Initialization and Transfer Learning of Stochastic Binary Networks from Real-Valued Ones

Anastasiia Livochka
Ukrainian Catholic University
anastasiia.livochka@gmail.com

Alexander Shekhovtsov
Czech Technical University in Prague
shekhovtsov@gmail.com

Abstract

We consider the training of binary neural networks (BNNs) using the stochastic relaxation approach, which leads to stochastic binary networks (SBNs). We identify that a severe obstacle to training deep SBNs without skip connections is already the initialization phase. While smaller models can be trained from a random (possibly data-driven) initialization, for deeper models and large datasets, it becomes increasingly difficult to obtain non-vanishing and low variance gradients when initializing randomly.

In this work, we initialize SBNs from real-valued networks with ReLU activations. Real valued networks are well established, easier to train and benefit from many techniques to improve their generalization properties. We propose that closely approximating their internal features can provide a good initialization for SBN. We transfer features incrementally, layer-by-layer, accounting for noises in the SBN, exploiting equivalent reparametrizations of ReLU networks and using a novel transfer loss formulation. We demonstrate experimentally that with the proposed initialization, binary networks can be trained faster and achieve a higher accuracy than when initialized randomly.

1. Introduction

Neural networks with binary weights and activations have much lower computation costs and memory consumption than their real-valued counterparts [8, 10, 11, 21]. They are therefore very attractive for applications in mobile devices, robotics and other resource-limited settings, in particular for solving vision and speech recognition problems [2, 27].

Training binary neural networks poses several additional challenges in comparison to real-valued ones. First, they are not differentiable. This has been overcome in seminal works [7, 12] by using straight-through (ST) estimators. A clear understanding of such estimators is possible when considering a stochastic relaxation of a binary network [25]. The associated costs are i) the need to introduce noises in each layer in order to make the loss function differentiable in the expectation and ii) ST estimators are biased, which may have a detrimental effect [25]. The training can take longer than for real-valued networks but, with appropriate hyper-parameters, succeeds in practice, achieving close to 100% training accuracy. In our view, there remain two challenges: how to make a good initialization so that deeper binary models can be trained and how to improve the generalization gap to close up the performance of real-valued networks.

The challenge of initialization can be seen from the following observations. At the first approximation, binary activations are similar to sigmoid activations and experience the problem of vanishing gradients. Batch normalization [13] helps a lot with this problem as it initially performs a data-driven standardization of pre-activations [16, 24]. Nevertheless, large-scale models are typically trained in stages where the initialization stage uses *e.g.* real-valued activations [15]. Moreover, real-valued residual paths are used to alleviate the difficulty of training [14]. To our knowledge, a successful training of non-residual deep convolutional architecture on large-scale dataset (*e.g.* ImageNet) has not been demonstrated yet. In the framework of stochastic binary networks (SBN) [19, 22, 23, 25] an additional challenge is to control the noise level of the stochastic relaxation so that the stochastic gradient has a meaningful signal-to-noise ratio.

We propose to address both the generalization and the initialization challenges by transferring intermediate feature representations from available real-valued networks. Lower-level and mid-level features are known to be generic across different tasks and transfering them, *e.g.*, in fine-tuning, leads to easier training and better generalization for the target task [28].

Contribution In this work we transfer features from a real-valued ReLU network to SBNs of the same architecture on the same dataset. We do not yet demonstrate superior results in large-scale problems or cross-domain trans-

fer. Rather we propose a detailed study on a small-scale CIFAR-10 dataset. ReLU networks are most commonly used, readily available for many tasks and possess a better performance than the best binary networks. Nevertheless, initializing binary networks from them is more difficult due to the unbounded response of ReLU activations. We study how to binarize them block-by-block in order to preserve intermediate feature representations. We find out that it is essential to use ternary weights to transfer ReLU networks. Incorporating ternary weights requires only two binary convolutions and can be seen as just a moderate increase of a binary neural network's width. For the feature transfer objective we consider several choices and propose to use the means squared error (MSE) of activations in the *next* layer to the one transferred. Since the transfer itself is performed by SGD optimization, it, in turn, needs to be initialized. As the first approximation, we propose and study a new method that aligns feature statistics while taking into account noises in the SBN model.

Experimentally we show that SBN layers can, to a large extend, substitute ReLU layers, preserving at least a part of functionality when initialized just based on simple statistics and can be further optimized to produce deep representations close to the original ones. Training an SBN network initialized with the proposed method has an advantage in the beginning and allows it to achieve higher accuracy than when initializing randomly. With this approach we improve over non-residual SOTA results on CIFAR-10 while using a smaller and more cost-efficient network.

2. Related Work

The SOTA methods for training large-scale binary networks [4, 5, 15] use the initialization strategy of Bulat et al. [3]: to firstly train a network with real weights and binary activations and then gradually anneal weights to binary. This is similar to starting by training models with real weights and clipping-like non-linearities [6]. Other works [7, 23] indicate that training with binary weights and ReLU activations is a relatively simpler problem. Thus we see the binarization of activations as the main difficulty. Alizadeh et al. [1] shows that initializing latent weights from real-valued weights can provide a good initialization resulting in a seeped-up for training. At the same time Bulat et al. [3] mention that the transition from a fully real-valued network to a binary one [both weights and activations] causes a catastrophic loss in accuracy often comparable to training from scratch. Martínez et al. [15] explore regularizing transferring knowledge from the teacher network by adding a dissimilarity of attention maps during training. These attention maps only align simple statistics of activation maps, and as we experimentally show are not sufficient for transferring features.

