Elastic-Link for Binarized Neural Network

Jie Hu,[1] **Ziheng Wu,** [3] **Vince Tan,** [4] **Zhilin Lu,** [2] **Mengze Zeng,** [4] **Enhua Wu**[*1,5]
[1] State Key Lab of Computer Science, ISCAS & University of Chinese Academy of Sciences
[2] Department of Electronic Engineering, Tsinghua University
[3] Alibaba Group
[4] ByteDance Inc.
[5] University of Macau
{hujie, weh}@ios.ac.cn

Abstract

Recent work has shown that Binarized Neural Networks (BNNs) are able to greatly reduce computational costs and memory footprints, facilitating model deployment on resource-constrained devices. However, in comparison to their full-precision counterparts, BNNs suffer from severe accuracy degradation. Research aiming to reduce this accuracy gap has thus far largely focused on specific network architectures with few or no 1×1 convolutional layers, for which standard binarization methods do not work well. Because 1×1 convolutions are common in the design of modern architectures (e.g. GoogleNet, ResNet, DenseNet), it is crucial to develop a method to binarize them effectively for BNNs to be more widely adopted. In this work, we propose an "Elastic-Link" (EL) module to enrich information flow within a BNN by adaptively adding real-valued input features to the subsequent convolutional output features. The proposed EL module is easily implemented and can be used in conjunction with other methods for BNNs. We demonstrate that adding EL to BNNs produces a significant improvement on the challenging large-scale ImageNet dataset. For example, we raise the top-1 accuracy of binarized ResNet26 from 57.9% to 64.0%. EL also aids convergence in the training of binarized MobileNet, for which a top-1 accuracy of 56.4% is achieved. Finally, with the integration of ReActNet, it yields a new state-of-the-art result of 71.9% top-1 accuracy.

Introduction

Convolutional Neural Networks (CNNs) have led to a series of breakthroughs for a variety of visual tasks (Krizhevsky, Sutskever, and Hinton 2012; Long, Shelhamer, and Darrell 2014; Ren et al. 2015; Toshev and Szegedy 2014; Zhu et al. 2016b). However, the challenge of resource constraints in terms of latency and memory storage is often faced when deploying CNNs on mobile or embedded devices. Previous work (Cai et al. 2017; Jacob et al. 2018; McKinstry et al. 2018; Jain et al. 2019; Shuang Wu 2016; Sung, Shin, and Hwang 2015; Yang et al. 2019; Zhou et al. 2016) has demonstrated that quantizing the real-valued weights and activations of CNNs into low-precision representations can reduce memory footprint while still achieving good performances.

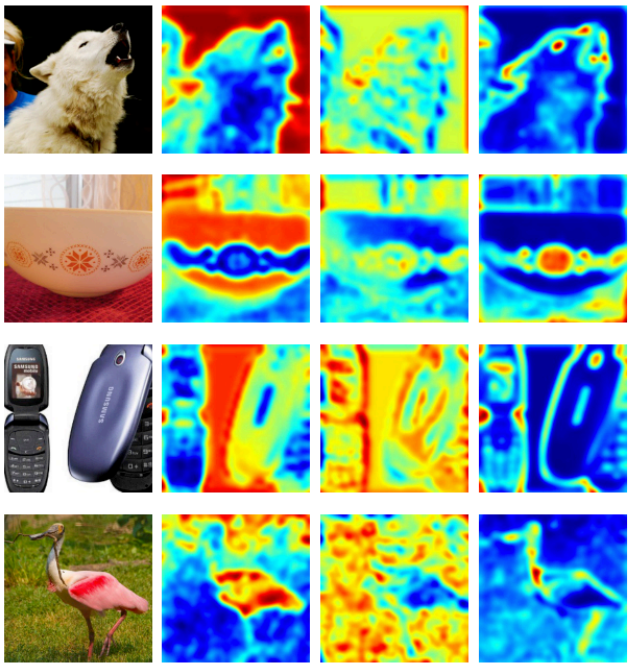

Figure 1: Example images illustrating the same features on full-precision ResNet26 (2nd column), Bi-Real-ResNet26 (3rd column) and our proposed EL-ResNet26 (4th column).

This class of methods allows fixed-point arithmetic to be applied, which substantially accelerates inference and reduces energy costs. Taken to an extreme, both the weights and the activations can be represented with binary tensors {-1, +1}. Such networks are termed Binarized Neural Networks (BNNs) (Courbariaux et al. 2016). In BNNs, arithmetic operations for convolutions can be replaced by the more efficient *xnor* and *bitcount* operations.

