Journal of Machine Learning Research 22 (2021) 1-20 Submitted 12/20; Revised 8/21; Published 8/21

CAT: Compression-Aware Training for Bandwidth Reduction

Chaim Baskin* CHAIMBASKIN@CAMPUS.TECHNION.AC.IL
Technion – Israel Institute of Technology, Haifa, Israel

Brian Chmiel* BRIAN.CHMIEL@INTEL.COM
Habana Labs – An Intel company, Caesarea, Israel

Evgenii Zheltonozhskii* EVGENIIZH@CAMPUS.TECHNION.AC.IL
Technion – Israel Institute of Technology, Haifa, Israel

Ron Banner RON.BANNER@INTEL.COM
Habana Labs – An Intel company, Caesarea, Israel

Alex M. Bronstein BRON@CS.TECHNION.AC.IL
Technion – Israel Institute of Technology, Haifa, Israel

Avi Mendelson AVI.MENDELSON@TECHNION.AC.IL
Technion – Israel Institute of Technology, Haifa, Israel

Editor: Sebastian Nowozin

Abstract

One major obstacle hindering the ubiquitous use of CNNs for inference is their relatively high memory bandwidth requirements, which can be the primary energy consumer and throughput bottleneck in hardware accelerators. Inspired by *quantization-aware training* approaches, we propose a compression-aware training (CAT) method that involves training the model to allow better compression of weights and feature maps during neural network deployment. Our method trains the model to achieve low-entropy feature maps, enabling efficient compression at inference time using classical transform coding methods. CAT significantly improves the state-of-the-art results reported for quantization evaluated on various vision and NLP tasks, such as image classification (ImageNet), image detection (Pascal VOC), sentiment analysis (CoLa) , and textual entailment (MNLI). For example, on ResNet-18, we achieve near baseline ImageNet accuracy with an average representation of only 1.5 bits per value with 5-bit quantization. Moreover, we show that entropy reduction of weights and activations can be applied together, further improving bandwidth reduction. Reference implementation is available.

Keywords: deep learning, neural network compression, efficient inference, entropy encoding, custom hardware for deep learning

1. Introduction

Deep Neural Networks (DNNs) have become a popular choice for a wide range of applications such as computer vision, natural language processing, autonomous cars, etc. Unfortunately, their vast demands for computational resources often prevent their use on power-challenged platforms. The desire for reduced bandwidth and compute requirements of deep learning models has driven research into quantization (Hubara et al., 2016; Yang et al., 2019b; Liu

∗. Equal contribution.

et al., 2019; Gong et al., 2019), pruning (LeCun et al., 1990; Li et al., 2017; Molchanov et al., 2019), and sparsification (Gale et al., 2019; Dettmers and Zettlemoyer, 2019).

In particular, quantization works usually focus on scalar quantization of the feature maps: mapping the activation values to a discrete set $\{q_i\}$ of size L. Such representation, while being less precise, is especially useful in custom hardware, where it allows more efficient computations and reduces the memory bandwidth requirement. In this work, we focus on the latter, which has been shown to dominate the energy footprint of CNN inference on custom hardware (Yang et al., 2017a). We show that the quantized activation values $\{q_i\}$ can further be coded to reduce memory requirements.

Raw quantized data require $\lceil \log_2(L) \rceil$ bits per value for storage, which quantity can be reduced by compressing the feature maps. In particular, in the case of element-wise compression of independent identically distributed values, the lower bound of the amount of bits per element is given by the entropy (Shannon, 1948):

$$H(\mathbf{q}) = -\sum_{i=1}^{L} p(q_i) \log_2 p(q_i) \tag{1}$$

of the quantized values $\{q_i\}$, where $p(q_i)$ denotes the probability of q_i.

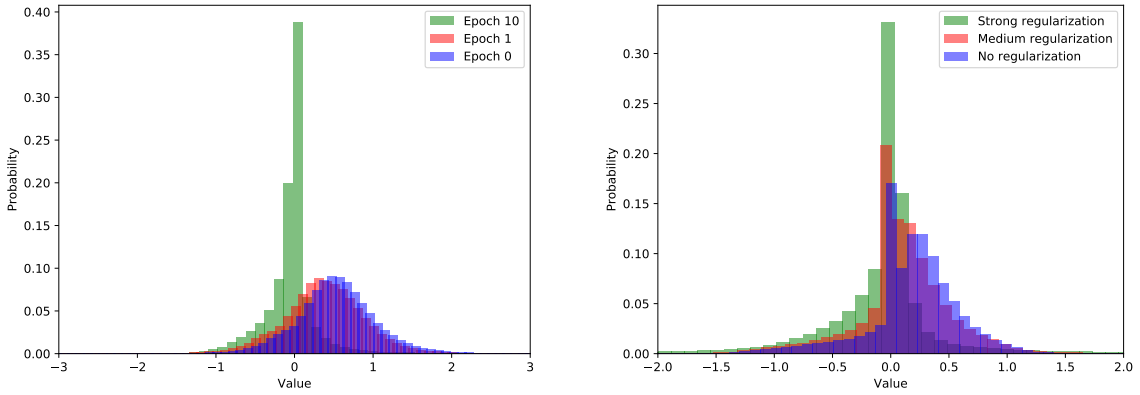
In this work, we take a further step by manipulating the distribution of the quantization values so that the entropy $H(q)$ is minimized. To that end, we formulate the training problem by augmenting the regular task-specific loss (the cross-entropy classifier loss in our case) with the feature map entropy serving as a proxy for the memory rate. The strength of the latter penalty is controlled through a parameter $\lambda > 0$. Fig. 1 demonstrates the effect of the entropy penalty on the compressibility of the intermediate activations.

