Sparsity Regret bounds for XNOR-nets++

Andrew Chee, Sébastien Loustau

Sparsity Regret bounds for XNOR-nets++

Andrew Chee[1] and Sébastien Loustau[2]

[1] *Cornell University, Operations Research and Information Engineering, Ithaca, New-York*

[2] *Université de Pau et des Pays de l'Adour, Laboratoire de Mathématiques et de leurs Applications de Pau, France.*

June 18, 2021

Abstract

Despite the attractive qualities of convolutional neural networks (CNNs), and the universality of architectures emerging now, CNNs are still prohibitive regarding environmental impact due to electric consumption or carbon footprint, as well as deployment in constrained platform such as micro-computers. We address this problem and sketch how PAC-Bayesian theory can be applied to learn lighter convolutional architectures in order to reduce training, inference complexity and environmental impact of machine learning. We propose two main contributions: (1) a first sparsity regret bounds to enforce specific network characteristics in terms of number of parameters to train as well as non-zero weights, and (2) the control of a low bitwidth architectura, where binarized operations are used to approximate standard real-valued convolutions. With this in mind, and gathering with [1], we have the long term objective to design adaptive CNN architectura that fits dynamically to the size of the data, the difficulty of the problem and the desire environmental constraints.

1 Introduction

Deep learning is nowadays the most used technique to address the problem of supervised learning. It is extensively developed in Computer Vision, Natural Language Processing or communication networks, where very often the architecture of the deep net is based on a cascade of convolutions and non-linear activations. Indeed, CNNs are based on a very reduced set of computations (essentially convolutions and activations), and can be used to cover numerous application domains (e.g., audio, video, biosignals). Many tasks can be automated by convolutional neural networks, from the vanilla image classification problem (see [2], [3]) to more complex tasks such as text classification (see [4]), long-range temporal dependencies such as speech generation (see [5]), or even in unsupervised task such as object detection (see [6]) or graph embeddings (see [7]). Despite the attractive qualities of CNNs, and the universality of architectures emerging now to solve a broad range of tasks (see [8]), CNNs are still prohibitively expensive to deploy in a tight constrained environment such as micro-computer, in the context of high-resolution datasets or more generally on low-power devices since they need power-hungry General Purpose - Graphic Processing Units (GP-GPUs). In the recent literature, a natural approach for reducing the computational effort of CNNs (training of smaller and faster models) is through binarized networks. The idea behind binarized neural network is to approximate real-valued convolutions with low bitwidth operations. In [9] (or more recently in [10]), CNNs are trained with binary weights during the forward and backward propagations, while retaining precision of the stored weights in which gradients are accumulated, whereas in [11, 12], or more recently in [13], both filters and signal activations are binarized, leading to the so-called XNOR-nets where convolutions are approximated using primarily binary operations. The latter results in 58x faster convolutional operations and offer the possibility of running state-of-the-art networks on CPUs (rather than GPUs) in real-time. Moreover, binarizing the input signals also provide a significant gain in the overall memory consumption, especially for large batch sizes. Finally, since binarized networks can lead to poor

approximations of real-valued convolutions, [14] proposes to use ternary weight networks whereas [15] proposes a more flexible low bitwidth approach using different bitwidth weights for weights (1-bit weights), activations (2-bit activations), and gradients (see below). [16] studies the effects of quantization for convolutional neural networks when the network complexity is changed. It shows a better resilience of deep nets when the number of layers is big. Finally, as low bitwidth convolutions can be implemented efficiently in standard CPU or GPU, but also on tight constrained field-programmable gate arrays (FPGAs) or even on Application Specific Integrated Circuits (ASICs), binarized networks are exploited on each specialized computer hardware. It leads in [17] to a energy-efficient and scalable CNN accelerator on ASICs, whereas [18] proposes a recent survey on hardware accelerators that uses FPGAs.

