Trainable pruned ternary quantization for medical signal classification models

Yamil Vindas, Blaise Kévin Guépié, Marilys Almar, Emmanuel Roux, Philippe Delachartre

Trainable pruned ternary quantization for medical signal classification models

Yamil Vindas[a,*], Blaise Kévin Guépié[b], Marilys Almar[c], Emmanuel Roux[a], Philippe Delachartre[a]

[a] *Univ Lyon, INSA-Lyon, Université Claude Bernard Lyon 1, UJM-Saint Etienne, CNRS, Inserm, CREATIS UMR 5220, U1294, F-69100, LYON, France*
[b] *Université de Technologie de Troyes / Laboratoire Informatique et Société Numérique, 10004 Troyes, France*
[c] *Atys Medical, 17 Parc Arbora, 69510 Soucieu-en-Jarrest, France*

Abstract

The field of deep learning is renowned for its resource-intensive nature, hence improving its environmental impact is crucial. In this paper, we propose a novel model compression method to mitigate the energy demands of deep learning for a greener, and more sustainable AI landscape. Our approach relies on an asymmetric weakly-differentiable pruning function that leverages weight statistics to directly incorporate adaptable pruning into the quantization mechanism. This enables us to achieve higher compression rates globally while simultaneously reducing energy consumption and minimizing classification performance degradation. The efficacy of our approach was evaluated using three distinct models on three distinct datasets: cerebral emboli (HITS), epileptic seizure recognition (ESR), and MNIST. Our method demonstrated a superior balance between compression, energy consumption, and classification performance compared to other state-of-the-art extreme quantization methods, across all models and datasets. In fact,

*Corresponding author
Email address: yamil.vindas@insa-lyon.fr (Yamil Vindas)

on the HITS dataset with a two-dimensional convolutional neural network, we achieved strong gains of 50.6%, 54.9%, 52.1% in compression rates (of the global model and the quantized layers only, respectively) and energy consumption, respectively, while improving the Matthews correlation coefficient by 2.5% compared to other approaches. The code is available at: https://github.com/yamilvindas/pTTQ

Keywords: Model compression, Model quantization, Model pruning, Signal classification, Transcranial Doppler

2010 MSC: 00-01, 99-00

---

1. Introduction

Deep learning models, such as convolutional neural networks (CNNs) and transformers, have demonstrated considerable success in a multitude of applications, including computer vision [1, 2], natural language processing [3], and signal processing [4, 5, 6]. However, these models often possess a considerable number of parameters, elevated energy consumption, and prolonged inference times [7], rendering them unsuitable for environmental constraints.

Recent works have sought to reduce the memory and resource requirements of these models by employing different model compression techniques [8]. Techniques such as quantization [9], pruning [10], and neural architecture search (NAS) [11] have demonstrated impressive results by reducing the number of parameters of deep learning models without a considerable compromise of the model performance. Quantization is a technique that reduces the precision of model parameters (weights and/or activations) in order to decrease the memory footprint, energy consumption and inference

time. Pruning involves removing (by setting to zero) some of the model parameters in order to eliminate redundancy, and it even acts as a regularizer [10]. NAS is a method that works towards finding efficient architectures that are suited for a desired task and that respect certain criteria. However, these techniques are often employed separately or sequentially, which can increase the development time of the models and may result in suboptimal performance in terms of classification, compression, and energy, due to the decorrelation of the tasks. Furthermore, NAS can facilitate the creation of highly efficient networks; however, it is often a time-consuming process that requires significant computational resources to achieve optimal outcomes [12]. A solution consists of combining pruning and extreme quantization[1], fostering the creation of sparser models without substantial performance drop. The sparsity in these models can be leveraged to reduce energy consumption, inference times, and memory requirements through efficient sparse encoding.

In this paper, we focus on extreme (ternary) quantization and pruning, and we propose a method to perform both simultaneously. On one side, trained ternary quantization (TTQ) [13] sets the weight to either zero, a negative value, or a positive one, using symmetric thresholds. On the other side, a pruning mechanism is integrated within the ternarization process to prune the parameters zeroed by TTQ. And we expand the search space of quantized models by employing asymmetric thresholds. This task is nontrivial, as pruning needs to be tailored to both the layers of the model and the dataset to prevent a significant decrease in the task-specific performance.

[1]Quantization methods that prioritize low bit-width quantized weights, often fewer than 4 bits.

To do this, we introduce a weakly-differentiable threshold function based on [14], but adding two asymmetric thresholds that can be learned throughout the training process. Finally, our method can be easily integrated into more general compression methods such as Deep Compression, facilitating seamless combinations with other model compression techniques. Our main contributions can be summarized as follows:

- Novel asymmetric weakly-differentiable threshold function based on the statistics of the weights with learnable parameters for pruning.
- New concurrent pruning and extreme quantization method, based on ternary asymmetric weights.
- Improving compression rates without significantly compromising the model performance.
- Extensive experimental validation highlighting the effectiveness of our method, with state-of-the-art results in terms of compression/energy/classification performance trade-off.

The rest of the paper is structured as follows. In section 2 we introduce the works related to our method. In section 3 we present in detail the different components of our method. In section 4 we describe the experimental setup that we use to validate our approach and present the results and their discussion. Finally, in section 6 we conclude and present the guidelines of our future work.

2. Related work

2.1. Model quantization

Model quantization has been an important research topic in the deep learning community in recent years. This has been driven by the need to reduce the energy resources consumed by deep learning, especially in edge applications where efficiency is critical.

Classic approaches can be based on matrix factorization (e.g., singular value decomposition) and vector quantization [15] or weight sharing and/or clustering [16, 17, 18, 19]. Weight sharing can be achieved using different approaches, such as soft weight sharing based on Gaussian mixture models [17], k-means clustering of the weights with or without identification of the important convolutional filters [16, 18], or cross-layer parameter sharing [19].

Quantization can also be performed at different granularities: channel-wise for CNNs [9], using subgroups of weights in the different layers [20], or at different levels in depth and width of the weight tensors based on the quantization error [21].

Furthermore, mixing different types of precision can be a solution to the performance drop attributable to quantization [9, 22, 23, 24]. These approaches can be expensive to apply, as the number of possible precision combinations increases exponentially with the number of layers. Some works have proposed solutions based on metrics extracted from the Hessian matrix [22]. The general idea is that in a flat zone of the loss landscape, one can carry out more aggressive pruning than in irregular areas. Some works have demonstrated that, quantization can be effectively performed if a pre-trained model is used, since near-optimal low-bit solutions exist close to the full-precision (FP) solutions [25]. Following the same idea, other works propose

adaptive quantization, where the bit-width used to encode the weights can be adapted to the hardware or device conditions [26].

Knowledge distillation [27] is also a popular method for model compression, and can be applied to quantization [28, 29, 30]. Some works use it directly to compensate for the performance drop due to quantization, without reducing the size of the network or modifying its architecture [29]. Other approaches use knowledge distillation to reduce the size of the model while performing quantization [28]. Bai et al. [30] propose binary quantization for BERT[2] models using quantization-aware training (QAT) and post-training quantization. First, they start by training with QAT a half-sized ternary BERT. Then, they initialize the weights of a full-size binary BERT using a ternary weight-splitting operator. Finally, they apply knowledge distillation to recover from the performance drop attributable to quantization.

Moreover, most QAT methods are difficult to optimize due to their non-differentiable nature. Consequently, the straight-through estimator (STE) [31] is often employed to approximate gradients. Zhang et al. [32] propose using learnable quantizers to minimize quantization errors. Bhalgat et al. [33] introduce LSQ+, an asymmetric quantization method suitable for skewed distributions with negative values, along with a trainable scale and offset. Additionally, some researchers attempt to reformulate the quantization problem as a differentiable task using non-linear functions [34].

Another important aspect of quantization is whether quantization is simulated or not. Indeed, in simulated quantization, all the computation is done in full-precision (FP), and thus the gain in inference time due to quantization can be negligible. Most quantization works use simulated quantization,

[2]Bidirectional Encoder Representations from Transformers.

which does not allow one to easily take advantage of arithmetic operations to improve latency and reduce energy consumption [9]. Some recent works propose the use of integer-only quantization, taking advantage of arithmetic operations [35, 36, 37] to reduce inference time and energy consumption, while reducing memory requirements.

Finally, even though numerous works focus on quantizing the model parameters using 4 bits or more, extreme quantization can be of interest as it can further reduce the memory requirements of the models. Different approaches have been studied to achieve extreme quantization, such as binary networks [38, 39, 40, 30], ternary models [13, 29, 21, 41], or mixed-precision quantization [42]. Most of these binary or ternary networks work with weight tensors having binary (0 or 1) or ternary (-1, 0, or 1) values multiplied by FP coefficients. The main difference in the various methods is the heuristics used to binarize or quantize the weights, which often depends on manually selected thresholds or thresholds based on the statistics of the weight tensors. Moreover, extreme quantization often creates significant noise during training because of STE approximation of the gradients, which can lead to difficult optimization. Fan et al. [43] proposed to tackle this issue by selecting a random subset of weights to quantize instead of quantizing all the weights, making it possible to keep some gradients without error, thus improving the gradient flow.

While several model quantization methods have successfully achieved significant compression rates, only a limited number of approaches capitalize on pruning to further enhance compression.

2.2. Model pruning

An alternative to parameter quantization is parameter pruning, where a subset of the parameters is set to zero based on different criteria. It has been shown that pruning can be a very effective technique for model compression without requiring a strong modification of the main architecture or training strategy [10]. Moreover, it has also been shown that, due to over-parameterization, there is often redundancy in the parameters of deep learning models; thus pruning can act as a regularizer, improving the generalization capabilities of the models [10]. This has been empirically shown and studied for different types of models, such as CNNs [44] a transformer models [45].