In contrast to previous works that employed entropy encoders (Agustsson et al., 2017; Aytekin et al., 2019) for weight compression, we focus on compression of activations. Activations are responsible for a significant part of the memory I/O during inference (Yang et al., 2017b; Siu et al., 2018), and their efficient encoding provides significant benefits in terms of power. Nevertheless, we show that the proposed method is compatible with the previously proposed weight compression approaches.

Our paper makes several contributions. Firstly, we introduce Compression-Aware Training (CAT), a novel technique for memory bandwidth reduction. The method works by introducing a loss term that penalizes the entropy of the activations at training time and by applying entropy encoding (e.g., Huffman coding) on the resulting activations at inference time. Similar to many successful and widely-used techniques in deep learning, the proposed scheme is almost straightforward to implement.

Since the only overhead of the method at inference time is entropy encoding, the improvement is universal for any hardware implementation, being especially efficient on computationally optimized ones, where memory I/O dominates the energy footprint (Yang et al., 2017a; Jouppi et al., 2017).

We demonstrate a two- to fourfold memory bandwidth reduction for multiple architectures: MobileNetV2 and ResNet on the ImageNet visual recognition task, SSD512 on the PASCAL VOC object detection task, BERT on CoLa sentiment analysis task and BERT on MNLI textual entailment task. We also investigate several differentiable loss functions that lead to activation entropy minimization and show a few alternatives that lead to the same effect.



(a) (b)

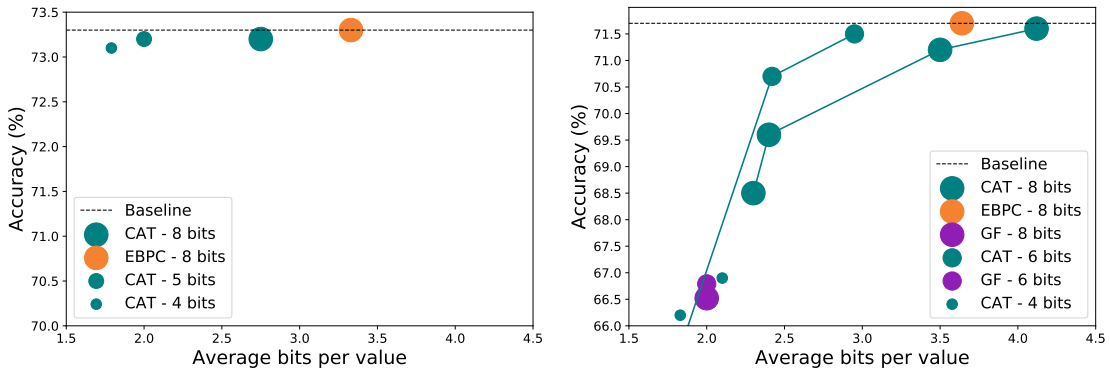
Figure 1: Pre-activation distributions of one layer in ResNet-18. **(a) Evolution at different epochs.** As training progresses, the probability of non-positive pre-activation values increases, zeroing more post-ReLU values. The sharp peak at zero reduces entropy and thus improves compressibility. **(b) Effect of entropy regularization.** Without regularization, the distribution has much heavier tails and thus has higher entropy. As regularization increases, the probability of extreme values is significantly reduced. The entropy penalty λ was selected so that the overall accuracy is not affected. The compression ratio in the strongly regularized case is 2.23 times higher compared to the unregularized the baseline.

Moreover, applying the same regularization to both the weights and the activations allows a further reduction in the memory bandwidth with only minor loss of accuracy. In addition, we show that entropy reduction of weights and activation can be efficiently applied together, further improving bandwith reduction.

Finally, we analyze the method's rate–distortion tradeoff, achieving even stronger compression at the expense of a minor reduction in accuracy: for ResNet-18, we manage to achieve entropy inferior to one bit per value, at the expense of losing 2% of the top-1 accuracy.

2. Related Work

Recent studies (Yang et al., 2017a; Wang et al., 2019) have shown that almost 70% of the energy footprint of custom hardware is due to the data movement to and from the off-chip memory. Nonetheless, techniques for memory bandwidth reduction have not received significant attention in the literature. One way to improve memory performance is by fusing convolutional layers (Xiao et al., 2017; Xing et al., 2019), reducing the number of feature map transfers. This reduces both runtime and energy consumption of the hardware accelerator. Alternatively, it is possible to use on-chip cache instead of external memory. Morcel et al. (2019) demonstrated order of magnitude improvement in power consumption using this technique. Another important system parameter dominated by the memory bandwidth is



(a) (b)

Figure 2: Comparison with other methods: EPBC (Cavigelli et al. (2019)) and GF (Gudovskiy et al. (2018)) in **(a) ResNet-34** and **(b) MobilenetV2**. Different marker sizes refer to different activation bitwidths before compression. For GF, the compression rate was averaged only over compressed layers.

latency. Jouppi et al. (2017) and Wang et al. (2019) showed that the state-of-the-art DNN accelerators are memory-bound, implying that increasing computation throughput without reducing the memory bandwidth barely affects the total system latency.

Quantization reduces computation and memory requirements; 16-bit fixed point has become a *de facto* standard for fast inference. In most applications, weights and activations can be quantized down to 8 bits without noticeable loss of precision (Lee et al., 2018; Yang et al., 2019a). Further quantization to lower precision requires non-trivial techniques (Mishra et al., 2018; Zhang et al., 2018), which are currently capable of reaching around 3–4 bits per entry without compromising precision (Choi et al., 2018b,a; Dong et al., 2019; Jin et al., 2019).

