An asymmetric heuristic for trained ternary quantization based on the statistics of the weights: an application to medical signal classification

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

Pattern Recognition Letters
journal homepage: www.elsevier.com

An asymmetric heuristic for trained ternary quantization based on the statistics of the weights: an application to medical signal classification

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

Article history:

Model compression, Model quantization, Model pruning, Signal classification, Transcranial Doppler *2010 MSC:* 00-01, 99-00

ABSTRACT

One of the main challenges in the field of deep learning and embedded systems is the mismatch between the memory, computational and energy resources required by the former for good performance and the resource capabilities offered by the latter. It is therefore important to find a good trade-off between performance and computational resources used. In this study, we propose a novel ternarization heuristic based on the statistics of the weights, in addition to asymmetric pruning. Our approach involves the computation of two asymmetric thresholds based on the mean and standard deviation of the weights. This allows us to distinguish between positive and negative values prior to ternarization. Two hyperparameters are introduced into these thresholds, which permit the user to control the trade-off between compression and classification performance. Following thresholding, ternarization is carried out in accordance with the methodology of trained ternary quantisation (TTQ). The efficacy of the method is evaluated on three datasets, two of which are medical: a cerebral emboli (HITS) dataset, an epileptic seizure recognition (ESR) dataset, and the MNIST dataset. Two types of deep learning models were tested: 2D convolutional neural networks (CNNs) and 1D CNN-transformers. The results demonstrate that our approach, aTTQ, achieves a superior trade-off between classification performance and compression rate compared with TTQ, for all the models and datasets. In fact, our method is capable of reducing the memory requirements of a 1D CNN-transformer model for the ESR dataset by over 21% compared to TTQ, while maintaining a Matthews correlation coefficient of 95%. The code is available at: https://github.com/attq-submission/aTTQ.

1. Introduction

In recent years, deep neural networks, such as convolutional neural networks (CNNs) or transformers, have achieved state-of-the-art performance in a number of tasks, including computer vision [1, 2], natural language processing [3], and signal processing (e.g., [4, 5]). However, these models are often characterised by high energy consumption, with thousands or millions of parameters, and they frequently necessitate significant computational resources, including memory and GPU, which prevent their common use in embedded systems.

This last point is of particular interest, given that in recent years deep learning has begun to be increasingly utilised in the medical field [6]. However, the computational and memory capabilities of medical devices, particularly those designed for portability, are often limited, making the use of large models challenging. This study focuses on the classification of medical signals, specifically transcranial Doppler (TCD) ultrasound for the classification of cerebral emboli (CE) and electroencephalograms (EEG) for the recognition of epileptic seizures (ESR). These two tasks are of particular relevance to public health, as the former can assist in the prevention of stroke, and both stroke and epilepsy are among the most common neurological disorders leading

**Corresponding author
e-mail: yamil.vindas@insa-lyon.fr (Yamil Vindas), emmanuel.roux@creatis.insa-lyon.fr (Emmanuel Roux), blaise_kevin.guepie@utt.fr (Blaise Kévin Guépié), marilys.almar@atysmedical.com (Marilys Almar), philippe.delachartre@creatis.insa-lyon.fr (Philippe Delachartre)

to disability or death worldwide [7, 8].

Moreover, recent works have used deep learning models to perform medical signal classification based on CNN and transformer models [9, 4, 10, 11]. However, although these models have achieved great performance, they often have hundreds of thousands or even millions of parameters. To tackle this problem, several authors have proposed to reduce the storage and memory requirements by using different model compression techniques [12] such as model quantization [13] and model pruning [14].

To take advantage of both techniques (quantization and pruning) some authors have suggested using them sequentially as they are compatible and independent [15]. This allows one to further increase the compression rate and reduce latency. Other techniques, such as trained ternary quantization (TTQ) [16], offer implicit pruning thanks to their quantization heuristic. However, to our knowledge, few works have attempted to directly incorporate pruning into the quantization mechanism.

In this paper, we propose a new ternarization heuristic based on asymmetric pruning and on the statistics of the weights, which increases the sparsity of the parameters of the weights while keeping the rest of the quantized weights at a reduced precision, without significantly degrading the classification performance. The rationale behind our approach is that asymmetric pruning enlarges the family of neural networks that can be explored during training, making it possible to obtain models with a better trade-off between compression, energy and classification performance. Moreover, a ternarization heuristic based on the statistics of the weights allows a better adaptation of the method to new datasets. Furthermore, our approach is parameterized by two hyperparameters, t_{min} and t_{max}, which allow us to control the sparsity rate of the quantized weights and thus the trade-off between compression and classification performance. Our main contributions can be summarized as follows:

- A new heuristic for trained ternary quantization, based on the statistics of the model weights.
- Asymmetric pruning before ternarization, allowing a better trade-off between compression and classification performance.
- Asymmetric parameterization of the sparsity rate (two hyperparameters), which allows us to control the trade-off between compression and classification performance.

The rest of the paper is structured as follows. In Section 2 we present some related work. In Section 3 we explain the proposed method. In Section 4 we present the experimental setup and discuss the different results. Finally, in Section 5 we conclude and give some perspectives for future work.

2. Related work

In this section, we present the main works related to our research problem, model quantization and model pruning. For a more general survey of model compression methods, we invite the reader to see [12].