Different approaches can be used to prune the weights of a model. Classic approaches are based on removing the smaller weights (according to a certain metric, e.g., the L1 norm) by setting them to zero [16]. More sophisticated models carry out gradual pruning during training until the desired sparsity rate is reached [46], or use determinantal point processes to select a subset of neurons in a layer and fuse them [47], or carefully select the layers/filters to prune, based on their importance [48] (which is computed using the statistics of the next layer's parameters).

Moreover, due to the threshold operator, pruning operations are often non-differentiable, which makes them difficult to automatically tune during training. Some approaches try to solve this by using different methods to optimize pruning, such as reinforcement learning (RL) [49] or genetic algorithms [50]. Manessi et al. [14] proposed a differentiable threshold function with learnable parameters, making it possible to automatically prune the models while increasing the compression rates.

Finally, quantization and pruning can be used together, as they are com-

plementary. Different works have shown that combining both strategies can be beneficial [16, 51, 17, 52]. This can be done by the sequential application of both techniques [16, 51], using Bayesian optimization techniques [52] or soft-weight sharing [17].

However, to the best of our knowledge, we propose the first incorporation of a learnable pruning mechanism into an extreme quantization process (ternarization), applied to several model architectures extensively evaluated on three different datasets. This could facilitate a more efficient optimization of both quantization and pruning, carried out concurrently, without significant performance degradation.

3. Methods

In this section we specify the different components of our method: the weakly-differentiable threshold function used for pruning, the concurrent ternarization and pruning process, and the strategy employed for selecting the layers to undergo quantization. An overview of our method can be found in Figure 1.

3.1. weakly-differentiable threshold function

We propose to use a weakly-diffedrentiable threshold function to achieve simultaneous pruning of weights and ternary quantization. To this end, we propose to use the following function p, inspired from [14]:

$$p(w; t_{min}, t_{max}, \alpha) = ReLU(w - \Delta_w(t_{max})) + \Delta_w(t_{max}) \times S(\alpha \times (w - \Delta_w(t_{max}))) - ReLU(-w - \Delta_w(t_{min})) - \Delta_w(t_{min}) \times S(\alpha \times (-w - \Delta_w(t_{min}))) \quad (1)$$

where $S : x \rightarrow \frac{1}{1+e^{-x}}$ is the sigmoid function, $t_{min}, t_{max} \geq 0$ are two hyperparameters controlling the minimum and maximum pruning thresholds

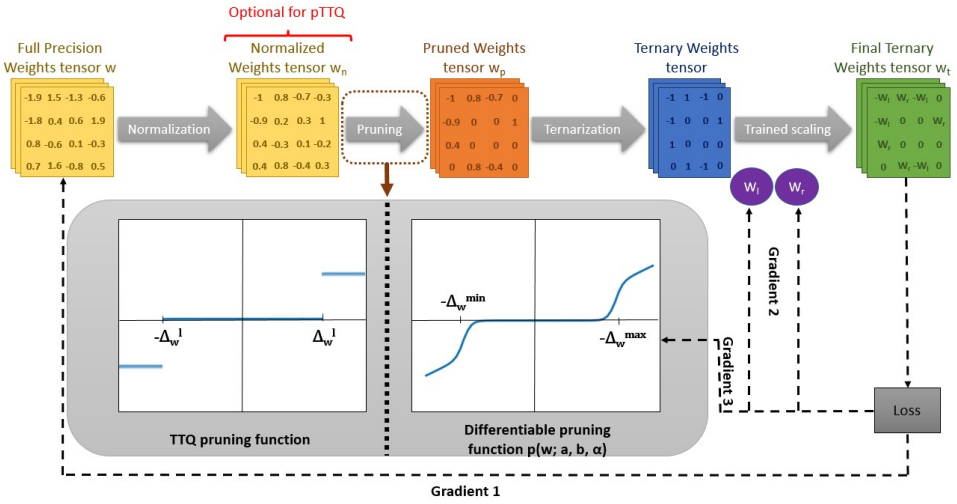

Figure 1: Training loop overview of the TTQ pruning function (on the left) compared to our proposed pruned TTQ (pTTQ) method (on the right). Unlike TTQ, the normalization step (first arrow) is not mandatory, and the model pruning is done direcly by a weakly-differentiable pruning function with learnable threshold parameters Δ_w^{min} and Δ_w^{max}

(denoted as Δ_w^{min} and Δ_w^{max} in Figure 1), and $\Delta_w : \mathbb{R} \to \mathbb{R}$ is a function of w (the weight tensor to be quantized) defined as $\Delta_w : t \to \mu_w + t \times \sigma_w$ where μ_w and σ_w are the mean and standard deviation of w. For the sake of clarity, we denote $\Delta_w^{min} = \Delta_w(t_{min})$ and $\Delta_w^{max} = \Delta_w(t_{max})$. This function is weakly-differentiable, and the weak derivatives of p with respect to t_{min} and t_{max} are the following:

$$\frac{\partial p}{\partial t_{min}}(w; t_{min}, t_{max}, \alpha) = \sigma_w \times H(-w - \Delta_w^{min}) - \sigma_w \times S(\alpha \times (-w - \Delta_w^{min})) + \sigma_w \times \alpha \times \Delta_w^{min} \times S(\alpha \times (-w - \Delta_w^{min})) \times (1 - S(-w - \Delta_w^{min})) \quad (2)$$

$$\frac{\partial p}{\partial t_{max}}(w; t_{min}, t_{max}, \alpha) = -\sigma_w \times H(w - \Delta_w^{max}) + \sigma_w \times S(\alpha \times (w - \Delta_w^{max})) \\ - \sigma_w \times \alpha \times \Delta_w^{max} \times S(\alpha \times (w - \Delta_w^{max})) \times (1 - S(w - \Delta_w^{max})) \quad (3)$$

where H is the Heaviside step function defined by $H(x) = 1$ if $x \geq 0$, $H(x) = 0$ otherwise.

As noted in [14], equations 2 and 3 are different from zero almost everywhere, allowing good gradient flow using gradient descent. The differences with respect to [14] lie in two key aspects: (1) our proposal incorporates two asymmetric thresholds as opposed to a single threshold, and (2) our thresholds depend on the statistics (mean and standard deviation) of the input tensor.

3.2. Pruned trained ternary quantization (pTTQ)

Unlike trained ternary quantization (TTQ) [13], our approach involves directly applying pruning during ternarization using the p function from Section 3.1. In contrast to the TTQ pruning method, which relies on fixed thresholds derived from potential outliers in the weights' distribution (maximum absolute value of the weights), our approach utilizes trainable pruning thresholds determined by the mean and standard deviation of the weights' distribution. This approach is better suited for asymmetric weights distributions, which is typically the case for a layer's weights, and it enables our method to adapt to new data distributions and models more effectively.

Let $w \in \mathbb{R}$ denote a FP weight, and $w_t \in \mathbb{R}$ its ternarization. We compute w_t from w as follows:

$$w_t = (\frac{sgn(p(w, a, b, \alpha)) - 1}{2} \times W_l + \frac{1 + sgn(p(w, a, b, \alpha))}{2} \times W_r) \times sgn(p(w, a, b, \alpha)) \quad (4)$$

where $W_l, W_r > 0$ are learnable scaling parameters, and sgn is the sign function. The (scaled) gradients of the loss function $\mathcal{L}$ with respect to w are computed in the same way as in TTQ, using the straight-through estimator:

$$\frac{\partial \mathcal{L}}{\partial w} = [(\frac{sgn(p(w, t_{min}, t_{max}, \alpha)) - 1}{2} \times W_l + \frac{1 + sgn(p(w, t_{min}, t_{max}, \alpha))}{2} \times W_r) \times sgn(p(w, t_{min}, t_{max}, \alpha)) + (1 - |sgn(p(w, t_{min}, t_{max}, \alpha))|)] \times \frac{\partial L}{\partial w_t} \quad (5)$$

Furthermore, unlike to TTQ, we do not necessarily need to normalize the FP weights before quantization. In fact, this is not necessary because our approach has learnable parameters (t_{min}, t_{max}, and α) that can adapt to the weights of a layer during training. These parameters can be global (i.e., the same for all layers in the model that are to be quantized), or local (i.e., one set of parameters per layer to be quantized). We refer to our method as "pTTQ" for pruned Trained Ternary Quantization.

3.3. Quantization layer selection

To select the layers to be quantized, we used the Hessian-based metric introduced in [23]. Therefore, in order to respect the assumptions of this metric, we used FP pre-trained models as initialization for the different quantization approaches.

The rationale behind this approach is that the Hessian metric provides information about the curvature of the loss landscape: small values indicate a flat loss landscape, while high values indicate a more curved landscape. Thus, extreme quantization of layers with a highly curved loss landscape (high values of the metric) is prone to higher performance degradation because the model can easily escape the local minima reached. On the contrary, extreme quantization of flat zones of the loss landscape (small values of the

metric) is more robust to the approach as it is more difficult to escape the local minima reached.

3.4. Model compression evaluation metrics

We introduce the notations and metrics that we used to compare different methods from three distinct perspectives: sparsity, compression, and energy consumption. We denote by $\mathcal{M}_{FP}$ the FP model composed of L layers, and $\mathcal{M}_Q$ a quantized model obtained from the FP model. We also assume that we have three functions: $nbits$, nqw, and $nzqw$. The first one, $nbits$, counts the number of bits necessary to store the (non-zero) weights of a model. The second metric, nqw, counts the number of weights within the selected quantization candidates of the model. Meanwhile, the third metric, $nzqw$, counts the number of weights that are equal to zero within the selected quantization candidates of the model. The main difference between nqw and $nzqw$ is that the former takes into account all the weights, while the latter only focuses on the zero weights.