It is also important to notice the recent growing interest for other main approaches to alleviate the computation of deep neural networks. Standard pruning methods (see [19] for the original paper, or more recently [20]) first train a feedforward or a convolutional neural network to convergence, and then network connections and / or neurons are pruned only subsequently. These techniques are applied and fined-tuned after training the entire network. Recent advances propose two different strategies in order to avoid the training of the entire network. It leads to significant improvements and accelerate both training and inference since the overall skeletonization is estimated before or during the training. One can prune the network at initialization, by estimating the important weights for a given task (see [21, 22] for moderate pruning levels up to 95%, or more recently [23] for higher level of compression, up to 99.5%). Another promising strategy is to select the connectivity of the network during the training. Interestingly, theoretical guarantees are proposed in [24] for an adaptive procedure selecting the architecture during training. It could be seen as a form of architecture designing, from the most general purpose of automated machine learning (AutoML, see [25]) to the problem of aggregation and design of efficient neural networks in terms of latency, memory size or carbon footprint, which lead finally to search for device-specific CNNs. As a seminal example, [26] proposes a neural architecture

search called FBNet to construct hardware efficient CNNs for mobile phones (see also [27]). The so-called Neural Architecture Search (NAS) uses a stochastic generator of architectures (that is a recurrent neural network named the controller) and train the proposed network with Reinforcement Learning (see for instance [28] for a nice introduction). Another approach is proposed in [29] where several budgeted super networks are selected to predict well in less than 100 milliseconds or to learn efficient models in terms of memory (for instance models that fit in a 50mb memory). More recently, [30] expends the search space to number of filters and channels dimension without prohibitive memory and computational cost with FBNetv2. Finally, [31] uses another approach based on a hierarchical neural ensemble to aggregate neural network blocks efficiently and control dynamically the inference latency depending on external constraints.

However, despite a growing interest, the study of computational-efficient deep neural networks is a work in progress in the community and affects several aspects of CNNs processing (i.e., training versus inference) as well as different key metrics and concepts to compare the various architectures, design and hardware (see [32] for a survey of deep nets processing). In this processing, power and energy consumption account for all aspects of the system including the size of the network, the hardware itself (such as CPUs, GPUs, FPGAs or ASICs) and the potentially external memory access. Recently open sources libraries have been proposed in order to evaluate globally the carbon footprint of deep learning algorithms as well as specific studies of particular algorithms. It has been shown for instance in [33] that significant gain in carbon emmission could be done without endowing the generalization performances. However, open sources available librairies in [34, 35, 36, 37, 38] are designed for massive usage of a large community of researchers and then constrain the estimations with strong hypothesis. As a consequence, we loose the specific impact of different hardware platform, as well as the cooling technique of the data center used for extensive computations. Moreover, carbon footprint are usually based on national coefficient. Finally, as discussed in [39], electric consumption based on a software estimation do not reflect the entire consumption of the system. Some contributions (see

[40, 39]) propose to conduct energy measurement experiments but are limited to a particular hardware distributor and specific algorithms.

With this in mind, we have the long term objective to train algorithms that reach a theoretical-based trade-off between accuracy of the network and power or energy consumption. For that purpose, we need to design penalties that cover precise attributes of the network such as - of course - the network architecture (including number of layers, number of filters, filter sizes, and number of channels), the number of non-zero weights (mainly for storage requirements) or the number of MACs (multiply-accumulate operations), but also metrics from the hardware design, depending on the number of cores or the amount of on-chip storage, the latency and throughput (depending on the batch size), the memory access and finally the power and energy consumption.

Finally, it is important to note that continuous weights are required for Stochastic Gradient Descent (SGD) to work at all since SGD computes small mooves and explores a real-valued space of parameters. Therefore, it is hard to promote low bitwidth for gradient updates (see a 6-bits attempt in [15]). In this paper, as well as in the companion paper [1], we use the PAC-Bayesian paradigm to learn distribution over a set of deep nets. It changes drastically the optimization procedure and lead to alternative to stochastic gradient updates for training - binarized - architectures. In the present paper, we use standard Kullback-Leibler divergences to get penalties proportional to the size of the network in terms of non-zero weights. In the companion paper [1] one offers more flexibility thanks to the introduction of Bregman divergences and optimal transport. It gives a new extension of the usual PAC-Bayesian theory presented in this paper and can be applied to deep learning with convolutional architectures in order to explore new strategies for penalizing training and/or inference complexity related with environmental impact. These theoretical based approaches could be applied to several data sources (images, time series, and graphs) and both generic and low bitwidth convolutional neural networks (see below), in order to design adaptive algorithms that fits dynamically to the size of the data, the difficulty of the problem and the desire environmental constraints.