However, BNNs suffer from significant accuracy degradation as a result of information loss at each binarized layer. To conduct binarization, a real-valued signal is passed through the $Sign$ activation, which eliminates the signal's amplitude and retains only its sign information. Because this process

*corresponding author

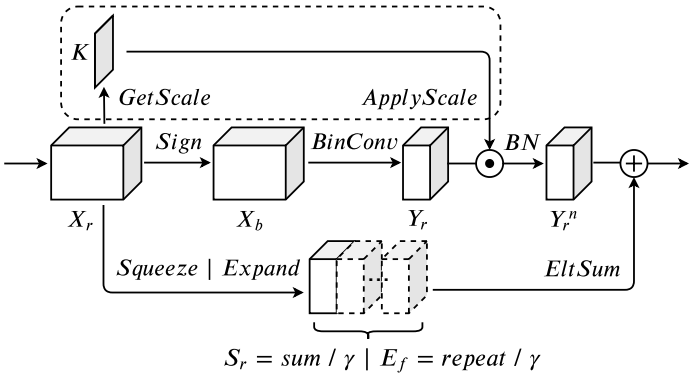

Figure 2: Diagram of the Elastic-Link module. $\oplus$ denotes element-wise summation. The process of applying the scaling factor to activations is depicted within the dashed box, indicating that it is omitted in some cases for better performance. $GetScale$ and $ApplyScale$ operations refer to XNOR-Net (Rastegari et al. 2016).

is irreversible, information loss increases with each layer of the BNN. Therefore, a central challenge to improving BNN accuracy is the reduction of this information loss.

One approach seeks to minimize the quantization error between the real-valued and binary forms of the weights and activations. (Rastegari et al. 2016) utilizes scaling factors to reduce the euclidean distances between the two forms. More recently, (Liu et al. 2018) employs a sophisticated fine-tuning strategy from a full-precision network with customized gradient approximation methods. (Liu et al. 2018) additionally proposes a shortcut connection to forward the real-valued activation, drastically reducing the extent of information loss. However, these methods apply best to networks which consist primarily of 3×3 or 5×5 convolutions, such as AlexNet (Krizhevsky, Sutskever, and Hinton 2012), VGGNet (Simonyan and Zisserman 2015) and Basic-Block ResNet (He et al. 2016).

Performing binarization with the above methods on networks in which 1×1 convolutions play a crucial role - for example, GoogleNet (Szegedy et al. 2015), Bottleneck ResNet or efficient networks with separated convolutions (Howard et al. 2017) - causes substantially greater accuracy degradations. 1×1 convolutions fuse information across channels and are already used to reduce computational cost via dimensionality reduction. We hypothesize that the marginal information loss from binarization is the proverbial last straw. (Howard et al. 2017) observed that the training or fine-tuning of binarized MobileNet fails to even converge, giving credence to this hypothesis.

In order to make binarization more widely applicable, we introduce an effective and universal module named *"Elastic-Link"* (EL). In order to compensate for the loss incurred by binarization, we adaptively add the real-valued input features (i.e. the features before feeding into the binarization function) to the output features of the subsequent convolution to retain the original real-valued signal. Liu et al. (Liu et al. 2018) demonstrated that adding extra shortcut connections, implemented by an element-wise summation, on the Basic-Block ResNet (He et al. 2016) produces considerable improvement in accuracy. This can be viewed as a special case of our proposed EL in which the input and output have the same shape. To generalize this finding, we develop a method to enable feature addition even if the feature size is changed by the convolution. EL uses a *Squeeze* or *Expand* operation to align the feature sizes between the input and the output. Furthermore, we do not simply perform a direct summation after the *Squeeze* or *Expand* operation, but rather learn a scaling factor to balance the relative extents of preserving the real-valued information and convolutional transformation, unifying these mechanisms and fusing them in a learnable, light-weight manner. EL is applicable to any architecture without structural limitations. To better visualize the effects of information preservation, we illustrate the feature-maps of a full-precision model, a Bi-Real model and our model with EL separately in Fig. 1. The feature maps of our EL model show clear object contours and the retention of important information for recognition, with less noise. By contrast, the foreground and background in the Bi-Real model are not as easily discriminated. We believe that the EL module benefits information flow in the binarized neural networks.

Moreover, as shown in Fig. 2, the design of the EL module is simple and can be directly applied to existing modern architectures. To assess the effectiveness of EL, we conduct extensive experiments on the ImageNet dataset. We outperform the current state-of-the-art result with a top-1 accuracy of 68.9%. We also contribute comprehensive ablation studies and discussions to further understanding of the intrinsic characteristics of BNNs.

Related Work

Quantized Weights with Real-Valued Activations. Restricting weights to be either +1 or -1 allows the time-consuming multiply-accumulate (MAC) operations to be replaced with simple addition operations. Recent studies (Courbariaux, Bengio, and David 2015; Rastegari et al. 2016) rely on this insight and employ the straight-through estimator (STE) (Bengio, LšŠonard, and Courville 2013) to tackle non-differentiability in the back-propagation of gradients during training. Courbariaux et al. proposed BinaryConnect (Courbariaux, Bengio, and David 2015) which drastically decreases computational complexity and storage requirements while achieving good results on the CIFAR-10 and SVHN datasets, but lacks experiments on large-scale datasets like ImageNet (Russakovsky et al. 2015). In Binarized Weight Networks (BWN) (Rastegari et al. 2016), the full-precision weights are binarized during each forward and backward pass on the fly, updating only the full-precision weights. BWN achieves a notable accuracy increase, especially on large-scale classification tasks. Finally, TWN (Li, Zhang, and Liu 2016) and TTQ (Zhu et al. 2016a) use ternary instead of binary weights $\{-\alpha^-,\ 0,\ \alpha^+\}$ to pass more information.

