Progressive Stochastic Binarization of Deep Networks

David Hartmann
Institute of Computer Science
Johannes Gutenberg-University of Mainz
Staudingerweg 9, 55128 Mainz, Germany
dahartma@uni-mainz.de

Michael Wand
Institute of Computer Science
Johannes Gutenberg-University of Mainz
Staudingerweg 9, 55128 Mainz, Germany
mwand@uni-mainz.de

Abstract

We propose a stochastic binarization scheme for deep networks that approximates scalar products of weights and activations using progressive sampling of stochastic shifts. This representation has bounded relative error and thereby permits high accuracies at moderate sampling costs. Further, it allows for the first time a fully dynamic and localized control of accuracy. This not only enables a choice of accuracy at run-time, but also provides a new tool for adaptively focusing computational attention. A reference implementation is provided under a free license (https://github.com/JGU-VC/progressive_stochastic_binarization).

1 Introduction

Computational efficiency of machine learning algorithms is an important problem: Resource demands are a major obstacle, especially for low-power embedded and mobile applications in deep networks. The bulk costs of current deep networks usually comes in the form of evaluating scalar products, i.e., addition and multiplication of real numbers. From a perspective of digital circuits, multiplication is more expensive at a gate-level, as costs grow faster with precision. All floating-point operations also incur substantial costs for alignment/normalization. While hardware vendors have already started the support of low-precision floating point and integer formats, custom hardware has the potential for substantial further cost reductions. [10]

Our approach: We propose a new method, *progressive stochastic binarization (PSB)*, that combines ideas from stochastic computing, quantization and binarization: Activations are represented by integers. We replace multiplications by stochastic gating that samples adjacent powers of two. Accumulation increases the precision as needed. This allows fine-grained dynamic accuracy control; we call this *computational attention* and propose a two-stage algorithm that first computes a rough estimate of accuracy demands and then uses higher precision more sparsely.
We apply our scheme to common deep networks for image classification. For the *ResNet50v2* model we obtain 94.4% of the full-precision accuracy at 16 random samples and 98.6% at 64 samples. Using PSB, we are able to quantize all multiplications in a deep network at high levels of precision without retraining. Retraining is, nonetheless, still possible with our method and can improve results further. As a key contribution, our approach is the first quantization scheme that allows for dynamic control of precision at run-time, introducing *computational attention* through adaptive sampling as a novel, additional cost reductions mechanism. We sped up a *ResNet18* model further by about 60% while losing about 1% of accuracy by switching between 4 and 16 accumulations dynamically.

2 Related Work

Quantization: One avenue for savings in chip area is reducing the precision and avoiding floating point circuits. Many quantization techniques require special training of the quantized model or perform fine-tuning of pretrained full-precision models [10]. More recently, adaptive quantization

5th Workshop on Energy Efficient Machine Learning and Cognitive Computing (co-located with NeurIPS 2019).

has also been considered, for example by using learned step size quantization [3] or by learning stochastic quantizations [7].
Our method is intended for in-place deployment, additional fine-tuning of the model is optional. In contrast to quantization methods, our method allows for choosing the quality of the representation at run-time. The closest related work is ShiftCNN [4], which transforms pretrained weights into sums-of-integer-shift operations. The main difference is that ShiftCNN is deterministic and not progressive; the precision of the network is fixed after deployment; dynamic control is not possible.

Binarization: The limit case of quantization is binarization, where weights and/or activations can only take two values. This reduces costs dramatically, replacing multiplications or even all arithmetics by logical operations, but has serious negative impact on accuracy [10]. Closest work to ours is ABC-Net; multiple scaled binary coefficients build a new number representation for weights [6].
Our stochastic binarization closes the performance gap of binary neural networks in most of our experiments at the cost of running multiple stochastic accumulations. To the best of our knowlegde, other binarization techniques require changes to the model hyperparameter or even add additional hyperparameters [1]. Our technique changes the number representations in-place, without retraining and without hyperparameter tuning.

Alternative Network Design: Stochastic computation (SC) is tightly related to our approach. SC uses sequences of random bits whose mean is the intended number. Logarithmic quantization has also been used in SC, similar in spirit to our importance sampling scheme. [9]
The biggest difference of our work to SC is the interpretation of data-streams and intermediate results. SC represent data as continuous random streams. In contrast, we use fixed-point numbers for incoming data and intermediate results and only weights are random variables.

