Classification Error Approximation of a Compressed Linear Softmax Layer

Diana Resmerita
Université Côte d'Azur, CNRS, I3S and Kalray, France

Rodrigo Cabral Farias and Lionel Fillatre
Université Côte d'Azur, CNRS, I3S
France

Benoît Dupont de Dinechin
Kalray
France

***Abstract*—Deep neural networks need to be compressed due to their high memory requirements and computational complexity. Previous papers have proposed numerous methods to quantize the weights with a single bit or more. However, the loss of accuracy involved in the compression is scarcely studied from a theoretical point of view. Motivated by the rate-distortion theory, we propose a new distortion measure which assesses the gap between the Bayes risk of a classifier before and after the compression. Since this distortion is not easily tractable, we derive a theoretical approximation when the last fully connected layer of a deep neural network is compressed under the assumption that the layer inputs follow a multivariate normal distribution. Numerical results show that the approximation performs well on both synthetic and real data. We also show that heuristic quantizers proposed in the literature may not be optimal.**

I. Introduction

Deep Neural Networks (DNNs) are used in many computer vision applications such as object recognition, object detection and segmentation [1]. Unfortunately, the outstanding performance of these models comes at the expense of a highly complex architecture, requiring large amounts of memory, processing power and energy. This makes them difficult to deploy on embedded systems (*e.g.* autonomous vehicles).

To deal with these issues, several approaches have been proposed to reduce the size of the network with little to no loss in accuracy. New architectures with a smaller number of parameters have shown good results [2], [3]. Other approaches focus on reducing large neural networks with compression methods by pruning [4], [5] and quantizing [6]–[11] their weights or by efficiently computing their low-rank tensor approximation [12]. In some works, these methods are applied during training [9], while in others, the compression is applied post-training [13]. The majority of works deals with deterministic methods. Stochastic methods [14] are harder to implement, as mentioned in [8]. In the case of binary quantization and uniform 8-bit quantization, state-of-the-art compression methods for image classification, like XNOR-NET [9] and TensorFlow Lite (TFLite) [10], [11], are proposed with rules for choosing adequate quantization parameters. What perturbation can we expect for a given model? And, furthermore, what affects the loss of accuracy in a model? When compressing a model, we are interested in obtaining a good trade-off between the number of bits used to represent the weights and the accuracy loss, viewed as a distortion measure, between the original and the compressed model. This fundamental problem has already been handled in rate-distortion theory [15] for data compression. Additionally, a number of works [14], [16], [17] emerged on the subject of neural network sensibility and error analysis for different purposes such as optimization, generalization, robustness and, of course, compression. We take inspiration from the rate distortion theory and state-of-the-art approaches to propose a theoretical approximation of the accuracy loss when the last fully connected layer of a deep neural network is compressed. We simplify the study by assuming that the inputs are drawn from Gaussian distributions. Through simulations with synthetic data and with a real classification dataset, Sonar [18], we show that when the perturbations are due to compression, either with 1-bit, 3-bit or 8-bit, quantization parameters minimizing the increase in the classification risk may not correspond to those given by XNOR-NET and TFLite.

The outline of this paper is the following. Section II presents the neural networks architecture, our working assumptions and the distortion measure we seek to minimize for neural network compression. Section III proposes an approximation of the distortion measure. Section IV shows the quality of our distortion approximation on synthetic and real data. Finally, in Section V, we conclude and present our future work.

II. Problem statement

A. Deep neural networks

We consider a classification problem between two classes C_0 and C_1. We process a couple $(\mathbf{x}, \mathbf{y})$ where $\mathbf{x} \in \mathbb{R}^n$ represents the input and $\mathbf{y} = (y_0, y_1) \in \mathbb{R}^2$ is the true label. The label satisfies $y_0 > y_1$ when the true class is C_0 and $y_0 \leq y_1$ when the true class is C_1. This corresponds to the usual one-hot encoding of a binary label [1]. To decode the label, we use the decoding rule:

$$\delta(\mathbf{y}) = \underset{i \in \{0,1\}}{\arg\max}\, y_i. \tag{1}$$

Let us consider a deep neural network $f(\mathbf{x})$ composed of $K+1$ layers with $\mathbf{f}_0$ being the input layer $\mathbf{f}_0(\mathbf{x}) = \mathbf{x} \in \mathbb{R}^{n_0}$ where $n_0 = n$. The hidden layers are from $\mathbf{f}_1$ to $\mathbf{f}_{K-1}$ and the output layer is denoted with $\mathbf{f}_K$. We define the model as follows

$$\mathbf{f}_k(\mathbf{x}) = \sigma(\mathbf{W}_k \mathbf{f}_{k-1}(\mathbf{x}) + \mathbf{b}_k),\ 1 \leq k < K, \tag{2}$$