Quantized Weights and Activations. Another approach which has recently gained popularity restricts both the

weights and the activations to {-1, +1}. This allows the convolutional operations to be completely replaced by the efficient *xnor* and *bitcount* operations (Courbariaux et al. 2016; Rastegari et al. 2016), thus gaining extreme efficiency. XNOR-Net (Rastegari et al. 2016) is one of the most representative works of this approach and achieved remarkable accuracy with various networks. Bi-Real net (Liu et al. 2018) introduced additional shortcuts to retain the information of the real-valued activations, so as to alleviate information loss from binarization. The paper additionally introduced a custom gradient approximation method with a sophisticated fine-tuning strategy to further increase accuracy. Recently, Bethge et al. (Bethge et al. 2019) demonstrated that the said fine-tuning strategy and gradient approximation methods are not necessary for training BNNs - even without these, the authors achieved superior accuracy training from scratch with the simple straight-through estimator. Another method proposed to counteract the information degradation phenomenon is the linear combination of multiple binary weights to approximate full-precision weights, as introduced by ABC-net (Lin, Zhao, and Pan 2017). Finally, TBN (Wan et al. 2018) takes ternary inputs{-1, 0, +1} and binary weights with scale factors $\{-\alpha, \alpha\}$, demonstrating the method on both image classification and object detection tasks.

Methodology

In this section, we first revisit the standard process for training BNNs, then subsequently introduce a novel module, "Elastic-Link" (EL), to reduce the information loss in BNNs. Lastly, we demonstrate the EL module on Bottleneck ResNet (He et al. 2016) and MobileNet (Howard et al. 2017).

Gradient Approximation

It is standard to use the $Sign$ function to binarize a CNN. Real values are converted to the binary set of {-1, +1} by the following equation:

$$Sign(x) = \begin{cases} +1 & if \quad x \geq 0 \\ -1 & otherwise \end{cases} \tag{1}$$

where x refers to a real-valued weight or input/activation. To facilitate training, binarization is typically executed on the fly and only the real-valued weights are updated by the gradients, as described in (Courbariaux et al. 2016; Rastegari et al. 2016). During inference, the real-valued weights are unused and binary weights are used as a drop-in replacement.

In the backward pass, since the $Sign$ function is non-differentiable everywhere, an approximation is used. In this work, we follow the conventional "straight through estimator" (STE) (Bengio, LšŠonard, and Courville 2013) unless otherwise stated. The approximated gradient in STE is formulated as:

$$\frac{\partial Sign(x)}{\partial x} = \begin{cases} 1 & if -1 \leq x \leq 1 \\ 0 & otherwise \end{cases} \tag{2}$$

Scaling factor for weights and activations

As proposed in XNOR-Net (Rastegari et al. 2016), a binary convolutional operation can be given as follows:

$$BinConv\left(\mathbf{A}, \mathbf{W}\right) \approx \left(Sign\left(\mathbf{A}\right) \otimes Sign\left(\mathbf{W}\right)\right) \odot \mathbf{K}\alpha \tag{3}$$

where $\mathbf{A} \in \mathbb{R}^{c \times h \times w}$ is the real-valued input activation and $\mathbf{W} \in \mathbb{R}^{c \times k_h \times k_w}$ is the real-valued convolutional kernel. Here (c, h, w, k_h, k_w) refer to *number of input channels*, *input height*, *input width*, *kernel height*, and *kernel width* respectively. $\otimes$ denotes the efficient XNOR-Bitcounting operation (Rastegari et al. 2016) that replaces the time-consuming arithmetic operations.

α is a scaling factor given by a vector L1-Normalization $\alpha = \frac{1}{n}\|\mathbf{W}\|_{\ell 1}$, which helps minimize the L2 error between the real-valued weights and the binary weights with scalar coefficient α. K is a two-dimensional scaling matrix for the input activation, whose shape corresponds to the convolutional output. It is given by setting each element with the same principle as α. XNOR-Net concluded that α is more effective than K, which can even be entirely ignored for simplicity due to the relatively small improvement realized. Similarly, Liu et al. (Liu et al. 2018) and Lin et al. (Lin, Zhao, and Pan 2017) validated the effectiveness of α across various networks and datasets. Recently, Bethge et al. (Bethge et al. 2019) found that these scaling factors did not result in accuracy gains when BatchNorm (Ioffe and Szegedy 2015) is applied after each convolutional layer. In our experiments, we did observe the same experimental phenomenon with the Basic-Block ResNet, which is constructed with only 3×3 convolutions. However, we find that this principle holds only for 3×3 convolutions. When binarizing 1×1 convolutions, the scaling factor is still effective, as elaborated upon in Sec. .

Elastic-Link

Inspired by the shortcut connection mechanism, we use an element-wise summation operation to add real-valued input features to the output features generated by a binary convolution. In our Elastic-Link module, instead of an identity shortcut, we apply either a *Squeeze* or an *Expand* operation when a convolution alters the feature shape. A diagram of our proposed Elastic-Link module is shown in Fig. 2. Formally, let X_r denote the real-valued input feature where $\mathbf{X_r} \in \mathbb{R}^{H_i \times W_i \times C_i}$. We binarize X_r through a *Sign* activation function and obtain the binary X_b. Next, a binary convolution and standard BatchNorm are applied to obtain the convolutional output feature $Y_r^n \in \mathbb{R}^{H_o \times W_o \times C_o}$.