Transferring knowledge from a real-valued network to a binary one has been considered by Du et al. [9]. The two networks are trained simultaneously in their work, interact both ways, however the interaction is only at the last layer. It therefore serves more as a regularization rather than allows to transfer generic intermediate representations. Mishra & Marr [17], Polino et al. [20] use knowledge distillation approach to train a quantized network, which utilizes the teacher's predictive distribution as soft targets.

In NN quantization, it is more natural to expect that a real-valued network can provide a good initialization. In particular MSE between quantized and non-quantized features has been considered [18]. Zhuang et al. [30] uses an auxiliary mixed-precision network that shares parameters and is trained jointly with the low-precision network. Zhou et al. [29] propose to regularize quantized weights by the MSE from the real-valued weights.

3. Background and Problem Statement

Binary neural networks are commonly represented in the literature with weights and activations in $\{-1, 1\}$ for mathematical convenience. At the test time, they can be equivalently implemented using $\{0, 1\}$ encoding, binary `XNOR` operations and few integer operations (summing bits in the output of the binary convolution and comparing it to a threshold). These operations are the most energy-efficient [8]. SOTA binary networks, however, also include floating-point multiplications [21] or floating-point parametric residual connections [5, 14, 15]. In the later architectures full-precision paths exist from any layer all the way to the network output layer. Such connections significantly mitigate the training difficulties but incur additional latency and energy costs.

Basic SBN We will aim at training a network that does not need floating point operations (except in the input and output layers). The basic BNN uses the following convolutional layers:

$$\textit{pre-activation} \quad a = W \circledast x + b, \tag{1a}$$

$$\textit{activation} \quad y = \operatorname{sign}(a), \tag{1b}$$

where x, y, W are $\{-1, 1\}$ binary, $\circledast$ denotes convolution with binary arguments and b is an integer.

Since binary activations are not differentiable and optimizing over binary weights creates a combinatorial problem, a basic stochastic relaxation of the network is considered. It is obtained by making binary entities stochastic:

$$W = \operatorname{sign}(V - Z), \tag{2}$$

$$y = \operatorname{sign}(a - \xi), \tag{3}$$

where V are real-valued *latent weights* and Z and ξ are (vectors) of independent noises with a simple distribution.

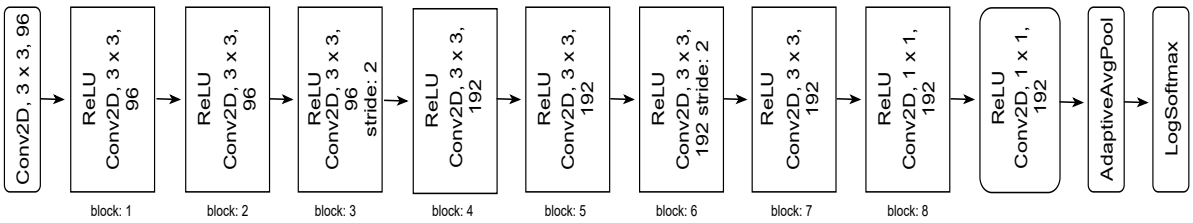


Figure 1: AllCNN network [26] used in our experiments and its partition in logical blocks. In order to convert this model to SBN, we process it sequentially, block by block, starting with the *block 1*. We first initialize a block to be converted analytically, append it to the SBN model and perform transfer learning to match features of the referenced network for this block and then proceed to the next block.

In this relaxation, the expected loss of the network is differentiable in latent weights V and the straight-through method provides a biased estimate of the true gradient [25]. We apply ST estimators as proposed by [25]: they recommend using logistic noises and identity straight-through estimator for weights while the activation noises can be chosen freely as long as the corresponding (non-identity) ST estimator is used for activations.

Problem Statement Considering the SBN model allows gradients and latent weights to have a mathematically precise meaning and allows us also mathematically model in which sense we want the SBN model to replicate the behavior of the ReLU network. As our working example we use the simple AllCNN architecture in Fig. 1 — it uses only convolutional layers with strides. We want to initialize an SBN model of the same structure with binary activations. Clearly, intermediate representations after binary and ReLU activations take values in different domains and cannot be aligned. We therefore define *internal features* of SBN at block k as pre-activation a^k and want them to approximate pre-activations $a^{\mathrm{R},k}$ of the ReLU network. Because a^k are stochastic due to injected noises, we pose the problem as finding parameters V, b such that the expected value of pre-activation over all SBN noises, $\mathbb{E}_{\mathrm{SBN}}[a^k]$, approximates the features $a^{\mathrm{R},k}$ in the teacher network. In doing so, we also need to keep the variance due to noises low enough in order to be able to capture the representation accurately with a small number of samples from SBN and to allow subsequent optimization with SGD.

