table>
Figure 2: The uniqueness bottleneck. Activation $A$ is convolved with all unique kernels $W_i$'s. The finite-alphabet feature maps $D$ are binarized by the $\text{sign}(.)$, which creates the bottleneck of multiple $D$'s mapping to the same binary feature map $B$. The equations (at the bottom) indicate the theoretical maximum number of unique combinations of a tensor.

5. Experimental Evaluation

In this section, we first reflect on the experimental setup. Next, we analyze the uniqueness bottleneck and how employing LAB can help alleviate the problem. We then show that LAB is beyond a one-off remedy but more of a universal module that can straightforwardly be plugged into existing baselines. Finally, we propose an end-to-end model architecture ($\text{LAB-BNN}$) revolving around LAB offering competitive performance against state-of-the-art.

5.1. Experimental Setup

To be able to compare against a variety of existing state-of-the-art baselines - also to allow for a wider adoption by the community - we implement LAB both in PyTorch and TensorFlow. For comparison with the state-of-the-art baselines, the TensorFlow-based Larq framework [3] is used.

Network structure. To show the versatility of LAB, we plug it into four different architectures. First, XNOR-Net [31], is chosen to examine the improved information flow on an AlexNet-based architecture [19] with no skip connections. Then, Bi-RealNet [26] and ReActNet [24] are used for their ResNet [15] and MobileNet [17] backbones, respectively. Finally, QuickNet [3], an improved version of Bi-RealNet, is used to assess whether LAB can make an impact on a top performing state-of-the-art network.

Proposed end-to-end network: $\text{LAB-BNN}$. Going beyond the proposed module LAB, we also design an end-to-end network based upon the architecture of Bi-RealNet-18\(^1\), which is further enhanced by combining PReLU in [7, 24, 27, 33] and initial layers (further referred to as the STEM) as proposed by QuickNet [3].

\(^1\)ReActNet-A was tried, but we could not reproduce in Larq.
Hyperparameters. All experiments are conducted using 4 NVIDIA GeForce GTX 1080 Ti GPUs and follow standard settings in Larq [3], unless otherwise mentioned. To reproduce nominal reported performance, we used a batch size of 128 and a learning rate of $1e^{-4}$ for the retraining of XNORNet. For Bi-RealNet a batch size of 256 and learning rate $2.5e^{-3}$ was used. For Quicknet we used a batch size of 512 and a learning rate of $2.5e^{-3}$. Lastly, we re-trained ReActNet-A from scratch with a batch size of 256 and learning rate of $2.5e^{-3}$.

To understand that this aim, we compare the binary feature maps of a trained Bi-RealNet-18 with a trained Bi-RealNet-18+LAB (where $\text{sign}(\cdot)$ is replaced with LAB). The results are shown in Fig. 4 and Table 2. The figure depicts the selected feature maps from layer 1 and 6 (out of 18 layers). As can be seen, more structural information is preserved in layer 1 of LAB compared to $\text{sign}(\cdot)$, even though the features become too abstract for human understanding as we go to the deeper layer 6.

The most important takeaway from Fig. 4 is to understand that LAB, in contrast to $\text{sign}(\cdot)$, can distinguish between similar pixel values (e.g., shades in the lemons and different tints of blue in the watch). Besides the qualitative demonstration, the impact of LAB is further quantified on the number of unique feature maps learned by both networks through two (dis)similarity measures: structural similarity loss (SSIM) [35] and a custom-designed metric we call Euclidean norm dissimilarity (ENDSIM), given by:

$$ENDSIM(A_i, A_j) = \sqrt{\frac{1}{W} \sum_{w \in W} |A_i(w) - A_j(w)|^2 + \frac{1}{H} \sum_{h \in H} |A_i(h) + A_j(h)|^2}.$$  \hspace{1cm} (3)

where $W$ and $H$ denote the width and height of the input images in pixels and $A_i$ and $A_j$ are different single channel feature maps being compared. The metric has two terms in which the first term measures discrepancy, and the second one ensures fully inverted feature maps are not penalized. The latter is because inverted feature map layers along the channels can occur but do not indicate structural differences. Both measures are computed and averaged over all possible combinations of feature map layer pairs across all the 16 convolutional layers of 10 randomly selected images of ImageNet passed through both networks. The results are summarized in Table 2, and as can be seen, except for the first layer, LAB shows higher dissimilarity values (a lower SSIM and a higher ENDSIM) indicating more uniqueness along the channels.