Sparsity. Sparsity is evaluated based only on the weights selected for quantization. Thus, we denote the sparsity rate of the quantized weights as $SRQW$, which is defined as follows:

$$SRQW(\mathcal{M}_{FP}, \mathcal{M}_Q) = \frac{nzqw(\mathcal{M}_Q)}{nqw(\mathcal{M}_{FP})}$$

Higher values of $SRQW$ indicate higher sparsity of the quantized model with respect to the FP model.

Compression. We measure the compression performance of the different methods based on the global (whole model) and local (only the weights

selected for quantization) compression rates. Therefore, we denote the compression rate as CR, which is defined as follows:

$$CR(\mathcal{M}_{FP}, \mathcal{M}_Q) = \frac{nbits(\mathcal{M}_Q)}{nbits(\mathcal{M}_{FP})}$$

Moreover, to facilitate the comparison between methods, we also work with the compression rate gain CR_G:

$$CR_G(\mathcal{M}_{FP}, \mathcal{M}_Q) = 1 - CR(\mathcal{M}_{FP}, \mathcal{M}_Q)$$

Finally, we distinguish the compression rate gain of the whole model, CR_G^T, from that of the layers selected for quantization, CR_G^Q.

Energy consumption. Accurately assessing model energy consumption is challenging because manual measurements depend on hardware and environmental factors, and because custom hardware can be designed to optimize sparse and quantized tensor operations, depending on the model and application requirements.

Therefore, we introduce our own energy consumption metric using the orders of magnitude of 32 floating point operations given in [53], and those of random access memory (RAM) data transfers given in [54]. Our aim is to provide orders of magnitude of the energy consumption (in Joules) of the different models in order to fairly compare the ternary quantization methods from both the sparsity and quantization perspectives. To do this, we make three assumptions : (1) only the non-zero parameters are counted for the multiplications and additions; (2) multiplications and additions have the same energy consumption[3], that of a 32 floating-point multiplication, i.e.,

[3]We take the worst case, as separating multiplications from additions is not straightforward.

3.7 pJ according to [53]; and (3) only the non-zero weights are transferred to RAM, by blocks of 32 bits (4 bytes). Hence, the energy consumption metrics, EC_{MA}, EC_{DT}, and EC_T, corresponding to the mult-adds, data transfer, and total consumption, respectively, are defined as follows:

$$EC_{MA}(\mathcal{M}) = N_{MA} \times 3.7 \times 10^{-12} J \tag{6}$$

$$EC_{DT}(\mathcal{M}) = 10^{-9} \times \sum_{i=1}^{L} (\lceil \frac{nnzw(\mathcal{M}_i) \times B_i}{32} \rceil + N_{SF}^i) J \tag{7}$$

$$EC_T(\mathcal{M}) = EC_{MA}(\mathcal{M}) + EC_{DT}(\mathcal{M}) \tag{8}$$

where $\mathcal{M}$ is the evaluated model, N_{MA} is the number of multiplications and additions necessary to obtain the output of the model, $nnzw$ is a function counting the number of non-zero weights in a given tensor, for all $i \in [1, L]$, $\mathcal{M}_i$ corresponds to layer i of $\mathcal{M}$, B_i is the number of bits used to encode one weight of that layer (32 for an FP model, 2 for a ternary one), and N_{SF}^i is the number of scaling factors used for that layer (0 for FP models, and 2 for ternary models[4]). Finally, we can define the energy consumption gain, EC_G^T, of a sparse quantized model $\mathcal{M}_Q$ with respect to its FP counterpart $\mathcal{M}_{FP}$ as follows:

$$EC_G^T(\mathcal{M}_Q) = \frac{|EC_T(\mathcal{M}_{FP}) - EC_T(\mathcal{M}_Q)|}{EC_T(\mathcal{M}_{FP})} \tag{9}$$

4. Experiments

4.1. Experimental setup

4.1.1. Datasets

TCD HITS dataset. Our main interest is the classification of transcranial Doppler (TCD) high-intensity transient signals (HITS) between artifacts

[4]FP models do not require scaling factors as no ternarization is performed.

(Art.), gaseous emboli (GE), and solid emboli (SE). This detection of solid emboli is an important task as it can help clinicians prevent ischaemic stroke because emboli can block cerebral arteries. To train the different models, we used a private TCD HITS dataset consisting of 1 545 HITS (569 SE, 569 GE, and 403 Art.), each with an associated audible TCD signal and its corresponding time-frequency representation. For more details on this dataset and the different pre-processing steps, we refer the reader to [55].

Epileptic seizure recognition (ESR) dataset. We propose to evaluate our method on another medical signal processing task, namely, epileptic seizure recognition (ESR), using the pre-processed ESR dataset of the UCI repository [56]. The dataset consists of 11 500 electroencephalogram (EEG) signals, equally distributed into five classes: (1) seizure activity and (2)–(5) no seizure activity. As in most works, we focus on binary classification where the first class consists EEG signals with seizure activity and the second class corresponds to the rest of the samples. For more details about this dataset and the different pre-processing steps, we refer the reader to [56, 57].

MNIST dataset. We also propose to use a subset of the MNIST dataset [58] to evaluate our method. To reduce the computational requirements, we propose to reduce the original MNIST training set from 60 000 images to 6 000 by keeping only 10% of the samples. Nevertheless, the test set remains untouched, with 6 000 test samples.

4.1.2. Optimization strategy

All the models (FP and quantized ones) were optimized using the Adamax optimizer, with different learning rates and weight decay values. As indicated in [13], the gradients are computed using the quantized weights, and

the FP weights are updated using these gradients.

Furthermore, for pTTQ, we propose to use different learning rates for the various hyperparameters: one for the weights of the model and one for the learned thresholds t_{min} and t_{max}. These parameters are also optimized using Adamax, with an additional cosine annealing learning rate scheduler using a maximum of 10 iterations.

4.1.3. Evaluation metrics

To measure the performance of the different FP and quantized models, we used several metrics. For the model compression metrics, we used those introduced in 3.4. For the classification performance, we used the Matthews Correlation Coefficient (MCC) and the MCC drop with respect to the FP model $\Delta MCC = MCC(\mathcal{M}_Q) - MCC(\mathcal{M}_{FP})$.

Finally, for statistical purposes, all the experiments were repeated 10 times. The reported metrics correspond to the mean and standard deviations of those repetitions.

4.1.4. Implementation details

All the codes were implemented using Pytorch and Scikit-Learn, and a batch size of 32 was used for all experiments. The different experiments were run on two high performance computing clusters: one with 25 heterogeneous machines (each machine with between 16 Gb and 128 Gb of RAM, CPUs with 8–32 cores, and different types of Nvidia Quadro RTX and Tesla GPUs), and another with NVIDIA Tesla V100 GPUs[5]. The GitHub for the MNIST and PTB experiments can be found at: https://github.com/pTTQSubmission/pTTQ.

[5]For a detailed description of this cluster, we refer the reader to http://www.idris.fr/jean-zay/jean-zay-presentation.html

4.2. Experiment 1: Comparison with state-of-the-art extreme quantization methods

The aim of this experiment is to compare our approach with three state-of-the-art (quantization) methods: a binary method (DoReFa [38])[6] using 1-bit encoding for quantized weights, a ternary method (TTQ [13]) using 2-bit encoding, and a FP model encoding weights with 32/64 bits.

To this end, we trained different models on the three aforementioned datasets. For the HITS and ESR datasets, we used the 2D CNN and the 1D CNN-Transformer from [57]. For the MNIST dataset, we used a vanilla 2D CNN model consisting of an encoder with two convolutional layers with 10 and 20 kernels of shape 5×5 (followed by max pooling and a ReLU activation function), and a classifier consisting of two linear layers with 80 and 50 input features, respectively. Moreover, we applied dropout with a ReLU activation function after the first linear layer, as well as dropout with a probability of 0.5, and a logarithmic softmax activation after the second linear layer.

The training parameters were adapted to the dataset and the model, as reported in Table 1. For the 2D CNN model on the HITS dataset, a learning rate of 10^{-3} was used for both the scaling factors and the thresholds. For the ESR dataset, the 2D CNN model was trained using a learning rate of 10^{-3} for the scaling factors and 10^{-5} for the thresholds, while the 1D CNN-Transformer was trained using a learning rate of 10^{-4} for both the scaling factors and the thresholds. For the rest of the models, the previously mentioned learning rates were identical to the global learning rate. The

[6]Binary methods yield non-sparse models as 0 is not a possible quantization value, but they tend to have higher compression rates for the non-zero weights.

results are presented in Table 2.

First, we can observe that in terms of energy consumption, pTTQ is the best performing method for all the tested models and datasets. This is evidenced by the improvements from 0.13% to 71.99% for EC_G^T with respect to DoReFa or TTQ. In particular, for the 2D CNNs on the HITS and ESR datasets (the ones with the highest percentage of weights to be quantized), this improvement is remarkable, with enhancements from 13.87% to 71.99%. This is an important advantage of our method, especially for energy-limited applications.

Second, we note that, as expected, in terms of compression the best performing models are obtained with the binarization method DoReFa, followed by ternarized models with our pTTQ approach. Indeed, DoReFa outperforms TTQ and pTTQ by margins ranging from 3.05% to 64.22% in terms of CR_G^T. However, while the discrepancy between TTQ and DoReFa is consistently above 10.79% CR_G^T, pTTQ comes closer to DoReFa in terms of compression, despite necessitating an additional bit to encode the quantized weights. Moreover, pTTQ is capable of outperforming DoReFa in terms of compression for the 1D CNN-Transformer model on the ESR dataset, with an improvement of CR_G^T of 0.40%. Additionally, despite going from 32 bits to 1 bit when using DoReFa, CR_G^Q is not 100%, as some 32-bit scaling factors are still required.