4. Method

We initialize an SBN model from a ReLU model incrementally, replacing one block of layers at a time as illustrated in Fig. 1. We keep the first and the very last convolutional transforms of the ReLU network as real-valued. Prior work on quantization has shown that they can be at least quantized to low bit-width without accuracy degradation in practice and are not our main focus. Initialization of one block consists of an optimization-free block conversion which matches statistics of features over a batch of data, and a more precise optimization-based feature matching.

4.1. Initialization Based on Statistics

It is a common approach, *e.g.*, in domain adaptation, to align distributions by matching their means and variances. The affine transform that achieves such alignment is found by a simple formula of these statistics. It is, therefore, a cheap and reasonable first step to transfer internal representations. To allow a sensible transfer from the existing ReLU network, we need to make certain adjustments to the SBN architecture. We will show afterwards that at the test time we can obtain a binary network equivalent to the basic model (2). The goal of aligning preactivations leads to the following definitions of blocks to be converted:

ReLU Block We define a *block* of ReLU network with input $x^R \in \mathbb{R}^n$ and output features $a^R \in \mathbb{R}^m$ to consist of the following operations applied sequentially:

$$\textit{input scaling} \quad u^R = x^R \odot t^R, \tag{4a}$$

$$\textit{activation} \quad y^R = \mathrm{ReLU}(u^R), \tag{4b}$$

$$\textit{conv} \quad v^R = W^R * y^R, \tag{4c}$$

$$\textit{output scaling} \quad a^R = v^R \odot s^R + b^R, \tag{4d}$$

where all entries are real-valued and $\odot$ denotes *channel-wise multiplication* (t^R is a vector of the number of channels of x^R). Common feed-forward NNs can be represented as a sequence of such blocks, where the input and or output scaling may be set to $t^R = s^R = 1$. These scaling factors will be used later on to perform equivalent transformations on a block.

SBN Block In the SBN model we propose the following representation of one building block:

$$\begin{aligned} \textit{input scaling} \quad & u = x \odot t, && \text{(5a)} \\ \textit{activation} \quad & y = [\![u - \xi \geq 0]\!], && \text{(5b)} \\ \textit{conv} \quad & v = (W^+ \circledast y - W^- \circledast y), && \text{(5c)} \\ \textit{output scaling} \quad & a = v \odot s + b, && \text{(5d)} \end{aligned}$$

where, compared to the basic model (2) we have rearranged the order of operations, introduce a pair of (stochastic) binary $\{0, 1\}$ weights W^+, W^- and real-valued parameters t, s, b and used indicator $[\![u - \xi \geq 0]\!]$ instead of sign.

The following properties are important. 1) When such blocks are composed, the adjacent affine transforms from two blocks compose into one. The resulting affine transform can be hidden at test time under binary threshold using that $[\![x \odot s + b - \xi \geq 0]\!] = [\![x \geq \theta]\!]$, where $\theta = (\xi - b)/s$ is the equivalent threshold that can be precomputed (for a fixed noise sample ξ or its mean value). Thus at the test time, extra affine transforms *do not induce any extra computation* compared to (2).

2) Using the indicator $[\![\cdot]\!]$ instead of sign for activations appears equivalent for BNN, but it is not equivalent for SBN due to the noises present in W. As will be seen below, using the indicator better corresponds to ReLU activations and leads to lower variances downstream.

3) The difference of two binary convolutions is equivalent to a convolution of the binary input y with a *ternary* weight kernel $W^+ - W^-$. On one hand this allows us to use all the formalism and methods developed for SBNs [25]. On the other hand, ternary weights can model the teacher real-valued weights significantly more accurately. In particular, most of the weights in the teacher network are close to zero and can be removed from the network (zerowed) as demonstrated by works on network sparsity and compression. Therefore for transferring such weights it is important to be able to represent 0. The basic SBN model necessarily has all inputs connected to all outputs (by ± 1 weights) and needs a combination of highly correlated inputs and anti-correlated weights in order to represent sparsity. Consider also that the ternary block eq. (5) can implement AND, OR and NOT logical operations on y, which the basic SBN block cannot. We should note that the methods in the literature often increase the number of channels compared to the reference real-valued architecture (motivated by the need to increase the representation power of SBN). Increasing the number of channels twice leads to a $\times 4$ increase in the number of weights and computation cost. In comparison, the proposed block structure merely doubles the computation complexity.

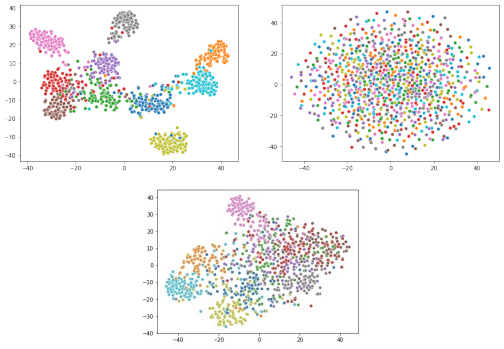


Figure 2: t-SNE embeddings of the last linear layer (*top left*), when converting one block using binary weights (*top right*) and when converting the same block using ternary weights (*bottom*) in section 4.1.2.