$$\hat{\mathbf{y}} = f(\mathbf{x}) = \mathbf{f}_K(\mathbf{x}) = \text{softmax}(\mathbf{W}\mathbf{f}_{K-1}(\mathbf{x}) + \mathbf{b}), \tag{3}$$

where $\mathbf{W}_k \in \mathbb{R}^{n_k \times n_{k-1}}$, $\mathbf{b}_k \in \mathbb{R}^{n_k}$ and $\sigma(\cdot)$ is a nonlinear activation function (typically, the ReLU function). The last layer, called the linear softmax layer, depends on $\mathbf{W} \in \mathbb{R}^{2 \times n_{K-1}}$ and $\mathbf{b} \in \mathbb{R}^2$. The output of the neural network $\hat{\mathbf{y}} = (\hat{y}_0, \hat{y}_1)$ is interpreted as a soft one-hot encoding vector. The decision rule, denoted $\delta_f(\mathbf{x})$, is

$$\delta_f(\mathbf{x}) = \delta(f(\mathbf{x})) = \delta(\hat{\mathbf{y}}), \tag{4}$$

and chooses the most probable component of $\hat{\mathbf{y}}$. The classification performance of a neural network is measured with the Bayes risk function $r(\delta_f)$:

$$r(\delta_f) = \mathbb{P}(\delta_f(\mathbf{x}) \neq \delta(\mathbf{y})) \tag{5}$$

where $\mathbb{P}(\cdot)$ stands for the joint probability measure of the couple $(\mathbf{x}, \mathbf{y})$. The Bayes risk is equal to $1 - \mathrm{acc}(\delta_f)$ where $\mathrm{acc}(\delta_f)$ is the usual accuracy of the neural network.

B. Assumption on the distribution of the last hidden layer

It has been shown in the literature that the first layers of deep neural networks are regularizing the input data such that the probability distribution becomes more similar to a normal distribution in the latest layers [13], [19]. In this paper, we consider that the vector $\mathbf{f}_{K-1}$ follows a multivariate normal distribution. In the case of two classes, we assume that

$$\mathbf{f}_{K-1} \sim \mathcal{N}(\mu_j, \Sigma_j) \text{ under } C_j, \tag{6}$$

where μ_j is a known mean vector and Σ_j is a known strictly positive definitive covariance matrix. We are aware that the distribution of weights and inputs depend on many different factors: the architecture of the network, initialization, training. These assumptions have been made to simplify the theoretical model. We have experimented and analyzed various architectures, especially the Fully Connected and Convolutional Neural Networks [20], [21]. For these networks, we found that both the weights and inputs tend towards a Gaussian distribution. To support this assumption, [19] shows the weight distribution and the density function curve of the corresponding Gaussian distribution for each layer in ResNet-18b. A normality test has also been conducted and the results indicate a good normal distribution fitting. Both [16] and [17] assume that the inputs and weights are drawn from a Gaussian distribution. In practice, the distribution is not always Gaussian. Fortunately, in the case of the softmax linear layer, the central limit theorem may ensure the Gaussian distribution.

C. Neural network compression

Consider an already trained neural network (3). Let $\mathbf{U}$ be the compressed form of $\mathbf{W}$. We assume that the weights of the matrix $\mathbf{W}$ are independent and identically distributed. Under this setting, classical rate-distortion theory [15, p. 301] describes the minimum transmission bit-rate R required for transmitting the weights $\mathbf{U}$ instead of $\mathbf{W}$ under a constraint on the maximum distortion D between them. Common distortion functions D correspond to measures of the gap between individual elements of $\mathbf{W}$ and $\mathbf{U}$. Note that the goal we pursue in the analysis of the effects of compression in deep neural networks is rather different than the one in the setting above. We are not interested simply in reconstructing $\mathbf{W}$ from $\mathbf{U}$, but in obtaining $\mathbf{U}$ such that the neural network accuracy will not change much. As a consequence, our distortion measure is defined over all the values $(\mathbf{W}, \mathbf{U})$ as follows.

Definition II.1. A distortion measure is a mapping

$$d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^+, \quad (\mathbf{W}, \mathbf{U}) \mapsto d(\mathbf{W}, \mathbf{U}). \tag{7}$$

It is a measure of the change in accuracy $d(\mathbf{W}, \mathbf{U})$ induced by representing the weights $\mathbf{W}$ by $\mathbf{U}$.

In [22], the rate-distortion theory is used to analyze the approximation of the posterior function involved in the Bayes classifier, not the accuracy loss. This theory is also used in [16] to analyze the Kullback-Leibler (KL) divergence between the classifier before and after the compression. However, as shown in [23], the KL divergence does not always reflect the accuracy loss when compressing a model. For this reason, we define a particular distortion measure based on the classification risk.