Third, it is observed that in terms of classification, the model and the dataset influence the performance of quantization methods. In particular, for the HITS dataset (our main application), the best performing quantization method is pTTQ with margins of up to 4.28%. In addition, it can be observed that the 1D CNN-Transformer model exhibits a notable enhancement in classification performance when pTTQ is employed, with an

Table 1: Experiment 1: Training parameters for the different models based on the dataset and the quantization method used. t_{min}^{init} and t_{max}^{init} correspond to the initial values chosen for t_{min} and t_{max} at the beginning of the training procedure. lr corresponds to the learning rate. In the last column, we specify the percentage of weights of the model to be quantized, selected using the Hessian-based metric outlined in section 3.3.

Dataset	**Model**	**Quant. method**	$\mathbf{t_{min}^{init}}$	$\mathbf{t_{max}^{init}}$	α	**lr**	**Epochs**	**No. params.**	% **weights to quantize**
HITS	2D CNN	FP	-	-	-	10^{-3}	50	1 681 923	-
		DoReFa [38]	-	-	-	3×10^{-3}	50		92.05
		TTQ [13]							
		pTTQ	2	2	10^4	5×10^{-3}	20		
	1D CNN-Trans.	FP	-	-	-	7×10^{-2}	150	766 271	-
		DoReFa [38]	-	-	-	10^{-4}	50		14.97
		TTQ [13]							
		pTTQ	3	3	10^3	10^{-5}	150		
ESR	2D CNN	FP	-	-	-	10^{-3}	100	1 555 842	-
		DoReFa [38]	-	-	-	10^{-3}	50		99.51
		TTQ [13]							
		pTTQ	4.5	4.5	10^3	10^{-3}	70		
	1D CNN-Trans.	FP	-	-	-	3×10^{-1}	100	109 942	-
		DoReFa [38]	-	-	-	10^{-3}	150		24.22
		TTQ [13]					100		
		pTTQ	1	1	800	10^{-3}	150		
MNIST	2D MNIST CNN	FP	-	-	-	10^{-3}	70	9 840	-
		DoReFa [38]	-	-	-	10^{-4}	200		53.35
		TTQ [13]							
		pTTQ	1	1	10^4	5×10^{-6}	50		

Table 2: Results of Experiment 1, in %. FP corresponds to the FP model where no quantization has been performed. ΔMCC corresponds to the difference between the MCC of the FP model and the MCC of the quantized model. CR_G^T, CR_G^Q, $SRQW$, and EC_G^T evaluate the compression and energy performance of each quantization method, and were introduced in 3.4. Values in bold correspond to the highest values.

Dataset	**Model**	**Quant. method**	$CR_G^T \uparrow$	$CR_G^Q \uparrow$	$SRQW$	$EC_G^T \uparrow$	**MCC** $\uparrow$	**Δ MCC** $\uparrow$
HITS	2D CNN	FP	-	-	-	-	89.84 ± 3.09	-
		DoReFa [38]	$\mathbf{89.18 \pm 0}$	$\mathbf{96.87 \pm 0}$	-	3.54 ± 0	85.05 ± 5.96	-4.79
		TTQ [13]	24.96 ± 2.25	27.12 ± 2.44	28.96 ± 2.12	*23.42 ± 1.30*	*86.82 ± 2.29*	*−3.02*
		pTTQ	*75.54 ± 3.39*	*82.06 ± 3.69*	$\mathbf{83.12 \pm 3.47}$	$\mathbf{75.53 \pm 1.53}$	$\mathbf{89.33 \pm 4.45}$	$\mathbf{-0.55}$
	1D CNN-Trans.	FP	-	-	-	-	82.64 ± 1.77	-
		DoReFa [38]	$\mathbf{14.50 \pm 0}$	$\mathbf{96.87 \pm 0}$	-	0.37 ± 0.03	*84.07 ± 3.11*	*+1.43*
		TTQ [13]	0.14 ± 0.04	0.91 ± 0.27	6.75 ± 0.26	*1.88 ± 0.03*	83.22 ± 2.36	$+0.58$
		pTTQ	*8.37 ± 0.05*	*55.89 ± 0.34*	$\mathbf{58.50 \pm 0.32}$	$\mathbf{2.01 \pm 0.05}$	$\mathbf{85.12 \pm 1.94}$	$\mathbf{+2.48}$
ESR	2D CNN	FP	-	-	-	-	92.81 ± 3.53	-
		DoReFa [38]	$\mathbf{96.40 \pm 0}$	$\mathbf{96.87 \pm 0}$	-	29.90 ± 0	*94.12 ± 0.87*	*+1.31*
		TTQ [13]	85.61 ± 1.37	86.03 ± 1.37	86.59 ± 1.29	*76.45 ± 1.13*	$\mathbf{95.00 \pm 1.11}$	$\mathbf{+2.19}$
		pTTQ	*93.35 ± 0.96*	*93.80 ± 0.96*	$\mathbf{94.17 \pm 0.91}$	$\mathbf{90.32 \pm 0.69}$	92.23 ± 2.32	-0.58
	1D CNN-Trans.	FP	-	-	-	-	94.33 ± 1.51	-
		DoReFa [38]	*23.46 ± 0*	*96.86 ± 0*	-	0.90 ± 0	$\mathbf{96.79 \pm 0.55}$	$\mathbf{+2.46}$
		TTQ [13]	11.40 ± 2.61	47.07 ± 10.79	50.22 ± 10.16	*3.21 ± 0.66*	96.25 ± 0.79	$+1.92$
		pTTQ	$\mathbf{23.86 \pm 0.04}$	$\mathbf{98.54 \pm 0.16}$	$\mathbf{98.67 \pm 0.15}$	$\mathbf{6.04 \pm 0.01}$	*96.35 ± 0.95*	*+2.02*
MNIST	2D MNIST CNN	-	-	-	-	-	94.39 ± 0.46	-
		DoReFa [38]	$\mathbf{51.67 \pm 0}$	$\mathbf{96.84 \pm 0}$	-	*3.28 ± 0*	87.03 ± 7.14	-7.36
		TTQ [13]	13.86 ± 2.33	25.97 ± 4.37	30.40 ± 4.12	2.58 ± 0.35	$\mathbf{92.09 \pm 0.89}$	$\mathbf{-2.30}$
		pTTQ	*33.92 ± 1.02*	*63.58 ± 1.92*	$\mathbf{65.79 \pm 1.80}$	$\mathbf{6.10 \pm 0.15}$	*91.01 ± 0.61*	*−3.38*

increase of 2.48% in MCC compared to the FP model, while the standard deviation is reduced.

Finally, based on the previous observations, we can see that pTTQ offers an interesting trade-off between compression, energy consumption, and classification. This is evidenced by the fact that it is the method with the best energy consumption metrics, while often achieving the best or second-best compression and classification performance.

4.3. Ablation study

We conducted a thorough study of pTTQ by performing an ablation study to better understand the different components of our contribution. We therefore studied three aspects. First, we analyzed the influence of having a pruning function depending on the statistics of the weights. Second, we examined the influence of having asymmetric pruning thresholds compared to symmetric ones (original pruning proposed by [14]). Third, we investigated the influence of having global or local hyperparameters for pruning and quantization. Lastly, we studied the impact of training the hyperparameters t_{min} and t_{max} in controlling the pruning thresholds.

4.3.1. Experiment 2: Influence of pruning on weight statistics

We propose to prune the weights before ternarization using a differentiable pruning function based on the statistics of the weights to be pruned. An alternative to this is to directly train the initial thresholds, similarly to what is done in [14]. Therefore, in this experiment, we trained the same models as in the previous experiment (section 4.2), but we directly learn Δ_w^{min} and Δ_w^{max}, without making them dependent on the statistics of the weights.

The training parameters of the pTTQ models are identical to those employed in the previous experiment. Those of the trained thresholds which are independent of weights' statistics can be found in Table 3. The results are presented in Table 4.

First, we notice that using the statistics of the weights for pruning in pTTQ helps to achieve higher energy consumption gains. Indeed, for the majority of the models and datasets, quantization using pruning based on the statistics of the weights is the method with the highest energy consump-

Table 3: Experiment 2: Training parameters for the different models quantized without using the weight statistics based on the dataset. Δ_{min}^{init} and Δ_{max}^{init} correspond to the initial values chosen for the thresholds Δ_{min} and Δ_{max} that will be directly learned. lr, lr_S, and lr_T correspond to the global learning rate, the learning rate for the scaling factors, and the learning rate for the learned thresholds (Δ_{min} and Δ_{max}), respectively.

Dataset	**Model**	$\mathbf{\Delta_{min}^{init}}$	$\mathbf{\Delta_{max}^{init}}$	α	**lr**	$\mathbf{lr_S}$	$\mathbf{lr_T}$	**Epochs**
HITS	2D CNN	1	1	90	10^{-5}	10^{-3}	10^{-2}	100
	1D CNN-Trans.	1	0.5	200	10^{-3}	10^{-3}	10^{-2}	200
ESR	2D CNN	1	1	100	10^{-3}	10^{-3}	10^{-2}	50
	1D CNN-Trans.	0.5	0.5	10^4	5×10^{-4}	5×10^{-4}	5×10^{-4}	150
MNIST	2D MNIST CNN	1	1	100	10^{-3}	10^{-4}	10^{-4}	100

Table 4: Results of Experiment 2, in %. FP corresponds to the FP model where no quantization has been performed. *Weight statistics* indicates whether the thresholds of the pruning function depend on the mean and standard deviation of the weights (Yes), or whether they are directly learned (No). CR_G^T and EC_G^T evaluate the compression and energy performance of each quantization method, and were introduced in 3.4. Values in bold correspond to the highest values.