4.1.1 Initialization of Activations

We make the following observation: if we choose ξ to be *uniform* in $[0, 1]$, the expected activation value expresses as

$$\begin{aligned} E_z[y|u] &= \mathbb{P}(u - \xi \geq 0) = \mathbb{P}(\xi \leq u) \\ &= F_\xi(0) = \min(\max(u, 0), 1), \end{aligned} \quad (6)$$

which matches ReLU activation on $[-\infty, 1]$ and saturates to 1 for $u \geq 1$. Therefore, supposing $x = x^R$ (blocks in SBN and ReLU networks receive the same inputs), we will achieve matching of activations $\mathbb{E}_{\text{SBN}}[y] = y^{\mathbb{R}}$ if u^R is in the range $(-\infty, 1)$. We thus chose the scaling t^R such that

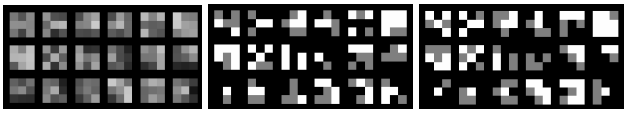
$$\mathbb{P}(x \odot t^R \leq 1) \geq \gamma, \quad (7)$$

where $\gamma < 1$ is a threshold and $\mathbb{P}$ is w.r.t. the empirical distribution over the inputs x in a batch as well as the spatial dimensions. Making γ closer to 1 reduces the approximation error, however, at the cost of a lower signal-to-noise ratio in y. We find γ experimentally in section 5.1. A common approach in the literature is to assume that the distribution of x is approximately Gaussian (*e.g.*, [8]). We have observed that it is a poor approximation *e.g.*, for networks with *Max-Pooling* such as VGG, leading to poor initialization performance for them. In contrast, we found that the empirical distribution works well in both cases (we tested AllCNN and VGG). When t^R is chosen, we divide the weight of the linear layer W^R channel-wise by t^R. Because ReLU is 1-homogenous, this preserves the equivalence of the whole block. The block so transformed is passed to the next step.

4.1.2 Initialization of Weights

Since the weights in SBN are stochastic, we match their expected values to real-valued weights W^R. With the logistic noise Z used for weights, we have, *e.g.* for W^+:

$$\mathbb{E}[W^+] = \mathbb{E}[[\![V^+ - Z \geq 0]\!]] = \mathbb{P}(V^+ \geq Z) = S(V^+), \quad (8)$$



(a) (b) (c)

Figure 3: Example of real-valued 3×3 kernels (a) and two samples of the initialized stochastic ternary model (b,c). It is seen that the initialization provides a reasonable set of filters with some entries more deterministic and some more noisy and amenable to changes in training.

where S is the logistic sigmoid function. Assuming W^R is positive, to achieve an expected value $\mathbb{E}[W] = W^R$, we set latent weights as $V^+ = \text{logit}(W^R)$ and $V^- = \text{logit}(10^{-13})$. The case when W^R is negative is symmetric. The matching of this expectation however makes sense only when $|W^R| \leq 1$. In order to meet these constraints we set

$$s_o^R = \max_{i,k,l} W^{\mathrm{R}}_{o,i,k,l}, \tag{9}$$

where i is input channel, o is output channel and k, l are spatial kernel coordinates. To preserve equivalence we must divide W^R by s^R in the *output channels*. As a result, the whole block of ReLU network preserves equivalence with the original one and we have $|W^R| \leq 1$.

As we discussed above many weights W^R in a pretrained network are often close to zero. If we were to approximate them with a single stochastic binary $\{-1, 1\}$ weight, we would obtain many weights "undecided", which are ± 1 with equal probability. If there are many such undecided states, the gradient of the model has a very high variance and a very small expectation. In the proposed ternary weight model, we can represent weights close to 0 with very low variance. Fig. 3 illustrates samples of stochastic ternary filters after the proposed initialization. Fig. 2 compares tSNE embeddings that we could achieve using binary weights and ternary weights and shows the loss of representation capability when using binary weights.

4.1.3 Initialization of Scales

Ideally, we want for all block inputs $x = x^R$ the expected SBN features a to match accurately the real-valued features a^R. So far we defined conversions of the first three operations of ReLU block eq. (4) and SBN block eq. (5) to preserve expectations of activations and weights separately. However, these steps are approximate and do not guarantee a good alignment of feature distributions. We therefore choose affine scaling parameters s, b in the last operation in order to align the first two moments of a and a^R. The non-standard part here, implied by the use of SBN model, is that we need to collect statistics over data and spatial dimensions while averaging out noises in a. Let

$$m_1^R = \mathbb{E}_{\text{data}}[a^R], \qquad m_2^R = \mathbb{E}_{\text{data}}[(a^R)^2], \tag{10a}$$

where m_1^R, m_2^R are vectors of the size of channels and the square is coordinate-wise. Note that these statistics do not change with the equivalent transforms applied to the ReLU block and can be computed on the initial network. Let $\bar{a} = \mathbb{E}_{\text{SBN}}[a]$ and let

$$m_1(\bar{a}) = \mathbb{E}_{\text{data}}[\bar{a}], \tag{11}$$

$$m_2(\bar{a}) = \mathbb{E}_{\text{data}}[\bar{a}^2]. \tag{12}$$

Recall that the notation $\mathbb{E}_{\text{SBN}}$ means averaging over all injected noises in the current as well as all preceding SBN blocks. Then we can compute s, b as:

$$s^2 = \frac{m_2(a^R) - (m_1(a^R))^2}{m_2(\bar{a}) - m_1(\bar{a})^2}, \tag{13a}$$

$$b = m_1(a^R) - m_1(\bar{a})s. \tag{13b}$$

In the experiments section 5.1 we show that the proposed initialization based on matching expectations and statistics of features allows to preserve a significant portion of accuracy, and especially so for deeper layers.