To distinguish the network before and after the compression, we use the notation $f_{\mathbf{W}}$ and, respectively $f_{\mathbf{U}}$, for the uncompressed, resp. compressed, neural network. We are not interested in how $\mathbf{U}$ was produced from $\mathbf{W}$, but we want to measure the gap between the risks $r(\delta_{f_{\mathbf{W}}})$ and $r(\delta_{f_{\mathbf{U}}})$. Hence, we consider the distortion function to be the absolute difference between the risks of the two classifiers:

$$d(\mathbf{W}, \mathbf{U}) = |r(\delta_{f_{\mathbf{W}}}) - r(\delta_{f_{\mathbf{U}}})|. \tag{8}$$

We are seeking the optimal number of bits that allows us to minimize the distortion error. Hence, this paper focuses on the study of the distortion measure (8). This will bring us closer to understanding the impact of compression methods on the last layer of a neural network and determine the minimal number of bits needed to ensure the quality of the classification.

III. Theoretical Analysis of the Distortion

Following [24], the Bayes risk (5) for δ_f is rewritten as

$$r(\delta_f) = \pi_0 \mathbb{P}_0(\delta_f(\mathbf{x}) = 1) + \pi_1 \mathbb{P}_1(\delta_f(\mathbf{x}) = 0), \tag{9}$$

where π_j is the known prior probability of class C_j and $R_j(\delta_f)$ is the conditional risk (knowing the true class) given by

$$R_j(\delta_f) = \mathbb{P}_j(\delta_f(\mathbf{x}) \neq j), \tag{10}$$

where $\mathbb{P}_j(\cdot)$ stands for the normal distribution $\mathcal{N}(\mu_j, \Sigma_j)$.

A. Compressed and uncompressed classifiers

Let us note first that we can rewrite $\delta_{f_{\mathbf{W}}}$ as a linear classifier without the operators argmax and softmax. The decision rule (4) is equivalent to the linear decision rule:

$$\delta_{f_{\mathbf{W}}}(\mathbf{f}_{K-1}) = \begin{cases} 0 & \text{if } \tilde{\mathbf{w}}^T \mathbf{f}_{K-1} > \lambda, \\ 1 & \text{otherwise,} \end{cases} \tag{11}$$

where $\tilde{\mathbf{w}} = \mathbf{w}_0 - \mathbf{w}_1$, $\lambda = b_1 - b_0$ and $\tilde{\mathbf{w}}^T$ denotes the transpose of $\tilde{\mathbf{w}}$. Note that $\mathbf{w}_0$ and $\mathbf{w}_1$ represent the first and the second row of $\mathbf{W}$ and b_0, b_1 are the two components of the bias vector $\mathbf{b}$. To simplify the notations, the vector $\mathbf{f}_{K-1}$

will be denoted $\mathbf{f}$ in the rest of the paper. We want to compare this classifier with the compressed version

$$\delta_{f_{\mathbf{U}}}(\mathbf{f}) = \begin{cases} 0 & \text{if } \tilde{\mathbf{u}}^T\mathbf{f} > \lambda, \\ 1 & \text{otherwise,} \end{cases} \tag{12}$$

where $\tilde{\mathbf{u}} = \mathbf{u}_0 - \mathbf{u}_1$, $\mathbf{u}_0$ and $\mathbf{u}_1$ represent the first and the second rows of $\mathbf{U}$. To simplify the study, we do not quantify $\mathbf{b}$ but our approach can be easily extended to this case.

Since $\mathbf{f}$ follows a Gaussian distribution, $\tilde{\mathbf{w}}^T\mathbf{f}$ also follows a Gaussian distribution

$$\tilde{\mathbf{w}}^T\mathbf{f} \sim \mathcal{N}(\tilde{\mathbf{w}}^T\mu_j, \tilde{\mathbf{w}}^T\Sigma_j\tilde{\mathbf{w}}). \tag{13}$$

Let $\Phi(\cdot)$ be the cumulative distribution function of the standard normal distribution and $\phi(\cdot)$ its probability density function. The risk $r(\delta_{f_{\mathbf{W}}})$ can be calculated from (9) by noting that

$$R_0(\delta_{f_{\mathbf{W}}}) = \Phi\left(\frac{\tilde{\mathbf{w}}^T\mu_0 - \lambda}{\sqrt{\tilde{\mathbf{w}}^T\Sigma_0\tilde{\mathbf{w}}}}\right), \tag{14}$$

$$R_1(\delta_{f_{\mathbf{W}}}) = 1 - \Phi\left(\frac{\tilde{\mathbf{w}}^T\mu_1 - \lambda}{\sqrt{\tilde{\mathbf{w}}^T\Sigma_1\tilde{\mathbf{w}}}}\right). \tag{15}$$

The calculation of (14)-(15) comes from the error analysis of a linear classifier as detailed in [24]. Similar results are obtained for the classifier with the compressed weights $\mathbf{U}$.