2 Context and notations

2.1 Deep nets architectura

In this paper, we are interested in convolutional neural networks. From the learning perspective, a CNN is uniquely determined by a potentially huge set of p weights $\mathbf{w} \in \mathbb{W}^p \subset \mathbb{R}^p$, where $\mathbb{W}$ depends on the bitwidth of the system. In what follows, we consider $\mathbb{W} = \mathbb{R}$ for standard deep nets, as well as $\mathbb{W} = \{-1, 1\}$ for binary XNOR-nets (the intermediate case $\mathbb{W} = \{-b, \dots, b\}$ for some integer $b > 1$ is not considered for simplicity). Then, we introduce a parametric set of decisions $\mathcal{G} := \{g_{\mathbf{w}} : \mathcal{X} \to \mathcal{Y}, \mathbf{w} \in \mathbb{W}^p\}$, where $\mathcal{X}$ is the input of the neural network (also called the input space of descriptors, such as images, time series, or graphs, see below), and $\mathcal{Y}$ is the output of the network (for instance a vector of probabilities in classification, or a scalar in a regression framework). We want to learn a set of weights $\mathbf{w} := (w_l, b_l)_{l=1}^k$, where each couple (w_l, b_l) denotes the weights of layer l of the convolutional neural network $g_{\mathbf{w}}(\cdot)$ defined as:

$$g_{\mathbf{w}}(x) = \text{softmax}\left((w_k * \sigma(w_{k-1} * \cdots * \sigma(w_1 * x + b_1) \cdots) + b_k)\right), \tag{1}$$

where $w * x$ is a linear transformation described precisely below as a matrix multiplication and $\sigma(\cdot)$ is an element-wise non-linear activation such as a rectified linear unit (ReLU in the sequel) defined as $\sigma(x) = \max(0, x)$. In (1), we arbitrarily choose a softmax$(\cdot)$ since we deal in the main theoretical bounds with classification with cross-entropy loss function. Of course other activations at the head of the CNN could be considered, for other learning tasks. In what follows, thanks to the general algorithm presented below, we can choose automatically the number of layers k, as well as the number and size of filters or matrices involved in the linear transformation at each layer. For the sake of simplicity, we fix the number and size of filters at each layer and consider the space of candidate architectures $\mathcal{W}$ as the following union:

$$\mathcal{W} := \bigcup_{k=1}^{L} \bigcup_{i=1}^{N} \bigcup_{j=1}^{F} \mathbb{W}^{k(i+1)j}, \tag{2}$$

where $L \geq 1$ is the maximal number of layers for $g_{\mathbf{w}}$, $N \geq 1$ is the maximum number of filters and $F \geq 2$ is the maximum size of filters. We can easily extend the main theoretical results to more complex decision spaces.

In the context of Computer Vision, $x \in \mathbb{R}^m$ represents a $m = r_1 \times r_2$ image (we restrict the number of channels to 1 for simplicity) and given a set of i filters $(f_u)_{u=1}^i$ of size $j = s \times s$, the convolution $w * x$ in (1) corresponds to the matrix product of the filters and a relaxed form of the Toeplitz matrix of input signal x as follows:

$$w * x := \begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \end{pmatrix} \begin{pmatrix} x_{11} & x_{12} & x_{21} & x_{22} \\ x_{12} & x_{13} & x_{22} & x_{23} \\ x_{21} & x_{22} & x_{31} & x_{32} \\ x_{22} & x_{23} & x_{32} & x_{33} \end{pmatrix} \in \mathbb{R}^{i \times 4},$$