3 Capacitor Units and PSB-Nets

Current deep networks typically consist of linear combinations followed by ReLU non-linearities. We consider a single activation in a network: $y = \mathrm{relu}\left(\sum_{i=1}^{d} x_i \cdot w_i\right)$. Here, x denotes the d-dimensional activation of the previous layer and ω_i denotes the corresponding i-th weight. We omit the bias term for clarity (w.l.o.g., we can set $x_0 \equiv 1$). In addition, architectures typically add batch normalization to each activation. This mapping can be "folded" into the preceding or following linear layer [5]. As our method approximates multiplications stochastically, it is crucial to fold successive multiplications into a single one to avoid high-variance estimates.

Stochastic multiplication: A multiplication $w \cdot x$ of a real number $w \in [0, 1]$ and an integer x can be estimated stochastically by $B_w \cdot x$ where $B_w \in \{0, 1\}$ is a Bernoulli random variable with probability $\Pr(B_w = 1) = w$. The expected value is equal to the original multiplication: $E[B_w \cdot x] = w \cdot x$. Thus, we substitute all multiplications $w \cdot x$ (after folding), $w \mapsto B_w$, and obtain a statistical approximation of the preactivation.

Value-based importance sampling: The limited range of binary weights is a drawback: probabilities w tend to gather around the borders 0 and 1 [2]. This reduces the amount of obtainable information of each gradient descent step, as gradients that point outside the weight-domain $[0, 1]$ get discarded. We propose an unbiased encoding that replaces floating point multiplications in-place. Floating point representations consist of three parts: $s \cdot 2^e \cdot m$. The sign $s \in \{-1, 1\}$ and the exponent $e \in \mathbb{N}$ adjust the coarse magnitude, the mantissa m fixes the more accurate decimal places. We replace the mantissa by a either the number 1 or 2, with probability chosen to match the true mantissa in expectation. Thus, we convert each weight w to an exponent $e \in \mathbb{Z}$, a sign $s \in \{-1, 1\}$ and a mantissa $p \in [0, 1]$:

$$w \mapsto \overline{w} := s \cdot 2^e \cdot (B_p + 1)\,, \tag{1}$$

where $s := \mathrm{sign}\,(w)\,,\ e := \lfloor \log_2 |w| \rfloor$, and $p := \frac{|w|}{2^e} - 1$. This can be implemented by stochastic bitshifts of $\cdot 2^{e+1}$ with probability p and $\cdot 2^e$ with probibility $1 - p$.
By construction, the mean is equal to w: $E[\overline{w}] = s \cdot 2^e \cdot (|w|/2^e - 1 + 1) = s \cdot |w| = w$.

Capacitor Units: We even out the stochasticity using additive accumulators: the variance of the binarization scheme in Equation (1) is reduced by drawing multiple samples and averaging the outcome. Due to the non-linearity of deep networks we run statistical averaging layer-wise, averaging over multiple samples *before* applying the non-linearity. This design utilizes only low-precision integer addition and does not require multiplication.

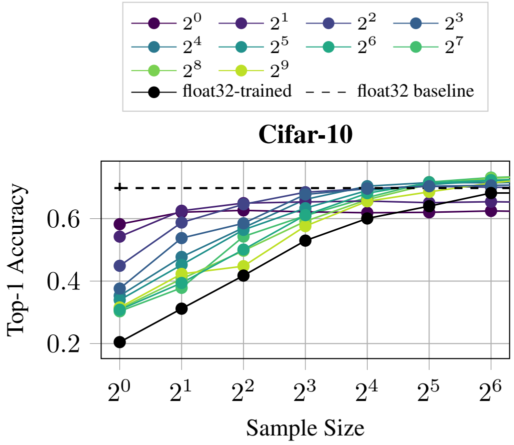

Figure 1: Training PSB from scratch on Cifar-10, with training specific to various sample sizes, evaluated at different sampling levels. Dashed black line: floating-point accuracy; solid black line: In-Place-PSB (no retraining).