2.1. Model quantization

Quantization [13] consists of reducing the precision of the weights of a model from 32 or 64 bits to a lower precision. This can be beneficial for memory resources and inference, especially when using aggressive quantization where one can benefit from efficient logic/arithmetic operations [17] or strategies [16, 18]. Early approaches were based on matrix factorisation and vector quantization [19, 20], but were mainly designed for dense layers. More recent works have quantized convolutional layers using extreme quantization with ternary [16] or binary [21] weights in $\{-1, 0, 1\}$ or $\{-1, 1\}$ multiplied by 32-bit scaling factors, or using weight sharing, which can be achieved by applying weight clustering [15] or Gaussian mixture models [22]. Due to compression, these methods often reduce the classification performance of the models. Some works have proposed techniques based on knowledge distillation [23] to guide the training of quantized models to achieve similar performance to their full-precision (FP) counterparts. The main idea is to train a quantized model with the same architecture as the FP model, but with quantized weights, and guide it to match the soft output probabilities of the FP model.

Finally, performance degradation can be reduced by combining different quantization methods to have different precision in each part of the model [13, 24]. The main difficulty with these mixed quantization methods is the choice of the layers of the model to be quantized and their quantization precision. To address this, some metrics have been proposed to evaluate the impact of quantization on the performance of the model. In fact, [24] used Hessian-based metrics that allowed them to evaluate the flatness of the loss landscape, thus avoiding the extreme quantization of layers with irregular landscapes.

For a more complete review of recent model quantization methods, we refer the reader to [13].

2.2. Model pruning

Pruning consists of removing the redundant parameters of a model by setting them to zero. This is interesting for neural network models, as they are typically over-parameterized, and thus pruning can act as a regularizer [14]. This family of methods can also improve memory and latency, as the resulting parameter tensors are sparse [25]. Furthermore, different approaches can be used to prune the parameters of a model. In some works, the weights with minimal norm (L1 or L2 norm) are removed, using a predefined threshold or number of weights to prune [15]. In more complex methods, the weights to be removed are chosen by calculating their importance based on the statistics of the parameters of the following layer [26], or by creating subsets of neurons to be merged based on determinantal point processes. Conversely, other approaches have tried to overcome the non-differentiability of threshold operators during pruning by using reinforcement learning [27], genetic algorithms [28], or differentiable threshold functions [29].

Finally, even if the above-mentioned methods make it possible to remove an important percentage of the network parameters by setting them to zero, the rest of the parameters remain in full precision (32 or 64 bits), which prevents the use of efficient operations and increases the memory requirements with respect to the quantized parameters. To overcome this, some authors have tried to combine pruning and quantization with different approaches. [15, 30] tried a sequential combination of pruning and quantization (combined with other techniques), while [31] used Bayesian optimization techniques and [22] used soft weight sharing to achieve both pruning and quantization.

For a more thorough review of model pruning techniques, the reader is referred to [14].

3. Methods

In this section, we present a new quantization heuristic for trained ternary quantization (TTQ), called asymmetric TTQ (aTTQ) (see figure 1). Our method is based on two assumptions: (1) asymmetric pruning can improve the trade-off between compression and classification performance, and (2) pruning and quantization based on the statistics of the weights allow a better adaptation to new datasets and models. The first assumption can be justified by the fact that asymmetric pruning increases the model search space during training. The second assumption is plausible because the thresholds used for pruning depend

on the statistics of the weights, which change during the training/fine-tuning step.

3.1. Pruning based on weights statistics

We propose a novel asymmetric ternarization heuristic where the quantization threshold does not depend on the maximum absolute value of the weights' tensor (as in [16]), but depends on the weights' statistics:

$$w_t = \begin{cases} W_l \text{ if } w < \Delta_{min} \\ 0 \text{ if } w \in [\Delta_{min}, \Delta_{max}] \\ W_r \text{ if } w > \Delta_{max} \end{cases} \quad (1)$$

where w and w_t are the FP and ternarized weights, $W_l, W_r \in \mathbb{R}$ are learnable scaling parameters, $\Delta_{min} = \mu_w + t_{min} \times \sigma_w$ and $\Delta_{max} = \mu_w + t_{max} \times \sigma_w$ are pruning thresholds, and t_{min} and t_{max} are two hyperparameters controlling the sparsity rate, with the constraint $t_{min} \leq t_{max}$. Indeed, if $t_{min} > t_{max}$ then $\Delta_{min} > \Delta_{max}$ and therefore $\forall w \in \mathbb{R}, w_t \neq 0$; thus, no pruning would be performed.

The gradients of $\mathcal{L}$, the loss to be optimized, can be computed using the straightforward estimator as in TTQ, with the only difference being that the threshold Δ_L used is replaced by the thresholds Δ_{min} and Δ_{max}:

$$\frac{\partial \mathcal{L}}{\partial w} = \begin{cases} W_l \times \frac{\partial \mathcal{L}}{\partial w_t} \text{ if } w < \Delta_{min} \\ 0 \text{ if } w \in [\Delta_{min}, \Delta_{max}] \\ W_r \times \frac{\partial \mathcal{L}}{\partial w_t} \text{ if } w > \Delta_{max} \end{cases} \quad (2)$$

3.2. Evaluation metrics

To compare the sparsity and compression of the obtained models, we introduce different metrics defined in the following paragraphs. We denote by $\mathcal{M}_{FP}$ the FP model and by $\mathcal{M}_Q$ a quantized model obtained from $\mathcal{M}_{FP}$. Similarly, we denote by *nbits* a function that allows us to count the number of bits needed to store the (non-zero) weights of a model, and *nqw*/*nzqw* two functions that allow us to count the number of weights/zero weights of a model among the weights selected for quantization using the Hessian-based metric of [24].