where we stand $s = 2$, $d = 2$, vertical and horizontal stride is let to 1 with no padding, and $r_1 = r_2 = 3$ for the sake of simplicity. Note that this representation could be extend to multiple channels and filters with redundant Toeplitz matrices. Of course this matrix multiplication above corresponds to convolutional layers and could be replaced by a standard matrix multiplication for fully connected layers. Moreover, concerning convolutions, it is also possible to consider similarly time series processing. If the input signal x is a univariate time serie of size m and we have at hand a set of filters, the one-dimensional convolution of x with filters $(f_u)_{u=1}^i$ with $i = 2$ can be written as a matrix product $w * x$ as follows:

$$w * x := \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} \begin{pmatrix} x_1 & x_2 & x_3 \\ x_2 & x_3 & x_4 \end{pmatrix} \in \mathbb{R}^{i \times 3}$$

where $m = 4$, stride is let to 1 with no padding. Finally, for completeness, we can also mention extensions of CNNs to graph real-world applications (see [41], [42], [7] or more recently [43]). In this framework, we observe a graph $G = (V, E)$ where $V = \{1, \ldots, N\}$ is the set of vertices and $E \subset \{(i, j), i, j \in \{1, \ldots, m\}^2\}$ is the set of edges, and a feature matrix X of size $m \times m'$ where each line i of X describes node i with m' features. Graph Convolutional Networks (GCNs) corresponds to an universal architecture able to process these inputs. It could be

written as follows:

$$\mathbf{H}^{l+1} = f(\mathbf{H}^l, A),$$

where A is the adjacency matrix of $G = (V, E)$, $\mathbf{H}^0 = X$ is the input matrix of feature, and $\mathbf{H}^L = Y$ is the output of the GCN. Then, the result of the present paper could be applied to the architectura presented in [7] where the layer-wise propagation rule has the following form:

$$\mathbf{H}^{l+1} = \sigma(\mathbf{D}^{-1/2} * \mathbf{A} * \mathbf{D}^{-1/2} * \mathbf{H}^l * \mathbf{W}^l),$$

where D is the diagonal node degree matrix, $\mathbf{W}^l$ is the weight matrix of the l-th layer and σ is a non-linear activation function like the ReLU, and $*$ stands for the classical matrix product. This propagation could be described intuitively as follows. Multiplication with A means that, for every node, we sum up all the feature vectors of all neighbor nodes. It extends the localization ensured by convolution to the graph topology. The presence of the diagonal node degree matrix arises technically to normalize the scale of the feature vectors for high degrees nodes.

Finally, in this paper, we also consider significant computational gain in convolution operations by considering XNOR-nets, where both parameters and activations are binary in the matrix multiplications considered above. In binarized CNNs, full precision weights and activations are binarized into 1 bit, so the multiplications and additions of the convolution are transformed into simple bitwise operations (XNOR-bitcounting operations), resulting in significant storage and computation requirements reduction. However, since much information has been discarded during binarization, the accuracy degradation is high. For that purpose, different techniques has been introduced to improve the approximation power of binary or low bitwidth convolutions. In [12], real-valued weights and activations are approximated by re-scaling the output of the binary convolution using real-valued positive scale factors. Whereas these scaling parameters are computed analytically in [12], [13] proposes to learn these factors directly via backpropagation. It leads to the following approximation of convolutions $\mathbf{w} * x$ defined

above:

$$w * x \approx \left(w^{\text{bin}} \bigoplus \text{sign}(x)\right) \bigodot w^{\text{scale}}, \tag{3}$$

where $\bigoplus$ stands for the bitwise convolution and $\bigodot$ defines the element-wise product of the output of the binary convolution $w^{\text{bin}} * \text{sign}(x)$ and the scaling factor w^{scale}. In [13], the set of parameters that is learnt during the training at a given layer is given by the couple $(w^{\text{bin}}, w^{\text{scale}})$, where the dimension of w^{scale} is varying. In this work, we adopt this strategy and consider (3) where w^{scale} corresponds to a sparse rank-1 tensor (see Section 3.2 and the proof of Theorem 3.2 for a precise description of the approximation, and [13] for details about this choice).