B. Distortion measure for classifiers

Using the risk from (9), we can rewrite (8) as follows

$$d(\mathbf{W},\mathbf{U}) = \left| \sum_{i=0}^{1} \pi_i [\mathbb{P}_i(f_{\mathbf{W}}(\mathbf{x}) > \lambda) - \mathbb{P}_i(f_{\mathbf{U}}(\mathbf{x}) > \lambda)] \right|. \tag{16}$$

The distortion $d(\mathbf{W},\mathbf{U})$ can be easily computed by using (14)-(15). However, a numerical computation does not offer any information on the joint role of $\mathbf{W}$ and $\mathbf{U}$. In other words, we do not gain any insight into the quality of the compression process applied to $\mathbf{W}$ in order to obtain $\mathbf{U}$. For this reason, we propose a bound which joins together $\mathbf{W}$ and $\mathbf{U}$.

Let us define the conditional gap

$$d_i(\mathbf{W},\mathbf{U}) = \mid \mathbb{P}_i(f_{\mathbf{W}}(\mathbf{x}) > \lambda) - \mathbb{P}_i(f_{\mathbf{U}}(\mathbf{x}) > \lambda) \mid \tag{17}$$

as the gap between the probability errors conditioned by the class C_i. It follows that

$$d(\mathbf{W},\mathbf{U}) \leq \pi_0 d_0(\mathbf{W},\mathbf{U}) + \pi_1 d_1(\mathbf{W},\mathbf{U}). \tag{18}$$

Using [25, Chap. 1, inequality 1.3.c], we get that

$$\begin{aligned} d_i(\mathbf{W},\mathbf{U}) \quad \leq \quad & \mathbb{P}_i(f_{\mathbf{W}}(\mathbf{x}) > \lambda, f_{\mathbf{U}}(\mathbf{x}) \leq \lambda) \\ & + \mathbb{P}_i(f_{\mathbf{W}}(\mathbf{x}) \leq \lambda, f_{\mathbf{U}}(\mathbf{x}) > \lambda). \end{aligned} \tag{19}$$

Hence, the conditional gap is bounded by the sum of the probabilities when the classifiers disagree. It is equivalent to

$$\begin{aligned} d_i(\mathbf{W},\mathbf{U}) \quad \leq \quad & \mathbb{P}_i(\tilde{\mathbf{w}}^T\mathbf{f} > \lambda, \tilde{\mathbf{u}}^T\mathbf{f} \leq \lambda) \\ & + \mathbb{P}_i(\tilde{\mathbf{w}}^T\mathbf{f} \leq \lambda, \tilde{\mathbf{u}}^T\mathbf{f} > \lambda). \end{aligned} \tag{20}$$

This form is simpler because it involves the couple of variables $(\tilde{\mathbf{w}}^T\mathbf{f}, \tilde{\mathbf{u}}^T\mathbf{f})$ that follows a bivariate normal distribution with a non-zero correlation coefficient. The distribution of this random couple is studied in the following lemma.

Lemma 1. Let $i \in \{0,1\}$. Then, we have the equalities

$$\mathbb{P}_i(\tilde{\mathbf{w}}^T\mathbf{f} > \lambda, \tilde{\mathbf{u}}^T\mathbf{f} \leq \lambda) = \mathbb{P}_i(X > \alpha_{i,\mathbf{W}}, Y \leq \alpha_{i,\mathbf{U}}), \tag{21}$$

$$\mathbb{P}_i(\tilde{\mathbf{w}}^T\mathbf{f} \leq \lambda, \tilde{\mathbf{u}}^T\mathbf{f} > \lambda) = \mathbb{P}_i(X \leq \alpha_{i,\mathbf{W}}, Y > \alpha_{i,\mathbf{U}}), \tag{22}$$

where X and Y denote two standard normal variables with correlation coefficient ϱ_i such as

$$\alpha_{i,\mathbf{W}} = \frac{\lambda - \tilde{\mathbf{w}}^T\mu_i}{\sqrt{\tilde{\mathbf{w}}^T\Sigma_i\tilde{\mathbf{w}}}}, \quad \alpha_{i,\mathbf{U}} = \frac{\lambda - \tilde{\mathbf{u}}^T\mu_i}{\sqrt{\tilde{\mathbf{u}}^T\Sigma_i\tilde{\mathbf{u}}}}, \tag{23}$$

$$\varrho_i = \varrho(i,\mathbf{W},\mathbf{U}) = \frac{\tilde{\mathbf{w}}^T\Sigma_i\tilde{\mathbf{u}}}{\sqrt{\tilde{\mathbf{w}}^T\Sigma_i\tilde{\mathbf{w}}}\sqrt{\tilde{\mathbf{u}}^T\Sigma_i\tilde{\mathbf{u}}}}. \tag{24}$$