4.2. Optimization Based Feature Transfer

Given the statistics-based initialization of SBN, we proceed to the full transfer of intermediate features using gradient-based optimization. Still considering one block at a time we want to achieve a closer match between $\bar{a}$ and a^R. The task can be formulated as the optimization problem

$$\min_{V^+, V^-, t, s, b} \mathbb{E}_{\text{data}}\big[D(a, a^R)\big], \tag{14}$$

where D is a dissimilarity function, for which we will consider several choices. The common MSE dissimilarity is defined as $D_{\text{MSE}}(a, a^R) = \big\|\bar{a} - a^R\big\|^2$. Notice we apply it to the expected SBN features $\bar{a} = \mathbb{E}_{\text{SBN}}[a]$. As alternatives we consider *attention loss* [15], denoted D_{ATT}, which we also apply to expected features $\bar{a}$ and our proposed innovation: the MSE loss *on the next layer* applied for optimizing the current layer, denoted $D_{\text{MSE}+}$. The rationale for the later is that it still enforces proximity of inner representations while being robust to deviations in representations that are of no importance to the next layer.

4.3. Transforming the Whole Network

Prior to convering to SBN, if the teacher network has *BatchNorm* layers, we compute their test-time equivalent affine transforms to obtain blocks of the form (4). We perform the whole network conversion as described above: we

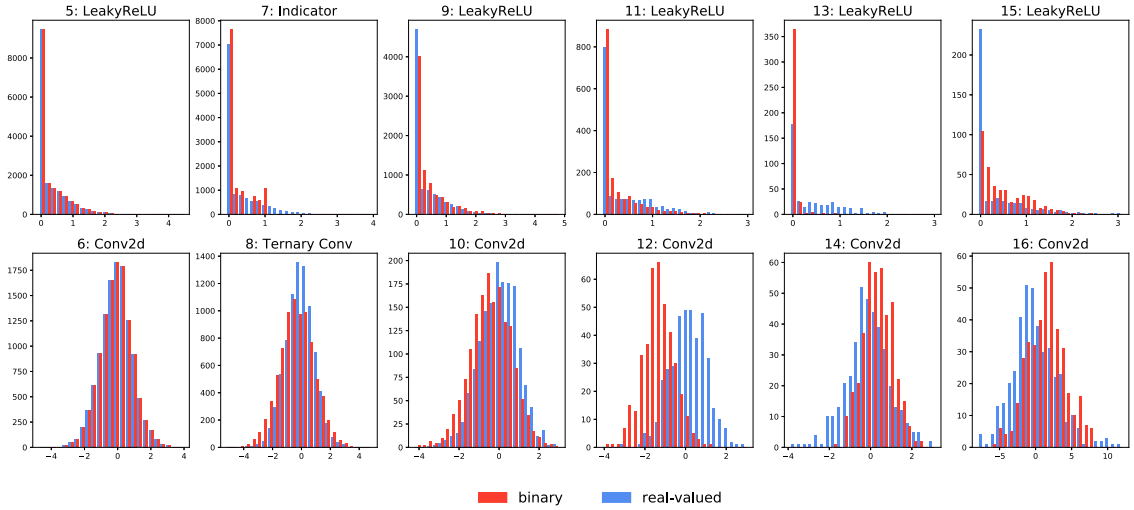


Figure 4: Statistics of units after each convolutional and linear layers following the converted block at layers (7,8), which correspond to *block # 4* at Fig. 1. The statistics with the SBN block (blue) are of the mean values over SBN noises approximated using 10 samples. Hence the histogram after binary activation (7) is not a discrete 0-1 distribution. Observe that even though the statistics after conv (8) are accurately aligned, more fine-grained differences in feature representations, not visible in the distributions, cause significant shifts downstream.

process it incrementally, starting from block 1, in each step initializing and re-optimizing the so-far converted part using the feature transfer formulation. For the optimization part we will experimentally explore possible options: which losss is better to use, should we adjust only the current block or including all the blocks below it, how long to train for the feature transfer, *etc*.

5. Experiments

For our experiments, we consider the CIFAR10 dataset and All-CNN architecture in Fig. 1 as a referenced ReLU network. This architecture has no max-pooling layers and no huge fully connected layers as in VGG. To study the individual steps of the initialization procedure we used a low-end model denoted AllCNN89, which was trained using common techniques and achieves 89% testing accuracy. In the final experiment we also transfer a high-end model of the exactly same architecture but trained using the method from [24] to 94.8% test accuracy[1], denoted AllCNN94.