<table>
<thead>
<tr>
<th>Layer</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIM ($\times 10^{-3}$)</td>
<td>94.8</td>
<td>21.3</td>
<td>14.5</td>
<td>13.1</td>
<td>11.9</td>
<td>21.6</td>
<td>16.6</td>
<td>13.5</td>
</tr>
<tr>
<td>LAB</td>
<td>1.5</td>
<td>9.0</td>
<td>4.2</td>
<td>5.4</td>
<td>8.9</td>
<td>5.1</td>
<td>6.6</td>
<td>6.4</td>
</tr>
<tr>
<td>ENDSIM ($\times 10^{-3}$)</td>
<td>0.9</td>
<td>0.93</td>
<td>0.85</td>
<td>0.84</td>
<td>0.78</td>
<td>0.96</td>
<td>0.94</td>
<td>0.8</td>
</tr>
<tr>
<td>LAB</td>
<td>0.92</td>
<td>0.96</td>
<td>0.7</td>
<td>1.1</td>
<td>1.3</td>
<td>1.4</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>SSIM ($\times 10^{-3}$)</td>
<td>12.5</td>
<td>25.8</td>
<td>19.7</td>
<td>17.6</td>
<td>14.6</td>
<td>29.3</td>
<td>22.4</td>
<td>18.0</td>
</tr>
<tr>
<td>LAB</td>
<td>8.5</td>
<td>4.2</td>
<td>2.9</td>
<td>2.4</td>
<td>8.8</td>
<td>5.6</td>
<td>8.2</td>
<td>7.1</td>
</tr>
<tr>
<td>ENDSIM ($\times 10^{-3}$)</td>
<td>0.76</td>
<td>0.92</td>
<td>0.88</td>
<td>0.78</td>
<td>0.81</td>
<td>0.98</td>
<td>0.93</td>
<td>0.99</td>
</tr>
<tr>
<td>LAB</td>
<td>1.6</td>
<td>1.4</td>
<td>1.4</td>
<td>1.4</td>
<td>1.7</td>
<td>1.4</td>
<td>1.4</td>
<td>1.4</td>
</tr>
</tbody>
</table>

5.2. The Uniqueness Bottleneck: Qualitative and Quantitative Analysis

In section 3, we have shown that $\text{sign}(\cdot)$ can introduce a bottleneck in reaching the theoretical maximum number of unique binary states, which we argued would limit the capacity of BNNs. Here, we adopt a small-scale experiment (followed by large-scale end-to-end experiments in the next subsection) to qualitatively and quantitatively demonstrate that LAB reduces this bottleneck. To this aim, we compare the binary feature maps of a trained Bi-RealNet-18 with a trained Bi-RealNet-18+LAB (where $\text{sign}(\cdot)$ is replaced with LAB). The results are shown in Fig. 4 and Table 2. The figure depicts the selected feature maps from layer 1 and 6 (out of 18 layers). As can be seen, more structural information is preserved in layer 1 of LAB compared to $\text{sign}(\cdot)$, even though the features become too abstract for human understanding as we go to the deeper layer 6. The most important takeaway from Fig. 4 is to understand that LAB, in contrast to $\text{sign}(\cdot)$, can distinguish between similar pixel values (e.g., shades in the lemons and different tints of blue in the watch). Besides the qualitative demonstration, the impact of LAB is further quantified on the number of unique feature maps learned by both networks through two (dis)similarity measures: structural similarity loss (SSIM) [35] and a custom-designed metric we call Euclidean norm dissimilarity (ENDSIM), given by:

$$ENDSIM(A_i, A_j) = \sqrt{\frac{1}{W} \sum_{w \in W} |A_i(w) - A_j(w)|^2 + \frac{1}{H} \sum_{h \in H} |A_i(h) + A_j(h)|^2}.$$  \hspace{1cm} (3)