Proof. We normalize the component $\tilde{\mathbf{w}}^T\mathbf{f}$ by removing its mean and dividing it by its standard deviation given in (13). We do the same for the component $\tilde{\mathbf{u}}^T\mathbf{f}$. A short calculation yields the correlation coefficient. □

It is well known that the bivariate normal distribution is not easy to compute except for some very specific cases. Fortunately, some accurate approximations exist. In this paper, we use the simple approximation given in [26] which is easy to interpret. By applying this approximation to (21), we get

$$\mathbb{P}_i(X > \alpha_{i,\mathbf{W}}, Y \leq \alpha_{i,\mathbf{U}}) \approx \Phi(-\alpha_{i,\mathbf{W}})\Phi\left(-\xi_{i,\mathbf{W},\mathbf{U}}\right), \tag{25}$$

$$\xi_{i,\mathbf{W},\mathbf{U}} = \frac{\varrho_i\, \mu(\alpha_{i,\mathbf{W}}) - \alpha_{i,\mathbf{U}}}{\sqrt{1-\varrho_i^2}},\ \mu(\alpha_{i,\mathbf{W}}) = \frac{\phi(\alpha_{i,\mathbf{W}})}{\Phi(-\alpha_{i,\mathbf{W}})}. \tag{26}$$

This approximation expresses the probability as a product of two simple terms. The first term depends only on $\mathbf{W}$ and is thus independent from the compression. The second term quantifies the dependency between $\mathbf{W}$ and its compressed form $\mathbf{U}$ through $\xi_{i,\mathbf{W},\mathbf{U}}$. We can do the same for the second inequality (22) in Lemma 1. Finally, we get an approximation $D_i(\mathbf{W},\mathbf{U})$ of the upper bound $d_i(\mathbf{W},\mathbf{U})$ in (20):

$$\begin{aligned} d_i(\mathbf{W},\mathbf{U}) &\leq \mathbb{P}_i(\tilde{\mathbf{w}}^T\mathbf{f} > \lambda, \tilde{\mathbf{u}}^T\mathbf{f} \leq \lambda) + \mathbb{P}_i(\tilde{\mathbf{w}}^T\mathbf{f} \leq \lambda, \tilde{\mathbf{u}}^T\mathbf{f} > \lambda) \\ &\approx \Phi(-\alpha_{i,\mathbf{W}})\Phi\left(-\xi_{i,\mathbf{W},\mathbf{U}}\right) + \Phi(-\alpha_{i,\mathbf{U}})\Phi\left(-\xi_{i,\mathbf{U},\mathbf{W}}\right) \\ &= D_i(\mathbf{W},\mathbf{U}), \end{aligned} \tag{27}$$

where $\xi_{i,\mathbf{U},\mathbf{W}}$ is similar to $\xi_{i,\mathbf{W},\mathbf{U}}$ provided that we swap the role of $\mathbf{W}$ and $\mathbf{U}$. Therefore, we get the following approximation $D(\mathbf{W},\mathbf{U})$ of $d(\mathbf{W},\mathbf{U})$:

$$d(\mathbf{W},\mathbf{U}) \approx \pi_0 D_0(\mathbf{W},\mathbf{U}) + \pi_1 D_1(\mathbf{W},\mathbf{U}) = D(\mathbf{W},\mathbf{U}). \tag{28}$$

This approximation is a closed form expression. Even if we cannot ensure that $D(\mathbf{W},\mathbf{U})$ is truly an upper bound, the advantage of using the approximation $D(\mathbf{W},\mathbf{U})$ over the true value $d(\mathbf{W},\mathbf{U})$ is to ease the interpretation of the effects of compression over the accuracy. By analyzing the expression of the approximation, one can note that its value mainly relies on three quantities: $\alpha_{i,\mathbf{W}}$ in (23), $\alpha_{i,\mathbf{U}}$ in (23) and the correlation ϱ_i between the compressed and uncompressed weights in (24). The constant $\alpha_{i,\mathbf{W}}$ depends on properties of the dataset (the means of the classes) and not on the

compressed network architecture. The value $\alpha_{i,\mathbf{U}}$ depends on the compressed network. Under some appropriate assumptions on the number of neurons and the compression bit-rate, $\alpha_{i,\mathbf{U}}$ can be approximated by $\alpha_{i,\mathbf{W}}$ weighted by a corrective term depending on ϱ_i. Under the same assumptions, the correlation ϱ_i can be approximated analytically as a function of the number of neurons of a layer and the compression bit-rate. Further details and a more in-depth analysis of these approximations in the case of uniform quantization are presented in [27].

IV. Experiments

Several experiments were carried out in order to analyze the proposed approximation $D(\mathbf{W}, \mathbf{U})$ in (28). The first experiment was done using a softmax classifier on synthetic data, while the second one was performed on a one-hidden-layer network trained on the Sonar dataset.