A different way to reduce memory bandwidth is by compressing the intermediate activations prior to their transfer to memory with some computationally cheap encoding such as Huffman (Chandra, 2018; Chmiel et al., 2020) encoding or run-length (RLE) encoding (Cavigelli et al., 2019). A similar approach of storing only non-zero values was utilized by Lin and Lai (2018). Chmiel et al. (2020) used linear dimensionality reduction (PCA) to increase the effectiveness of Huffman coding, while Gudovskiy et al. (2018) proposed to use nonlinear dimensionality reduction techniques.

Lossless coding was previously utilized in a number of ways for DNN compression: Han et al. (2015) and Zhao et al. (2019) used Huffman coding to compress weights, while Wijayanto et al. (2019) used the more complicated DEFLATE (LZ77 + Huffman) algorithm for the same purpose. Aytekin et al. (2019) proposed to use compressibility loss, which induces sparsity and has been shown (empirically) to reduce entropy of the non-zero part of the activations.

3. Method

We consider a feed-forward DNN $\mathcal{F}$ composed of L layers. Each subsequent layer processes the output of the previous one: $x^i = \mathcal{F}_i\left(x^{i-1}\right)$, using the parameters $\mathbf{w}_i \in \mathbb{R}^{N_i \times N_{i-1}}$. We denote by $x^0 = x$ and $x^L = y$ the input and output of the network, respectively, and the number of elements of x^i as N_i. The parameters $\mathbf{w}$ of the network are learned by minimizing $\mathcal{L}(x, y; \mathbf{w}) + \lambda\mathcal{R}(\mathbf{w})$, with the former term $\mathcal{L}$ being the task loss, and the latter term $\mathcal{R}$ being a regularizer (e.g., $\|\mathbf{w}\|_2$) inducing some properties on the parameters $\mathbf{w}$.

3.1 Entropy Encoding and Rate Regularization

Entropy encoders are a family of lossless data encoders that compress each symbol independently. In this case, assuming i.i.d. distribution of the input, it has been shown that optimal code length is $-\log_b p$, where b is the size of the alphabet and p_i is the probability of the i$^{\text{th}}$ symbol (Shannon, 1948). Thus, for a discrete random variable X, we define an entropy $H(X) = -\mathbb{E}\log_2 X = -\sum_i p(x_i)\log_2 p(x_i)$, which is a lower bound on the amount of information required for lossless compression of X. The expected total space required to encode the message is $N \cdot H$, where N is the number of symbols. Since we encode the activations with the entropy encoder before writing them into memory, we would like to minimize the entropy of the activations to improve the compression rate.

One example of an entropy encoder is Huffman coding – a prefix coding that assigns shorter codes to the more probable symbols. The simplicity along with the high compression rate (Szpankowski, 2000), bounded by $H(X) \leq R \leq H(X) + 1$, renders it especially useful in performance-critical applications. The comparison between the compression achieved by Huffman coding and the lower bound, entropy, in case of neural networks is shown Figs. 3 and 4. For large values of bits per value, the difference between Huffman coding and entropy is negligible. Other entropy encoders, such as arithmetic coding or asymmetric numeral systems (Duda et al., 2015), can provide even better compression rates; for instance, for large enough inputs, arithmetic coding achieves optimal rates. These schemes, however, require more computational resources for encoding and are harder to implement.

3.2 Differentiable Entropy-Reducing Loss

Since the empirical entropy is a discrete function, it is not differentiable and thus cannot be directly minimized with gradient descent. Nevertheless, there exist a number of differentiable functions which either approximate entropy or have same minimizer. Thus, we optimize

$$\mathcal{L} = \mathcal{L}_{\text{p}} + \lambda\mathcal{L}_H, \tag{2}$$

where $\mathcal{L}_{\text{p}}$ is a target loss function and $\mathcal{L}_H$ is some regularization that minimizes the entropy.

3.2.1 Soft Entropy

First, we consider the differentiable entropy estimation suggested by Agustsson et al. (2017). We start from the definition of the entropy,

$$H(X) = -\sum p(x_i)\log(p(x_i)) \tag{3}$$

$$p(x_i) = \frac{\left|\{x|x = q_i\}\right|}{N}, \tag{4}$$

where $\mathbf{q}$ is a vector of quantized values. Let m be an index of the bin to which the current value is mapped, and $\mathbf{Q}$ a one-hot encoding of this index, i.e.,

$$q_m = \arg\min_{q_i \in \mathcal{Q}} |x - q_i| = \arg\max_{q_i \in \mathcal{Q}} (-|x - q_i|) \tag{5}$$

$$\mathbf{Q}_i = \delta_{im}, \tag{6}$$

where δ_{im} denotes Kroneker's delta. To make the latter expression differentiable, we can replace argmax with softmax:

$$\tilde{\mathbf{Q}}(x) = \text{softmax}(-|\mathbf{x} - \mathbf{q}|, T), \tag{7}$$

where T is the temperature, and $\tilde{\mathbf{Q}}(x) \to \mathbf{Q}(x)$ as $T \to 0$. Finally, the soft entropy $\hat{H}$ is defined as

$$\hat{H}(X) = -\sum \hat{p}(x_i)\log(\hat{p}(x_i)) \tag{8}$$

$$\hat{p}(x_i) = \frac{\sum_j \tilde{\mathbf{Q}}_i(x_j)}{N}. \tag{9}$$

To improve both memory requirements and time complexity of the training, we calculate the soft entropy only on part of the batch, reducing the amount of computation and the gradient tensor size.In particular, we try to take each k^{th} pixel of every feature map or a random subset of the activation, both leading to the same performance. We empirically confirm that this choice gives a reasonable approximation of the real entropy (Section 4.1.5).

