MeliusNet: An Improved Network Architecture for Binary Neural Networks

Joseph Bethge[1], Christian Bartz[1], Haojin Yang[1,2], Ying Chen[2], Christoph Meinel[1]
[1]Hasso Plattner Institute, University of Potsdam, Germany {firstname.surname}@hpi.de
[2]AI Labs, Alibaba Group {haojin.yhj,chenying.ailab}@alibaba-inc.com

Abstract

Binary Neural Networks (BNNs) are neural networks which use binary weights and activations instead of the typical 32-bit floating point values. They have reduced model sizes and allow for efficient inference on mobile or embedded devices with limited power and computational resources. However, the binarization of weights and activations leads to feature maps of lower quality and lower capacity and thus a drop in accuracy compared to their 32-bit counterparts. Previous work has increased the number of channels or used multiple binary bases to alleviate these problems. In this paper, we instead present an architectural approach: MeliusNet. It consists of alternating a DenseBlock, which increases the feature capacity, and our proposed ImprovementBlock, which increases the feature quality. Experiments on the ImageNet dataset demonstrate the superior performance of our MeliusNet over a variety of popular binary architectures with regards to both computation savings and accuracy. Furthermore, BNN models trained with our method can match the accuracy of the popular compact network MobileNet-v1 in terms of model size and number of operations. Our code is published online:
https://github.com/hpi-xnor/BMXNet-v2

1. Introduction

The success of deep convolutional neural networks in a variety of machine learning tasks, such as image classification [17, 26], object detection [35, 36], text recognition [23], and image generation [2, **?**], has led to the design of deeper, larger, and more sophisticated neural networks. However, the large size and high number of operations of these accurate models severely limit the applicability on resource-constrained platforms, such as those associated with mobile or embedded devices. There are many existing works aiming to solve this problem by reducing memory requirements and accelerating inference. These approaches can be roughly divided into a few research directions: knowledge distillation [9, 33, 40], network pruning techniques [15, 16], compact network designs [18, 19, 22, 37, 44], and low-bit quantization [8, 34, 45], wherein the full-precision 32-bit floating point weights (and in some cases also the activations) are replaced with lower-bit representations, e.g. 8 bits or 4 bits. The extreme case, Binary Neural Networks (BNNs), was introduced by [21, 34] and uses only 1 bit for weights and activations.

It was shown in previous work that the BNN approach is especially promising, since a binary convolution can be sped up by a factor higher than 50× while using only less than 1% of the energy compared to a 32-bit convolution on FPGAs and ASICs [32]. This speed-up can be achieved by replacing the multiplications (and additions) in matrix multiplications with bit-wise `xnor` and `bitcount` operations [32, 34], processing up to 64 values in one operation. However, BNNs still suffer from accuracy degradation compared to their full-precision counterparts [13, 34]. To alleviate this issue, there has been work to approximate full-precision accuracy by using multiple weight bases [27, 48] or increasing the channel number in feature maps [32, 38]. We briefly review the related work in more detail in Section 2.

Furthermore, alternative approaches to BNNs, such as the compact network structure MobileNet-v1 [19] have achieved higher accuracy in the past. More recent work on compact network structures, such as MobileNet-v2 or -v3 [18, 37] further widened the gap in accuracy between BNNs and compact networks. Since this gap has reduced the practical applicability of BNNs, one of the goals in this work is to show that it is possible to achieve the milestone of MobileNet-v1-level accuracy.

Prior work has been using full-precision architectures, e.g., AlexNet [26] and ResNet [17], without specific adaptations for BNNs. To the best of our knowledge, only two works are exceptions: Liu *et al*. added additional residual shortcuts to the ResNet architecture [30] (the resulting model was reused in more recent work [13, 31, 48]) and Bethge *et al*. adapted a DenseNet architecture with dense shortcuts for BNNs [5]. In our work, we use another architectural approach *MeliusNet* with designated building blocks to increase *quality* and *capacity* of features throughout the whole network (see Section 3). Further, a large share of operations in previous BNNs stems from a few lay-

ers which use 32-bit instead of 1-bit. To solve this issue, we propose a redesign of these layers which saves operations and improves the accuracy at the same time (see Section 3.2).

We evaluate MeliusNet on the ImageNet [10] dataset and compare it with the state-of-the-art (see Section 4). To confirm the effectiveness of our methods, we also provided extensive ablation experiments. During this study, we found that our training process with Adam [25] achieves much better results than reported in previous work. To allow for a fair comparison, we also trained the original (unchanged) networks and clearly separated the accuracy gains between the different factors in our ablation study (also within Section 4). Finally, we conclude our work in Section 5.

Summarized our main contributions in this work are:

- A novel BNN architecture MeliusNet which counters the lower quality and lower capacity of binary feature maps efficiently. It achieves state-of-the-art performance on ImageNet compared to other BNN networks.
- A more accurate and efficient initial set of grouped convolution layers for all binary networks.
- The first BNN models that match the accuracy of MobileNet-v1 0.5, 0.75, and 1.0.

2. Related Work

Alternatives to binarization, such as compact network structures [18, 19, 22, 37, 44], knowledge distillation [9, 33, 40], and quantized approaches [8, 24, 34, 41, 45, 43] have been introduced in the past. In this section, we take a more detailed look at approaches that use BNNs with 1-bit weights and 1-bit activations. These networks were originally introduced by Courbariaux *et al*. [21] with *Binarized Neural Networks* and improved by Rastegari *et al*. who used channel-wise scaling factors to reduce the quantization error in their *XNOR-Net* [34]. The following works tried to further improve the network accuracy, which was much lower than the accuracy of common 32-bit networks, with different techniques:

For instance, they modified the loss function (or added new loss functions) instead of using a simple cross-entropy loss to train more accurate BNNs [13, 31, 41].