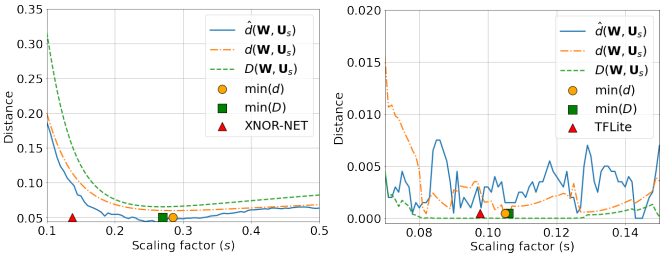
We used the standard binary and uniform quantization to compress the weights $\mathbf{W}$ into $\mathbf{U}_s$ where $\mathbf{U}_s$ underlines that the compressed weights $\mathbf{U}$ depend on a scaling factor $s > 0$ to tune. The binarization process produces $\mathbf{U}_s = s \cdot \text{sign}(\mathbf{W})$, where $\text{sign}(\mathbf{W})$ means that the element-wise sign function is applied to each element of $\mathbf{W}$. The uniform quantization produces $\mathbf{U}_s = \text{round}\left(\frac{\mathbf{W}}{s}\right)$, where the element-wise round$(\cdot)$ operation approximates its input with the closest integer.

We are looking for the best scaling factor s that minimizes $d(\mathbf{W}, \mathbf{U}_s)$. For both experimental settings, we iterated over s from 0 to 2 by a step of 10^{-3}. At each iteration, we compressed the weights with the methods mentioned above. When we use the synthetic data, we computed the theoretical distortion $d(\mathbf{W}, \mathbf{U}_s)$ and its approximation $D(\mathbf{W}, \mathbf{U}_s)$. For the real dataset, we cannot compute the theoretical distortion because we do not know the exact parameters of the assumed normal distribution. We estimated the values of μ_0, μ_1, σ_0 and σ_1 in (6) from the samples. Then, we computed $\tilde{d}(\mathbf{W}, \mathbf{U}_s)$ and $\tilde{D}(\mathbf{W}, \mathbf{U}_s)$ by replacing the true values μ_0, μ_1, σ_0 and σ_1 by their estimates in the definition of $d(\mathbf{W}, \mathbf{U}_s)$ and $D(\mathbf{W}, \mathbf{U}_s)$. In both datasets, we evaluated the empirical distortion $\hat{d}(\mathbf{W}, \mathbf{U}_s)$ by computing the empirical Bayes risks. The minimum of $d(\mathbf{W}, \mathbf{U}_s)$ with respect to s is denoted min(d). We use the same notation min$(\cdot)$ for the other distortions. We also take a look at the true Bayes risk $r(\delta_{f_{\mathbf{U}_s}})$ for the synthetic data and at the empirical risk $\hat{r}(\delta_{f_{\mathbf{U}_s}})$ for the real data.

Additionally, we compared our results to two state-of-the-art methods that have shown promising results, namely XNOR-NET [9] for binary quantization and Tensorflow Lite [10], [11] for uniform quantization:

XNOR-NET. XNOR-NET quantizes $\mathbf{W}$ by solving the optimization problem $\min ||\mathbf{W} - s\mathbf{W}_b||_2^2$ where $\mathbf{W}_b$ is a binary matrix and $s > 0$. The optimal solution $\mathbf{U}_{s_{\text{XNOR-NET}}}$ is the product of the optimal binary matrix $\mathbf{W}_b^* = \text{sign}(\mathbf{W})$ with the optimal scaling factor $s_{\text{XNOR-NET}} = ||\mathbf{W}||_1/n$ where $||\mathbf{W}||_p$ is the p-norm.

TFLite. The TFLite approach is based on the standard uniform scalar quantizer. The quantization of an entry w of $\mathbf{W}$ into a quantized value u of $\mathbf{U}_{s_{\text{TFLITE}}}$ proceeds as follows: $u = \text{round}(\frac{w}{s_{\text{TFLITE}}}) + z$, where the scaling factor is defined as $s_{\text{TFLITE}} = (w_{\max} - w_{\min})/N$ with $N = 2^R - 1$ and the parameter z represents the quantized value of the real value 0. This method does not involve any optimization.



(a) Binary quantization. (b) Uniform quantization (3 bits).

Fig. 1. Distortions $\hat{d}(\mathbf{W}, \mathbf{U}_s)$, $d(\mathbf{W}, \mathbf{U}_s)$ and $D(\mathbf{W}, \mathbf{U}_s)$ for synthetic data experiment with XNOR-NET (left) and TFLite (right) as a function of the scaling factor. The orange circle and the green square represent respectively the minimum of $d(\mathbf{W}, \mathbf{U}_s)$ and $D(\mathbf{W}, \mathbf{U}_s)$ and, in red, we show the scaling factors of XNOR-NET and TFLite.