3.2.2 Compressibility Loss

An alternative loss promoting entropy reduction was proposed by Aytekin et al. (2019) under the name of *compressibility loss* and based on earlier work by Hoyer (2004):

$$\mathcal{L}_c = \frac{\|\mathbf{x}\|_1}{\|\mathbf{x}\|_2}. \tag{10}$$

This loss has the advantage of computational simplicity, and has been shown both theoretically and practically to promote sparsity and low entropy in input vectors. While originally applied to the weights of the network, here we apply the same loss to the activations. As shown in Section 4.1, both the soft entropy and the compressibility loss lead to similar results.

We summarize the proposed method for reducing memory bandwidth as follows: at training time, we fine-tune (training from scratch is also possible but was not performed in our

Architecture	Compute (bits)	Memory (bits)	Compression ratio	Top-1 accuracy (%)
ResNet-18, CAT	32	32	1	69.70
	5	1.5	3.33	69.20
	4	1.51	2.65	68.08
ResNet-50, CAT	32	32	1	76.1
	5	1.60	3.125	74.90
	4	1.78	2.25	74.50
SSD512-SqueezeNet (Gudovskiy et al., 2018)	32	32	1	68.12
	8	2	4	64.39
	6	2	3	62.09
SSD512-VGG, CAT	32	32	1	80.72
	6	2.334	2.57	77.49
	4	1.562	2.56	77.43

Table 1: Results for ResNet-18, ResNet-50, and SSD512. We include the results by Gudovskiy et al. (2018) for the SSD512 model on the same task but with a different backbone, for which we obtain a better compression with a lower accuracy degradation. Compute denotes the activation bitwidth used for arithmetic operations. Memory denotes the average number of bits for memory transactions (after compression). Compression ratio denotes the reduction in representation size. Weight bitwidth is 8 except for the full-precision experiments. Additional results are provided in Tables A.1 and A.2 in the Appendix.

experiments) the pre-trained network $\mathcal{F}$ with the regularized loss (2), with $\mathcal{L}_H = \sum N_i \cdot \hat{H}(x^i)$ in the case of differentiable entropy and $\mathcal{L}_H = \sum \mathcal{L}_c(x^i)$ in the case of compressability loss, where the sum is running over network layers. At test time, we apply entropy coding on the activations (on a per layer basis) before writing them to memory, thus reducing the amount of memory transactions. In contrast to Chmiel et al. (2020), who avoided fine-tuning by using test-time transformation to reduce entropy, our method does not requires complex transformations at test time because it induces low entropy during training.

4. Experimental Results

We evaluate the proposed scheme on common CNN architectures for image classification (ResNet-18/34/50, MobileNetV2) on ImageNet (Russakovsky et al., 2015) , object detection (SSD512[1], Liu et al., 2016) on Pascal VOC (Everingham et al., 2010) as well as Transformers (BERT, Devlin et al., 2019) for sentimental analysis on CoLA (Warstadt et al., 2019) and for textual entailment on MNLI (Williams et al., 2018).

1. Our code is based on an implementation by Li (2018).

Data set	Architecture	Compute (bits)		Memory, embeddings (bits)		Memory, fully connected (bits)		(%)
		weights	activ.	weights	activ.	weights	activ.	
CoLa	Baseline	32	32	–	–	–	–	49.23
	No regularization	8	8	5.06	6.3	5.1	5.4	49.00
	CAT	8	8	2.64 ± 0.07	3.17 ± 0.10	2.47 ± 0.07	2.97 ± 0.08	49.10
MNLI	Baseline	32	32	–	–	–	–	83.91/84.1
	No regularization	8	8	5.4	6.7	5.33	6.3	83.88/84
	CAT	8	8	2.92 ± 0.08	3.4 ± 0.10	2.87 ± 0.07	3.37 ± 0.06	83.9/83.9

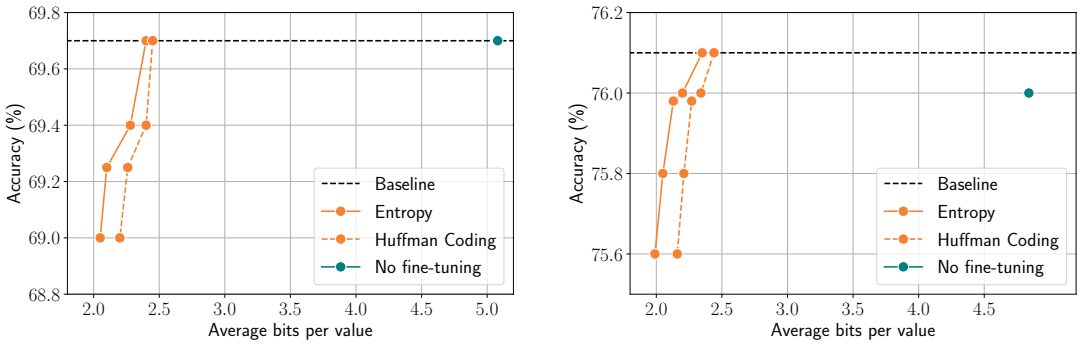
Table 2: Results of compression of both weights and activations for sentiment analysis task on CoLa data set and textual entailment on MNLI data set using BERT model. The primary performance metric for CoLa dataet is Matthews correlation coefficient (MCC) and matched/mismatched accuracy (MA/MI) for MNLI data set. The notation is the same as in Table 1, as compared to full precision baseline and Huffman coding without regularization. We provide standard deviation over different samples in form mean $\pm$ std.