First goals of experiments are to verify how the individual steps of the proposed initialization perform, where are the biggest losses in the accuracy and which choices in these steps lead to a better individual step performance as well as that of the full feature transfer. We then set to answer the following questions. Does a better initialization allow to achieve a better test performance and not merely speed up the training? Can the initialization by transferring intermediate features lead to improved generalization capabilities for the trained SBN?

5.1. Analytical Initialization

In Fig. 4 we illustrate how replacing one ReLU block (4) with SBN block (5) via proposed statistics based initialization aligns the distributions at this block, but at the same time causes shifts in the downstream distributions.

In the next experiment we study how the threshold γ for the activation initialization affects the accuracy of the network after the analytical matching. Fig. 5 illustrates the tendencies using several values of γ. We observe that converting blocks in the beginning of the network results in a significantly larger accuracy degradation ($\approx$ 40%) compared to that of converting deeper blocks ($\approx$ 10%). We also see that values close to 1 are sub-optimal for all blocks. We empirically chose threshold $\gamma \approx 0.85$ as a robust value for all layers.

5.2. Optimization-Based Feature Transfer

The next step after the analytical initialization of the parameters in the block is the matching of its output features. Here we conduct a series of experiments to determine a suitable loss for feature transfer, and the incremental optimization strategy.

In Fig. 6 we compare different losses applied at the feature transfer step. For all losses we used Adam optimizer with learning rate 0.001 and 5 samples of the SBN noises for estimating E_{SBN}. From Fig. 6 we see that the accuracy increases faster and achieves higher final value when optimizing $D_{\mathrm{MSE}+}$ rather than D_{MSE} and the previously used

[1]The model selection was made by the validation accuracy.

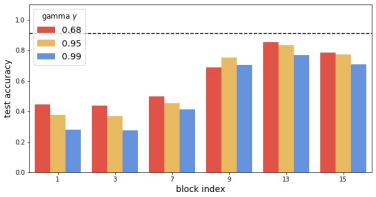


Figure 5: Test accuracy after statistics-based conversion of one block at given index (shown on x-axis) for different values of γ. Converting more shallow blocks incurs more significant loss of accuracy and γ has somewhat different effect at different depth. The dashed line represents the referenced network test accuracy.

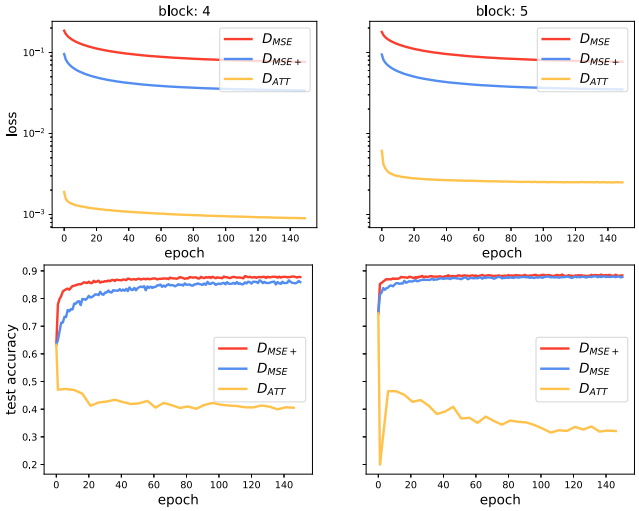


Figure 6: Comparison of different losses for feature transfer: D_{MSE}, $D_{\mathrm{MSE}+}$, D_{ATT}. We transfer a single block 4 (*left*) or 5 (*right*) in AllCNN89 by optimizing the respective loss, and measure the validation accuracy of the whole network. The network accuracy improves when optimizing the first two losses, but more so when optimizing $D_{\mathrm{MSE}+}$. The difference is substantial after epoch 1 and is significant even with more epochs. In contrast, while D_{ATT} can be easily optimized (top plots), its lower value does not secure a better accuracy (bottom plots).

attention loss [15] actually has a negative impact on accuracy.

In the next step we experiment with the incremental optimization strategy. The proposed incremental procedure leaves a few free choices: how many epochs to optimize and whether to optimize parameters of the already converted layers as we go deeper. In Fig. 7 we show that a large portion of the accuracy gap is closed in just one epoch of $D_{\mathrm{MSE}+}$ optimization. Training further improves the results only marginally. At the same time we found that one

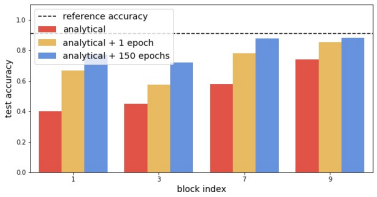


Figure 7: Comparison of the test accuracy with 1) analytical statistics-based initialization, 2) after one features matching epoch and 3) after 150 epochs. Optimization-based feature transfer improves the accuracy without using the class labels.