Sparsity. To quantify the sparsity achieved during the quantization of the different models, we introduce the sparsity rate over the quantized weights, denoted SRQW and defined as

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

where higher values of $SRQW$ indicate sparser models.

Compression. We want to quantify the compression achieved by ternarization and pruning (simultaneously). To do this, we introduce the compression rate, *CR*, defined as follows:

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

To facilitate the comparison between methods, we work with the compression rate gain CR_G, defined as:

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

We denote as CR_G^T and CR_G^Q the compression rate gains of the whole model and the layers selected for quantization, respectively.

Energy consumption. We estimate the energy consumption of the different models based on the number of multiplications and additions (mult-adds) and the number of data transfers to RAM. To do this, we make three assumptions: (1) only non-zero weights are considered for the mult-adds and data transfers; (2) multiplications and additions have the same energy cost, which is the cost of a multiplication (the most expensive one) [32]; and (3) the transfers of weights to RAM memory are done in chunks of 32 bits. The proposed metric for the total energy consumption in Joule of a model $\mathcal{M}$ composed of p layers $L_1, ..., L_p$ is then defined as follows

$$EC_T(\mathcal{M}) = N_{MA} \times 3.7 \times 10^{-12} + 10^{-9} \times \sum_{i=1}^{p} (\lceil \frac{nnzw(L_i) \times B_i}{32} \rceil + N_{SF}^i) \quad (3)$$

where the factors 3.7×10^{-12} and 10^{-9} are taken from [32] and [33], respectively, N_{MA} is the number of (non-zero) mult-adds, N_{SF}^i is the number of quantization scaling factors used for the i^{th} layer (0 for an FP layer, 2 for a ternarized layer), and B_i is the number of bits needed to encode a weight of that layer (32 for an FP layer, and 2 for a ternarized layer). Moreover, the first term corresponds to the energy consumed by mult-adds, and the second term corresponds to the energy consumed by data transfers.

Finally, we measure the energy consumption gain, EC_G^T, of a quantized and sparse model $\mathcal{M}_C$ with respect to its FP counterpart, $\mathcal{M}_{FP}$, as follows:

$$EC_G^T(\mathcal{M}_{FP}, \mathcal{M}_C) = \frac{|EC_T(\mathcal{M}_{FP}) - EC_T(\mathcal{M}_C)|}{EC_T(\mathcal{M}_{FP})} \quad (4)$$

where higher values of EC_G^T indicate lower global energy consumption compared to the FP model.

4. Experiments

We perform two experiments to evaluate our approach. The first experiment compares our proposed weight-statistic quantization heuristic, aTTQ, with state-of-the-art quantization methods. We focus on extreme ternarization methods (bit width less than 4 bits) as they allow higher compression rates than higher bit width quantization methods (8, 16 or 32 bits). The second experiment studies the influence of the two hyperparameters of our approach, t_{min} and t_{max}.

4.1. Data

We train and evaluate our proposed method using repeated holdout evaluation on three datasets:

- MNIST [34]: composed of 70 000 greyscale 28×28 images (60 000 for training and 10 000 for testing) distributed in 10 balanced classes. To reduce computational resources, we used only 10% of the training samples (sampled uniformly at random). In summary, we used 37.5% samples for training and 62.5% for testing.
- HITS dataset [9, 4]: consisting of 1541 Doppler signals with a sampling frequency of 4.4 kHz distributed in three classes: 403 artefacts, 569 gaseous emboli, and 569 solid emboli. We used 63% of the samples for training and 37% for testing (subject-wise fixed split).
- ESR [35]: composed of 11 500 pre-processed[1] electroencephalograms (EEGs). Like most authors using this dataset, we perform binary classification between seizure activity (2300 samples) and no seizure activity signals (9200 samples). Finally, we used 90% of the samples for training and 10% for testing.

[1] We used the publicly available version found at `https://www.kaggle.com/datasets/harunshimanto/epileptic-seizure-recognition`

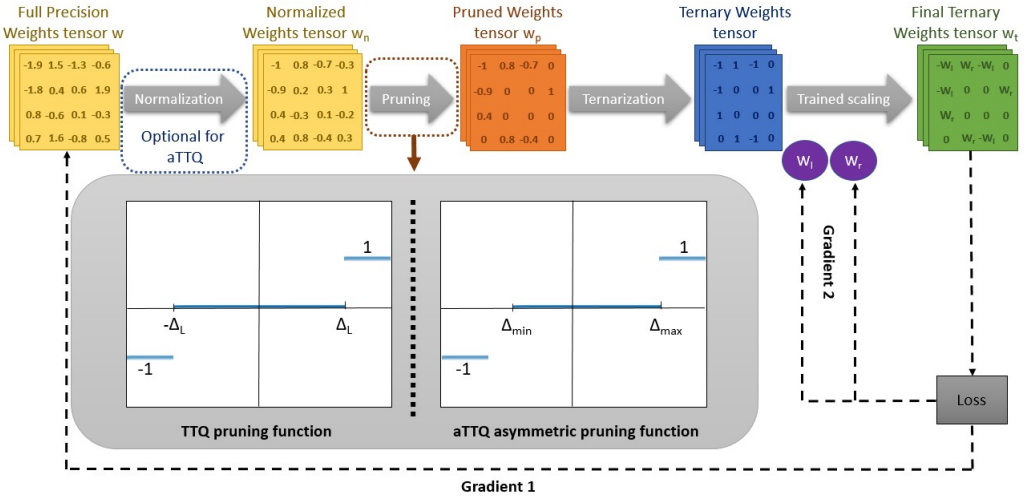

Fig. 1: Proposed asymmetric TTQ (aTTQ) method. The main difference with respect to TTQ lies in the pruning mechanism used before ternarization: We use two asymmetric thresholds, Δ_{min} and Δ_{max}, instead of one symmetric threshold $\Delta_{min} = -\Delta_{max} = -\Delta_L$. The normalization step is optional in our approach.

