Supplementary Material for 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

Supplementary material

Our supplementary material contains the following information:

- Section 1 briefly explains the structure of the experiment data also contained within this archive
- Section 2 shows a comparison between MeliusNet and the naive approach of simply alternating Residual Blocks and Dense Blocks
- Section 3 contains data that shows some of the observed differences between the different optimizers (SGD, Adam, RAdam)

1. Detailed Experiment Data

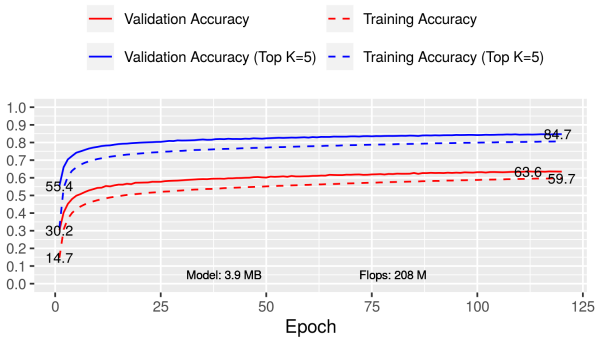
We include the experiment logs (`experiment.log`), accuracy curves (`accuracy.png`) and detailed plots (`network.pdf`) of our model architectures in one folder per experiment result within the parent folder "main_experiment_data". The accuracy curves also include the model size and number of operations of the corresponding model. An example of the accuracy curve of MeliusNet22 can be seen in Figure 1.

2. Comparing the Naive Approach and MeliusNet

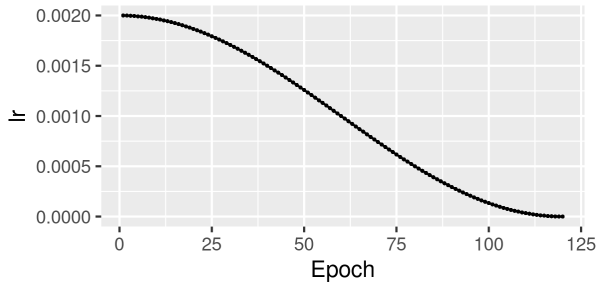
The direct approach to combining residual and dense shortcut connections could lead to a result as shown in Figure 2a. In this case the combination of a Dense Block and a Residual Block are repeated throughout the network. However, the residual shortcut connection requires that feature map sizes between the input and output of the convolution match. This means the number of channel contributes to the number of of operations quadratically. This makes achieving a reasonable number of operations difficult with this approach, since increasing the channel number (as is done in every Dense Block) leads to a quadratic increase of operations. Therefore, increasing the capacity of feature maps with this approach is not practical, especially for larger binary networks.

Figure 2b shows the MeliusNet for comparison. The design of our Improvement Block keeps the number of operations lower, since increasing the channel number with Dense Blocks only linearly increases the number of operations required for later blocks.

We also empirically evaluated both models. These experiments were trained for only 40 epochs and a different learning rate schedule (base learning rate is 0.001, decaying by 0.1 at epochs 35 and 37). However, since both models

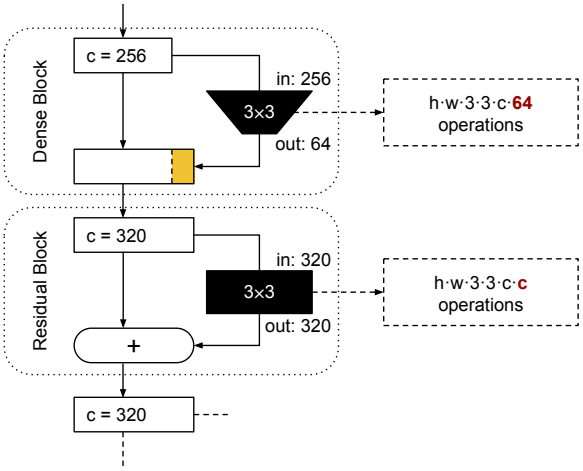


(a) The accuracy curve.

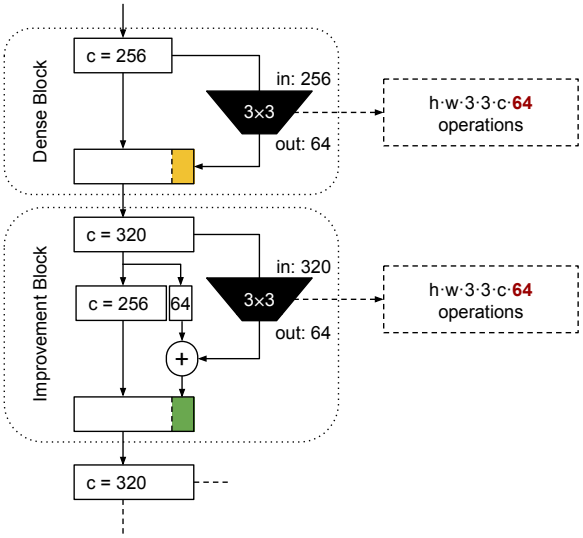


(b) The "cosine" learning rate scheduling.

Figure 1: A visualization of the training process of MeliusNet22.