2.2 PAC-Bayesian Theory

PAC-Bayesian machine learning theory can be traced back to the work of Mac Allester and the first so-called Pac-Bayesian bounds ([44] and [45]). It was first introduced in learning theory to give a theoretical framework for proving generalization abilities of algorithms that combine the advantages of both PAC, that is generalization bounds, and Bayesian statistics (that is, the introduction of a prior domain knowledge on the set of candidate models). The main objective of PAC-Bayesian bounds is to extract a procedure that optimizes a tradeoff between complexity, or structure, such as sparsity of the candidate learners, with a goodness of fit, or likelihood of the observed sample. In [44], the first theoretical bound claims that with high probabilities over the sample distribution, there exists an optimal posterior distribution that realizes this trade-off, whereas [45] extends the previous result to an uncountable set of decisions, thanks to the introduction of the Kullback-Leibler divergence as an upper bound. It also shown that the optimal posterior is actually a Gibbs distribution. More recently, many authors have used these ingredients in the high dimensional setting to get interesting alternatives to standard sparsity-induced penalized empirical risk procedures such as lasso estimators. It leads to sparsity oracle inequalities for different aggregating procedures based on exponential weighted averages, with significant weaker

assumptions over the gram matrix of the predictors, as well as augmented stability in the procedures. For instance, in the i.i.d. setting, [46] extends to an infinite set of weak learners the previous results stated in [47]. [48] goes one step further considering a deterministic setting, tits the online learning setting, where no assumption are provided over the data mechanism. Thanks to similar argues, namely convex duality and a cancellation argument originated in [49], regret bounds are stated for recursive algorithm inspired from the previous cited litterature. Sparsity regret bounds for exponential weighted averages are then derived in [50] in an online regression setting, as well as in online clustering in [51]. In this paper, we focus on the sparsity scenario and apply the PAC-Bayesian theory to different families of CNNs whereas the companion paper [1] proposes a more ambitious theory where Kullback-Leibler divergence is replaced by Bregman or optimal transport divergences and give rise to more flexible penalties for revisited optimization procedures.

2.3 Supervised online learning

In the online supervised learning scenario[1], we observe sequentially a set of couples $\{(x_t, y_t) \in \mathcal{X} \times \mathcal{Y},\, t = 1, \ldots, T\}$. At each round $t \geq 1$, the data mechanism and the learning machine interact as follows:

- the environment reveals an new input (or mini-batch) $x_t \in \mathcal{X}$,
- a stochastic algorithm proposes $\mathbf{w}_t \in \mathcal{W}$ drawn from a suitable distribution π_t over the space of candidate architectures (2) and predicts $g_{\mathbf{w}_t}(x_t)$,
- the environment reveals the true output $y_t \in \mathcal{Y}$ and the stochastic algorithm computes π_{t+1}.

Based on the deep nets architecture introduced in Section 2.1, and the usual online PAC-Bayesian theory introduced in Section 2.2, the construction of the sequence

[1]In the sequel, we study the online learning paradigm to avoid any assumption about the data mechanism. It is important to note that thanks to [48], our procedure in Algorithm 1 could be slightly modified (by introducing a mirror averaging, see [46]) in order to get risk bounds in a classical statistical i.i.d. framework.