WRPN [32] and Shen *et al*. [38] increased the number of channels for a better performance. Their work only increases the number of channels in the convolutions and the feature maps, but does not change the architecture.

Another way to increase the accuracy of BNNs was presented by *ABC-Net* [27] and *GroupNet* [48]. Instead of using a single binary convolution, they use a set of k binary convolutions to approximate a 32-bit convolution (this number k is sometimes called the number of binary bases). This achieves higher accuracy but increases the required memory and number of operations of each convolution by the factor k. These approaches optimize the network *within* each building block.

The two approaches most similar to our work are Bi-RealNet [30] and BinaryDenseNet [5]. They use only a single binary convolution, but adapt the network architecture compared to full-precision networks to improve the accuracy of a BNN. However, they did not test whether their proposed architecture changes are specific for BNNs or whether they would improve a 32-bit network as well.

3. MeliusNet

The motivation for MeliusNet comes from the two main disadvantages of using binary values instead of 32-bit values for weights and inputs.

On the one hand, the number of possible *weight* values is reduced from up to 2^{32} to only 2. This leads to a certain quantization error, which is the difference between the result of a regular 32-bit convolution and a 1-bit convolution. This error reduces the *quality* of the features computed by binary convolutions compared to 32-bit convolutions.

On the other hand, the value range of the *inputs* (for the following layer) is reduced by the same factor. This leads to a huge reduction in the available *capacity* of features as well, since fine-granular differences between values, as in 32-bit floating point values, can no longer exist.

In the following section, we describe how MeliusNet increases the quality and capacity of features efficiently. Afterwards, we describe how the number of operations in the remaining 32-bit layers of a binary network can be reduced. Finally, we show the implementation details of our BNN layers.

3.1. Increasing Capacity and Improving Quality

The core building block of MeliusNet consists of a *Dense Block* followed by an *Improvement Block* (see Figure 1). The *DenseBlock* increases feature *capacity*, whereas the *Improvement Block* increases feature *quality*.

The *Dense Block* was inspired by the BinaryDenseNet architecture [5], which is a binary variant of the DenseNet architecture [20]. It consists of a binary convolution which derives 64 channels of new features based on the input feature map, with, for example, 256 channels. These features are concatenated to the feature map itself, resulting in 320 channels afterwards, thus increasing feature *capacity*.

The novel *Improvement Block* increases the quality of these newly concatenated channels. It uses a binary convolution to compute 64 channels again based on the input feature map of 320 channels. These 64 output channels are added to the previously computed 64 channels through a residual connection, without changing the first 256 channels of the feature map (see Figure 1). Thus, this addition improves the last 64 channels, leading to the name of our

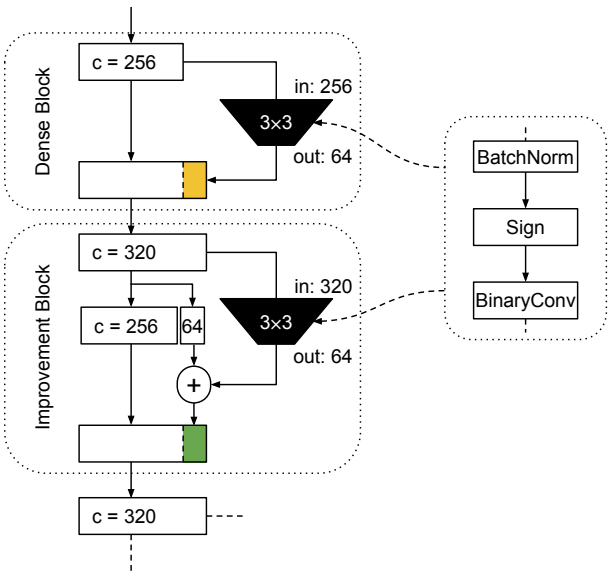


Figure 1: Building block of MeliusNet (c denotes the number of channels in the feature map). We first increase the feature *capacity* by concatenating 64 newly computed channels to the feature map (yellow area) in the Dense Block. Then, we improve the *quality* of *those newly added channels* with a residual connection (green area) in the Improvement Block. The result is a balanced increase of *capacity* and *quality*.

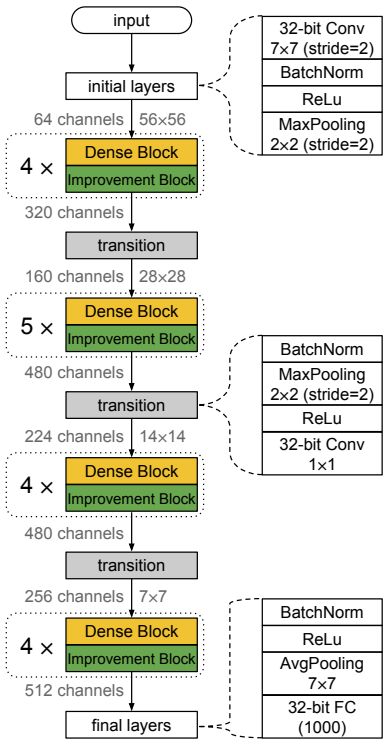


Figure 2: A depiction of our MeliusNet 22 with a configuration of 4-5-4-4 blocks between transitions. Details of the *Dense Block* and *Improvement Block* can be seen in Figure 1.

network (*melius* is latin for *improvement*). With this approach each section of the feature map is improved exactly once.