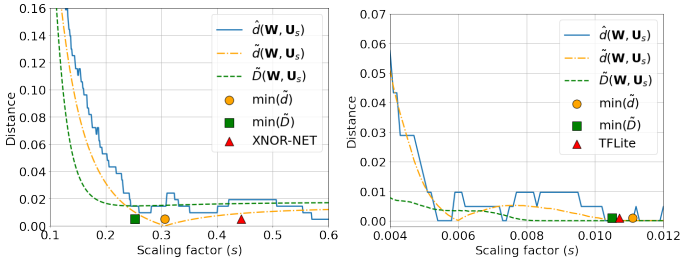
A. Softmax classifier on synthetic data

The experiments in this subsection were performed on synthetic data by employing a binary softmax classifier without any hidden layers. In order to train our model, we generated a two-class dataset with $N = 2 \times 10^3$ samples, 10^3 samples per class. Each sample of the generated data has $n = 10$ features and was drawn from a multivariate normal distribution with a given mean and covariance for each class. Following [28], means were drawn from a 10 dimensional sphere with radius of 1 for C_0 and 5 for C_1. The variance is considered spherical with the intensity 4 for C_0 and 2.25 for C_1. The training leads to a Bayes risk $r(\delta_{f_{\mathbf{W}}}) = 0.1152$. We quantized the weights of the trained model using XNOR-NET (binary weights) and TFLite with only 3 bits due to the small number of weights the model contains (20 values).

Fig. 1, on the left, shows the results with the binarization (XNOR-NET) and, on the right, the results for the uniform quantization (TFLite). It must be noted that $d(\mathbf{W}, \mathbf{U}_s)$ overlaps with $\hat{d}(\mathbf{W}, \mathbf{U}_s)$. We observe that $D(\mathbf{W}, \mathbf{U}_s)$ follows the same shape as the actual error. Our approximation is able to outperform XNOR-NET in terms of the obtained optimal scaling factor: $s_D = 0.267$ is reasonably close to the theoretical minimum $s_d = 0.285$ and better than $s_{\text{XNOR-NET}} = 0.139$. The Bayes risks for each scaling factor are the following: $r(\delta_{f_{\mathbf{U}_{s_d}}}) = 0.1751$, $r(\delta_{f_{\mathbf{U}_{s_D}}}) = 0.1752$ and $r(\delta_{f_{\mathbf{U}_{s_{\text{XNOR-NET}}}}}) = 0.2255$. On the other hand, TFLite proposes a scaling factor that is close to the optimal one, but our approximation is closer: $s_{\text{TFLITE}} = 0.0976$, $s_D = 0.106$ and $s_d = 0.105$. By looking at the risk, we see that our approximation is extremely close to the theoretical minimum giving a risk of 0.1153 and better than TFLite $r(\delta_{f_{\mathbf{U}_{s_{\text{TFLITE}}}}}) = 0.1175$.

B. One-hidden-layer ReLU neural network on Sonar

We also performed experiments on a one-hidden-layer neural network trained on the Sonar dataset [18]. It is composed of $N = 208$ instances, $n = 60$ attributes and two classes.



(a) Binary quantization. (b) Uniform quantization (8 bits).

Fig. 2. Distortions $\hat{d}(\mathbf{W}, \mathbf{U}_s)$, $\tilde{d}(\mathbf{W}, \mathbf{U}_s)$ and $\tilde{D}(\mathbf{W}, \mathbf{U}_s)$ for Sonar dataset with XNOR-NET (left) and TFLite (right) as a function of the scaling factor. The orange circle and the green square represent the minimum of $\tilde{d}(\mathbf{W}, \mathbf{U}_s)$ and $\tilde{D}(\mathbf{W}, \mathbf{U}_s)$ and the red triangles show the scaling factors of XNOR-NET and TFLite.

The network we trained had one fully connected ReLU layer with 60 neurons and a final softmax layer. After training, we obtained an empirical risk $\hat{r}(\delta_{f_{\mathbf{W}}}) = 0.0727$. We quantized the weights of the last layer using the same methods as in the previous subsection, except, for the uniform quantization where we used 8 bits.