of distributions $\{\pi_t\}_{t=1}^T$ expects to minimize the following notion of regret:

$$\sum_{t=1}^{T} \mathbb{E}_{\pi_t} \ell(y_t, g_{\mathbf{w}_t}(x_t)) - \inf_{\mathbf{w}\in\mathcal{W}} \left\{ \sum_{t=1}^{T} \ell(y_t, g_{\mathbf{w}}(x_t)) + \lambda \text{pen}(\mathbf{w}) \right\}, \tag{4}$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is a task-dependent loss function, $\mathcal{W}$ is the space of candidate weights defined above and pen($\mathbf{w}$) depends on the network size and the number of non-zero weights. Parameter $\lambda > 0$ governs the trade-off between complexity of the solution and accuracy with respect to the training set. To prove regret bounds such as (4), we consider the standard PAC-Bayesian theory with sparsity priors introduced in [52] (see also [48, 50, 51]) and lead to ℓ_1 and ℓ_0 penalties. Depending on the choice of $\lambda > 0$ and the penalty above, our procedure reaches automatically the desire trade-off. In the same spirit, we can mention several papers dealing with neural networks sparsity (see [53] and the references therein). Several authors also use reinforcement learning to conduct Neural Architecture Search (NAS, see [28]), such as in FBNets (see [26, 30, 54]). Others techniques are presented in [29] or [31].

Algorithm 1 General Algorithm

init. $\lambda > 0$, $\pi = \pi_1$ a prior distribution on $\mathcal{W}$ and $t = 1$. Let $h_0 \equiv 0$.

Observe x_1 and draw $\hat{\mathbf{w}}_1$ from π_1. Predict y_1 according to $g_{\hat{\mathbf{w}}_1}(x_1)$.

repeat Observe y_t and compute

$$h_t(\mathbf{w}) = h_{t-1}(\mathbf{w}) + \ell(y_t, g_{\mathbf{w}}(x_t)) + \frac{\lambda}{2}\left(\ell(y_t, g_{\mathbf{w}}(x_t)) - \ell(y_t, g_{\hat{\mathbf{w}}_t}(x_t))\right)^2, \text{ for all } \mathbf{w} \in \mathcal{W}.$$

return $\pi_{t+1} = \text{Gibbs}_\lambda(h_t)$ where:

$$d\text{Gibbs}_\lambda(h_t)(\mathbf{w}) := Z_\lambda \exp\left\{-\lambda h_t(\mathbf{w})\right\} d\pi(\mathbf{w}),$$

for Z_λ the normalizing constant.

Draw $\hat{\mathbf{w}}_{t+1}$ from π_{t+1} and predict y_{t+1} according to $g_{\hat{\mathbf{w}}_{t+1}}(x_{t+1})$.

$t = t + 1$

until $t = T + 1$

3 Sparsity regret bounds for CNNs

In this section, we state regret bounds for convolutional neural networks (CNNs) described above. The main agenda is to show how theoretical results can be applied to convolutional architectures in order to get sparsity regret bounds for Algorithm 1 and control the training and/or inference complexity of the solution. In Section 3.1, we apply the general PAC-Bayesian paradigm to choose automatically the architecture of a generic CNN. We penalize the size of the network in terms of magnitude and non-zero weights. In Section 3.2, we apply these results to a particular low bitwidth architectura, where a precise bitwise approximation is used for the convolutional layers. It leads to a sparse and Bayesian version of the XNOR-nets described in [13].

3.1 Sparsity Regret bound for Classification

We state regret bounds in the context of classification. In this case, considering a set of weights $\mathbf{w} \in \mathcal{W}$ and the associated convolutional neural network $g_{\mathbf{w}}(\cdot)$ defined in (1), we define the last k-th layer by:

$$\text{softmax}(w_k * x_{k-1} + b_k) = \begin{pmatrix} \frac{e^{x_k(1)}}{C_k} \\ \frac{e^{x_k(2)}}{C_k} \\ \vdots \\ \frac{e^{x_k(K)}}{C_k} \end{pmatrix} \in [0,1]^K, \tag{5}$$

where x_{k-1} is the output of layer $k-1$, $x_k = w_k * x_{k-1} + b_k$, $w_k \in \mathbb{W}^{p_k}$ is the weight matrix of last layer k, $C_k = \sum_{j=1}^{K} e^{x_{k,j}}$, K is the number of classes, $*$ is the standard matrix multiplication and $x_k(j)$ stands for the j-th coordinate of vector x_k.