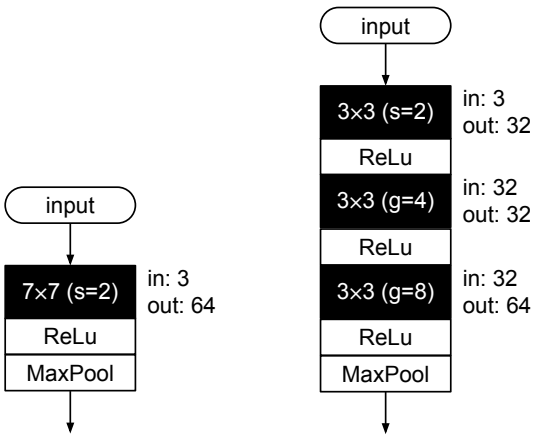
Note that we could also use a residual connection to improve the *whole* feature map instead of using the proposed *Improvement Block*. However, with this naive approach, the number of times each section of the feature map is improved would be highly skewed towards the initially computed features. It would further incur a much higher number of operations, since the number of output channels needs to match the number of channels in the feature map. With the proposed *Improvement Block*, we can instead save computations and get a feature map with balanced quality improvements. Our experiments showed that using a regular residual connection instead of our *Improvement Block* leads to $\sim$3% *lower* accuracy on ImageNet for equally sized networks (see the supplementary material for details).

As stated earlier, alternating between a *Dense Block* and an *Improvement Block* forms the core part of the network. Depending on how often the combination of both blocks is repeated, we can create models of different size and with a different number of operations. Our network progresses through four stages, with transition layers in between, which halve the height and width of the feature map with a MaxPool layer. Furthermore, the number of channels is also roughly halved in the 1×1 downsampling convolution during the transition (see Table 1 on page for the exact factors). We show an example in Figure 2, where we repeat the blocks 4, 5, 4, and 4 times between transition layers and achieve a model which is similar to Bi-RealNet18 [30] in terms of model size.

3.2. Creating an Efficient Stem Architecture

We follow previous work and do not binarize the first convolution, the final fully connected layer, and the 1×1 ("downsampling") convolutions in the network to preserve accuracy [5, 30, 48]. However, since these layers contribute a large share of operations, we propose a redesign of the first layers. We hypothesize that improving the first set of layers in an efficient way should generalize well to all BNN architectures. Note that we refer to the ImageNet classification task [10] in the following examples.

We compared previous BNNs to the compact network architecture MobileNet-v1 0.5 [19], which only needs $1.49 \cdot 10^8$ operations in total and can achieve 63.7% accuracy on ImageNet. We found, that the closest BNN result (regarding model size and operations) is Bi-RealNet34, which achieves lower accuracy (62.2%) with a similar model size, but it also needs more operations ($1.93 \cdot 10^8$). We presume, that because of this difference, compact model architectures are



(a) The 7×7 convolution with $1.18 \cdot 10^8$ operations.

(b) Our proposed *grouped stem* with $0.69 \cdot 10^8$ operations.

Figure 3: A depiction of the two different versions of initial layers of a network (s is the stride, g the number of groups, we use 1 group and stride 1 otherwise). Our *grouped stem* in (b) can be applied to all common BNN architectures, e.g., Bi-RealNet [30] and BinaryDenseNet [5], as well as our proposed MeliusNet to save operations by replacing the expensive 7×7 convolution in the original layer configuration (a) without an increase in model size.

more popular for practical applications than BNNs, especially with more recent (and improved) compact networks appearing [18, 37]. To find a way to close this gap, we analyze the required number of operations in the following.

The first 7×7 convolution layer in a Bi-RealNet18 *alone* needs 65% $(1.18 \cdot 10^8)$ of the total operations of the whole network. The three 1×1 downsampling convolutions account for another 10% $(0.18 \cdot 10^8)$ of operations. Since in total about 75% of all $1.81 \cdot 10^8$ operations are needed for these 32-bit convolutions, we focused on them to reduce the number of operations.

In previous work the 7×7 32-bit convolution uses 64 channels. We propose to replace the 7×7 convolution with three 3×3 convolutions, similar to the stem network used by Szegedy *et al*. [39]. However, their stem network uses four times as many operations compared to the regular 7×7 convolution. We use grouped convolutions [26] instead of regular convolutions for a reduction in operations (resulting in the name *grouped stem*) and a lower number of channels. The first convolution has 32 output channels (with a stride of 2), the second convolution uses 4 groups and 32 output channels, and the third convolution has 8 groups and 64 output channels (see Figure 3). This layer combination needs the same number of parameters (and thus model size) as the 7×7 convolution, but only $0.69 \cdot 10^8$ instead of the original $1.18 \cdot 10^8$ operations, which is a reduction of more than 40%.

Similarly to adapting the stem structure, the 1×1 downsampling convolution can also use a certain number of groups, e.g., 2 or 4. Since the features in the feature map are created consecutively with Dense Blocks, we add a channel shuffle operation before the downsampling convolution [44], but only if we use groups in our downsampling convolution. This allows the downsampling convolution to combine features from earlier layers and later layers together.

Even though there are certainly other ways to change the 32-bit layers to reach an even lower number of operations, e.g., using quantization, a different set of layers, etc., our main goal is to highlight the high influence of these 32-bit layers on the number of operations in BNNs. The 75% share of operations in these layers was not clear in previous cost analyses of BNNs. We hope this insight can direct future work into considering them for further improvement and investigate alternatives. However, our proposed redesign already enables previous works on BNNs to reach a similar number of operations as MobileNet-v1 and we can test whether their accuracy can also achieve a similar level (see Section 4.3 for the results).