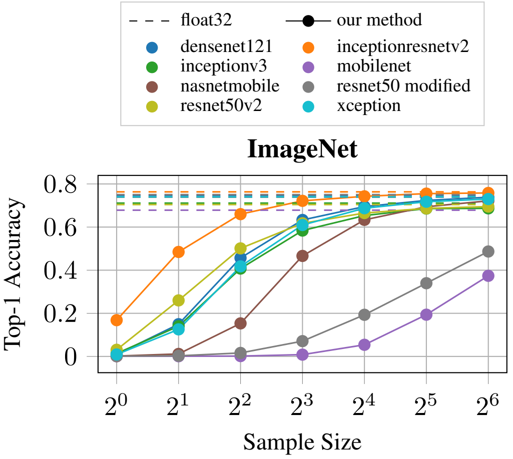

Figure 2: Pretrained ImageNet models after binarizing using In-Place-PSB with different sample sizes. Dashed lines: original floating-point performance. No retraining is used in any ImageNet test.

		Base sample size						Base sample size						Base sample size						Base sample size					
		1	2	4	8	16	32	1	2	4	8	16	32	1	2	4	8	16	32	1	2	4	8	16	32
Δ Accuracy Refinement	1	-54%						-53%						-51%						-54%					
	2	-50%	-28%					-40%	-26%					-36%	-25%					-31%	-29%				
	4	-47%	-25%	-12%				-34%	-19%	-10%				-24%	-15%	-10%				-15%	-13%	-11%			
	8	-46%	-23%	-10%	-4%			-28%	-14%	-7%	-3%			-17%	-10%	-5%	-3%			-7%	-5%	-4%	-4%		
	16	-47%	-24%	-9%	-3%	-1%		-25%	-12%	-5%	-2%	0%		-14%	-7%	-3%	-1%	0%		-4%	-2%	-1%	-1%	-1%	
	32	-45%	-23%	-9%	-3%	0%	0%	-24%	-11%	-4%	-1%	0%	0%	-12%	-5%	-2%	0%	0%	0%	-2%	-1%	0%	0%	0%	0%
Sampl. cost Refinement	1	-96%						-96%						-96%						-96%					
	2	-95%	-93%					-95%	-93%					-94%	-93%					-94%	-93%				
	4	-93%	-91%	-87%				-93%	-91%	-87%				-89%	-88%	-87%				-89%	-88%	-87%			
	8	-89%	-87%	-83%	-75%			-89%	-87%	-83%	-75%			-80%	-79%	-77%	-75%			-79%	-79%	-78%	-75%		
	16	-81%	-79%	-74%	-66%	-50%		-80%	-79%	-74%	-66%	-50%		-61%	-60%	-58%	-55%	-50%		-60%	-60%	-59%	-56%	-50%	
	32	-64%	-62%	-58%	-50%	-33%	0%	-63%	-62%	-58%	-49%	-33%	0%	-22%	-22%	-20%	-17%	-12%	0%	-21%	-22%	-21%	-18%	-12%	0%
		random 34% (a)						entropy (b)						random 76% (c)						entropy + border (d)					
		34% covering												**76% covering**											

Table 1: Computatation Attention — loss of accuracy (upper row) and loss of sampling costs (lower row) for a given choice of base sample size for a first rough estimate. Interesting regions are refined with the help of the random or entropy-based heuristics. The left half represents two heuristics that cover about 34% of the test set images, the right one cover about 76%.

We call this step a *capacitor unit*. Thus, we adapt Equation (1) to

$$w \mapsto \overline{w}_n := s \cdot 2^e \cdot \left(\frac{B_{n,p}}{n} + 1\right), \tag{2}$$

where n denotes the sample size, and $B_{n,p}$ the n-fold binomial distribution with probability p.

Variance: The variance of n samples is $\mathrm{Var}(\overline{w}_n) \leq \frac{w^2}{n \cdot 8}$. It is locally maximal between two shifts, i.e. whenever $p = 0.5$. The relative error of the number system is bounded by a constant: $\frac{\sigma_{\overline{w}_n}}{|E[\overline{w}_n]|} \leq \frac{1}{\sqrt{n \cdot 8}}$.

4 Experimental Results

To evaluate accuracy-vs-sampling trade-offs, we implemented Equation (2), sampling the corresponding filters directly as Binomial distributions. We quantize intermediate results to 16-bit integers.