(a) Naive approach



(b) MeliusNet

Figure 2: The basic building blocks of MeliusNet and the naive approach of repeating alternating Dense Blocks and Residual Blocks (c denotes the number of channels in the feature map). (a) With the naive approach, the number of operations in the Residual Block increases by the factor of c instead of the constant number of output channels (64) compared to the Dense Block. This means the Residual Block needs between 2 and 10 times the number of operations of the Dense Block, depending on the number of layers and depth of the layer in the network. Furthermore the number of weights and operations increases quadratically, depending on c, making anything except very shallow networks unfeasible. (b) Our MeliusNet for comparison. The number of operations between both blocks is similar (only the number of input channels changes slightly between the blocks).

were trained with the same hyperparameters this should not affect the comparison between both. Since we struggled to construct a model which could match in both model size and number of operations, we only made the number of operations equal. In the comparison we can see that the naive approach is much worse, with a 3% different in Top 1 accuracy on ImageNet (see Figure 3). Even with the slightly smaller model (3.3 MB instead of 4 MB) this drop in accuracy is too much compared to other binary models, e.g. Bi-RealNet or BinaryDenseNet. Therefore, we concluded that this approach is not useful for BNNs and have not pursued it further. The details of these experiments are in the folder "naive_vs_MeliusNet".

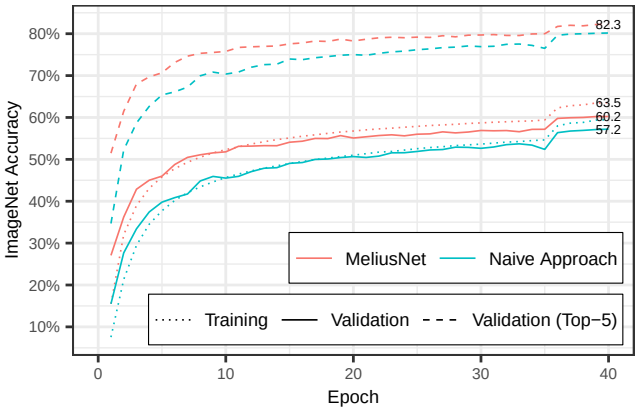
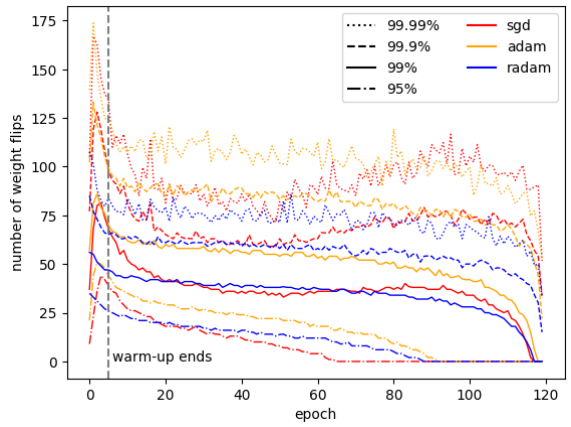


Figure 3: A comparison between MeliusNet and the naive approach of simply alternating a Residual Block and a Dense Block (see Figure 2a). Both models need about 258 million operations (factoring in the speedup of binary operations). The 3% accuracy drop of the naive approach is too large and the high number of operations needed for larger models make the naive architecture unfeasible for BNNs.

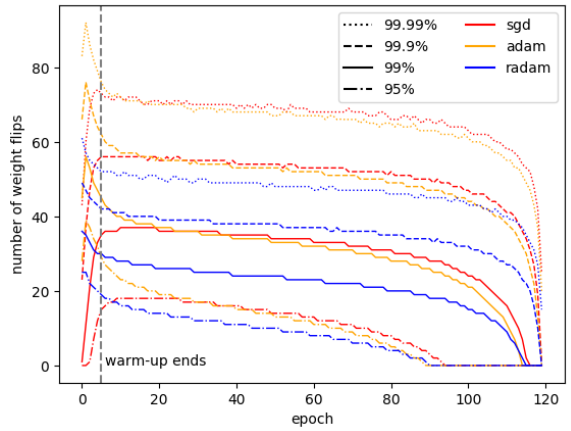
3. Optimizer Comparison

As written in the paper, we found, that both Adam and RAdam optimize better than SGD. We tried different learning rates and learning rate schedules, however, the accuracy on ImageNet when training with SGD still was about 1% lower than Adam (with warmup). Therefore, we counted the number of sign "flips" for each individual weight between batches (accumulated per epoch) for each optimizer during the training of ResNetE18 on ImageNet (see Figure 4). If a weight was updated from -1 to $+1$ when updating the weights after processing one batch its weight flip count would increase by one. This can happen several times per epoch and intuitively reflects the "stability" of the training process regarding the binary weights.

First of all, the data showed, that surprisingly, after about 90 epochs, 95% of all binary weights are stable during a single given epoch. Note that this does not mean that 95% of weights are stable for the *whole time* after the 90th epoch, since the 95% of stable weights are not necessarily identical between the different epochs.



(a) Data from the *first* binary convolution of the *first* network stage



(b) Data from the *last* binary convolution of the *last* network stage

Figure 4: We show the n-th percentile of the number of weight "flips" for each optimizer for the binary weights of two different convolution layers over the whole training process of 120 epochs for a ResNetE. The first 5 epochs are warm-up epochs for Adam and SGD, where the learning rate is increased linearly to the base learning rate. We can see, for example, that after the 100th epoch during a single given epoch 95% of weights are stable in these layers. Furthermore, for Adam and RAdam the stability increases during the training. This is not the case for SGD in the earlier layers of the network (e.g., in (a)), where the number of flips increases starting around epoch 60.

During the training with Adam and RAdam, the average stability increases during the training, while for SGD the stability decreases after about 50 epochs. However, this is only true for the earlier layers in the network (see Figure 4a), but does not apply to later layers (see Figure 4b). Although this is an indication for a more unstable training process with SGD it does not yet conclusively explain the performance difference to RAdam and Adam.