3.3. Implementation Details

We follow the general principles to train binary networks as presented in previous work [5, 30, 34]. The weights and activations are binarized by using the *sign* function:

$$\operatorname{sign}(x) = \begin{cases} +1 \text{ if } x \geq 0, \\ -1 \text{ otherwise.} \end{cases} \tag{1}$$

The non-differentiability of the sign function is solved with a Straight-Through Estimator (STE) [3] coupled with gradient clipping as introduced by Hubara *et al*. [21]. Therefore, the forward and backward passes can be described as:

$$\text{Forward: } r_o = \operatorname{sign}(r_i)\ . \tag{2}$$

$$\text{Backward: } \frac{\partial l}{\partial r_i} = \frac{\partial l}{\partial r_o} 1_{|r_i| \leq t_{\text{clip}}}\ . \tag{3}$$

In this case l is the loss, r_i a real number input, and $r_o \in \{-1, +1\}$ a binary output. We use a clipping threshold of $t_{\text{clip}} = 1.3$ as used by [5]. Furthermore, the computational cost of binary neural networks at runtime can be highly reduced by using the `xnor` and `popcount` CPU instructions, as presented by Rastegari *et al*. [34].

Previous work [30] has suggested a different backward function to approximate the *sign* function more closely, however, we found no performance gain during our experiments, similar to the results of [4]. Channel-wise scaling factors have been proposed to reduce the difference between a regular and a binary convolution [34]. However, it was also argued, that they are mostly needed to scale the gradients [30], that a single scaling factor is sufficient [45], or that neither of them is actually needed [4]. Recent work

Table 1: Details of our different MeliusNet configurations, including the number of floating point and binary operations (FLOPs/BOPs), and their accuracy on the ImageNet classification task [10]. The combined number of operations (OPs) is based on the speed-up factor of previous work [5, 30]: OPs = $\left(\frac{1}{64} \cdot \mathrm{BOPs} + \mathrm{FLOPs}\right)$. The channel reduction factors are chosen at such specific fractions to keep the number of channels as multiples of 32.

Name (block numbers)/ (groups in 1×1 conv)	Channel reduction factors in transitions	BOPs $(\cdot 10^9)$	FLOPs $(\cdot 10^8)$	OPs $(\cdot 10^8)$	Size (MB)	OPs and Size similar to	Top-1 (Top-5) accuracy
MeliusNet22 (4,5,4,4)	$\frac{160}{320}, \frac{224}{480}, \frac{256}{480}$	4.62	1.35	2.08	3.9	BDenseNet28 [5]	63.6% (84.7%)
MeliusNet29 (4,6,8,6)	$\frac{128}{320}, \frac{192}{512}, \frac{256}{704}$	5.47	1.29	2.14	5.1	BDenseNet37 [5]	65.8% (86.2%)
MeliusNet42 (5,8,14,10)	$\frac{160}{384}, \frac{256}{672}, \frac{416}{1152}$	9.69	1.74	3.25	10.1	MobileNet-v1 0.75[19]	69.2% (88.3%)
MeliusNet59 (6,12,24,12)	$\frac{192}{448}, \frac{320}{960}, \frac{544}{1856}$	18.3	2.45	5.25	17.4	MobileNet-v1 1.0[19]	71.0% (89.7%)
MeliusNetA (4,5,5,6)/(4)	$\frac{160}{320}, \frac{256}{480}, \frac{288}{576}$	4.85	0.86	1.62	4.0	Bi-RealNet18 [30]	63.4% (84.2%)
MeliusNetB (4,6,8,6)/(2)	$\frac{160}{320}, \frac{224}{544}, \frac{320}{736}$	5.72	1.06	1.96	5.0	Bi-RealNet34 [30]	65.7% (85.9%)
MeliusNetC (3,5,10,6)/(4)	$\frac{128}{256}, \frac{192}{448}, \frac{224}{704}$	4.35	0.82	1.50	4.5	MobileNet-v1 0.5[19]	64.1% (85.0%)

suggests, that the effect of scaling factors might be neutralized by BatchNorm layers [5]. For this reason, and since we have not observed a performance gain by using scaling factors, we did not apply them in our convolutions. We use the typical layer order (BatchNorm $\rightarrow$ sign $\rightarrow$ BinaryConv) of previous BNNs [5, 30]. Finally, we replaced the bottleneck structures, consisting of a 1×1 and a 3×3 convolution, which are often used in full-precision networks, as it was done in previous work [4, 48] and used a single 3×3 (1-bit) convolution instead.

4. Results and Discussion

We selected the challenging task of image classification on the ImageNet dataset [10] to test our new model architecture and perform ablation studies with our proposed changes. Our implementation is based on BMXNet 2[1] [42] and the model implementations of Bethge *et al*. [5]. Note that experiment logs, accuracy curves, and plots of model structures for all trainings can be found in the supplementary material.