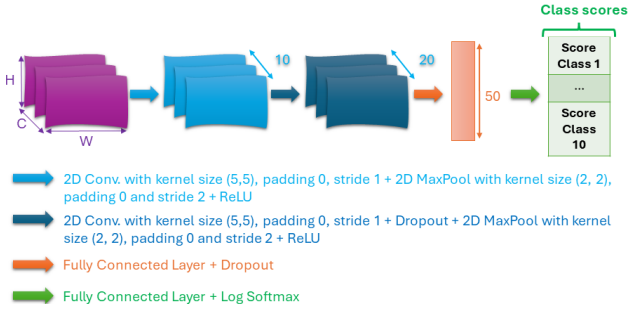

Fig. 2: 2D CNN architecture used for the MNIST classification models. C is the number of channels in the input image, W is the width and H is the height.

4.2. Architectures

We used three different models, depending on the dataset: one vanilla 2D CNN for the MNIST dataset (Figure 2), and one 2D CNN (Figure 3) and 1D CNN-transformer (Figure 4) for the medical signal datasets [5].

The vanilla 2D CNN MNIST model is composed of two convolutional layers, followed by 2D max pooling and ReLU activation applied to both, and dropout after the second convolutional layer. A fully connected (FC) classifier is then applied, consisting of two linear layers, followed by dropout and ReLU activation for the first linear layer, and logarithmic softmax for the second linear layer.

Moreover, we used the same single-feature architectures for the HITS and ESR datasets as [5][2]. For the HITS dataset, the hyperparameters of the 1D CNN-transformers were the same as [5]. For the ESR dataset, the following hyperparameters were used: The last 1D convolutional layer is applied twice, four attention heads (per multi-head attention) are used for the four transformer encoder layers, a transformer intermediate hidden dimension of 8 is used, and a dimension of 4 for the projected representation is used for the final classification. Finally, for the 2D CNN, all hyperparameters were the same as [4], except for the number of initial convolutional filters, which was set to 64.

4.3. Training parameters

Table 1 shows the training parameters used for the different models on the different datasets. All models were trained with a cross-entropy loss function, optimized using an Adamax optimizer for the 2D CNN models and Noam for the 1D CNN-Transformer models, with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and 4000 warm-up steps for all models except the TTQ and aTTQ quantized HITS models, which were trained with 700 warm-up steps. To deal with class imbalance, class weights [36, 37] were applied to all models. We used a weight decay of 10^{-7} for almost all models, except for the 1D CNN-transformer models trained on the ESR dataset, the 2D CNN trained on the MNIST dataset, which used a weight decay of 0, and the full-precision ESR 2D CNN, which used a weight decay of 10^{-5}. In addition, we used a batch size of 32 for all models except the ESR 1D CNN-transform models, which used a batch size of 64.

Furthermore, we used the Hessian-based metric of [24] to select the different layers to be quantized, and we quantized only their weights and not the biases. For the MNIST 2D CNN, all convolutional layers were quantized; for the 2D CNN used for the medical datasets, we quantized all convolutional layers except the first one; for the 1D CNN-transformer, we quantized the second convolutional layer plus the second linear layer of all encoder layers of the transformer encoder. The percentage of selected weights to be quantized (for the whole model) can be found in the last column of Table 1.

4.4. Baseline

We compare our aTTQ approach against three different baselines: FP model, TTQ [16], and DoReFa [21]. For the last one, we use the extreme binary quantization approach, where the quantized weights of the model can take the values -1 or 1 multiplied by a single scaling factor corresponding to the mean of the weights of a given layer or convolutional filter.

4.5. Evaluation metrics

We used the Matthew correlation coefficient (MCC) to measure the classification performance of the models (well suited for imbalanced

[2]All architectures described are available in the associated GitHub repository: https://github.com/attq-submission/aTTQ

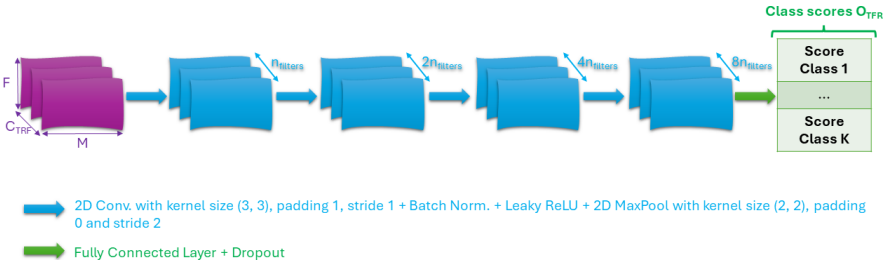

Fig. 3: 2D CNN TFR architecture used for the HITS and ESR classification models. C_{TRF} is the number of channels in the time-frequency representation, F is the number of frequency bins, and M is the number of time bins.