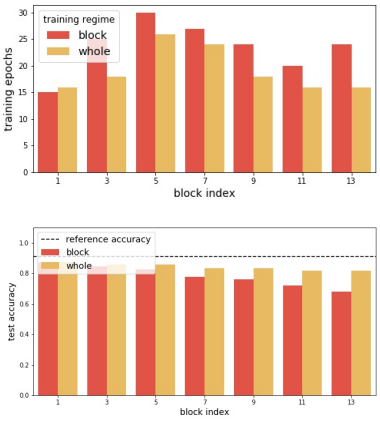


Figure 8: (*Top*) The number of training epochs needed in order to achieve an approximately stationary point of D_{MSE} when converting blocks incrementally. The result shows that more work is needed in the middle of the network. The graph also compares optimizing the current block parameters (red) or the whole current SBN network (yellow). (*Bottom*) Respective validation accuracy retained after the respective conversion stage (at block 13 the whole network is converted).

epoch per block was not sufficient for converting the whole network. In Fig. 8 (top) we show the number of epochs needed to optimize the target $D_{\mathrm{MSE}+}$ loss (achieve its gradient value $\leq \varepsilon$) depending on the block depth. We therefore use this strategy as a stopping criterion instead of a predefined number of epochs. It is found to be significantly larger for blocks in the middle. From Fig. 8 we also see that optimizing parameters of all the preceding blocks gives both faster training and more accurate final results as compared to optimizing only parameters of the current block.

5.3. Training with Class Labels

We now verify how supervised training with class labels and cross-entropy loss benefits from the proposed initial-

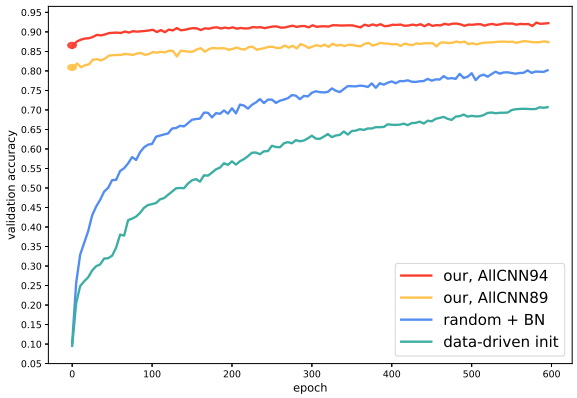


Figure 9: Validation accuracy of SBN models (using 10 samples) of the same network structure during their training when started from different initialization points. Our initialization has a significant advantage at the start and over 10% higher accuracy at the end of the training. The yellow and red circles indicate the accuracy when initialized (81.1% for *AllCNN89* and 86% for *AllCNN94*).

ization method. Towards this end we compare: 1) randomly initialized SBN with data-driven initialization [16] of scales and biases; 2) randomly initialized SBN with batch normalization layers; 3) Initialization with our method from *AllCNN89* and 4) initialization with our method from *AllCNN94*. Note that with the exception of 1) we do not use batch normalization, which appears to be a key component in SOTA BNN training.

The training setups was the same for all four cases: we use the same architecture obtained by replacing blocks of AllCNN with the respective blocks (5), denoted **AllCNN-T** (T for ternary), same gradient estimators and Adam optimizer with learning rate 0.001.

Based on the results in Fig. 9 and table 1 we make the following observations. **I)** Our initialization using either of the real-valued networks has a significant advantage over a random initialization, giving a well-performing starting point and keeping the advantage even after 600 epochs. **II)** A higher accuracy of the teacher network leads to nearly proportionally higher accuracy of the initial point as well as of the final obtained model. **III)** When initializing from a high-performance point, the trained SBN surpasses many previous SOTA results (that use non-residual models) in accuracy with an architecture that has a smaller computation cost. It also has a comparable accuracy in the deterministic mode (noises replaced by their mean values). This indicates that we have transferred some useful representations that lead to an easier training and a better generalization. It is also closing the gap in the performance ($\sim$ 2%), compared to the teacher network, suggesting that it has a sufficient representation capability. We expect that the results could be further improved using complementary techniques applied during final training, such as batch normalization or activation regularization [8].

Table 1: Comparison with SOTA. We compare the computation cost (memory, floating point, integer and binary operations) of test-time models and the test accuracy they achieve.

BINARY MODELS SIZE AND COMPLEXITY				
Model	Weights	FLOPS	IOPS	BOPS
AllCNN-T	**0.3** Mb	$\mathbf{5 \cdot 10^6}$	$\mathbf{4.2 \cdot 10^6}$	$\mathbf{2.7 \cdot 10^8}$
VGGSmall [12]	1.7 Mb	$7.5 \cdot 10^6$	$1.9 \cdot 10^7$	$6.1 \cdot 10^8$
Net 5 [8]	5 Mb	$2.3 \cdot 10^7$	$1.7 \cdot 10^8$	$5.4 \cdot 10^9$

CIFAR-10 TEST ACCURACY			
Method	Model	Deterministic	10-sample
Our	AllCNN-T	**91**	$\mathbf{92.6 \pm 0.1}$
Shekhovtsov & Yanush [25]	VGGSmall	89.7	90.5
Peters et al. [19]	VGGSmall	88.61	91.2
Hubara et al. [12]	VGGSmall	89.85	-
Rastegari et al. [21]	VGGSmall +scaling	89.83	-
Ding et al. [8]	Net 5	91.56	-
Bulat et al. [4]	17-cell*	93.7	-

* The quoted result if for basic data augmentation. The network has real-valued skip connections with parameters and a complicated structure; we could not estimate its complexity.