To compare to other state-of-the-art networks we created different configurations of MeliusNet with different model sizes and number of operations (see Table 1). Our main goal was to reach fair comparisons to previous architectures by using a similar model size and number of operations. For example, we chose the configurations of MeliusNet22 and MeliusNet29 to be similar to BinaryDenseNet28 and BinaryDenseNet38, respectively. Note that we calculated the number of operations in the same way as in previous work, factoring in a $64\times$ speed-up factor for binary convolutions [5, 30]. For a comparison to Bi-RealNet we further reduced the amount of operations, by using 2 or 4 groups in the downsampling convolutions for MeliusNetA and MeliusNetB, respectively and added a channel shuffle operation beforehand as described in Section 3.2. Finally, we created the networks MeliusNetC, MeliusNet42 and MeliusNet59 to be able to fairly compare to MobileNet-v1 0.5, 0.75 and 1.0, respectively. This also shows, that the basic network structure of MeliusNet can be adapted easily to create networks with different sizes and number of operations by tuning the number of blocks.

[1] https://github.com/hpi-xnor/BMXNet-v2

4.1. Grouped Stem Ablation Study and Training Details

When training models based on previous architectures with our proposed *grouped stem* structure, we discovered a large performance gain compared to previously reported results. To verify the source of these gains we ran an ablation study on ResNetE18 [4] (which is similar to Bi-RealNet18 [30], except for the addition of a single ReLu layer and a single BatchNorm), Bi-RealNet34 [30], BinaryDenseNet28/37[5], and our MeliusNet22/29 with and without our proposed grouped stem structure (see Table 2).

On the one hand, the results show, that using grouped stem instead of a regular 7×7 convolution increases the model accuracy for all tested model architectures. The actual increase by using the grouped stem structure is between 0.4% and 1.1% for each model in addition to saving a constant amount $(0.49 \cdot 10^8)$ of operations. We conclude, that not only is using our grouped stem structure highly efficient, but it also generalizes well to different BNN architectures.

On the other hand, we also recognized that our training process performs significantly better than previous training strategies. Therefore, we give a brief overview about our training configuration in the following:

For data preprocessing we use channel-wise mean subtraction, normalize the data based on the standard deviation, horizontally flip the image with a probability of $\frac{1}{2}$ and finally select a random resized crop, which is the same augmentation scheme that was used in XNOR-Net [34]. We

Table 2: Ablation study on ImageNet [10] separating the gains between the training process and *grouped stem*. It shows the generic applicability of both.

Size	Network Architecture	Training Procedure	Group Stem	OPs ($\cdot 10^8$)	Top-1 acc.
~4.0MB	ResNetE18 [5]	Original	✗	1.63	58.1%
		Ours	✗	1.63	60.0%
		Ours	✓	1.14	60.6%
	BDenseNet28 [5]	Original	✗	2.58	60.7%
		Ours	✗	2.58	61.7%
		Ours	✓	2.09	62.6%
	MeliusNet22 (ours)	Ours	✗	2.57	62.8%
		Ours	✓	2.08	63.6%
~5.1MB	Bi-RealNet34 [30]	Original	✗	1.93	62.2%
		Ours	✗	1.93	63.3%
		Ours	✓	1.43	63.7%
	BDenseNet37 [5]	Original	✗	2.71	62.5%
		Ours	✗	2.71	63.3%
		Ours	✓	2.20	64.2%
	MeliusNet29 (ours)	Ours	✗	2.63	64.9%
		Ours	✓	2.14	65.8%

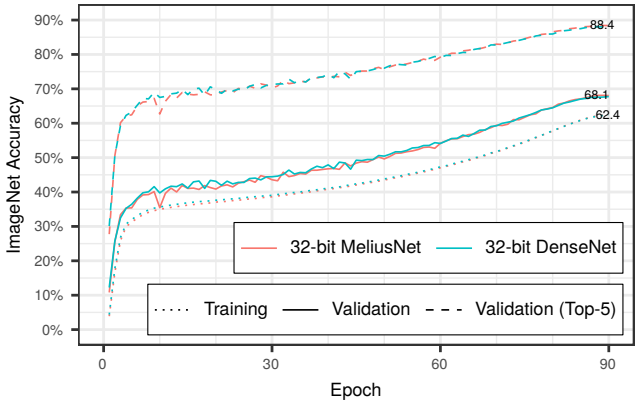


Figure 4: A comparison between the 32-bit versions of MeliusNet and DenseNet (best viewed in color). We tuned the number of building blocks to achieve models of similar complexity: MeliusNet uses 4,4,4,3 (4.5×10^9 FLOPs, 20.87 MB) and DenseNet uses 6,6,6,5 (4.0×10^9 FLOPs, 19.58 MB). We used the off-the-shelf Gluon-CV training script for ImageNet [14] with identical hyperparameters to train both models. The accuracy curves are almost indistinguishable for the whole training process and our 32-bit MeliusNet is not able to improve the result compared to a 32-bit DenseNet, even though it uses slightly more FLOPs and memory.

initialize the weights with the method of [12] and train our models from scratch (without pre-training a 32-bit model) for 120 epochs with a base learning rate of 0.002. We use the RAdam optimizer proposed by Liu *et al*. [29] and the default ("cosine") learning rate scheduling of the GluonCV toolkit [14]. This learning rate scheduling steadily decreases the learning rate based on the following formula (t is the current step in training, with $0 \leq t \leq 1$): $\mathrm{lr}(t) = \mathrm{lr}_{\mathrm{base}} \cdot \left(\frac{1+\cos(\pi \cdot t)}{2}\right)$. However, we achieved similar (only slightly worse) results with the same learning rate scheduling and the Adam [25] optimizer, if we use a warm-up phase of 5 epochs in which the learning rate is linearly increased to the base learning rate. Using SGD led to the worst results overall and even though we did some initial investigation into the differences between optimizers (included in supplementary material) we could not find a clear reason for the performance difference (a similar observation was made by Alizadeh *et al*. [1]).