In order to compensate for the information loss, we add the real-valued input X_r to the normalized convolutional output Y_r^n. If the input size is equal to the convolutional output size, an identity shortcut connection with element-wise summation is applied as proposed in Bi-Real net (Liu et al. 2018). However, this condition is rarely true. In practice, a convolution operation usually changes the number of channels, and occasionally changes the height and width as well. In the channel-reduction case, we design a *Squeeze* operation in which the real-valued input X_r is split into multiple groups along the channel axis without overlap. The number

of channels for each group is $\lceil \frac{C_i}{C_o} \rceil$. We additionally zero-pad the features on input X_r to ensure that C_i can be exactly divided by C_o. Next, we sum these feature groups together to yield the squeezed features which will be of the same shape as Y_r^n. To reduce the effect of amplitude increase from the summation, and to offer a self-balancing tradeoff between information preservation and transformation, we divide the squeezed feature by a learnable scalar γ initialized as the number of groups. We take a similar approach for the channel expansion case. In an *Expand* operation, the real-valued input feature is repeated several times and then concatenated to match the feature size of the convolutional output. The expanded feature is correspondingly divided by same learnable factor of γ. Finally, the output of the *Squeeze* or *Expand* operation is added to the convolutional output feature Y_r^n, giving the overall output of the binarized convolution module. If spatial downsampling is required, a 2×2 max-pooling with stride 2 is applied before *Squeeze* or *Expand* to ensure spatial compatibility. The Elastic-Link module is formulated as follows:

$$EL\left(\mathbf{X_r}, \mathbf{W}, \gamma\right) = BN\left(BinConv\left(\mathbf{X_r}, \mathbf{W}\right)\right) + SEI\left(\mathbf{X_r}, \gamma\right) \tag{4}$$

Where $\mathbf{X_r}$ refers to the real-valued input activation and $\mathbf{W}$ refers to the convolutional weight. BN and $Sign$ refer to the BatchNorm and Sign function respectively. SEI refers to $Squeeze$, $Expand$ or $Identity$ operation, depending on the ratio of input and output channels. γ is the aforementioned learnable parameter that balances information preservation and transformation. We initialize γ by the following equation and optimize it through back-propagation:

$$\gamma = \begin{cases} \lceil \frac{C_i}{C_o} \rceil, & C_i >= C_o \\ \lceil \frac{C_o}{C_i} \rceil, & C_i < C_o \end{cases} \tag{5}$$

Where C_i and C_o refer to the number of channels for the input and output of a convolution respectively. $\lceil\ \rceil$ denote the ceiling or round-up operation. The max-pooling operation in the downsample case as well as the additional ReLU for efficient networks are omitted here for clarity.

Instantiations. The Elastic-Link module easily plugs into many modern architectures. Taking Bottleneck ResNet as an example, we integrate the Elastic-Link module into ResNet26 which consists of 8 bottleneck blocks. All bottleneck blocks are replaced by EL-Bottlenecks, as depicted in Fig. 3. The first convolution (of kernel size 7×7) and the classification layers remain full-precision to keep essential information at the input and output of the whole network. The downsampling shortcut in the first block of each stage, originally a 1×1 convolution of stride 2, is replaced by a 2×2 average pooling with stride 2 followed by a 1×1 convolution in full-precision. By integrating an EL module into the first 1×1 convolution of each *Bottleneck* block, more full-precision information flows to the middle 3×3 convolution, which is essential for capturing features at larger receptive fields.

Next we apply the EL module to efficient networks composed of separable convolutions. To the best of our knowledge, efficient networks have so far been considered incompatible with binarization. MobileNet (Howard et al. 2017) is one of the most representative efficient architectures. By adding Elastic-Link to each pointwise convolution and depthwise convolution (see Fig. 3), we are able to overcome the non-convergence problem typically encountered in training binarized MobileNet. We additionally find that keeping the ReLU activation achieves better performance. Similar to the ResNet case, we keep the first convolution, classifier and downsample components at full-precision.

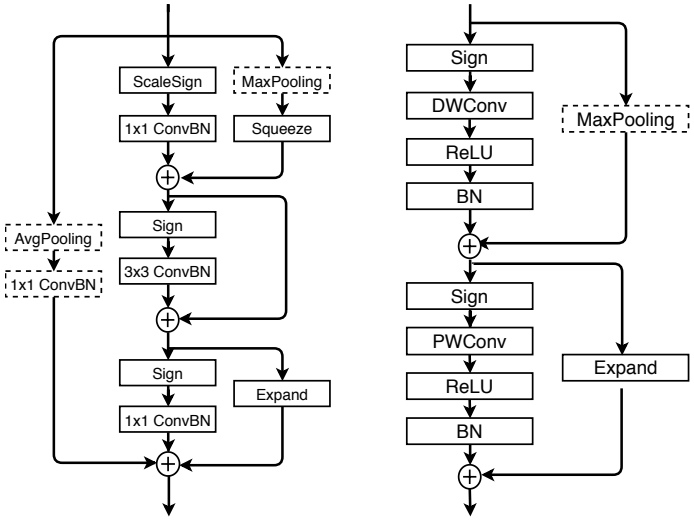

Figure 3: Schema of EL-Bottleneck module (Left) and EL-MobileNet module (Right).