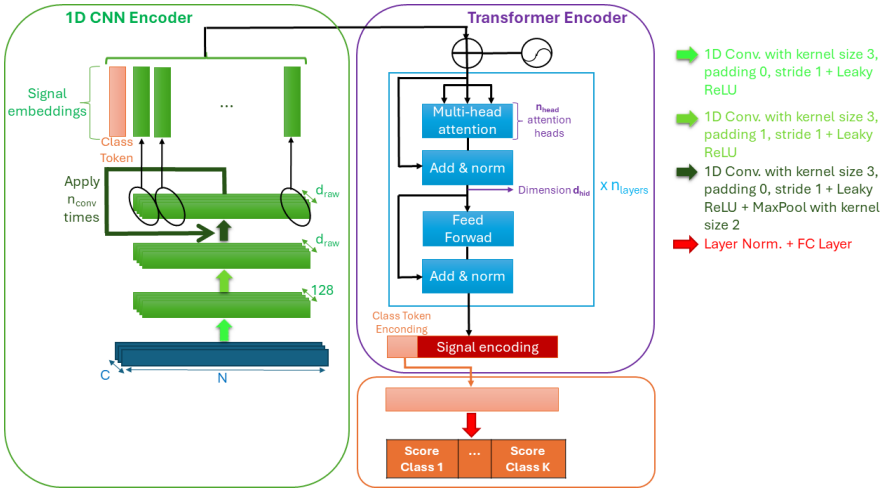

Fig. 4: 1D CNN-transformer architecture used for the HITS and ESR datasets. C is the number of channels of the input signal and N is its number of samples.

datasets) and $SRQW$, CR_G^T, CR_G^Q and EC_G^T to measure the compression and energy performance. Finally, for statistical purposes, Experiment 1 was repeated 10 times and Experiment 2 five times; the metrics reported correspond to the mean and standard deviation computed on the test sets.

4.6. *Experiment 1: Comparison with the state-of-the-art*

The aim of this experiment is to compare the performance of the proposed quantization heuristic, aTTQ, with that of TTQ and DoReFa. This comparison is made in terms of three main aspects: compression, energy and classification performance. The results are presented in Table 2.

First, we can see that aTTQ outperforms the other quantization methods in terms of energy consumption by a wide margin. In fact, aTTQ improves the energy gain EC_G^T by up to 54.59%. This is particularly notable for the 2D CNN models, which have the higher percentage of weights that are quantized. The main reason for the good energy performance of aTTQ compared to the other methods is that aTTQ offers higher sparsity rates than TTQ and DoReFa (in fact, due to the nature of DoReFa, the quantized weights are not sparse). Since energy can be saved by ignoring zero mult-adds and data transfers (with specialized hardware), aTTQ and TTQ tend to give higher energy gains.

Second, we find that DoReFa outperforms aTTQ and TTQ in terms of compression and sparsity metrics by at least 0.56% CR_G^T. This is normal behavior, since DoReFa performs a more extreme binary quantization, which allows higher compression by using one bit less to encode the quantized weights with respect to TTQ and aTTQ. However, among the ternarization methods, aTTQ outperforms TTQ by a wide margin. Indeed, aTTQ improves the sparsity rate of the quantized weights ($SRQW$) by at least 2.7% (and up to 86.8%) compared to TTQ. With respect to DoReFa, aTTQ is able to achieve similar compression performance. This is due to the high sparsity offered by aTTQ, which makes it possible to partially compensate for the extra bit used to encode the ternary weights compared to DoReFa. Furthermore, similar results are observed for the compression rate gain of the quantized layers, CR_G^Q. However, although a similar behavior is observed for the compression rate gain of the whole model, the increase of aTTQ is smaller compared to that of TTQ. This can be explained by the fact that the quantized layers do not always contain the majority of the model's parameters. Therefore, even if all the parameters of these layers were removed, the compression would not be significant.

Thirdly, TTQ and aTTQ achieve a similar classification performance compared to the FP model. Indeed, for the HITS dataset, we observe a maximum MCC decrease compared to the FP model of 3.70% and 3.02% for aTTQ and TTQ, respectively. On the contrary, for the 1D CNN-transformer models on the ESR dataset, we observe an increase in MCC of 1.01% and 1.92% for aTTQ and TTQ, respectively. This can be explained by three factors. The first factor is that sparsity can act as a regularization, as several papers have shown [14]. The second factor is that quantization can also help regularization, since deep neural networks are highly over-parametrized. Finally, the quantized models are obtained from pre-trained FP models, and the choice of layers to be quantized is based on a Hessian-based quantization sensitivity metric. Therefore, the fine-tuning quantization step could help the models to get closer to a local minimum, as the loss landscape should be relatively flat for the chosen layers to be quantized.

Thirdly, we can observe that the improvement of the compression rate with aTTQ comes at the cost of a drop in classification performance. However, for most datasets and models, this performance loss is of the same order as that of TTQ. Globally, TTQ slightly outperforms

Table 1: Training parameters for the different models based on the dataset and the ternarization method used. In the last column, we indicate the percentage of weights of the model that will be quantized, selected using the Hessian-based metric introduced in [24].