4.2. Ablation Study on 32-bit Networks

We performed another ablation study to determine whether our proposed MeliusNet is indeed specifically better for a BNN or whether it would also increase the performance of a 32-bit network. Since our proposed MeliusNet without the Improvement Blocks is similar to a DenseNet, we compared these two architectures and trained two 32-bit models based on a DenseNet and a MeliusNet. We used the off-the-shelf Gluon-CV training script for ImageNet and their DenseNet implementation as a basis for our experiment [14]. To achieve a fair comparison, we constructed two models of similar size and operations. We used 4-4-4-3 blocks (Dense Block and Improvement Block) between the transition stages for MeliusNet and 6-6-6-5 blocks (Dense Blocks only) for a DenseNet. The models need 4.5 billion FLOPs with 20.87 MB model size and 4.0 billion FLOPs with 19.58 MB model size, respectively. Therefore, we expect MeliusNet to definitely achieve a slightly better result, since it uses slightly more FLOPs and has a higher model size, unless our designed architecture is only specifically useful for BNNs. Both models were trained with SGD with momentum ($\mathrm{lr} = 0.1$) and equal hyperparameters for 90 epochs (with a warm-up phase of 5 epochs and "cosine" learning rate scheduling). Note that additional augmentation techniques (HSV jitter and PCA-based lightning noise) were used (in this study only), since we did not change the original Gluon-CV training script for the 32-bit models.

The result shows basically identical training curves between both models for the whole training (see Figure 4). At the end of training, the training accuracy is even between both architectures at 62.4%. Even though the validation accuracy does not match for the whole training, this is probably caused by randomized augmentation and shuffling of the dataset. The accuracy gain of 1% to 1.6% that could be observed between a binary DenseNet and a binary MeliusNet (see Table 2) does not occur for the 32-bit version of networks. Therefore, we conclude, that using our MeliusNet architecture for 32-bit models does not lead to an improvement, and our architecture is indeed only an improve-

Table 3: Comparison to state-of-the-art quantized and binary CNNs on ImageNet [10]. All models were trained with cross-entropy loss. Methods include low-bit quantization (first section) and approaches with multiple binary bases (second section). The comparison is in parallel for two size categories (with differing number of layers). The best result in each section is bold.

Method	Bitwidth (W/A)	ImageNet (≈18 layers)			ImageNet (≈34 layers)		
		Top-1 Acc.	Model Size	OPs ($\cdot 10^8$)	Top-1 Acc.	Model Size	OPs ($\cdot 10^8$)
BWN[34]	1/32	60.8	4MB	18.1	-	-	-
TTQ[46]	2/32	**66.6**	5.3MB	18.1	-	-	-
HWGQ[7]	1/2	59.6	4MB	∼2.4	64.3	5.1MB	∼3.4
LQ-Net[43]	1/2	62.6	4MB	∼2.4	**66.6**	5.1MB	∼3.4
SYQ[11]	1/2	55.4	4MB	∼2.4	-	-	-
DoReFa[45]	2/2	62.6	5.3MB	∼2.4	-	-	-
Ensemble[47]	(1/1)×6	61.0	-	-	-	-	-
Circulant-CNN[28]	(1/1)×4	61.4	-	-	-	-	-
ABC-Net[27]	(1/1)×5	65.0	8.7MB	7.8	-	-	-
GroupNet[27]	(1/1)×5	**67.0**	9.2MB	2.68	**70.5**	15.3MB	4.13
BNN[21]	1/1	42.2		1.57	-		-
XNOR-Net[34]	1/1	51.2		1.59	-		-
Bi-RealNet[30]	1/1	56.4		1.63	62.2		1.93
XNOR-Net++[6]	1/1	57.1		1.59	-		-
Bi-RealNet (our baseline)	1/1	60.6	∼4MB	1.14	63.7	∼5.1MB	1.43
BinaryDenseNet[5]	1/1	60.7		2.58	62.5		2.71
Strong Baseline[31]	1/1	60.9		1.82	-		-
BinaryDenseNet (our baseline)	1/1	62.6		2.09	64.2		2.20
MeliusNetA,B (ours)	1/1	**63.4**		1.62	**65.7**		1.96
32-bit baseline (ResNet)	32/32	69.3	46.8MB	18.1	73.3	87.2MB	36.6

ment for BNNs.

4.3. Comparison to State-of-the-art

The results of MeliusNetA and MeliusNetB compared to the state-of-the-art can be seen in Table 3. Previous work has often compared two different size categories, BiRealNet18 and BiRealNet34 [30], without taking into account the cost in operations and model size. For a proper cross-domain comparison to quantized approaches and approaches with multiple binary bases we included these numbers. In those cases where the authors did not reveal the exact numbers, we calculated them to the best of our knowledge. In addition to the accuracy reported by the original authors, we also report our baselines of Bi-RealNet [30] and BinaryDenseNet (BDenseNet) [5] with *grouped stem* and our training strategy for a fair comparison between these works focused on architecture design.

Comparison to other binary networks (one base):

Overall, MeliusNetA and B achieve the best accuracy compared to other approaches with 1-bit activations and 1-bit weights without additional cost. However, we recognize that by applying *grouped stem* to the Bi-RealNet architecture it can also achieve a much lower cost than our MeliusNet, which could be useful for certain applications despite its lower accuracy.