In Fig. 2, on the left, we present the results obtained using binarization and, on the right, the results with uniform quantization. Although the 1-bit quantization performs slightly worse than the 8-bit quantization, both quantizations perform well. We observe that the XNOR-NET scaling factor $s_{\text{XNOR}-\text{NET}} = 0.4432$ is far from the estimated theoretical minimum $s_{\tilde{d}} = 0.3062$, while our approximation is closer $s_{\tilde{D}} = 0.2522$. XNOR-NET has a higher Bayes risk $\hat{r}(\delta_{f_{\mathbf{U}_{s_{\text{XNOR}-\text{NET}}}}}) = 0.0913$. Our approximation gives the same empirical risk as the one obtained with the estimated theoretical $\hat{r}(\delta_{f_{\mathbf{U}_{s_{\tilde{D}}}}}) = \hat{r}(\delta_{f_{\mathbf{U}_{s_{\tilde{d}}}}}) = 0.0865$, which are closer to the original risk. Using the uniform quantization, we observe that our approximation with $s_{\tilde{D}} = 0.0105$ is almost the same as TFLite $s_{\text{TFLITE}} = 0.0107$, both close to $s_{\hat{d}} = 0.0111$. The empirical risk values are all three at 0.0721. Although the normal assumption is not perfectly satisfied in the last layer (because of the ReLU), it is worth noting that our approximation still performs well.

V. Conclusion

In this paper, we propose a theoretical analysis of the accuracy loss of a deep neural network when the last softmax layer is compressed. We derive an accurate closed form approximation of this accuracy loss. Our future work will focus on highlighting the relationship between the quantization step and the error introduced by the approximation. We also plan to extend our analysis to a multiclass model.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[2] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size," 2016.

[3] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, 2017.

[4] H. Mao, S. Han, J. Pool, W. Li, X. Liu, Y. Wang, and W. J. Dally, "Exploring the regularity of sparse structure in convolutional neural networks," *CoRR*, 2017.

[5] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," *JETC*, vol. 13, no. 3, pp. 32:1–32:18, 2017.

[6] Y. Gong, L. Liu, M. Yang, and L. D. Bourdev, "Compressing deep convolutional networks using vector quantization," *CoRR*, 2014.

[7] P. M. Gysel, "Ristretto: Hardware-oriented approximation of convolutional neural networks," Ph.D. dissertation, University of California Davis, 2016.

[8] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *NIPS*, 2016, pp. 4107–4115.

[9] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: ImageNet classification using binary convolutional neural networks," in *Comput. Vis. ECCV*. Springer, 2016, pp. 525–542.

[10] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," *CoRR*, vol. abs/1806.08342, 2018.

[11] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. G. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *IEEE CVPR 2018*, 2018, pp. 2704–2713.

[12] J. Cheng, P.-s. Wang, G. Li, Q.-h. Hu, and H.-q. Lu, "Recent advances in efficient computation of deep convolutional neural networks," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, no. 1, pp. 64–77, 2018.

[13] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *4th ICLR*, Y. Bengio and Y. LeCun, Eds., 2016.

[14] S. Arora, R. Ge, B. Neyshabur, and Y. Zhang, "Stronger generalization bounds for deep nets via a compression approach," in *International Conference on Machine Learning*. PMLR, 2018, pp. 254–263.

[15] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley-Interscience, 2006.

[16] W. Gao, Y.-H. Liu, C. Wang, and S. Oh, "Rate distortion for model compression: From theory to practice," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2102–2111.

[17] S. W. Piché, "The selection of weight accuracies for madalines," *IEEE Transactions on Neural Networks*, vol. 6, pp. 432 – 445, 1995.

[18] R. P. Gorman and T. J. Sejnowski, "Analysis of hidden units in a layered network trained to classify sonar targets," *Neural Networks*, vol. 1, no. 1, pp. 75–89, 1988.

[19] Z. He and D. Fan, "Simultaneously optimizing weight and quantizer of ternary neural network using truncated gaussian approximation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 438–11 446.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd ICLR*, 2015.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of IEEE conference on CVPR*, 2016, pp. 770–778.

[22] M. Nokleby, A. Beirami, and R. Calderbank, "Rate-distortion bounds on bayes risk in supervised learning," in *2016 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2016, pp. 2099–2103.

[23] D. Resmerita, R. C. Farias, B. D. de Dinechin, and L. Fillatre, "Compression des réseaux de neurones profonds à base de quantification uniforme et non-uniforme," in *Colloque GRETSI*, 2019.

[24] H. V. Poor, *An Introduction to Signal Detection and Estimation*, 2nd ed. Springer-Verlag New York, 1994.

[25] Z. Lin and Z. Bai, *Probability Inequalities*. Springer-Verlag Berlin Heidelberg, 2011.

[26] D. R. Cox and N. Wermuth, "A simple approximation for bivariate and trivariate normal integrals," *International Statistical Review*, vol. 59, no. 2, pp. 263–269, 1991.

[27] D. Resmerita, R. C. Farias, B. D. de Dinechin, and L. Fillatre, "Distortion approximation of a compressed softmax layer," in *2021 IEEE Statistical Signal Processing Workshop (SSP)*, 2021.

[28] M. E. Muller, "A note on a method for generating points uniformly on n-dimensional spheres," *Comm. Assoc. Comput. Mach.*, vol. 2, pp. 19–20, 1959.