Computational complexity

For the Elastic-Link module to be considered practical, it must offer a good tradeoff between improved performance and additional computational burden. To illustrate the increased complexity associated with the Elastic-Link module, we compare Bi-ResNet50 with EL-ResNet50. The additional computational cost incurred by the EL module originates from the γ scaling after the *Squeeze*, *Expand* or *Identity* operation as well as the element-wise summation of each 1×1 convolution, because the *Squeeze* and *Expand* can be implemented with address mapping without any overhead. In total, EL-ResNet50 requires an extra $\sim$8M FLOPs over Bi-ResNet50's $\sim$300M FLOPs for a single forward pass with an input image of 224×224, corresponding to a 2.6% increase. The number of FLOPs is computed as described in (Liu et al. 2018). For a practical comparison, we use the BMXNet library (Yang et al. 2017) on an Intel Core i7-9700K CPU to measure the actual time taken. Bi-Real-ResNet50 takes on average 22.2 ms for a single forward pass (over 10 runs), compared to 22.9 ms for our proposed EL-ResNet50. We believe that this small additional cost is justified by the increase in model performance.

Experiments

In this section, we conduct extensive evaluations to demonstrate the effectiveness of Elastic-Link for binarized neural networks on the large scale image classification task ImageNet (Russakovsky et al. 2015). ImageNet is challenging and often used to validate the performance of proposed methods in BNNs. The dataset consists of about 1.28 million training images and 50 thousand validation images, annotated for 1000 classes.

Training Details

During training, we perform binarization on-the-fly in the forward pass and follow the STE gradient approximation strategy. Input images are resized such that the shorter edge is 256 pixels, then randomly cropped to 224×224 pixels, followed by a random horizontal flip. Mean channel subtraction is used to normalize the input. All networks are trained from scratch using the Adam optimizer without weight decay normalization. The entire training process consists of 100 epochs with a mini-batch size of 256. The initial learning rate is set to 1e-3 and decreases after the 50th and 80th epochs by a factor of 10 each.

During inference, we center-crop patches of size 224×224 from each image on the validation set and report the top-1 accuracy for comparison.

Effect of Scaling Activation

We have previously discussed (in Sec.) the importance of the scaling weights and activations in binarized neural networks. Scaling is intended to mitigate the differences between the distributions of full-precision tensors and their binary counterparts. However, we obverse that scaling only the weights results in a minimal increase of accuracy when training from scratch, as previously observed in (Bethge et al. 2019). We expect that scaling the activations will greatly improve performance.

To investigate the effects of the scaling factor K on activations, we conduct experiments on Bi-Real nets - binarized ResNet26 with an additional shortcut at the middle 3×3 convolution of each block - here named S-ResNet26 for convenience. ResNet26 is constructed from a set of homogenous *Bottleneck* blocks which each comprises three convolutions: a first 1×1 channel-reduction convolution to reduce computation burden, a middle 3×3 convolution to capture spatial information, and a final channel-expansion 1×1 convolution to align the channel count with the input features for residual connection.

The experimental results are shown in Table 1. Applying the scaling factor to only the first channel-reduction convolution of each bottleneck block results in a top-1 accuracy of 59.8%, a significant increase over the original 58.7% wherein no scaling factors are applied. When scaling is also applied to the other two convolutions of each block, a slight accuracy drop is observed. Based on this phenomenon, we speculate that the scaling factor for activations facilitates optimization of the channel-reduction convolution. Noting that improper application of the scaling factor can cause adverse effects, we apply scaling on only the first (channel-reduction) convolution of each block in our subsequent experiments on *Bottleneck* ResNet.

Integration with Basic-Block ResNet

We next apply the EL module to the Basic-Block ResNet to validate its benefits. A Basic-Block is constructed by two successive isomorphic convolutions, and thus a feature passing through the block maintains its shape, except where downsampling is applied. In a Basic-Block, the EL module resolves to the identity shortcut connection as proposed in Bi-Real nets (Liu et al. 2018). The binarized ResNet model constructed using Basic-Blocks with the EL module is almost the same as the Bi-Real net in terms of model architecture. However, the Bi-Real net utilizes a multi-stage finetuning strategy and the custom differentiable approximation ApproxSign, whereas the EL models are trained from scratch with simple STE. As can be observed in the results listed in table 3, both EL-ResNet18 and EL-ResNet34 are superior to their Bi-Real variants by a margin of 2.2% and 1.0% in top-1 accuracy respectively, demonstrating that a simple training schedule suffices to achieve a well-trained binarized model. The experiments in Sec. **??** further validate this point.