We limited the comparison to other works that use cross-entropy loss as a training objective. We note that Martinez *et al*. [31] showed that their approach with multi-stage knowledge distillation training can further enhance the accuracy and achieve 64.4% over their 60.9% accuracy of the "Strong Baseline". However, their approach is orthogonal to ours, since we focus on the architectural improvement and purposely use only cross-entropy loss for training. Similarly we have not included other work in Table 3, which uses more sophisticated training techniques, such as CI-Net [41] and BONN [13]. These works achieved 59.9% and 59.3% accuracy on ImageNet (for a 4 MB model) with their improved training strategy, respectively.

Comparison to quantized networks:

MeliusNet compares favorably to most quantized approaches (first section in Table 3), achieving a higher accuracy and lower resource cost than DoReFa [45] and HWGQ [7]. Some quantized approaches can achieve a higher accuracy than MeliusNet, but they also require a significantly higher model size or higher number of operations. TTQ [46] with 2-bit weights and 32-bit activations achieves

66.6% accuracy, but does not save any operations and has a higher model size. LQ-Net [43] achieves 0.9% higher accuracy, but also needs about 75% more operations.

Comparison to other binary networks (multiple bases):

Comparing MeliusNetA and B to approaches with multiple bases (second section in Table 3) reveals that both ABC-Net [27] and GroupNet [48] achieve better results. However, they come at a significant increase in model size and operations and represent a different approach of using multiple binary convolutions instead of a single binary convolution in each layer. Still, the exceptionally high accuracy of GroupNet partly achieves the level of MobileNet-v1, hence we examined it further in the next section with a comparison to the larger MeliusNet models.

Cross-domain comparison between BNNs and compact networks:

For another challenging comparison, we compared our results based on Bi-RealNet34, MeliusNetC, MeliusNet42, and MeliusNet59 to the compact network architecture MobileNet-v1 [19] in Table 4. Furthermore, we included the GroupNet approach [48] as an alternative BNN approach that uses 5 binary bases.

First of all, the comparison between MobileNet-v1 and MeliusNet shows small accuracy improvements between 0.4% and 0.8% across three different model sizes. For a model size of $\leq$5.1 MB, a Bi-RealNet34 trained with *grouped stem* also shows the potential to reach the same accuracy with a lower amount of operations. This shows that our proposed *grouped stem* structure can effectively reduce the gap between MobileNet-v1 and previous BNN work.

We note that the GroupNet approach can also achieve an accuracy similar to MobileNet-v1 1.0, although they have not shown the same level of accuracy for smaller model sizes, e.g., MobileNet-v1 0.5 and 0.75. In addition, GroupNet and MeliusNet differ in their approach. GroupNet replaced a single binary convolution with multiple ones while reusing a regular Bi-RealNet architecture, whereas MeliusNet uses a novel architecture but with a single binary convolution per layer. This also means both approaches could be combined in future work to achieve even more accurate BNNs.

We conclude that MeliusNet is a valid alternative to the decomposition strategy described in GroupNet, since it is more flexible for creating models with different size and number of operations. MeliusNet also shows very promising results to be comparable to MobileNet-v1 since it surpasses their accuracy for three different model sizes.

Table 4: Comparison of MobileNet-v1 [19], the GroupNet approach [48], which uses multiple binary bases, and our results, based on Bi-RealNet34 [30] and our binary MeliusNet on the ImageNet dataset [10]. With our method BNNs can achieve an accuracy similar to or even higher than MobileNet 0.5, 0.75 and 1.0.

Model size	Architecture	Bitwidth (W/A)	OPs ($\cdot 10^8$)	Top-1 acc.
9.2MB	GroupNet18 [48]	(1/1)×5	2.68	67.0%
15MB	GroupNet34 [48]	(1/1)×5	4.13	70.5%
5.1MB	Bi-RealNet34 [30]	1/1	1.93	62.2%
5.1MB	MobileNet-v1 0.5 [19]	32/32	1.49	63.7%
5.1MB	Bi-RealNet34* [30]	1/1	1.43	63.7%
4.5MB	**MeliusNetC**	1/1	1.50	**64.1%**
10MB	MobileNet-v1 0.75 [19]	32/32	3.25	68.4%
	MeliusNet42	1/1	3.25	**69.2%**
17MB	MobileNet-v1 1.0 [19]	32/32	5.69	70.6%
	MeliusNet59	1/1	5.32	**71.0%**

* This result is based on our training using grouped stem.

5. Conclusion

Previous work has shown different techniques to increase the accuracy of BNNs by increasing the channel numbers or replacing the binary convolutions with convolutions with multiple binary bases. The Bi-RealNet and the BinaryDenseNet approaches were the first to change the architecture of a BNN compared to a 32-bit network. In our work, we showed a novel architecture *MeliusNet*, which is specifically designed to amend the disadvantages of using binary convolutions. In this architecture, we repeatedly add new features and improve them to compensate for the lower quality and lower capacity of binary feature maps. Our experiments with different model sizes on the challenging ImageNet dataset show that MeliusNet is superior to previous BNN approaches, which adapted the architecture.

Further, we presented *grouped stem*, an optimized set of layers that can replace the first convolution. This has reduced the accuracy gap between BNNs and compact networks. With our method both previous architectures and our proposed MeliusNet can reach an accuracy similar to MobileNet-v1 0.5, 0.75, and 1.0 based on the same model size and a similar amount of operations. This provides a strong basis for BNNs to gain popularity and possibly achieve future milestones, such as reaching an accuracy similar to MobileNet-v2 or -v3. The higher energy saving potential of BNNs (based on customized hardware) could then make them the favorable choice for many applications.