6. Conclusion and Future Work

We have proposed an initialization scheme for SBNs which incrementally converts ReLU network block-by-block, first using basic statistics and then re-optimizing for better feature transfer. This initialization scheme takes into account noises in SBNs and is invariant to the equivalent transformations of the original ReLU network. In contrast to using a pretraining phase [3, 6, 15] where models are in between real-valued and binarized networks and are not trained to high performance, we showed that with a ternary architecture and the proposed method it is possible to reuse already available high-performance ReLU networks to initialize SBNs by transferring their internal features and narrow the accuracy gap to them. However, further experimental verification of our observations on a large-scale data needs to be conducted.

Acknowledgment

The authors gratefully acknowledge support by Czech OP VVV project "Research Center for Informatics "(CZ.02.1.01/0.0/0.0/16019/0000765)".

References

[1] Alizadeh, M., Fernandez-Marques, J., Lane, N. D., and Gal, Y. An empirical study of binary neural networks' optimisation. In *ICLR*, 2019.

[2] Bulat, A. and Tzimiropoulos, G. Binarized convolutional landmark localizers for human pose estimation and face alignment with limited resources. In *ICCV*, Oct 2017.

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

[4] Bulat, A., Martinez, B., and Tzimiropoulos, G. BATS: Binary architecture search, 2020.

[5] Bulat, A., Martinez, B., and Tzimiropoulos, G. High-capacity expert binary networks. In *ICLR*, 2021.

[6] Choi, J., Wang, Z., Venkataramani, S., Chuang, P. I., Srinivasan, V., and Gopalakrishnan, K. PACT: parameterized clipping activation for quantized neural networks. *CoRR*, abs/1805.06085, 2018.

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

[8] Ding, R., Chin, T.-W., Liu, Z., and Marculescu, D. Regularizing activation distribution for training binarized deep networks. In *CVPR*, June 2019.

[9] Du, K., Zhang, Y., Guan, H., Tian, Q., Wang, Y., Cheng, S., and Lin, J. FTL: A universal framework for training low-bit DNNs via feature transfer. In *ECCV*, pp. 700–716, 2020.

[10] Esser, S. K., Merolla, P. A., Arthur, J. V., Cassidy, A. S., Appuswamy, R., Andreopoulos, A., Berg, D. J., McKinstry, J. L., Melano, T., Barch, D. R., di Nolfo, C., Datta, P., Amir, A., Taba, B., Flickner, M. D., and Modha, D. S. Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences*, 113(41):11441–11446, 2016.

[11] Horowitz, M. Computing's energy problem (and what we can do about it). In *International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 10–14, 2014.

[12] Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks. In *NeurIPS*, pp. 4107–4115, 2016.

[13] Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, volume 37, pp. 448–456, 2015.

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

[15] Martínez, B., Yang, J., Bulat, A., and Tzimiropoulos, G. Training binary neural networks with real-to-binary convolutions. In *ICLR*, 2020.

[16] Mishkin, D. and Matas, J. All you need is a good init. In *ICLR*, May 2016.

[17] Mishra, A. and Marr, D. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. In *ICLR*, 2018.

[18] Nagel, M., Amjad, R. A., van Baalen, M., Louizos, C., and Blankevoort, T. Up or down? adaptive rounding for post-training quantization, 2020.

[19] Peters, J. W., Genewein, T., and Welling, M. Probabilistic binary neural networks, 2019.

[20] Polino, A., Pascanu, R., and Alistarh, D. Model compression via distillation and quantization. In *ICLR*, 2018.

[21] Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. XNOR-Net: Imagenet classification using binary convolutional neural networks. In *ECCV*, pp. 525–542. Springer, 2016.

[22] Roth, W., Schindler, G., Fröning, H., and Pernkopf, F. Training discrete-valued neural networks with sign activations using weight distributions. In *European Conference on Machine Learning (ECML)*, 2019.

[23] Shayer, O., Levi, D., and Fetaya, E. Learning discrete weights using the local reparameterization trick. In *ICLR*, 2018.

[24] Shekhovtsov, A. and Flach, B. Stochastic normalizations as Bayesian learning. In *Asian Conference on Computer Vision*, pp. 463–479. Springer, 2018.

[25] Shekhovtsov, A. and Yanush, V. Reintroducing straight-through estimators as principled methods for stochastic binary networks, 2021.

[26] Springenberg, J., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for simplicity: The all convolutional net. In *ICLR (workshop track)*, 2015.

[27] Xiang, X., Qian, Y., and Yu, K. Binary deep neural networks for speech recognition. In *INTERSPEECH*, 2017.

[28] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? In *NeurIPS*, volume 27, 2014.

[29] Zhou, A., Yao, A., Wang, K., and Chen, Y. Explicit loss-error-aware quantization for low-bit deep neural networks. In *CVPR*, June 2018.

[30] Zhuang, B., Liu, L., Tan, M., Shen, C., and Reid, I. Training quantized neural networks with a full-precision auxiliary module. In *CVPR*, June 2020.