To perform an evaluation in conditions close to real-life setups, we chose to quantize the weights to 8 bits and activations to the range of 4–8 bits. Adding a complicated quantization method to CAT would require more resources and might add additional noise to the measurements, without contributing much to the analysis of CAT. Thus, we chose a simple quantization method and did not attempt to achieve state-of-the-art quantization performance for the baseline. Compute bits refers to the bitwidth the activations or weights during computation, while memory bits refers to the number of bits used for storage, i.e., after applying Huffman coding.

The weights were initialized with a pre-trained model and quantized with uniform quantization using the shadow weights, i.e. applying updates to a full precision copy of quantized weights (Hubara et al., 2016; Rastegari et al., 2016). The activations were clipped with a learnable parameter and then uniformly quantized as suggested by Baskin et al. (2018). Similarly to previous works (Zhou et al., 2016; Rastegari et al., 2016), we used the straight-through estimator (Bengio et al., 2013) to approximate the gradients. We quantize all layers in the network, in contrast to the common practice of leaving the first and last layers in high precision (Zhou et al., 2016; Baskin et al., 2018).

For optimization, we used SGD with a learning rate of 10^{-4}, momentum 0.9, and weight decay 4×10^{-5} for up to 30 epochs (usually, 10 to 15 epochs were sufficient for convergence). Our initial choice of temperature was $T = 10$, which performed well. We tried to apply exponential scheduling to the temperature (Jang et al., 2017), but it did not have any noticeable effect on the results.

In Fig. 2 we compare our method with EPBC (Cavigelli et al. (2019)) and GF (Gudovskiy et al. (2018)). EPBC is based on a lossless compression method that maintains the full precision accuracy while reducing the bit rate to approximately 3.5 bits/value in both models. GF, on the other hand, provides strong compression at the expense of larger accuracy degradations. In addition, Gudovskiy et al. (2018) compressed only part of the layers. Unlike these two methods, CAT allows more flexible tradeoff between compression and accuracy. CAT shows better results in ResNet-34 and shows either better accuracy



(a) (b)

Figure 3: Tradeoff between rate and accuracy for **(a) ResNet-18** and **(b) ResNet50** in weight compression. The activations are quantized to 8 bits for fair comparison. "No fine-tuning" refers to pre-trained model with activations and weights quantized to 8 bits and the Huffman coding applied to the weights.

or compression for MobileNetV2. We also ran our method on additional architectures: ResNet-18, ResNet-50, and SSD512 with VGG backbone; the results are listed in Table 1. Even though we cannot directly compare detection results with Gudovskiy et al. (2018), the drop in accuracy is lower in our case. Additional experimental results are presented in the Appendix.

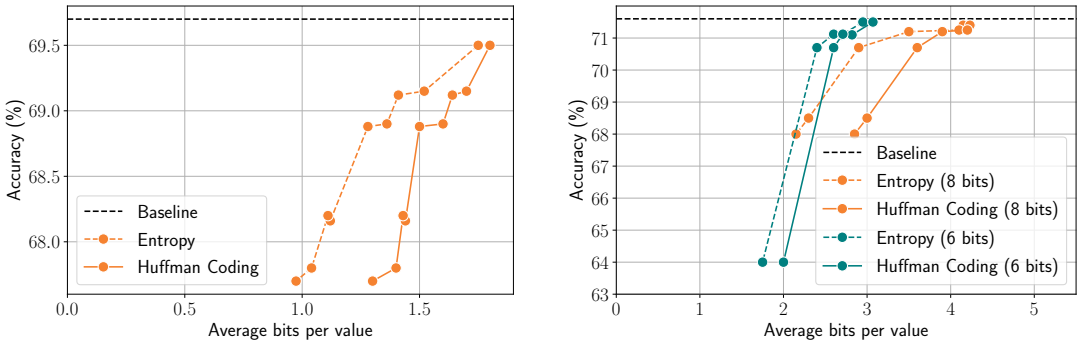
In Table 2 we evaluate our method on BERT. Since those models are more challenging for quantization, we used 8-bit quantization for both weights and activations. We used sentiment analysis task on CoLa data set and textual entailment task on MNLI data set for evaluation. The primary performance metric for CoLa dataet is Matthews correlation coefficient (MCC) and matched/mismatched accuracy (MA/MI) for MNLI data set. CAT was able to compress feature maps in CoLa data set to ~3-bit, while Huffman coding alone requires almost twice higher bandwidth. In both cases there is only minor performance degradation.

4.1 Ablation study

To better study the proposed approach and its properties, we perform multiple ablation studies.

4.1.1 Rate–Accuracy Tradeoff

The proposed algorithm tries to balance between the compression and the accuracy of the network by means of the parameter λ in Eq. (2). To evaluate this tradeoff, we trained ResNet-18 and MobileNetV2 with different values of λ in the range of $0 - 0.3$, with results shown in Fig. 4. Increasing the value of the regularization term results in weights that produce lower-entropy activations and thus allow a better compression rate, at the expense of accuracy degradation. We show the values of the theoretical entropy and the average



(a) (b)

Figure 4: Tradeoff between rate and accuracy for different values of λ (ranged between 0 and 0.3) in **(a) ResNet-18** and **(b) MobileNetV2** in activations compression. In ResNet-18, the activations are quantized to 5 bits; in MobileNet we show results for activation quantized to 6 and 8 bits.