References

[1] Milad Alizadeh, Javier Fernández-Marqués, Nicholas D Lane, and Yarin Gal. An Empirical study of Binary Neu-

ral Networks' Optimisation. *International Conference on Learning Representations*, 2019.

[2] Martin Arjovsky and Léon Bottou. Towards Principled Methods for Training Generative Adversarial Networks. *International Conference on Learning Representations (ICLR)*, 2017.

[3] Yoshua Bengio, Nicholas Léonard, and Aaron C Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *CoRR*, abs/1308.3, 2013.

[4] Joseph Bethge, Marvin Bornstein, Adrian Loy, Haojin Yang, and Christoph Meinel. Training competitive binary neural networks from scratch. *arXiv preprint arXiv:1812.01965*, 2018.

[5] Joseph Bethge, Haojin Yang, Marvin Bornstein, and Christoph Meinel. BinaryDenseNet: Developing an Architecture for Binary Neural Networks. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*, 2019.

[6] Adrian Bulat and Georgios Tzimiropoulos. XNOR-Net++: Improved binary neural networks. In *30th British Machine Vision Conference*, 2019.

[7] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5918–5926, 2017.

[8] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[9] Elliot J Crowley, Gavin Gray, and Amos J Storkey. Moonshine: Distilling with cheap convolutions. In *Advances in Neural Information Processing Systems*, pages 2888–2898, 2018.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. Ieee, 2009.

[11] Julian Faraone, Nicholas Fraser, Michaela Blott, and Philip H W Leong. Syq: Learning symmetric quantization for efficient deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[12] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[13] Jiaxin Gu, Junhe Zhao, Xiaolong Jiang, Baochang Zhang, Jianzhuang Liu, Guodong Guo, and Rongrong Ji. Bayesian Optimized 1-Bit CNNs. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[14] Jian Guo, He He, Tong He, Leonard Lausen, Mu Li, Haibin Lin, Xingjian Shi, Chenguang Wang, Junyuan Xie, Sheng Zha, Aston Zhang, Hang Zhang, Zhi Zhang, Zhongyue Zhang, and Shuai Zheng. GluonCV and GluonNLP: Deep Learning in Computer Vision and Natural Language Processing. *arXiv preprint arXiv:1907.04433*, 2019.

[15] Song Han, Huizi Mao, and William J Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In *International Conference on Learning Representations (ICLR)*, 2016.

[16] Song Han, Jeff Pool, John Tran, and William Dally. Learning both Weights and Connections for Efficient Neural Networks. In *Advances in Neural Information Processing Systems*, pages 1135–1143, 2015.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[18] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V Le, and Hartwig Adam. Searching for MobileNetV3. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[19] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. 2017.

[20] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pages 2261–2269, 2017.

[21] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016.

[22] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[23] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Deep Features for Text Spotting. In *Computer Vision – ECCV 2014*, pages 512–528, Cham, 2014. Springer International Publishing.

[24] Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. Learning to Quantize Deep Networks by Optimizing Quantization Intervals With Task Loss. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[27] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards Accurate Binary Convolutional Neural Network. In *Advances in Neural Information Processing Systems*, number 3, pages 344–352, 2017.

[28] Chunlei Liu, Wenrui Ding, Xin Xia, Baochang Zhang, Jiaxin Gu, Jianzhuang Liu, Rongrong Ji, and David Doermann.

Central circulant binary convolutional networks : enhancing the performance of 1-bit DCNNs with central circulant back propagation. *Cvpr*, pages 2691–2699, 2019.

[29] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the Variance of the Adaptive Learning Rate and Beyond. *arXiv preprint arXiv:1908.03265*, 2019.

[30] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-Real Net: Enhancing the Performance of 1-bit CNNs with Improved Representational Capability and Advanced Training Algorithm. In *The European Conference on Computer Vision (ECCV)*, sep 2018.

[31] Brais Martinez, Jing Yang, Adrian Bulat, and Georgios Tzimiropoulos. Training binary neural networks with real-to-binary convolutions. In *International Conference on Learning Representations*, 2020.

[32] Asit Mishra, Eriko Nurvitadhi, Jeffrey J Cook, and Debbie Marr. WRPN: Wide Reduced-Precision Networks. *International Conference on Learning Representations (ICLR)*, 2018.

[33] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *International Conference on Learning Representations*, 2018.

[34] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[35] Joseph Redmon, Santosh Kumar Divvala, Ross B Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[36] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems 28*, pages 91–99, 2015.

[37] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[38] Mingzhu Shen, Kai Han, Chunjing Xu, and Yunhe Wang. Searching for Accurate Binary Neural Architectures. *The IEEE International Conference on Computer Vision (ICCV) Workshops*, 2019.

[39] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *AAAI*, volume 4, page 12, 2017.

[40] Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1365–1374, 2019.

[41] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. HAQ: Hardware-Aware Automated Quantization With Mixed Precision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[42] Haojin Yang, Martin Fritzsche, Christian Bartz, and Christoph Meinel. BMXNet: An Open-Source Binary Neural Network Implementation Based on MXNet. In *Proceedings of the 2017 ACM on Multimedia Conference*, pages 1209–1212. ACM, 2017.

[43] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. LQ-Nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 365–382, 2018.

[44] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[45] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients. *arXiv preprint arXiv:1606.06160*, 2016.

[46] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *International Conference on Learning Representations (ICLR)*, 2017.

[47] Shilin Zhu, Xin Dong, and Hao Su. Binary Ensemble Neural Network: More Bits per Network or More Networks per Bit? In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[48] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian Reid. Structured Binary Neural Networks for Accurate Image Classification and Semantic Segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.