where $W$ and $H$ denote the width and height of the input images in pixels and $A_i$ and $A_j$ are different single channel feature maps being compared. The metric has two terms in which the first term measures discrepancy, and the second one ensures fully inverted feature maps are not penalized. The latter is because inverted feature map layers along the channels can occur but do not indicate structural differences. Both measures are computed and averaged over all possible combinations of feature map layer pairs across all the 16 convolutional layers of 10 randomly selected images of ImageNet passed through both networks. The results are summarized in Table 2, and as can be seen, except for the first layer, LAB shows higher dissimilarity values (a lower SSIM and a higher ENDSIM) indicating more uniqueness along the channels.
Figure 4: Qualitative comparison between sign(.) and LAB on two layers of Bi-RealNet-18. LAB illustrates higher amount of texture (especially in Layer 1), which indicates allowing more information to pass through.

Table 3: Results of applying LAB on the corresponding baselines on ImageNet.

<table>
<thead>
<tr>
<th>Network</th>
<th>Backbone</th>
<th>Epochs</th>
<th>Method</th>
<th>Top-1 [%]</th>
<th>Top-5 [%]</th>
<th>Model Size [MB]</th>
<th>Latency [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>XNOR-Net [31]</td>
<td>AlexNet [19]</td>
<td>60</td>
<td>Sign</td>
<td>64.0</td>
<td>68.1</td>
<td>23.9</td>
<td>50.8</td>
</tr>
<tr>
<td>BiRealNet [26]</td>
<td>ResNet-18 [15]</td>
<td>150</td>
<td>LAB</td>
<td>59.1</td>
<td>81.2</td>
<td>6.6</td>
<td>100.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network</th>
<th>Method</th>
<th>W/A</th>
<th>Top-1 [%]</th>
<th>Top-5 [%]</th>
<th>BOPs [\times 10^9]</th>
<th>FLOPs [\times 10^9]</th>
<th>OPs [\times 10^9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full-precision</td>
<td>32/32</td>
<td></td>
<td>69.6</td>
<td>89.2</td>
<td>0</td>
<td>18.1</td>
<td>18.1</td>
</tr>
<tr>
<td>Bi-RealNet [26]</td>
<td>1/1</td>
<td></td>
<td>56.4</td>
<td>79.5</td>
<td>1.68</td>
<td>1.39</td>
<td>1.63</td>
</tr>
<tr>
<td>BNN-UAD [18]</td>
<td>1/1</td>
<td></td>
<td>57.2</td>
<td>80.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DCNet [29]</td>
<td>1/1</td>
<td></td>
<td>58.1</td>
<td>80.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SM-BNN [33]</td>
<td>1/1</td>
<td></td>
<td>59.7</td>
<td>81.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SiMaN [20]</td>
<td>1/1</td>
<td></td>
<td>60.1</td>
<td>82.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>QuickNet [3]</td>
<td>1/1</td>
<td></td>
<td>63.3</td>
<td>84.6</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LAB-BNN</td>
<td>1/1</td>
<td></td>
<td>64.2</td>
<td>85.0</td>
<td>1.68</td>
<td>0.66</td>
<td>0.92</td>
</tr>
<tr>
<td>ReActNet [24]</td>
<td>1/1</td>
<td></td>
<td>65.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

5.3. LAB: A Universal Module

The proposed method LAB can readily be applied to any BNN, with no architectural adjustments. In this subsection, we demonstrate this by replacing the sign(.) with LAB in four seminal BNN architectures and evaluate the impact on the downstream classification task of ImageNet. The results are illustrated in Table 3. As can be seen, at the cost of negligible increase in model size and tolerable increase in delay (max of 2x in [ms]), the boosted architectures XNORNet, Bi-RealNet, QuickNet, and ReActNet offer 3.3%, 7.9%, 3.8% and 1.7% improved Top-1 accuracies, respectively. The performance boost is slightly less pronounced for Top-5 accuracies. Note that the number of required training epochs to reach the nominal performance is dependent upon the architecture itself. The key message from this experiment is that irrespective of the backbone and architecture design, LAB makes a considerable impact on all architectures.