Network	**Accuracy, % (mean ± std)**	**Memory, bits (mean ± std)**
ResNet-18	69.122 ± 0.016	1.5150 ± 0.0087
ResNet-34	73.025 ± 0.095	1.7875 ± 0.033

Table 3: Mean and standard deviation over five runs of ResNet-18 and ResNet-34 with 5 bit compute.

bitwidth of the Huffman-coded activations. The main advantage of Huffman coding is computational efficiency: even the naive implementation of the Huffman coding introduces only 4% overhead for inference time. For high bitwidth, Huffman coding is close (~3% overhead) to the theoretical entropy, while for lower entropy there is a larger difference (in particular, Huffman coding is bounded from below by 1 bit per value) – in this case, different lossless coding schemes such as arithmetic coding or asymmetric numeral systems (Duda et al., 2015) can provide better results.

4.1.2 Robustness

To check the robustness of our method, we performed several runs with the same hyperparameters and a different random seed. The results, reported in Table 3, suggest that the method is robust and stable under random initialization.

4.1.3 Soft Entropy vs. Compressibility Loss

Replacing the soft entropy with a different entropy-minimizing loss has a minor effect on the results (Table 4). In contrast, using ℓ_1 regularization, which promotes sparsity, shows

Compute, bits	Loss	Accuracy	Memory, bits
4	entropy	67.86%	1.43
4	comp.	67.84%	1.50
4	ℓ_1	65.3%	2.4
5	entropy	69.49%	1.79
5	comp.	69.36%	1.73
5	ℓ_1	67.9%	3.7

Table 4: Performance of soft entropy (8), compressibility loss (10) and ℓ_1 regularization on ResNet-18.

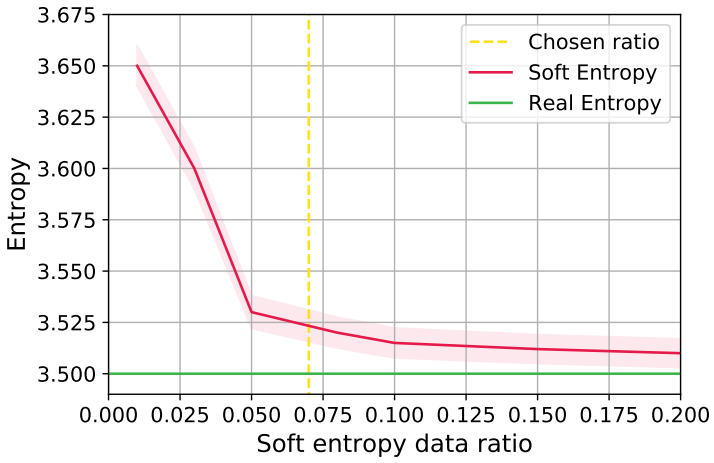


Figure 5: Soft entropy with different sample sizes compared to a real entropy of a single batch. The shaded region covers the standard deviation over three runs.

lower accuracy for higher bitwidth. This suggests that the desired effect is the result of an entropy reduction rather than a particular form of regularization promoting it.

4.1.4 Batch Size

We noticed that training ResNet-50 on a single GPU mandated the use of small batches, leading to performance degradation. Increasing the batch size from 16 to 64 without other changes increased accuracy by more than 0.5% with an entropy increase of less than 0.1 bits/value.

4.1.5 Sample Size in Soft Entropy Calculation

To check whether the number of values used to calculate soft entropy is enough, we ran a soft entropy evaluation on a single tensor and compared it to real values. Since the tensors

Architecture	Compute (bits)		Memory, original (bits)		Memory, CAT (bits)		Top-1 accuracy
	weights	activ.	weights	activ.	weights	activ.	(%)
ResNet-18	32	32	32	32	–	–	69.70
	8	6	5.1	3.5	2.2 ± 0.06	2.4 ± 0.07	68.90
	8	5	5.1	3	1.9 ± 0.03	2.24 ± 0.04	66.80
ResNet-50	32	32	32	32			76.15
	8	6	4.6	3.6	2.4 ± 0.08	2.6 ± 0.09	75.1
	8	5	4.6	2.95	2.3 ± 0.06	1.9 ± 0.04	74.5

Table 5: Results of compression of both weights and activations for ResNet-18 and ResNet-50. The notation is the same as in Table 1. We add the baseline ("Memory, original") that employs Huffman coding but no regularization to pre-trained model. We provide standard deviation over different samples in form mean $\pm$ std.

are large (hundreds of thousands of elements), even 5% of the values already provide a reasonable approximation of the real entropy, as shown in Fig. 5.

4.1.6 Weight Compression

We also show that the proposed method can be applied along with weight compression. First, we performed experiments with regularization applied only to weights. Fig. 3 shows the result for ResNet-18 and ResNet-50 and Table 2 show the result of BERT on CoLa and MNLI data set. Both activations and weights are quantized to 8 bits. The regularization provides a significant improvement, reducing the entropy of weights to ~2.5 bits without a loss in accuracy or MCC score in CoLa data set. A further increase in regularization allows us to achieve 2-bit entropy with less than 1% loss in accuracy.

Finally, we apply the entropy reduction to both weights and activations. The results are presented in Table 5. With less than 1% loss in accuracy, we get an almost twofold improvement over unregularized version in both weight and activation entropy.

5. Discussion

Quantization of activations reduces memory access costs that are responsible for a significant part of the energy footprint of DNN accelerators. Conservative quantization approaches, known as post-training quantization, take a model trained for full precision and directly quantize it to 8-bit precision. These methods are simple to use and allow for quantization with limited data. Unfortunately, post-training quantization below 8 bits usually incurs significant accuracy degradation. Quantization-aware training approaches involve some sort of training either from scratch (Hubara et al., 2016), or as a fine-tuning step from a pre-trained floating point model (Han et al., 2015). Training usually compensates significantly for a model's accuracy loss due to quantization.