Training PSB from Scratch on Cifar-10: Retraining vs. In-Place-PSB: We train a simple convolutional network on Cifar-10 (eight layers of 3x3 conv, BN and ReLU). Figure 1 compares the results of the model trained with floating point arithmetics to the same model trained with our method (PSB). We train a full precision network as a baseline and use this pretrained model to evaluate our number representation with varying number of samples. Next, we evaluate the effect of training directly on PSB using training sample sizes ranging from 2^1 to 2^5 and then reevaluating the pretrained weights

Experiment		Number System	Accuracy Top-1 [%]
baseline		float32	69.7
		psb5	68.2
		psb4	67.1
		psb2	54.7
+ pruning	25%	float32	69.0
		psb4	65.8
	50%	float32	41.5
		psb4	35.3
+ discrete	4-bit	psb4	66.7
p-values	2-bit	psb4	62.7
	1-bit	psb4	31.3
+ attention	random 37%	psb1/5	44.7
	entropy	**psb1/5**	**57.1**
	random 76%	psb2/5	65.7
	entropy + b	**psb2/5**	**67.7**
= combined		**psb1/5**	**57.4**
		psb2/5	**67.8**

Table 2: Classification error for a `float32`-pretrained ResNet18. Note: `psb`k refers to our method with 2^k-fold sampling, `psb`j/k refers to our method with 2^j-fold base sampling and 2^{k-j} fold additional refinement.

Method	**Number of Bits** Weights, Act.	**Accuracy** **Top-1** [%]
LSQ [3]	4, 4	70.9
DoReFa [10]	4, 4	68.1
INQ [10]	2, ·	66.6
BWN [10]	1, ·	60.8
XNOR-Net [10]	1, 1	51.2
ABC-Net [6]	5, 5	65.0
ABC-Net [6]	1, 1	42.0

Table 3: Classification Top 1-Accuracy; ResNet18 on ImageNet for a selection of quantization and binarization methods.

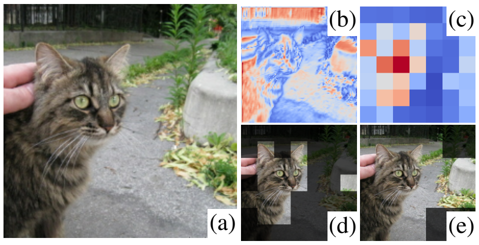

Figure 3: Attention. (a) ImageNet input; (b) first and (c) last layer errors: psb5 vs. float32; (d) entropy and (e) entropy with 2px border attention. We show the mean error over 100 independent repetitions of psb5 inference.

with other sample sizes. Results show higher accuracies when trained directly with PSB, compared to converting float32-weights. This is in accordance with other quantization schemes.

ImageNet: Next, we evaluate PSB on several (float32-) pretrained ILSVRC-models (Figure 2) without retraining. We use several pretrained state-of-the-art architectures, mostly *Inception-* or *ResNet*-Types [10], that provide easily foldable batch normalizations (BN). In most cases, PSB already yields half of the accuracy of an unbinarized network with only four samples. With increasing sample size the results approach full precision quickly. One of two exceptions is *MobileNet* [10] that uses separable convolutions with intermediate BNs. These problems are in line with previous work that combines MobileNet and quantization [8]. To emphasize the requirement of avoiding direct successive stochastic multiplications, we have added an experiment with a modified ResNet model, *Resnet50 modified*, with BNs after each shortcut (named "BN after addition" in the original ResNet paper). Here, data that follows the shortcut undergoes multiple unfolded multiplications, leading to large variance. A small architectural modification (standard ResNet) avoids this issue. We have added a selection of methods from literature (Table 3), however, a direct comparison should be made with caution: The cited methods include retraining or training from scratch, thus achieve good results in a low-precision setting, whereas our method does not require any fine-tuning of the pretrained models.

Weight Pruning & Discretization of Probabilities: We now evaluate modifications that allow for more computational or memory efficiency. We focus on the ResNet18 model again and evaluate it with float32 and with PSB of 2^4 samples. First, we remove the 25% and 50% of all weights closest to zero. As shown in Table 2, pruning of 25% has only moderate influence. Next, we reduce the memory footprint by quantizing the probabilities in Table 2 regularly on $p \in [0, 1)$ to to 4, 2 and 1 bits. The accuracy does not change much with discretized probabilities in a stochastic setting, but drops significantly for the discrete case (1-bit). As we use 16-bits fixed point numbers for intermediate results, we conclude that 4-bit exponents and 4-bit probabilities are sufficient for our scheme.