Integration with Bottleneck ResNet

We next investigate the effectiveness of the proposed EL module on Bottleneck ResNet. Using the activation scaling strategy reported above, we obtain a strong baseline on S-ResNet26. Subsequently, we apply the EL module to each bottleneck block (see Fig. 3) and conduct extensive ablation studies. The results are reported in Table 2 and we can obverse the following phenomena:

- From the result of model EL_1, EL_2 and EL_3, we can see that applying Elastic-Link on more convolutions within a network can monotonically bring benefits ($59.8\% \rightarrow 61.8\% \rightarrow 62.1\% \rightarrow 63.2\%$), demonstrating the effectiveness of the EL module.
- Comparing model EL_3 with EL_4, when the original residual shortcuts are removed throughout the network, there is a significant drop of top-1 accuracy from 63.2% to 59.2%. This proves that although EL provides more information via the additional connection for each convolution, residual connections are still essential to forward primitive uncompressed information.
- Comparing model EL_6 with EL_3, allowing γ to be learnable further increases the top-1 accuracy from 63.2% to 64.0%.
- From the results of models EL_5 and EL_6, it can be observed that EL is compatible with the scaling factor strategy and achieves better accuracy when used with other techniques.

We term EL_6 in Table 2 as EL-ResNet26, and this is constructed using EL-Bottleneck (see Fig. 3). Including EL with learnable γ in the convolutional layers results in dramatic gains, increasing top-1 accuracy by an absolute value of 4.2%.

K_s	K_i	K_e	Top-1
			58.7
		✓	58.5
✓		✓	59.3
✓	✓		59.8
✓	✓	✓	59.4
✓			**59.8**

Table 1: Top-1 accuracy (%) of S-ResNet26 variants on ImageNet. K_s, K_i and K_e mean applying activation scaling to the first convolution (channel-reduction), middle convolution (spatial) and last convolution (channel-expansion) of each bottleneck block respectively.

	K_s	EL_s	EL_i	EL_e	$Id.$	γ_l	Top-1
Baseline	✓				✓		59.8
EL_1	✓	✓			✓		61.8
EL_2	✓	✓		✓	✓		62.1
EL_3	✓	✓	✓	✓	✓		63.2
EL_4	✓	✓	✓	✓			59.2
EL_5		✓	✓	✓	✓	✓	63.6
EL_6	✓	✓	✓	✓	✓	✓	**64.0**

Table 2: Top-1 accuracy (%) of binarized ResNet26 with different configurations of EL on the ImageNet validation set. EL_s, EL_i and EL_e denotes applying Elastic-Link to the first convolution (channel-reduction), middle convolution (spatial) and the last convolution (channel-expansion) of each bottleneck block respectively. $Id.$ means keeping residual connections, and γ_l means setting γ to be learnable in the EL module.

Results on deeper and efficient networks

In order to validate the generalizability of the EL module, we apply it to deeper networks, such as ResNet50 (He et al. 2016), which are rarely reported in existing literature. The results in Table 3 shows that EL-ResNet50 (65.6% top-1 accuracy) is superior to the Bi-Real-ResNet50 (62.7% top-1 accuracy) by a significant margin, proving that the EL module maintains strong performance even as the network grows deeper.

We next demonstrate the efficacy of the EL module on MobileNet. This is much more challenging as separable convolutions are weak at capturing spatial features. More concretely, the depthwise convolutions lack inter-channel information, while the pointwise convolutions which are expected to perform the inter-channel aggregation are particularly sensitive to information loss. As such, binarizing a MobileNet causes serious degradation in the network's ability to extract strong features. We add additional shortcut connection on depthwise convolutions and also perform the multi-stage fine-tuning strategy as discussed in Bi-Real net (Liu et al. 2018). However, this binarized MobileNet still failed to converge in its training loss. Subsequently, we included the EL module into MobileNet (see Fig. 3), and obtained the results shown in Table 3. The EL module not only enabled convergence, but also achieved an excellent performance of 56.4% top-1 accuracy, a strong baseline for future work on binarized efficient networks.

Comparison with the state-of-the-art

Finally, in order to demonstrate the superiority of our proposed EL in BNNs, we compare our results with other work on binary weights and activations. The main results on ImageNet are listed in Table 3. We demonstrate that binarized networks integrating with our EL module obtain considerable gains and also achieve the best performance against previous methods. At the same time, our approach integrates well with the Real-to-Binary (Brais, Bulat, and Tzimiropoulos 2020) approach, which is important for practical applications. Remarkably, we also integrated EL with the state-of-the-art result ReActNet-C (Liu et al. 2020) with top-1 accuracy of 71.4% on the Reduction block by replacing the duplicate activation parts with our EL modules. We obtained a new state-of-the-art result with a top-1 accuracy of 71.9%.

Discussion

Effect of binary 1×1 convolution. An EL module propagates the real-valued input signal without a change of receptive field regardless of the change in channel depth. Similarly, a 1×1 convolution linearly fuses inter-channel information also without a change of receptive field. Our experiments have comprehensively demonstrated the effectiveness of the EL module as a solution to the information loss incurred by binary 1×1 convolutions. We next investigate whether the EL module is effective outside of 1×1 convolutions. We set up a comparison on S-ResNet26 by replacing the first convolution (channel-reduction) of each block with the following alternatives: 1) a full-precision convolution, 2) an EL module and 3) an EL module without the inner 1×1 convolution. The result in Table 4 shows that the model with the complete EL module reaches almost the accuracy of the model with the full-precision convolution. Removing the 1×1 convolution from the EL module still results in an absolute 1.2% improvement in top-1 accuracy. We therefore conclude that between the two mechanisms of fusing inter-channel information by binary 1×1 convolution and forwarding real-valued information, the latter is more crucial to improve accuracy in BNNs.