Dataset	Model	Quant. method	t_{min}	t_{max}	Learning rate	Epochs	No. params.	% weights to quantize
HITS	2D CNN	FP	-	-	10^{-3}	50	1 681 923	-
		TTQ [16]/DoReFa [21]	-	-	3×10^{-3}	50		92.05
		aTTQ	-4	0	10^{-4}	150		
	1D CNN-trans.	FP	-	-	7×10^{-2}	150	766 271	-
		TTQ [16]/DoReFa [21]	-	-	10^{-4}	50		14.97
		aTTQ	-2	1.5	5×10^{-5}	100		
ESR	2D CNN	FP	-	-	10^{-3}	100	1 555 842	-
		TTQ [16]/DoReFa [21]	-	-	10^{-3}	50		
		aTTQ	-3	1	10^{-3}	200		99.51
	1D CNN-trans.	FP	-	-	3×10^{-1}	100	109 942	-
		TTQ [16]/DoReFa [21]	-	-	10^{-3}	100		
		aTTQ	-2	1	5×10^{-4}	100		24.22
MNIST	2D MNIST CNN	FP	-	-	10^{-3}	70	9 840	-
		TTQ [16]/DoReFa [21]	-	-	10^{-4}	200		53.35
		aTTQ	-1	0.5	10^{-3}	200		

aTTQ in terms of classification performance on almost all datasets and for almost all models, with an MCC margin increasing from 0.68% to 2.77%. This is not the case for the 2D CNN in the MNIST dataset, where aTTQ outperforms TTQ by an MCC margin of 1.53%. Therefore, aTTQ offers a better trade-off between classification and compression performance. Indeed, in our approach we carefully remove weights that are far from the mean for each layer, and this is done in an asymmetric way for weights on either side of the mean. On the contrary, TTQ removes weights by keeping values relatively close to the maximum of the absolute value (up to a scaling factor), and this is done in a symmetric way. However, in some cases, the maximum absolute value may be an outlier in the distribution of weights. In addition, positive and negative weights are treated equally, although there is no reason for them to have the same importance for the final classification, and this may penalize the sparsity of the quantized weights. On the other hand, in some cases a slight reduction in classification performance may be justified by lighter models. Good examples are (medical) embedded applications, where if the best model does not fit the device, the good (not the best) classification performance is appropriate. Our extreme quantization method allows us to achieve trade-offs that are compatible with real (clinical) applications, compared to other methods such as TTQ or higher bit-width quantization methods such as 8, 16 or 32-bit (full precision) quantization. In addition, compared to more aggressive pruning strategies such as binarization, ternarization makes it possible to reduce the drop in classification performance while maintaining good compression rates, as shown in [16]. This is an important point, because with the compression comes an acceleration of the inference (similar to the sparsity rate if we ignore the zero operations with specific hardware), and in our application of interest (HITS classification) the recordings can last from 30 to 180 minutes, generating thousands of HITS (on average 14 per minute) that must be analyzed quickly to be compatible with clinical practice.

4.7. Experiment 2: Influence of t_{min} and t_{max}

The objective of this experiment is twofold: (1) to show the interest of using asymmetric thresholds, and (2) to study the influence of t_{min} and t_{max} on the performance of the models. To do this, we trained the 1D CNN-transformer model from the same experiment on the ESR dataset, varying the values of x and y in $\{-2, -1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2\}$. The results are shown in Figure 5. The results are displayed in Figure 5.

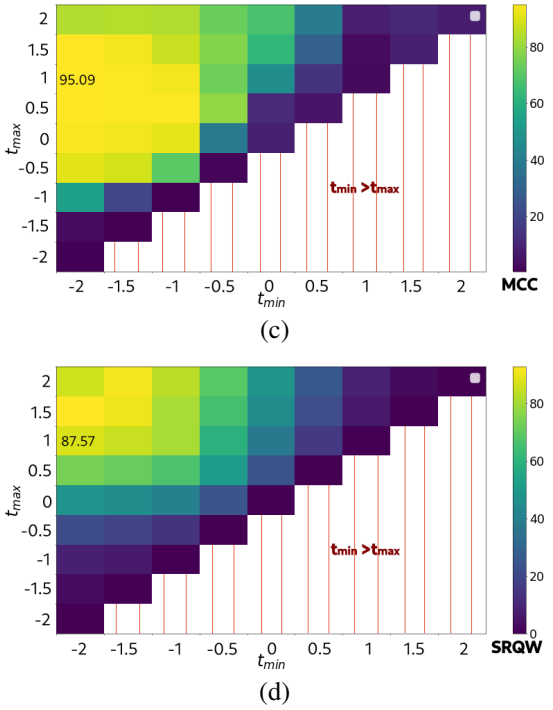

Fig. 5: Results of Experiment 2. (a) MCC for 1D CNN-transformer model trained on the ESR dataset. (b) SRQW for the 1D CNN-transformer model trained on the ESR dataset. The x-axis corresponds to the different tested values of t_{min} and the y-axis corresponds to the different values of t_{max}. All the values are given in %.

First, we can observe that the best classification performance is not obtained for symmetric values of t_{min} and t_{max}. In fact, the best results are obtained when $t_{min} \leq 0$ and $t_{max} \geq 0$, where 95.09% MCC is obtained for the 1D CNN-transformer on the ESR dataset. Furthermore, the best SRQW values are also obtained for the same ranges of (asym-

Table 2: Results of experiment 1, in %. FP corresponds to the full-precision model where no quantization was performed. ΔMCC corresponds to the difference between the MCC of the full-precision 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 3.2.