Dataset	**Model**	**Weight statistics**	$CR_G^T \uparrow$	$EC_G^T \uparrow$	**MCC** $\uparrow$
HITS	2D CNN	No	50.55 ± 0.74	42.21 ± 0.43	88.08 ± 3.67
		Yes	$\mathbf{75.54 \pm 3.39}$	$\mathbf{75.53 \pm 1.53}$	$\mathbf{89.33 \pm 4.45}$
	1D CNN-Trans.	No	$\mathbf{12.36 \pm 0.04}$	$\mathbf{7.83 \pm 0.02}$	81.81 ± 4.39
		Yes	8.37 ± 0.05	2.01 ± 0.05	$\mathbf{85.12 \pm 1.94}$
ESR	2D CNN	No	$\mathbf{95.80 \pm 1.16}$	88.46 ± 3.56	$\mathbf{92.47 \pm 4.48}$
		Yes	93.35 ± 0.96	$\mathbf{90.32 \pm 0.69}$	92.23 ± 2.32
	1D CNN-Trans.	No	22.94 ± 3.55	5.79 ± 0.89	94.80 ± 4.52
		Yes	$\mathbf{23.86 \pm 0.04}$	$\mathbf{6.04 \pm 0.01}$	$\mathbf{96.35 \pm 0.95}$
MNIST	2D MNIST CNN	No	$\mathbf{37.48 \pm 2.32}$	5.81 ± 0.43	$\mathbf{93.57 \pm 0.50}$
		Yes	33.92 ± 1.02	$\mathbf{6.10 \pm 0.15}$	91.01 ± 0.61

tion gain of EC_G^T, with improvements ranging from 0.25% to 33.62%. The only exception is the 1D CNN-Transformer, where the absence of weight statistics for pruning results in an improvement of the energy consumption gain by 5.82%. However, this gain comes at the expense of classification performance, which is reduced by approximately 3.31% MCC (on average) with respect to the quantized models using pruning based on weight statistics.

Following this, we also observe that, for the majority of models and datasets, pruning based on the statistics of the weights also allows for important gains in terms of classification performance, with an improvement of up to 3.13% MCC.

up to 3.13% MCC. Despite being quantized and pruned without using the weights' statistics, the 2D CNN model outperforms the others by a margin of 0.24% and 2.56% MCC on the ESR and MNIST datasets, respectively. Nevertheless, these models demonstrate lower energy consumption gains than those quantized using weight-statistics-based pruning.

Lastly, in terms of compression, it can be observed that when pruning is not performed based on the statistics of the weights, there is a tendency for compression performance to increase. However, in the majority of cases, this improvement is relatively small, with an increase of up to 3.99% in CR_G^T. On the other hand, the greatest gap in terms of CR_G^T is obtained when pruning the weights based on their statistics, with an increase in CR_G^T of 24.99%.

4.3.2. Experiment 3: Influence of asymmetric pruning

Another important aspect of our proposed approach is asymmetric pruning. Indeed, we make the assumption that by enlarging the function search space (using asymmetric thresholds instead of symmetric ones), we can find

neural networks with a better trade-off between compression, energy consumption, and classification performance. To do this, we modify pTTQ by using the same (symmetric) pruning function as that described in [14].

The training parameters are the identical to those employed in the previous experiment. The only difference between the two experiments is that in this experiment the models are trained with the pruning function of [14], resulting in a single threshold being learned (initialized using Δ_{min}^{init} as in the previous experiment). The results are presented in Table 5.

Table 5: Results of Experiment 3, in %. FP corresponds to the FP model where no quantization has been performed. *Asymmetry* indicates whether the thresholds of the pruning function are asymmetric (Yes) or symmetric (No). CR_G^T and EC_G^T evaluate the compression and energy performance of each quantization method, and were introduced in 3.4. Values in bold correspond to the highest values.

Dataset	**Model**	**Asymmetry**	$\mathbf{CR_G^T}\uparrow$	$EC_G^T \uparrow$	**MCC** $\uparrow$
HITS	2D CNN	No	50.55 ± 0.74	42.21 ± 0.43	88.08 ± 3.67
		Yes	$\mathbf{75.54 \pm 3.39}$	$\mathbf{75.53 \pm 1.53}$	$\mathbf{89.33 \pm 4.45}$
	1D CNN-Trans.	No	$\mathbf{14.96 \pm 0.01}$	$\mathbf{6.90 \pm 0.32}$	26.44 ± 33.07
		Yes	8.37 ± 0.05	2.01 ± 0.05	$\mathbf{85.12 \pm 1.94}$
ESR	2D CNN	No	$\mathbf{96.00 \pm 1.01}$	88.70 ± 3.45	$\mathbf{93.78 \pm 1.29}$
		Yes	93.35 ± 0.96	$\mathbf{90.32 \pm 0.69}$	92.23 ± 2.32
	1D CNN-Trans.	No	$\mathbf{24.16 \pm 0.02}$	$\mathbf{6.092 \pm 0.004}$	94.62 ± 2.21
		Yes	23.86 ± 0.04	6.04 ± 0.01	$\mathbf{96.35 \pm 0.95}$
MNIST	2D MNIST CNN	No	33.21 ± 12.22	5.29 ± 1.78	$\mathbf{93.26 \pm 0.83}$
		Yes	$\mathbf{33.92 \pm 1.02}$	$\mathbf{6.10 \pm 0.15}$	91.01 ± 0.61

First, we observe that in terms of classification, asymmetric pruning tends to yield a higher and more stable performance than symmetric pruning before ternarization. Indeed, for all the models and datasets, the classification performance of asymmetric quantized models is relatively good

(similar to the FP models), whereas the performance of the symmetrically quantized models can drop drastically. This is the case for the 1D CNN-Transformer model on the HITS dataset, where the symmetrically quantized model achieves an MCC of $26.44 \pm 33.07\%$, which corresponds to a decrease of 58.68% MCC with respect to the FP and asymmetrically quantized models.

Second, asymmetric and symmetric pruning tend to give similar results in terms of energy consumption. However, asymmetric pruning outperforms symmetric pruning by a large margin of 33.32% EC_G^T for the 2D CNN model on the HITS dataset. In addition, the symmetrically quantized 1D CNN-Transformer on the HITS dataset outperforms the one using asymmetric pruning by a margin of 4.89% EC_G^T, but this comes at the expense of a very low classification performance ($26.44 \pm 33.07\%$ MCC vs. 85.12 ± 1.94 MCC).

Lastly, both asymmetric and symmetric pruning perform similarly in terms of compression. When symmetric pruning outperforms asymmetric pruning, we observe two behaviors: either the classification performance is low (e.g., the 1D CNN-Transformer on the HITS dataset with an MCC of $26.44 \pm 33.07\%$), or the gap is relatively small, from 0.30% to 2.65% CR_G^T. Conversely, when asymmetric pruning outperforms symmetric pruning by a large margin (24.99% CR_G^T for the 2D CNN on the HITS dataset), the other metrics (EC_G^T and MCC) also exhibit a notable improvement (33.32% EC_G^T and 1.25% MCC for the 2D CNN on the HITS dataset).

4.3.3. Experiment 4: Global vs. local threshold parameters

The objective of this experiment is to compare two configurations of our method: (a) when we have local pruning hyperparameters, namely, one t_{min} and one t_{max} per quantized layer; and (b) when we have global pruning

hyperparameters, namely, one t_{min} and one t_{max} for all the quantized layers. To do this, we trained the different models on the three datasets, using pTTQ with local and global hyperparameters (the same as in the previous experiment). The results can be found in Table 6.

Table 6: Results of Experiment 4, in %. *Granularity* indicates whether the thresholds of the pruning function are the same for all the layers (Global), or whether they are unique for each layer (Local). CR_G^T and EC_G^T evaluate the compression and energy performance of each quantization method, and were introduced in 3.4. Values in bold correspond to the highest values.

Dataset	**Model**	**Granularity**	$\mathbf{CR_G^T}\uparrow$	$EC_G^T \uparrow$	**MCC** $\uparrow$
HITS	2D CNN	Global	$\mathbf{92.05 \pm 0.00}$	$\mathbf{87.53 \pm 0.00}$	0 ± 0.00
		Local	75.54 ± 3.39	75.53 ± 1.53	$\mathbf{89.33 \pm 4.45}$
	1D CNN-Trans.	Global	$\mathbf{14.96 \pm 0.00}$	$\mathbf{7.01 \pm 0.37}$	43.57 ± 28.78
		Local	8.37 ± 0.05	2.01 ± 0.05	$\mathbf{85.12 \pm 1.94}$
ESR	2D CNN	Global	$\mathbf{99.51 \pm 0.01}$	$\mathbf{94.85 \pm 0.00}$	5.27 ± 12.03
		Local	93.35 ± 0.96	90.32 ± 0.69	$\mathbf{92.23 \pm 2.32}$
	1D CNN-Trans.	Global	11.50 ± 0.34	3.09 ± 0.09	93.59 ± 1.10
		Local	$\mathbf{23.86 \pm 0.04}$	$\mathbf{6.04 \pm 0.01}$	$\mathbf{96.35 \pm 0.95}$
MNIST	2D MNIST CNN	Global	0.10 ± 0.73	0.88 ± 0.14	59.43 ± 4.61
		Local	$\mathbf{33.92 \pm 1.02}$	$\mathbf{6.10 \pm 0.15}$	$\mathbf{91.01 \pm 0.61}$

First, we note that, in terms of classification, locally learned quantization parameters allow us to obtain a significantly better performance, with up to 89.33% MCC improvement. Indeed, the performance of the quantized models with local quantization parameters is consistent across datasets and models, close to their FP counterparts. Moreover, when using global quantization, the performance tends to vary considerably from one dataset and model to another, often achieving extremely low performance, compared with locally quantized models or FP models.