The role of the γ factor. The learnable factor γ for the EL module controls the proportion of information fusion between the real-valued input features and the convolutional output features. In order to investigate the behavior of the γ factor, we study the distribution of γ from EL-ResNet26 with respect to the depth within the model. Fig. 4 illustrates the value changes of all 16 γ across 8 EL-Bottleneck blocks after training. Relative to their initial values, all the γ in the EL module that were applied to the channel-reduction convolutions with kernel size 1×1 increased significantly. This indicates that less original information is retained as input into the subsequent spatial convolution. In contrast, the γ values in the EL module that were applied to the channel-expansion convolutions slightly decreased, indicating that more information from previous spatial convolution is forwarded to the next block. This phenomenon is within

	Bottleneck Block		Efficient Block	Basic Block	
	RN26	RN50	MobileNet	RN18	RN34
Full-Precision	72.5	75.9	70.6	69.3	71.5
XNOR (Rastegari et al. 2016)	52.1	54.2	*Not Converge*	51.2	53.2
ABCNet (Lin, Zhao, and Pan 2017)	45.2	52.9	*Not Converge*	42.7	52.4
TBN (Wan et al. 2018)	-	-	-	55.6	58.2
Bi-Real (Liu et al. 2018)	57.8	62.7	*Not Converge*	56.4	62.2
BinaryE (Bethge et al. 2019)	57.9	61.2	*Not Converge*	56.7	59.5
CI-Net (Wang et al. 2019)	-	-	-	56.7	62.4
XNOR++ (Bulat and Tzimiropoulos 2019)	-	-	-	57.1	-
GBCN (Liu et al. 2019)	-	-	-	57.8	-
MoBiNet (Phan et al. 2020)	-	-	54.4	-	-
EL (Ours)	64.0	65.6	56.4	60.1	63.2
Real-to-Bin (Brais, Bulat, and Tzimiropoulos 2020)	64.8	65.9	54.8	65.4	66.1
EL† (Ours)	**67.1**	**68.9**	**61.2**	**65.7**	**66.5**

Table 3: Comparison of top-1 accuracy (%) on the ImageNet validation set. EL is specifically designed to improve 1×1 convolution, and its superiority compared to other works is correspondingly obvious across networks constructed with Bottleneck or Efficient Block. For reference, we also include networks built with Basic Block (which lack 1×1 convolutions). "RN" is short for ResNet. EL† refers to including Real-to-Bin (Brais, Bulat, and Tzimiropoulos 2020) in our proposed EL networks.

expectations because if all the original information before a convolution is forwarded, the said convolution would be of relatively little utility as a feature transformer. It is also worth noting that all γ applied to the channel-reduction convolutions in which downsampling occurred increased substantially during training (i.e. little original information is retained), as the convolutions in these shortcuts are full-precision, and thus are able to output more accurate information.

FP 1×1	Binary 1×1	EL^{-}	Top-1
	✓		59.4
✓			62.0
		✓	60.6
	✓	✓	61.8

Table 4: Top-1 accuracy (%) of various S-ResNet26 with different forms for the first convolutions of each block (the 1×1, channel-reduction ones). FP 1×1 refers to full-precision 1×1 convolution. Binary 1×1 refers to binarized 1×1 convolution. EL^{-} refers to removing the 1×1 convolution from the original EL module, retaining only the *Squeeze* operation.

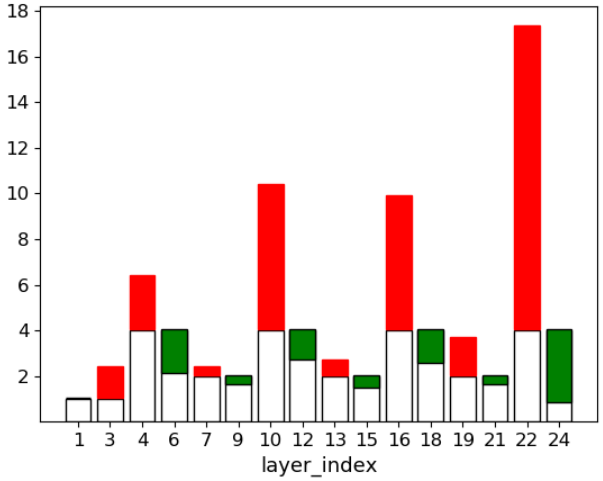

Figure 4: Learnable factor γ in all the EL module of EL-ResNet26 on ImageNet. The red histogram indicates an increase and the green histogram indicates a decrease compared to the initial value after training.