Computational Attention: In this section, we take advantage of the adaptive sampling property of our method using the same ResNet18 model as above, pretrained on ImageNet. First, we observe that the approximation error, $|\frac{x_{\texttt{psb5}} - x_{\texttt{float32}}}{x_{\texttt{float32}}}|$, using our method (with 2^5 samples) compared to floating point calculations seems to follow local features in the last layer (Figure 3c, red regions indicate higher error). We optimize the overall computational attention by evaluating the network with a varying sample precision. First, we obtain interesting regions by using the network in a low-precision mode on the full image. Then we refine those regions using a high-precision mode of the same network. As a heuristic for these regions, we use the estimated pixelwise entropy, h_{xy} of the image in the last convolutional layer by $h_{xy} \approx \sum_c -\text{softmax}(a_{xyc}) \cdot \log(\text{softmax}(a_{xyc}))$, where a_{xyc} denotes the activation of the last layer in the pixel (x, y) and channel c. Note that this is a more expensive computation, but it is only run on the very coarse-resolution top layer. For our mechanism, we use a threshold at the mean entropy in the image to estimate the interesting regions (Figure 3e). For the ImageNet testset this results in a ratio of about 34% of interesting regions of higher entropy and 66% of regions of lower entropy. If we surround the prediction by two pixels, we get a ratio of 76% interesting regions (due to the relatively small picture size of ImageNet). We use these regions to adapt precision of each filter individually using scaled masks (Table 2, "+attention"). In Table 1, we evaluate our method for all combinations of base sampling and refinement up to 2^5 samples and compare the results with a random proposal of same ratio. While the random heuristic with a coverage of 34% naturally fails to identify the interesting regions (poor performance for few base sample sizes), refinement of high-entropy regions increases accuracy significantly.

5 Conclusion

We have introduced progressive stochastic binarization (psb), an unbiased floating-point approximator. Our method converts network weights into stochastic shift operations, yielding a promising tool for future deep network hardware. PSB converts pretrained models in-place and thus, is particularly advantageous when training is expensive or unrepeatable (e.g. online learning). However, a thorough evalution of compute savings and accuracy-vs-energy trade-offs is still subject of future work. Performance wise, we match the accuracy on ImageNet-classification of previous binarized approaches in a low-precision setting, and in a high-precision regime our method is accuracy-wise competitive with previous quantization schemes. The method also permits localized, dynamic accuracy control within a single network, providing a new tool for adaptively focusing computational attention; only few quantization schemes provide this feature. We use this feature to direct computational costs adaptively using the network itself as an attention proposal mechanism for better classification results.

References

[1] Bethge, J., Yang, H., Bartz, C., and Meinel, C. Learning to train a binary neural network. *CoRR*, abs/1809.10463, 2018.

[2] Darabi, S., Belbahri, M., Courbariaux, M., and Nia, V. P. BNN+: improved binary network training. *CoRR*, abs/1812.11800, 2018.

[3] Esser, S. K., McKinstry, J. L., Bablani, D., Appuswamy, R., and Modha, D. S. Learned step size quantization. *CoRR*, abs/1902.08153, 2019.

[4] Gudovskiy, D. A. and Rigazio, L. Shiftcnn: Generalized low-precision architecture for inference of convolutional neural networks. *CoRR*, abs/1706.02393, 2017.

[5] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[6] Lin, X., Zhao, C., and Pan, W. Towards accurate binary convolutional neural network. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 344–352, 2017.

[7] Shayer, O., Levi, D., and Fetaya, E. Learning discrete weights using the local reparameterization trick. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[8] Sheng, T., Feng, C., Zhuo, S., Zhang, X., Shen, L., and Aleksic, M. A quantization-friendly separable convolution for mobilenets. *CoRR*, abs/1803.08607, 2018.

[9] Sim, H. U. and Lee, J. Log-quantized stochastic computing for memory and computation efficient dnns. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference, ASPDAC 2019, Tokyo, Japan, January 21-24, 2019*, pp. 280–285, 2019.

[10] Sze, V., Chen, Y., Yang, T., and Emer, J. S. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.