Second, if we focus only on compression and energy, the advantage of locally over globally learned parameters is not obvious, because depending on the dataset or the model, the quantization method giving the best performance is not always the same. Indeed, for some datasets and models, such as the 2D CNN models on the HITS dataset, global quantization parameters allow us to achieve a CR_G^T 16.51% higher than local quantization parameters. On the other hand, on the ESR dataset, the 1D CNN-Transformer model quantized with local parameters achieves a CR_G^T 12.36% higher than the one quantized with global parameters. Nevertheless, the classification performance shows that using global quantization parameters yields models that are useless in practice as each time that they achieve better compression and energy performance, their classification performance is relatively lower than that of the FP and locally quantized models.

4.3.4. Experiment 5: Influence of training the pruning thresholds

One important characteristic of pTTQ is that the two hyperparameters controlling the pruning thresholds, t_{min} and t_{max}, are learnable. In this experiment, we investigate the influence of learning these hyperparameters on compression, energy consumption, and classification performance. To this end, we trained different models on the three datasets using pTTQ, both with and without learning the hyperparameters t_{min} and t_{max}. In the case where the hyperparameters were not trained, we performed a grid search in the range $[-4, 4]$ and selected those that yielded the highest classification performance. The training parameters for the pTTQ quantized models with trained thresholds are the same as those used in Experiment 1, while the parameters for models without trained thresholds are listed in Table 7. The results of these experiments are presented in Table 8.

Table 7: Experiment 5: Training parameters for the different models based on the dataset, using fixed t_{min} and t_{max} hyperparameters for pTTQ. Here, lr denotes the learning rate employed during training.

Dataset	**Model**	$\mathbf{t_{min}}$	$\mathbf{t_{max}}$	**lr**	**Epochs**
HITS	2D CNN	-4	0	10^{-4}	150
	1D CNN-Trans.	-2	1.5	5×10^{-5}	100
ESR	2D CNN	-3	1	10^{-3}	200
	1D CNN-Trans.	-2	1	5×10^{-4}	100
MNIST	2D MNIST CNN	-1	0.5	10^{-3}	200

Table 8: Results of Experiment 5, in %. *Trained thresholds* specifies whether the hyperparameters t_{min} and t_{max}, which control the pruning thresholds, were trained or kept fixed. CR_G^T and EC_G^T, defined in Section 3.4, assess the compression and energy performance of each quantization method. Values in bold represent the highest values.

Dataset	**Model**	**Trained thresholds**	$\mathbf{CR_G^T}\uparrow$	$\mathbf{EC_G^T}\uparrow$	**MCC** $\uparrow$
HITS	2D CNN	No	42.98 ± 0.23	44.04 ± 0.19	86.14 ± 3.37
		Yes	$\mathbf{75.54 \pm 3.39}$	$\mathbf{75.53 \pm 1.53}$	$\mathbf{89.33 \pm 4.45}$
	1D CNN-Trans.	No	$\mathbf{13.94 \pm 0.02}$	$\mathbf{7.64 \pm 0.11}$	81.66 ± 4.17
		Yes	8.37 ± 0.05	2.01 ± 0.05	$\mathbf{85.12 \pm 1.94}$
ESR	2D CNN	No	88.48 ± 0.44	84.49 ± 0.33	$\mathbf{92.41 \pm 2.22}$
		Yes	$\mathbf{93.35 \pm 0.96}$	$\mathbf{90.32 \pm 0.69}$	92.23 ± 2.32
	1D CNN-Trans.	No	21.02 ± 0.15	5.37 ± 0.04	95.34 ± 0.79
		Yes	$\mathbf{23.86 \pm 0.04}$	$\mathbf{6.04 \pm 0.01}$	$\mathbf{96.35 \pm 0.95}$
MNIST	2D MNIST CNN	No	28.98 ± 1.26	4.97 ± 0.22	$\mathbf{93.62 \pm 0.96}$
		Yes	$\mathbf{33.92 \pm 1.02}$	$\mathbf{6.10 \pm 0.15}$	91.01 ± 0.61

We observe that for almost all datasets and models, training t_{min} and t_{max} results in higher compression and energy efficiency, achieving improvements in CR_G^T and EC_G^T of up to 32.56% and 31.49%, respectively. The only exception is the HITS dataset with the 1D CNN-Transformer, where not training t_{min} and t_{max} yields a marginal benefit in compression and

energy efficiency, with CR_G^T and EC_G^T improvements of 5.57% and 5.63%, respectively. However, in this case, classification performance declines compared to the FP model, whereas using pTTQ with trained t_{min} and t_{max} enhances classification performance.

Regarding classification performance, both strategies yield similar results, with pTTQ using trained thresholds often outperforming its non-trained counterpart. On average, models quantized with trained thresholds surpass those with fixed thresholds by a margin of 0.97% in terms of MCC. In cases where models with fixed thresholds outperform those with trained thresholds in terms of MCC, their compression and energy consumption performance is inferior. However, the opposite is not necessarily true. This demonstrates that training the pruning thresholds t_{min} and t_{max} allows for a better trade-off between compression, energy consumption, and classification performance.

5. Discussion

5.1. Experiment 1: Comparison with state-of-the-art extreme quantization methods

The results of this experiment showed the value of our method in terms of trade-off between compression, energy, and classification performance compared to other state-of-the-art extreme quantization methods.

In terms of energy consumption, pTTQ outperforms DoReFa and TTQ for all the datasets and models. This is an important property of our method, as it is of significant importance in certain applications, especially considering the increasing energy constraints imposed by environmental factors. In specific scenarios, devices may possess limited energy resources or must

adhere to predetermined energy consumption constraints. This limitation discourages the use of alternative models that cannot meet these specified constraints. Our pTTQ approach achieves energy saving of up to 90.32% compared to FP models, with only a small drop in classification performance (at most -3.38% MCC), and often improves the MCC. The excellent energy efficiency of our quantization approach can be attributed to several factors. First, we applied asymmetric pruning based on the statistics of the weights before ternarization, which allows us to meticulously eliminate weights that deviate significantly from the mean in each quantized layer. Conversely, in TTQ, weights are removed by maintaining values relatively close to the maximum of the absolute value, (following a symmetric approach), which raises the following concern: the maximum absolute value could sometimes be an outlier in the distribution of the weights. Secondly, thanks to the employed pruning technique, pTTQ eliminates a significant number of parameters. This enables a substantial reduction in the number of non-zero operations, which in turn reduces lowering energy consumption.

On the other hand, in terms of energy consumption, DoReFa does not perform as well as pTTQ or TTQ, yielding smaller energy consumption gains. This is due to the nature of the DoReFa approach, which involves binarizing the weights (with a scaling factor), excluding zero as a possible quantized value. As a result, pruning is not applied in this process. Thus, DoReFa can barely save energy through efficient sparse operations, as it performs similarly to the FP model in terms of mult-adds, but is more efficient than the FP model in terms of data transfers. Nevertheless, in terms of compression, DoReFa tends to give better results than the other methods. This is to be expected, since binary DoReFa is a more aggressive quantization method than pTTQ and TTQ, where only one bit is needed to

encode the binarized weights (plus 32 bits for the scaling factors), whereas pTTQ and TTQ require 2 bits (plus the 32 bits of the scaling factors). This means that, ignoring sparsity, DoReFa should offer compression rates that are at least twice as high. Nevertheless, pTTQ is able to achieve similar or even better compression rates (for the 1D CNN-Transformer on the ESR dataset) than DoReFa, despite using an extra bit to store the quantized weights. This is due to the high sparsity rates achieved by pTTQ, which allows a reduction in the memory requirements when only the non-zero weights are stored (e.g., by COO sparse coding). This property of our method is interesting as the incorporation of the additional bit in pTTQ (compared to DoReFa) introduces asymmetry, which enlarges the function search space. This expansion facilitates achieving a more favorable trade-off between compression, energy consumption, and classification performance.

Finally, in terms of classification, our pTTQ approach yields more consistent results across datasets and models, with results relatively close to the FP models, making it easier to use in different heterogeneous applications. In some cases, classification can be improved, which can be justified by three factors. First, sparsity itself serves as a form of regularization, a fact supported by several studies [10]. Second, the process of quantization also aids in regularization since deep neural networks tend to be highly over-parameterized and contain redundancies. Lastly, the quantized models are derived from pre-trained FP models, with the specific layers selected for quantization on the basis of a Hessian-based quantization sensitivity metric. As a result, the fine-tuning quantization step plays a role in guiding the models towards a local minimum, as the loss landscape is relatively flat for the chosen layers undergoing quantization.

5.2. Ablation study

5.2.1. Experiment 2: Influence of pruning on weight statistics

This experiment demonstrated the value of employing asymmetric pruning based on weight statistics during ternarization, enhancing the balance between compression, energy efficiency, and classification performance.

As discussed in Experiment 1, employing asymmetric pruning based on weight statistics enables the preservation of important weights, namely those that are relatively close to the mean of the distribution of the weights. Therefore, by carefully choosing the weights to be removed, it is possible to generally achieve better energy consumption and compression with only a small drop in classification performance. What is more, we also observe that, even though in some cases ternarization based on the statistics of the weights leads to smaller compression rate gains, a lower energy consumption is achieved. This can be attributed to the fact that, in contrast to ternarization without weight statistics pruning, our approach eliminates weights used in computationally more intensive operations (convolutions). This results in greater energy consumption gains, although the compression rate gains are comparatively smaller.