Conclusion

In this work, we proposed a novel Elastic-Link module for binarized neural networks. The Elastic-Link module introduces a connectivity mechanism to adaptively fuse real-valued input features and convolutional output features regardless of whether or not the feature-size is altered in the convolution. A learnable scaling factor enables an optimized tradeoff between information preservation and feature transformation, significantly reducing information degradation in the binarized form. The Elastic-Link module can be easily embedded into any architecture. It greatly enhances the representational ability of BNNs, especially for networks in which 1×1 convolutions are indispensable, such as Bottleneck ResNet and MobileNet. Extensive experiments demonstrate that binarized networks with Elastic-Link achieve considerable performance gains with negligible computational overhead. Moreover, the module is compatible with other techniques that have been shown to improve accuracy, e.g. the application of a scaling factor to activations. Combining Elastic-Link with such techniques achieves a new state-of-the-art result. A key challenge remaining is the observed increase in the degree of information degradation with network depth. We plan to explore more effective approaches to counteract this phenomenon in future work.

Acknowledgment

The work is supported in part by NSFC Grants (62072449).

References

Bengio, Y.; LšŠonard, N.; and Courville, A. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. In *CoRR*.

Bethge, J.; Yang, H.; Bornstein, M.; and Meinel, C. 2019. Back to Simplicity: How to Train Accurate BNNs from Scratch? *arXiv preprint arXiv:1906.08637*.

Brais, M.; Bulat, J. Y. A.; and Tzimiropoulos, G. 2020. Training binary neural networks with real-to-binary convolutions. *ICLR*.

Bulat, A.; and Tzimiropoulos, G. 2019. XNOR-Net++: Improved binary neural networks. *BMVC*.

Cai, Z.; He, X.; Sun, J.; and Vasconcelos, N. 2017. Deep learning with low precision by half-wave gaussian quantization. In *CVPR*, 5918–5926.

Courbariaux, M.; Bengio, Y.; and David, J.-P. 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NeurIPS*, 3123–3131.

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

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *CVPR*.

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv:1704.04861*.

Ioffe, S.; and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*.

Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; and Kalenichenko, D. 2018. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In *CVPR*.

Jain, S. R.; Gural, A.; Wu, M.; and Dick, C. H. 2019. Trained Quantization Thresholds for Accurate and Efficient Fixed-Point Inference of Deep Neural Networks. *arXiv:1903.08066*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NeurIPS*.

Li, F.; Zhang, B.; and Liu, B. 2016. Ternary weight networks. *arXiv preprint arXiv:1605.04711*.

Lin, X.; Zhao, C.; and Pan, W. 2017. Towards accurate binary convolutional neural network. In *NeurIPS*, 345–353.

Liu, C.; Ding, W.; Hu, Y.; Zhang, B.; Liu, J.; and Guo, G. 2019. GBCNs: Genetic Binary Convolutional Networks for Enhancing the Performance of 1-bit DCNNs. *arXiv preprint arXiv:1911.11634*.

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

Liu, Z.; Wu, B.; Luo, W.; Yang, X.; Liu, W.; and Cheng, K.-T. 2018. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *ECCV*, 722–737.

Long, J.; Shelhamer, E.; and Darrell, T. 2014. Fully Convolutional Networks for Semantic Segmentation. *TPAMI*.

McKinstry, J. L.; Esser, S. K.; Appuswamy, R.; Bablani, D.; Arthur, J. V.; Yildiz, I. B.; and Modha, D. S. 2018. Discovering low-precision networks close to full-precision networks for efficient embedded inference. *arXiv preprint arXiv:1809.04191*.

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

Rastegari, M.; Ordonez, V.; Redmon, J.; and Farhadi, A. 2016. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In *ECCV*, 525–542.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *NeurIPS*.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *IJCV*.

Shuang Wu, F. C. L. S., Guoqi Li. 2016. Training and Inference with Integers in Deep Neural Networks. In *ICLR*.

Simonyan, K.; and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. *ICLR*.

Sung, W.; Shin, S.; and Hwang, K. 2015. Resiliency of Deep Neural Networks Under Quantization. *arXiv preprint arXiv:1511.06488*.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going Deeper with Convolutions. In *CVPR*.

Toshev, A.; and Szegedy, C. 2014. DeepPose: Human Pose Estimation via Deep Neural Networks. In *CVPR*.

Wan, D.; Shen, F.; Liu, L.; Zhu, F.; Qin, J.; Shao, L.; and Tao Shen, H. 2018. TBN: Convolutional Neural Network with Ternary Inputs and Binary Weights. In *ECCV*.

Wang, Z.; Lu, J.; Tao, C.; Zhou, J.; and Tian, Q. 2019. Learning channel-wise interactions for binary convolutional neural networks. *CVPR*.

Yang, H.; Fritzsche, M.; Bartz, C.; and Meinel, C. 2017. BMXNet: An open- source binary neural network implementation based on mxnet. *arXiv preprint arXiv:1705.09864*.

Yang, J.; Shen, X.; Xing, J.; Tian, X.; Li, H.; Deng, B.; Huang, J.; and Hua, X.-s. 2019. Quantization networks. In *CVPR*, 7308–7316.

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

Zhu, C.; Han, S.; Mao, H.; and Dally, W. J. 2016a. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*.

Zhu, W.; Hu, J.; Sun, G.; Cao, X.; and Qiao, Y. 2016b. A Key Volume Mining Deep Framework for Action Recognition. In *CVPR*.