5.4. Comparison Against State-of-the-Art

Now that the universality of LAB has been shown, the efficacy of the proposed end-to-end network (LAB-BNN) with LAB sitting at its core, is evaluated. We compare the performance of LAB-BNN against the state-of-the-art baselines focused on activation binarization. For fair comparison, we focus on reported Top-1 and 5 validation accuracies on ImageNet with a ResNet-18 backbone. The outcome is summarized in Table 4. Both Top-1 and 5 results are competitive with the state-of-the-art and come short of the full precision equivalent network by only about 5%. Notably, ReActNet reports a higher accuracy (that we could not reproduce in TensorFlow) in part owing to adopting a multi-stage training strategy, whereas LAB-BNN is trained from scratch.

Aside from accuracy, the time and computational complexity of a BNN is just as important, even though most existing baselines overlook the importance of these metrics and sometimes do not even report them. Following [24], in Table 4 we report the total binary and floating-point operations (OP = BOP/64 + FLOP). Here, we achieve a lower total computational complexity (in FLOPs) compared to other baselines including the original Bi-RealNet. To further break this down, in Table 5 we set Bi-RealNet as our reference and state the added (+) complexity per operation listed in the second column. Even though the mechanics of LAB adds slight complexity, we compensate for this with a more efficient implementation of the STEM layer [3] leading to an overall decrease of 73.16 \times 10^6 in total FLOPs.

Table 4: Comparison against reported state-of-the-art on ImageNet. A dash indicates no value was reported by the original authors.

Table 5: FLOPs comparison: LAB-BNN vs. Bi-RealNet.
Post-binarization distribution. To illustrate that the post-binarization distribution is not a telling factor about the performance of the network (in contrast to conclusions drawn in [9, 18, 24]), Fig. 5 shows the post-binarization distribution of $+1$’s and $−1$’s for original Bi-RealNet-18 and Bi-RealNet-18+LAB averaged over all channels of 1000 input images. As can be seen, while the distribution of binary values remains almost the same across both networks, the performance of the two (reported in Subsection 5.3) is apart by 7.9%, supporting our claim that the binary distribution is not important, but the spatial arrangement of pixels is.

Going deep or going LAB? Plugging LAB into the standard Bi-RealNet (our base to build LAB-BNN) adds additional complexity to the network, and one could argue that adding the kernel of LAB per layer could virtually help make the network deeper. However, as a counter-argument, we compare LAB-BNN against a Bi-RealNet with a deeper backbone. More specifically, in the current construct, our method adds a depthwise convolution for each binary layer in Bi-RealNet-18. This is (roughly) equivalent to addition of 16 convolution layers, yielding a total of $18 + 16 = 34$ layers. According to [26], Bi-RealNet-34 and Bi-RealNet-50 respectively achieve 62.2% and 62.6% validation accuracies on ImageNet which are both still below the accuracy of LAB-BNN. Furthermore, the model size of BiRealNet-18+LAB is 4.24MB (FP32 weights for depthwise kernel), whereas BiRealNet-34 occupies 5.23MB. Thus, we are offering smaller model size and higher accuracy. To assess the impact of precision on the accuracy gain of LAB, we evaluated the depthwise convolution kernel of LAB for different bit widths. The results summarized in Table 6 indicate that going down from FP32 to INT8, the impact of LAB does not degrade, and even INT4 still offers a significant gain over no-LAB baseline.

Comparison against local thresholding We compare LAB against classical local binarization techniques, to show that LAB adheres to the common wisdom that learnable approaches outperform statistical approaches. We implement Niblack [28] and Sauvola [32] and use a window size of 3 and common values for $k$. Table 7 shows LAB has a gap of 11.5% with Niblack and improves with 2.7% over the best configuration for Sauvola.

### Table 6: LAB-BNN (Imagenette, 100 epochs) with multiple precisions for the depthwise convolution for LAB, and a comparison with other adaptive binarization techniques.