Lastly, in terms of the compression/energy/classification trade-off, our approach achieves better results, as it generally outperforms ternarization without pruning based on the statistics of the weights in terms of energy consumption and classification, while achieving similar compression rates. This is important because our objective is to maintain the highest classification performance while reducing energy and memory requirements. This should improve the usability of the models in a wide range of applications.

5.2.2. Experiment 3: Influence of asymmetric pruning

This experiment confirmed that asymmetry has a positive influence on the compression/energy/classification trade-off.

We saw that asymmetry can produce more consistent classification results across models and datasets, with similar and even better compression rates and energy consumption gains. This can be explained by two factors. First, asymmetry allows us to increase the function search space, allowing for a more accurate approximation of a global/local minimizer of the loss function. Second, positive and negative weight values within a neural network may not have an equivalent impact on the final classification performance, therefore there is no reason to process them under a symmetry constraint.

Lastly, the trade-off obtained with asymmetry is better than that obtained with symmetry as, globally, the classification performance is better, for similar (or even better) compression and energy consumption performance. This is particularly interesting for applications where the three factors (compression, energy, and classification) are crucial, such as medical applications (e.g., HITS classification using portable TCD devices).

5.2.3. Experiment 4: Global vs. local threshold parameters

This last experiment also confirmed that working with local quantization parameters (one set of t_{min}, t_{max}, and α per layer) instead of global ones yields a better compression/energy/classification trade-off.

In fact, we observed that local quantization parameters always give better classification performance. The cases where CR_G^T and EC_G^T are higher for models quantized with global quantization parameters are not valuable, as they often reach a useless classification performance. This phenomenon can be attributed to the different distributions and characteristics of the

different layers, indicating that each layer does not follow a uniform pattern. This implies that each layer does not employ the same information for feature extraction or classification purposes. Consequently, the quantization parameters should be adapted to each quantized layer to obtain optimal results (which is shown in this experiment by the poor classification performance of the models quantized with global quantization parameters).

Finally, following on from last point, it is important to note that high compression and energy consumption performance are not relevant if the classification performance (which determines whether the model can be used in practice) is not acceptable for the desired task. For instance, medical applications typically require high classification performance. Hence, if a model falls short in classification performance, it may not find practical usage. However, the opposite scenario should also be avoided: even if a model demonstrates outstanding classification performance, its impracticality arises if it exceeds memory constraints, exhibits slow inference times or high energy consumption).

5.2.4. Experiment 5: Influence of training the pruning thresholds

This experiment highlights the benefits of training the threshold parameters t_{min} and t_{max} in achieving a better trade-off among compression, energy consumption, and classification performance. On average, trained thresholds yield improvements of 7.93%, 6.70%, and 0.97% over their non-trained counterparts in terms of CR_G^T, EC_G^T, and MCC, respectively. One possible explanation for this phenomenon is that training these parameters allows for the automatic adaptation of thresholds to the specific data distribution and model, which is a crucial advantage as none of the other methods compared in this work offer this level of adaptability.

Furthermore, it is interesting to note that pTTQ with fixed pruning thresholds alone can achieve remarkable results, often surpassing other state-of-the-art methods such as binary DoReFa or TTQ from Experiment 1. This underscores the significance of pruning based on weights' statistics before ternarization, in contrast to TTQ, where implicit pruning occurs through the maximum weight value in the layer being quantized. Our approach is more robust to outliers, as the maximum value can be easily affected by them.

5.2.5. Limitations

Although our proposed approach offers an interesting trade-off between compression, energy consumption, and classification performance, some limitations must be highlighted.

First, in the different experiments conducted here, some of the layers and parameters of the model were not quantized, which means that in practice we can compress the different models even further. The main inconvenience with this is that, since we are working with extreme quantization methods, a complete quantization of the models will tend to considerably reduce the classification performance of the model. One solution is to use mixed quantization approaches, such as those in [22, 23, 24], which can help reduce memory, energy, and computational resources without significantly degrading of the classification performance.

Second, even though we studied our approach on several datasets and models, further experimentation can be performed on more models and datasets to validate the generality of the approach.

Thirdly, in our approach, all the quantization parameters are learned, but they still need to be initialized with good values in order to avoid small

compression and energy gains or a significant drop in classification performance. Further experimentation is needed to propose a general initialization strategy, more efficient than random or grid search.

Fourthly, in this work we presented an energy consumption metric allowing us to easily compare different quantized models from an energy perspective. The advantage of this metric is that it is not hardware dependent and does not require manual measurements, which can be difficult to perform without any bias. However, this metric is based on some theoretical assumptions and has not been empirically validated[7]. Such a validation could help us to obtain an order of magnitude of the approximation error committed by our metric.

Lastly, despite the fact that our quantization approach offers interesting compression, energy, and compression properties, it is not straightforward to highlight them in practice. Indeed, in order to truly take advantage of our method, specialized hardware should be developed to take advantage of sparse operations and low bit-width operations combined with high bit-width operations (32 or 64 bits).

6. Conclusion and future work

In this work, we presented a new extreme quantization (ternarization) approach that directly incorporates asymmetric pruning into the ternariza-

[7]The assumptions that we have made are reasonable, especially since dedicated hardware can be designed to leverage quantized and sparse models (pertaining to the first and third assumptions). The second assumption does not pose a significant problem, as we err on the side of caution by overestimating the total energy consumption, recognizing that multiplication operations typically consume more energy than addition operations.

tion mechanism, allowing us to achieve a better compression/energy/classification trade-off than other state-of-the-art methods such as TTQ or DoReFa. To do so, we proposed a novel quantization heuristic using an asymmetric differentiable pruning function, based on the weights' statistics of the layers to be quantized.

Extensive experimentation was performed to study and compare our method with other state-of-the-art methods, achieving improvements of up to 4% in MCC, with energy consumption gains of over 71% and compression rate gains of over 50%.

In our upcoming research, we intend to focus on designing dedicated hardware optimized for the operations required by our quantized models. This hardware will enhance the inference efficiency and substantially lower energy consumption in practical applications.

Acknowledgments

This work was carried out in the context of the CAREMB project funded by the Auvergne-Rhône-Alpes region, within the *Pack Ambition Recherche* program. This work was performed within the framework of the LABEX CELYA (ANR-10-LABX-0060) and PRIMES (ANR-11-LABX-0063) of Universite de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR). This work was performed using HPC resources from GENCI-IDRIS (Grant 2022-AD011013616)

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used ChatGPT, Copilot, and Deepl in order to improve the readability and language. After using this tools, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

[1] Z. Li, F. Liu, W. Yang, S. Peng, J. Zhou, A survey of convolutional neural networks: Analysis, applications, and prospects, IEEE Transactions on Neural Networks and Learning Systems 33 (12) (2022) 6999–7019. doi:10.1109/TNNLS.2021.3084827.

[2] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: Transformers for image recognition at scale, ICLR (2021).

[3] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, A. Rush, Transformers: State-of-the-art natural language processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Online, 2020, pp. 38–45. doi:10.18653/v1/2020.emnlp-demos.6.

[4] S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. Yalta, R. Yamamoto, X. fei Wang, S. Watanabe,

T. Yoshimura, W. Zhang, A comparative study on transformer vs rnn in speech applications, 2019, pp. 449–456.

[5] A. Tjandra, C. Liu, F. Zhang, X. Zhang, Y. Wang, G. Synnaeve, S. Nakamura, G. Zweig, Deja-vu: Double feature presentation and iterated loss in deep transformer networks, in: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, pp. 6899–6903.

[6] C. Che, P. Zhang, M. Zhu, Y. Qu, B. Jin, Constrained transformer network for ecg signal processing and arrhythmia classification, BMC Medical Informatics and Decision Making 21 (2021).

[7] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, F. Kawsar, An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices, in: Proceedings of the 2015 International Workshop on Internet of Things towards Applications, IoT-App '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 7–12. doi:10.1145/2820975.2820980.

[8] Y. Cheng, D. Wang, P. Zhou, T. Zhang, Model compression and acceleration for deep neural networks: The principles, progress, and challenges, IEEE Signal Processing Magazine 35 (1) (2018) 126–136. doi:10.1109/MSP.2017.2765695.

[9] A. Gholami, S. Kim, D. Zhen, Z. Yao, M. Mahoney, K. Keutzer, A Survey of Quantization Methods for Efficient Neural Network Inference, 2022, pp. 291–326. doi:10.1201/9781003162810-13.

[10] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, A. Peste, Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks, J. Mach. Learn. Res. 22 (1) (jul 2022).

[11] T. Elsken, J. H. Metzen, F. Hutter, Neural architecture search: A survey, J. Mach. Learn. Res. 20 (1) (2021) 1997–2017.

[12] L. Lu, B. Lyu, Reducing energy consumption of neural architecture search: An inference latency prediction framework, Sustainable Cities and Society 67 (2021) 102747. doi:https://doi.org/10.1016/j.scs.2021.102747.

[13] C. Zhu, S. Han, H. Mao, W. J. Dally, Trained ternary quantization, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, OpenReview.net, 2017.

[14] F. Manessi, A. Rozza, S. Bianco, P. Napoletano, R. Schettini, Automated pruning for deep neural network compression, in: 2018 24th International Conference on Pattern Recognition (ICPR), 2018, pp. 657–664. doi:10.1109/ICPR.2018.8546129.

[15] Y. Gong, L. Liu, M. Yang, L. D. Bourdev, Compressing deep convolutional networks using vector quantization, CoRR abs/1412.6115 (2014). arXiv:1412.6115.
URL http://arxiv.org/abs/1412.6115

[16] S. Han, H. Mao, W. J. Dally, Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding, in: Y. Bengio, Y. LeCun (Eds.), 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings, 2016.