Dataset	Model	Quant. method	$CR_G^T \uparrow$	$CR_G^Q \uparrow$	$SRQW \uparrow$	$\mathbf{EC_G^T} \uparrow$	MCC ↑	Δ MCC ↑
HITS	2D CNN	FP	-	-	-	-	89.84 ± 3.09	-
		DoReFa [21]	**89.18 ± 0**	**96.87 ± 0**	-	3.54 ± 0	85.05 ± 5.96	−4.79
		TTQ [16]	24.96 ± 2.25	27.12 ± 2.44	28.96 ± 2.12	*23.42 ± 1.30*	**86.82 ± 2.29**	**−3.02**
		aTTQ	*42.98 ± 0.23*	*46.69 ± 0.25*	**45.95 ± 0.21**	**44.04 ± 0.19**	*86.14 ± 3.37*	*−3.70*
	1D CNN-trans.	FP	-	-	-	-	82.64 ± 1.77	-
		DoReFa [21]	**14.50 ± 0**	**96.87 ± 0**	-	0.37 ± 0.03	**84.07 ± 3.11**	**+1.43**
		TTQ [16]	0.14 ± 0.04	0.91 ± 0.27	6.75 ± 0.26	*1.88 ± 0.03*	*83.22 ± 2.36*	*+0.58*
		aTTQ	*13.94 ± 0.02*	*93.17 ± 0.16*	**93.53 ± 0.15**	**7.64 ± 0.11**	81.66 ± 4.17	−0.98
ESR	2D CNN	FP	-	-	-	-	92.81 ± 3.53	-
		DoReFa [21]	**96.40 ± 0**	**96.87 ± 0**	-	29.90 ± 0	*94.12 ± 0.87*	*+1.31*
		TTQ [16]	85.61 ± 1.37	86.03 ± 1.37	86.59 ± 1.29	*76.45 ± 1.13*	**95.00 ± 1.11**	**+2.19**
		aTTQ	*88.48 ± 0.44*	*88.91 ± 0.45*	**89.30 ± 0.42**	**84.49 ± 0.33**	92.41 ± 2.22	−0.40
	1D CNN-trans.	FP	-	-	-	-	94.33 ± 1.51	-
		DoReFa [21]	**23.46 ± 0**	**96.86 ± 0**	-	0.90 ± 0	**96.79 ± 0.55**	**+2.46**
		TTQ [16]	11.40 ± 2.61	47.07 ± 10.79	50.22 ± 10.16	*3.21 ± 0.66*	*96.25 ± 0.79*	*+1.92*
		aTTQ	*21.02 ± 0.15*	*86.78 ± 0.63*	**87.59 ± 0.59**	**5.37 ± 0.04**	95.34 ± 0.79	+1.01
MNIST	2D MNIST CNN	FP	-	-	-	-	94.39 ± 0.46	-
		DoReFa [21]	**51.67 ± 0**	**96.84 ± 0**	-	*3.28 ± 0*	87.03 ± 7.14	−7.36
		TTQ [16]	13.86 ± 2.33	25.97 ± 4.37	30.40 ± 4.12	2.58 ± 0.35	*92.09 ± 0.89*	*−2.30*
		aTTQ	*28.98 ± 1.26*	*54.32 ± 2.36*	**57.08 ± 2.22**	**4.97 ± 0.22**	**93.62 ± 0.96**	**−0.77**

metric) values of t_{min} and t_{max}. This can be explained by the fact that within a neural network, positive and negative values of the weights do not necessarily have the same impact on the final classification performance.

Moreover, we observe a trade-off between compression and classification performance, as the best performing models in terms of MCC are not the ones with the higher SRQW (and thus the higher compression rate). In terms of sparsity, the higher the gap between t_{min} and t_{max}, the higher the sparsity (larger range of values of weights mapped to zero). If this gap is large enough, the classification performance is often worse than with smaller gaps (which translates into lower sparsity rates). In fact, very high sparsity rates tend to decrease classification performance. However, this decrease is not always significant, whereas the gain in sparsity rate of the quantized layers is. Therefore, depending on the application, if memory requirements are an important factor, higher sparsity rates could be chosen despite the decrease in classification performance. In our case, we decided to choose the models that gave the higher classification performance without considering the sparsity rates. This is an advantage of our method, since the two hyperparameters t_{min} and t_{max} allow us to control this trade-off between compression and classification performance.

Similarly, tuning t_{min} and t_{max} can help to better adapt to new domains and tasks. Indeed, our method offers great flexibility thanks to the parameterization of the ternarization heuristic based on the statistics of the weights. The results show that for different datasets and models, the behavior of t_{min} and t_{max} is similar, but the values that allow to maximize the classification performance are not the same. This can be explained by the fact that, based on the model and the dataset, the important weights with respect to the mean are not the same; in some cases a larger dispersion is required (e.g. HITS 2D CNN), and in others a smaller dispersion is required (e.g. MNIST 2D CNN). Therefore, we recommend tuning t_{min} and t_{max} according to the desired objective: For higher compression rates[3] the user should choose t_{min} and t_{max} with a large gap, while for better classification performance the user should reduce this gap.

Finally, it is interesting to note that small sparsity rates (small gap between t_{min} and t_{max}) do not always yield models with the highest classification performance. In fact, we observe that a sparsity rate close to 0% tends to yield models with worse classification performance than higher sparsity rates. This highlights the regularization effect of pruning: as the sparsity rate of the quantized weights increases, the classification performance tends to increase up to a certain point.

4.8. Limitations

Although our approach strikes a good balance between compression, energy consumption and classification performance, we acknowledge certain limitations.

First, in this work, certain layers and parameters were not quantized, leaving room for further compression. However, full quantization may significantly reduce classification performance. Mixed quantization approaches, like those in [38, 24, 39], can mitigate this issue.

Second, although we tested our approach with multiple datasets and models, further experiments on additional models, compression methods, and datasets are needed to highlight the generalizability and significance of our approach.