<table>
<thead>
<tr>
<th>LAB depthwise convolution</th>
<th>Top 1 [%]</th>
<th>Top 5 [%]</th>
<th>Model Size [MB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>No-LAB</td>
<td>76.4</td>
<td>97.2</td>
<td>2.04</td>
</tr>
<tr>
<td>LAB FP32</td>
<td>88.5</td>
<td>99.2</td>
<td>2.30</td>
</tr>
<tr>
<td>LAB INT8</td>
<td>89.1</td>
<td>99.2</td>
<td>2.13</td>
</tr>
<tr>
<td>LAB INT4</td>
<td>85.3</td>
<td>98.7</td>
<td>2.10</td>
</tr>
</tbody>
</table>

### Table 7: LAB-BNN (Imagenette, 150 epochs, batch size 32) A comparison with adaptive binarization techniques.

<table>
<thead>
<tr>
<th>Binarization method</th>
<th>Top 1 [%]</th>
<th>Top 5 [%]</th>
<th>Model Size [MB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>STE-sign</td>
<td>87.0</td>
<td>98.7</td>
<td>2.06</td>
</tr>
<tr>
<td>Niblack, $k = −0.2$</td>
<td>77.8</td>
<td>97.0</td>
<td>2.06</td>
</tr>
<tr>
<td>Sauvola, $k = 0.2$</td>
<td>82.5</td>
<td>98.3</td>
<td>2.06</td>
</tr>
<tr>
<td>Sauvola, $k = 0.5$</td>
<td>86.6</td>
<td>98.8</td>
<td>2.06</td>
</tr>
<tr>
<td>LAB</td>
<td>89.3</td>
<td>99.1</td>
<td>2.09</td>
</tr>
</tbody>
</table>

6. Ablation Studies

In this section, the effect of different components of LAB-BNN is further inspected. In what follows, we use the same settings listed in Section 5.1, except for the number of epochs which are shortened to 30. We start with the original Bi-RealNet-18 (model A) and update the components progressively towards LAB-BNN (model D). We first upgrade model A by incorporating PReLU activation [33] resulting in B. Model B is then extended by replacing sign(.) by the proposed LAB module leading to model C. Lastly, we replace the initial convolution in Bi-RealNet with the STEM layer of QuickNet [3], which forms LAB-BNN (model D). The results are illustrated in Table 8. Plugging LAB into model B results in an improvement of about 6% on the Top-1 validation accuracy (and roughly 5% for Top-5), at an increase of 0.46MB in model size and 28.5ms in latency. To make up for the added latency, we apply the STEM layers as proposed by QuickNet [3] which results in decreasing the latency with respect to model A down to 9.7ms.

![Figure 5: Post-binarization distribution of original Bi-RealNet-18 vs LAB-BNN averaged over all channels of 1000 input images. Rows indicate the blocks in the ResNet structure, columns indicate layer number within each block.](image-url)
6.1. Where to apply LAB.

So far, when applying LAB, all instances of $\text{sign}(.)$ in every binary layer of the network were replaced with LAB. For the architecture Bi-RealNet-18, this means for all of the four residual blocks, in all four convolutions per block. As we saw in Table 1, the uniqueness bottleneck is mainly visible in later layers. Therefore, in Table 9 we assess in which combination of blocks (out of 16 possible combinations) it is most impactful to apply LAB as a binarization function. In doing so, we apply LAB on all layers per selected block. The networks are trained for 30 epochs and we report Top-1 and Top-5 validation accuracies, model size and and latencies. We conclude from the table that omitting LAB (using the original $\text{sign}(.)$ everywhere) leads to the worst results, and see that as we apply LAB in more blocks progressively, the accuracy increases, and applying LAB to every block gives the highest accuracy. Interestingly, the model with LAB applied to the last 3 blocks (in the second row) has an exceptional accuracy-latency trade-off. Table 9 shows that by applying LAB in different blocks across the network, it is possible to design a network with a certain accuracy-latency trade-off, which gives useful design freedom for the practical use cases of BNN’s.

7. Concluding Remarks

We have shown that the commonly adopted binarization operation $\text{sign}(.)$ imposes a uniqueness bottleneck on BNNs, making it a sub-optimal choice for binarization. As a remedy, we introduce learnable activation binarizer (LAB), a novel binarization function that allows BNNs to learn a flexible binarization kernel per layer. We have demonstrated that LAB can readily be plugged into existing baseline BNNs boosting their performance regardless of their architecture design. Beyond that, we have also built a new end-to-end network (LAB-BNN) based upon LAB that offers competitive performance on par with the state-of-the-art. For future work, we will investigate applying learnable binarization to weights in addition to activations. We also plan to further extend our experimentation especially around LAB-BNN to push the performance boundaries and advance the state-of-the-art.
References