[17] K. Ullrich, E. Meeds, M. Welling, Soft weight-sharing for neural network compression., in: ICLR (Poster), OpenReview.net, 2017.
URL http://dblp.uni-trier.de/db/conf/iclr/iclr2017.html#

UllrichMW17

[18] A. Dubey, M. Chatterjee, N. Ahuja, Coreset-based neural network compression, in: Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part VII, Springer-Verlag, Berlin, Heidelberg, 2018, p. 469–486. doi:10.1007/978-3-030-01234-2_28.

[19] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut, ALBERT: A lite BERT for self-supervised learning of language representations, in: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020, OpenReview.net, 2020.

[20] G. Prato, E. Charlaix, M. Rezagholizadeh, Fully quantized transformer for machine translation, in: Findings of the Association for Computational Linguistics: EMNLP 2020, Association for Computational Linguistics, Online, 2020, pp. 1–14. doi:10.18653/v1/2020.findings-emnlp.1.

[21] Y. Xu, Y. Wang, A. Zhou, W. Lin, H. Xiong, Deep neural network compression with single and multiple level quantization, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, AAAI'18/IAAI'18/EAAI'18, AAAI Press, 2018.

[22] Z. Dong, Z. Yao, A. Gholami, M. W. Mahoney, K. Keutzer, Hawq: Hessian aware quantization of neural networks with mixed-precision, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019.

[23] Z. Dong, Z. Yao, D. Arfeen, A. Gholami, M. W. Mahoney, K. Keutzer, Hawq-v2: Hessian aware trace-weighted quantization of neural net-

works, in: Advances in Neural Information Processing Systems, Vol. 33, Curran Associates, Inc., 2020, pp. 18518–18529.

[24] Z. Yao, Z. Dong, Z. Zheng, A. Gholami, J. Yu, E. Tan, L. Wang, Q. Huang, Y. Wang, M. Mahoney, K. Keutzer, Hawq-v3: Dyadic neural network quantization, in: M. Meila, T. Zhang (Eds.), Proceedings of the 38th International Conference on Machine Learning, Vol. 139 of Proceedings of Machine Learning Research, PMLR, 2021, pp. 11875–11886.

[25] J. L. McKinstry, S. K. Esser, R. Appuswamy, D. Bablani, J. V. Arthur, I. B. Yildiz, D. S. Modha, Discovering low-precision networks close to full-precision networks for efficient embedded inference, CoRR abs/1809.04191 (2018).

[26] Q. Jin, L. Yang, Z. Liao, Adabits: Neural network quantization with adaptive bit-widths, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Los Alamitos, CA, USA, 2020, pp. 2143–2153. doi:10.1109/CVPR42600.2020.00222.

[27] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, in: NIPS Deep Learning and Representation Learning Workshop, 2015.

[28] A. Polino, R. Pascanu, D. Alistarh, Model compression via distillation and quantization, CoRR abs/1802.05668 (2018). arXiv:1802.05668.

[29] W. Zhang, L. Hou, Y. Yin, L. Shang, X. Chen, X. Jiang, Q. Liu, Ternarybert: Distillation-aware ultra-low bit bert, in: Conference on Empirical Methods in Natural Language Processing, 2020.

[30] H. Bai, W. Zhang, L. Hou, L. Shang, J. Jin, X. Jiang, Q. Liu,

M. Lyu, I. King, BinaryBERT: Pushing the limit of BERT quantization, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Association for Computational Linguistics, Online, 2021, pp. 4334–4348. doi:10.18653/v1/2021.acl-long.334.

[31] P. Yin, J. Lyu, S. Zhang, S. J. Osher, Y. Qi, J. Xin, Understanding straight-through estimator in training activation quantized neural nets, in: International Conference on Learning Representations, 2019.

[32] D. Zhang, J. Yang, D. Ye, G. Hua, Lq-nets: Learned quantization for highly accurate and compact deep neural networks, in: Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VIII, Springer-Verlag, Berlin, Heidelberg, 2018, p. 373–390. doi:10.1007/978-3-030-01237-3_23.

[33] Y. Bhalgat, J. Lee, M. Nagel, T. Blankevoort, N. Kwak, Lsq+: Improving low-bit quantization through learnable offsets and better initialization, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2020.

[34] J. Yang, X. Shen, J. Xing, X. Tian, H. Li, B. Deng, J. Huang, X.-s. Hua, Quantization networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

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

[36] O. Zafrir, G. Boudoukh, P. Izsak, M. Wasserblat, Q8BERT: quantized 8bit BERT, in: Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition, EMC2@NeurIPS 2019, Vancouver, Canada, December 13, 2019, IEEE, 2019, pp. 36–39. doi:10.1109/EMC2-NIPS53020.2019.00016.

[37] S. Kim, A. Gholami, Z. Yao, M. W. Mahoney, K. Keutzer, I-bert: Integer-only bert quantization, International Conference on Machine Learning (Accepted) (2021).

[38] S. Zhou, Z. Ni, X. Zhou, H. Wen, Y. Wu, Y. Zou, Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients, CoRR abs/1606.06160 (2016). arXiv:1606.06160.

[39] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, Xnor-net: Imagenet classification using binary convolutional neural networks, in: B. Leibe, J. Matas, N. Sebe, M. Welling (Eds.), Computer Vision – ECCV 2016, Springer International Publishing, Cham, 2016, pp. 525–542.

[40] X. Lin, C. Zhao, W. Pan, Towards accurate binary convolutional neural network, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 30, Curran Associates, Inc., 2017.

[41] L. Hou, J. T. Kwok, Loss-aware weight quantization of deep networks, in: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, OpenReview.net, 2018.

[42] S. Shen, Z. Dong, J. Ye, L. Ma, Z. Yao, A. Gholami, M. W. Mahoney, K. Keutzer, Q-bert: Hessian based ultra low precision quantization of bert, in: AAAI Conference on Artificial Intelligence, 2019.

[43] A. Fan, P. Stock, B. Graham, E. Grave, R. Gribonval, H. Jégou, A. Joulin, Training with quantization noise for extreme model compression, CoRR abs/2004.07320 (2020). arXiv:2004.07320.

[44] T. Liang, J. Glossner, L. Wang, S. Shi, X. Zhang, Pruning and quantization for deep neural network acceleration: A survey, Neurocomputing 461 (2021) 370–403. doi:https://doi.org/10.1016/j.neucom.2021.07.045.

[45] T. Ji, S. Jain, M. Ferdman, P. Milder, H. A. Schwartz, N. Balasubramanian, On the distribution, sparsity, and inference-time quantization of attention values in transformers, ArXiv abs/2106.01335 (2021).

[46] M. Zhu, S. Gupta, To prune, or not to prune: Exploring the efficacy of pruning for model compression, in: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 2018, Workshop Track Proceedings, 2018.

[47] Z. Mariet, S. Sra, Diversity networks: Neural network compression using determinantal point processes, in: International Conference on Learning Representations (ICLR), 2016.

[48] J.-H. Luo, J. Wu, W. Lin, Thinet: A filter level pruning method for deep neural network compression, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 5068–5076. doi:10.1109/ICCV.2017.541.

[49] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, S. Han, Amc: Automl for model compression and acceleration on mobile devices, in: V. Ferrari, M. Hebert, C. Sminchisescu, Y. Weiss (Eds.), Computer Vision – ECCV 2018, Springer International Publishing, Cham, 2018, pp. 815–832.

[50] K. Xu, D. Zhang, J. An, L. Liu, L. Liu, D. Wang, Genexp: Multi-

objective pruning for deep neural network based on genetic algorithm, Neurocomputing 451 (2021) 81–94. doi:https://doi.org/10.1016/j.neucom.2021.04.022.

[51] M. S. Park, X. Xu, C. Brick, Squantizer: Simultaneous learning for both sparse and low-precision neural networks, CoRR abs/1812.08301 (2018). arXiv:1812.08301.

[52] F. Tung, G. Mori, Deep neural network compression by in-parallel pruning-quantization, IEEE Transactions on Pattern Analysis and Machine Intelligence 42 (3) (2020) 568–579. doi:10.1109/TPAMI.2018.2886192.

[53] M. Horowitz, 1.1 computing's energy problem (and what we can do about it), in: 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014, pp. 10–14. doi:10.1109/ISSCC.2014.6757323.

[54] D. Molka, D. Hackenberg, R. Schöne, M. S. Müller, Characterizing the energy consumption of data transfers and arithmetic operations on x86-64 processors, in: International Conference on Green Computing, 2010, pp. 123–133. doi:10.1109/GREENCOMP.2010.5598316.

[55] Y. Vindas, B. K. Guépié, M. Almar, E. Roux, P. Delachartre, An hybrid cnn-transformer model based on multi-feature extraction and attention fusion mechanism for cerebral emboli classification, in: Proceedings of the 7th Machine Learning for Healthcare Conference, Proceedings of Machine Learning Research, PMLR, 2022.

[56] R. Andrzejak, K. Lehnertz, F. Mormann, C. Rieke, P. David, C. Elger, Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording

region and brain state, Physical review. E, Statistical, nonlinear, and soft matter physics 64 (2002) 061907.

[57] Y. Vindas, E. Roux, B. K. Guépié, M. Almar, P. Delachartre, Guided deep embedded clustering regularization for multifeature medical signal classification, Pattern Recognition (2023) 109812doi:https://doi.org/10.1016/j.patcog.2023.109812.

[58] L. Deng, The mnist database of handwritten digit images for machine learning research, IEEE Signal Processing Magazine 29 (6) (2012) 141–142.