Third, in our approach, the hyperparameters t_min and t_max which control the sparsity rate, have to be carefully chosen based on the application. This could be avoided by learning them, e.g. using for instance a differentiable threshold function as in [29].

Lastly, although our quantization approach offers compelling compression, energy, and computational properties, highlighting them in practice is not straightforward. To fully leverage our method, specialized hardware must be developed to exploit sparse operations and low bit-width operations, in conjunction with high bit-width (32 or 64 bits) operations.

[3] This is often associated with reduced energy consumption and inference time.

5. Conclusion

In this paper, we proposed to modify the quantization heuristic of trained ternary quantization (TTQ) in order to improve the trade-off between compression and classification performance. In fact, instead of using symmetric thresholds for the positive and negative weights to quantize, we proposed the use of asymmetric thresholds computed with the statistics of the weights (mean and standard deviation) and two hyperparameters, t_{min} and t_{max}, which control the sparsity rate of the quantized weights. Extensive experiments on three datasets and two types of models demonstrate the effectiveness of our method, which is able to improve the compression and energy performance while maintaining similar classification results to TTQ.

In future work, we plan to develop specialized hardware to efficiently perform the operations required by our quantized models, allowing us to accelerate inference and reduce energy consumption in practice.

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).

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] 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.

[5] 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.

[6] F. Piccialli, V. D. Somma, F. Giampaolo, S. Cuomo, G. Fortino, A survey on deep learning in medicine: Why, how and when?, Information Fusion 66 (2021) 111–137. doi:https://doi.org/10.1016/j.inffus.2020.09.006.

[7] V. Feigin, E. Nichols, T. Alam, M. Bannick, E. Beghi, N. Blake, W. Culpepper, E. Dorsey, A. Elbaz, R. Ellenbogen, J. Fisher, C. Fitzmaurice, G. Giussani, L. Glennie, S. James, C. Johnson, N. Kassebaum, G. Logroscino, B. Marin, T. Vos, Global, regional, and national burden of neurological disorders, 1990-2016: a systematic analysis for the global burden of disease study 2016, The Lancet Neurology 18 (2019) 459–480. doi:10.1016/S1474-4422(18)30499-X.

[8] W. H. Organization, Neurological disorders: public health challenges, World Health Organization, 2006.

[9] Y. Vindas, B. K. Guépié, M. Almar, E. Roux, P. Delachartre, Semi-automatic data annotation based on feature-space projection and local quality metrics: an application to cerebral emboli characterization, Medical Image Analysis (2022) 102437doi:https://doi.org/10.1016/j.media.2022.102437.

[10] G. Xu, T. Ren, Y. Chen, W. Che, A one-dimensional cnn-lstm model for epileptic seizure recognition using eeg signal analysis, Frontiers in Neuroscience 14 (2020).

[11] A. M. Hilal, A. A. Albraikan, S. Dhahbi, M. K. Nour, A. Mohamed, A. Motwakel, A. S. Zamani, M. Rizwanullah, Intelligent epileptic seizure detection and classification model using optimal deep canonical sparse autoencoder, Biology 11 (8) (2022).

[12] 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.

[13] 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.

[14] 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).

[15] 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.

[16] C. Zhu, S. Han, H. Mao, W. J. Dally, Trained ternary quantization, in: International Conference on Learning Representations, 2017.

[17] 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.

[18] A. Trusov, E. Limonova, D. Nikolaev, V. V. Arlazarov, Fast matrix multiplication for binary and ternary cnns on arm cpu, in: 2022 26th International Conference on Pattern Recognition (ICPR), IEEE Computer Society, Los Alamitos, CA, USA, 2022, pp. 3176–3182. doi:10.1109/ICPR56361.2022.9956533.

[19] Y. Gong, L. Liu, M. Yang, L. D. Bourdev, Compressing deep convolutional networks using vector quantization (2014).

[20] H. Kim, M. U. K. Khan, C.-M. Kyung, Efficient neural network compression, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 12561–12569. doi:10.1109/CVPR.2019.01285.

[21] 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.
URL http://arxiv.org/abs/1606.06160

[22] K. Ullrich, E. Meeds, M. Welling, Soft weight-sharing for neural network compression, in: International Conference on Learning Representations, 2017.

[23] 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.

[24] Z. Dong, Z. Yao, D. Arfeen, A. Gholami, M. W. Mahoney, K. Keutzer, Hawq-v2: Hessian aware trace-weighted quantization of neural networks, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, Vol. 33, Curran Associates, Inc., 2020, pp. 18518–18529.

[25] A. Gondimalla, N. Chesnut, M. Thottethodi, T. N. Vijaykumar, Sparten: A sparse tensor accelerator for convolutional neural networks, in: Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO '52, Association for Computing Machinery, New York, NY, USA, 2019, p. 151–165. doi:10.1145/3352460.3358291.

[26] J.-H. Luo, J. Wu, W. Lin, Thinet: A filter level pruning method for deep neural network compression, 2017, pp. 5068–5076. doi:10.1109/ICCV.2017.541.

[27] 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.

[28] 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.

[29] F. Manessi, A. Rozza, S. Bianco, P. Napoletano, R. Schettini, Automated pruning for deep neural network compression (12 2017).

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] Y. LeCun, C. Cortes, MNIST handwritten digit database (2010).

[35] 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.

[36] G. King, L. Zeng, Logistic regression in rare events data, Political Analysis 9 (2001) 137–163.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[38] 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.

[39] 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.