In this work, we take a further step and propose a compression-aware training method to aggressively compress activations to as low as 2 average bit/value representations without

harming accuracy. Our method optimizes the average bit per value needed to represent activation values by minimizing the entropy. We demonstrate the applicability of our approach and its compatibility with other compression methods, such as quantization and weight entropy reduction, on classification tasks using the MobileNetV2 and various ResNet models, as well as an object detection task using the model SSD512, sentimental analysis and textual entailment tasks using BERT model. Also, we show that entropy reduction of weights and activations can be applied together, further improving bandwidth reduction. Because of the low overhead, the method provides universal improvement for any custom hardware, being especially useful for accelerators with efficient computations, where memory transfers are a significant part of the energy budget. We show that the effect is universal among loss functions and robust to random initialization.

Acknowledgments

This research was partially supported by the Technion Hiroshi Fujiwara Cyber Security Research Center and the Israel National Cyber Directorate.

Appendix A. Additional experimental results

We provide additional experimental results in Tables A.1 and A.2.

Architecture	Batch size	lr	λ	Compute (bits)	Memory (bits)	Top-1 accuracy (%)
ResNet-18	96	0.001	0	4	2.050	68.000
			0.05		1.540	67.950
			0.08		1.430	67.860
			0.05	5	1.790	69.490
			0.05		1.750	69.400
			0.1		1.410	69.120
			0.12		1.361	68.900
			0.15		1.280	68.914
			0.18		1.120	68.160
			0.2		1.110	68.300
			0.25		1.040	67.800
			0.3		0.974	67.700
			0	6	3.100	70.000
			0.05		1.930	69.710
			0.08		1.700	69.500
			0.05	7	2.280	69.660
			0	8	5.100	69.900
			0.05		2.460	69.820
			0.08		2.410	69.110
ResNet-34	96	0.001	0.05	8	2.750	73.200
			0.05	6	2.000	73.200
			0.05	5	1.790	73.100
ResNet-50	16	0.0001	0	4	2.500	73.700
	16		0.05		1.720	73.800
	64		0.05		1.78	74.5
	48		0.08		1.67	74.2
	16		0	5	2.950	75.500
	16		0.05		1.920	75.460
	16		0.08		1.700	75.200
	16		0.1		1.600	74.900

Table A.1: Experimental results for ResNet.

Architecture	Batch size	lr	λ	Compute (bits)	Memory (bits)	Top-1 accuracy (%)
MobileNetV2	64	0.0001	0	4	2.200	66.150
	64	0.001	0		2.800	66.200
	64	0.0001	0.05		2.100	66.900
	64	0.001	0.05		2.080	66.400
	64	0.001	0.08		1.830	66.200
	64	0.0001	0.08		1.980	66.450
	64	0.001	0	6	3.900	69.600
	64	0.0001	0		3.700	71.000
	96	0.0001	0.05		2.950	71.500
	32	0.0001	0.08		2.700	70.930
	64	0.0001	0.1		2.700	70.950
	32	0.001	0.15		1.750	64.000
	64	0.0001	0.15		2.600	71.200
	64	0.0001	0.2		2.450	70.700
	64	0.0001	0	8	4.750	71.300
	64	0.0001	0.05		4.150	71.400
	64	0.001	0.08		2.600	70.000
	64	0.0001	0.08		4.120	71.600
	64	0.001	0.1		2.400	69.600
	64	0.0001	0.1		4.100	71.250
	64	0.001	0.15		2.300	68.500
	64	0.001	0.2		2.150	68.000
	64	0.0001	0.2		3.500	71.200
	32	0.0001	0.3		2.900	70.700

Table A.2: Experimental results for MobileNet.

References

Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc Van Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 1141–1151. Curran Associates, Inc., 2017.

Caglar Aytekin, Francesco Cricri, and Emre Aksu. Compressibility loss for neural network weights. *arXiv preprint arXiv:1905.01044*, 2019.

Chaim Baskin, Natan Liss, Yoav Chai, Evgenii Zheltonozhskii, Eli Schwartz, Raja Giryes, Avi Mendelson, and Alexander M. Bronstein. NICE: noise injection and clamping estimation for neural network quantization. *arXiv preprint arXiv:1810.00162*, 2018.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Lukas Cavigelli, Georg Rutishauser, and Luca Benini. EBPC: Extended bit-plane compression for deep neural network inference and training accelerators. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(4):723–734, 2019. doi: 10.1109/JETCAS.2019.2950093.

Mahesh Chandra. Data bandwidth reduction in deep neural network SoCs using history buffer and Huffman coding. In *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*, pages 1–3. IEEE, 2018.

Brian Chmiel, Chaim Baskin, Evgenii Zheltonozhskii, Ron Banner, Yevgeny Yermolin, Alex Karbachevsky, Alex M. Bronstein, and Avi Mendelson. Feature map transform coding for energy-efficient cnn inference. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9, 2020. doi: 10.1109/IJCNN48605.2020.9206968.

Jungwook Choi, Pierce I-Jen Chuang, Zhuo Wang, Swagath Venkataramani, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Bridging the accuracy gap for 2-bit quantized neural networks (QNN). *arXiv preprint arXiv:1807.06964*, 2018a.

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018b.

Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840*, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423.

Zhen Dong, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. HAWQ: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Jarek Duda, Khalid Tahboub, Neeraj J. Gadgil, and Edward J. Delp. The use of asymmetric numeral systems as an accurate replacement for huffman coding. In *2015 Picture Coding Symposium (PCS)*, pages 65–69, May 2015. doi: 10.1109/PCS.2015.7170048.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Denis Gudovskiy, Alec Hodgkinson, and Luca Rigazio. DNN feature map compression using learned representation over GF(2). In *The European Conference on Computer Vision (ECCV) Workshops*, September 2018.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

Patrik O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov):1457–1469, 2004.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *arXiv preprint arXiv:1609.07061*, 2016.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-softmax. In *International Conference on Learning Representations*, 2017.

Qing Jin, Linjie Yang, and Zhenyu Liao. Towards efficient training for neural network quantization. *arXiv preprint arXiv:1912.10207*, 2019.

Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-datacenter performance analysis of a tensor processing unit. *ACM SIGARCH Computer Architecture News*, 45(2):1–12, June 2017. ISSN 0163-5964. doi: 10.1145/3140659.3080246.

Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 598–605. Morgan-Kaufmann, 1990.

Jun Haeng Lee, Sangwon Ha, Saerom Choi, Won-Jo Lee, and Seungwon Lee. Quantization for rapid deployment of deep neural networks. *arXiv preprint arXiv:1810.05488*, 2018.

Congcong Li. High quality, fast, modular reference implementation of SSD in PyTorch. `https://github.com/lufficc/SSD`, 2018.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2017.

Chien-Yu Lin and Bo-Cheng Lai. Supporting compressed-sparse activations and weights on SIMD-like accelerator for sparse convolutional neural networks. In *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 105–110, Jan 2018. doi: 10.1109/ASPDAC.2018.8297290.

Chunlei Liu, Wenrui Ding, Xin Xia, Yuan Hu, Baochang Zhang, Jianzhuang Liu, Bohan Zhuang, and Guodong Guo. Rectified binary convolutional networks for enhancing the performance of 1-bit DCNNs. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 854–860. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/120.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *The European Conference on Computer Vision (ECCV)*, pages 21–37, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46448-0.

Asit Mishra, Eriko Nurvitadhi, Jeffrey J. Cook, and Debbie Marr. WRPN: Wide reduced-precision networks. In *International Conference on Learning Representations*, 2018.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Raghid Morcel, Hazem Hajj, Mazen A. R. Saghir, Haitham Akkary, Hassan Artail, Rahul Khanna, and Anil Keshavamurthy. FeatherNet: An accelerated convolutional neural network design for resource-constrained FPGAs. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 12(2):6:1–6:27, March 2019. ISSN 1936-7406. doi: 10.1145/3306202.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: Imagenet classification using binary convolutional neural networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *The European Conference on Computer Vision (ECCV)*, pages 525–542, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46493-0.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, July 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x.

Kevin Siu, Dylan Malone Stuart, Mostafa Mahmoud, and Andreas Moshovos. Memory requirements for convolutional neural network hardware accelerators. In *2018 IEEE International Symposium on Workload Characterization (IISWC)*, pages 111–121. IEEE, 2018.

Wojciech Szpankowski. Asymptotic average redundancy of Huffman (and Shannon-Fano) block codes. In *2000 IEEE International Symposium on Information Theory (Cat. No.00CH37060)*, pages 370–, June 2000. doi: 10.1109/ISIT.2000.866668.

Erwei Wang, James J. Davis, Peter Y. K. Cheung, and George A. Constantinides. LUTNet: Rethinking inference in FPGA soft logic. In *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 26–34, 2019. doi: 10.1109/FCCM.2019.00014.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural Network Acceptability Judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 09 2019. ISSN 2307-387X. doi: 10.1162/tacl_a_00290.

Arie Wahyu Wijayanto, Jun Jin Choong, Kaushalya Madhawa, and Tsuyoshi Murata. Towards robust compressed convolutional neural networks. In *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 1–8. IEEE, 2019.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics, 2018.

Qingcheng Xiao, Yun Liang, Liqiang Lu, Shengen Yan, and Yu-Wing Tai. Exploring heterogeneous algorithms for accelerating deep convolutional neural networks on FPGAs. In *Proceedings of the 54th Annual Design Automation Conference 2017*, DAC '17, pages 62:1–62:6, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4927-7. doi: 10.1145/3061639.3062244.

Yu Xing, Shuang Liang, Lingzhi Sui, Xijie Jia, Jiantao Qiu, Xin Liu, Yushun Wang, Yi Shan, and Yu Wang. Dnnvm: End-to-end compiler leveraging heterogeneous optimizations on fpga-based cnn accelerators. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1–1, 2019. doi: 10.1109/TCAD.2019.2930577.

Guandao Yang, Tianyi Zhang, Polina Kirichenko, Junwen Bai, Andrew Gordon Wilson, and Chris De Sa. SWALP : Stochastic weight averaging in low precision training. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7015–7024. PMLR, 09–15 Jun 2019a.

Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua. Quantization networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019b.

Tien-Ju Yang, Yu-Hsin Chen, Joel Emer, and Vivienne Sze. A method to estimate the energy consumption of deep neural networks. In *2017 51st Asilomar Conference on Signals, Systems, and Computers*, pages 1916–1920. IEEE, 2017a.

Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017b.

Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. LQ-Nets: Learned quantization for highly accurate and compact deep neural networks. In *The European Conference on Computer Vision (ECCV)*, September 2018.

Yiren Zhao, Xitong Gao, Daniel Bates, Robert Mullins, and Cheng-Zhong Xu. Efficient and effective quantization for sparse DNNs. *arXiv preprint arXiv:1903.03046*, 2019.

Shuchang Zhou, Zekun Ni, Xinyu Zhou, He Wen, Yuxin Wu, and Yuheng Zou. DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.