In what follows, we state a first regret bound considering the cross-entropy between the output of the CNNs and $\mathbf{y}_t = \mathbf{e}_t$ the orthonormal vector of $\mathbb{R}^K$ with coordinate 1 at y_t.

Theorem 3.1. *Consider Algorithm 1 with prior $\pi \in \mathcal{P}(\mathcal{W})$ defined as:*

$$\pi(\mathbf{w}) = \sum_{k=1}^{L} \alpha_k \sum_{i=1}^{N} \beta_i \sum_{j=1}^{F} \gamma_j \pi_{ijk}(\mathbf{w}) \mathbb{1}\left\{\mathbf{w} \in \mathbb{R}^{k(i+1)j}\right\}, \tag{6}$$

where α (respectively β and γ) is the prior probability on the number of layers $\{1, \dots, L\}$ (respectively on the number of filters and the size of the filters) and π_{ijk} is a sparsity prior for a given architecture of k layers, i filters of size j whose density is given by:

$$\pi_{ijk}(\mathbf{w}) = \prod_{u=1}^{p} \frac{c_{R,\tau}}{(1 + |w_u|/\tau)^4} \mathbb{1}\left(|w_u| \leq R\right), \mathbf{w} \in \mathbb{R}^p,$$

where constant $c_{R,\tau} := \frac{3}{2\tau}\left(1 - \frac{1}{(1+R/\tau)^3}\right)^{-1}$ and with a slight abuse of notations $\mathbf{w} = (w_1, \dots, w_p)$ for $p = k(i+1)j$.

Let $\{(x_t, y_t), t = 1, \dots, T\}$ a deterministic sequence and $(\pi_t)_{t=1}^T$ the sequence of distribution defined in Algorithm 1 with prior π. Then, for any $\lambda, \tau > 0$, we have:

$$\sum_{t=1}^{T} \mathbb{E}_{\pi_t} \mathcal{H}(y_t, g_{\mathbf{w}_t}(x_t)) \leq \inf_{\mathbf{w} \in \mathcal{W}(R)} \left\{ \sum_{t=1}^{T} \mathcal{H}\left(y_t, g_{\mathbf{w}}(x_t)\right) + \lambda \operatorname{pen}(\mathbf{w}) \right\}, \tag{7}$$

where $\operatorname{pen}(\mathbf{w}) = 2\tau \operatorname{pen}_1(g_{\mathbf{w}}) \sum_{t=1}^{T} \|x_t\| + 2\tau T \operatorname{pen}_2(g_{\mathbf{w}}) + \lambda \operatorname{pen}_3(g_{\mathbf{w}})$ *for*

$$\operatorname{pen}_1(g_{\mathbf{w}}) = \prod_{l=1}^{k} C_l^1 + \sum_{l \neq l'} C_l^1 \|w_{l'}\| + \lambda C'(R) T, \tag{8}$$

$$\operatorname{pen}_2(g_{\mathbf{w}}) = \sum_{l=1}^{k-1} \left(\prod_{l'=l+2}^{k} \|w_{l'}\| \right) \left[C_{l+1}^1 (\|ReLU(b_l)\| + C_l^2) + C_l^2 \right], \tag{9}$$

$$\operatorname{pen}_3(g_{\mathbf{w}}) = 4\|\mathbf{w}\|_0 \log\left(1 + \frac{\|\mathbf{w}\|_1}{\tau \|\mathbf{w}\|_0}\right) - \log(\alpha_k \beta_i \gamma_j), \tag{10}$$

where k is the number of layer of $\mathbf{w}$ and $C_l^{1,2}$ are constants depending on the distribution π in its layer l and:

$$\mathcal{W}(R) = \{\mathbf{w} \in \mathcal{W} : |w_u| \leq R, \forall u \in \{1, \dots, p\}\}.$$

The proof uses the general PAC-Bayesian theory with several computations related with the definition of CNNs architecture in (2) gathered with Kullback-Leibler divergences with sparsity prior (6) (see Section 4 for the proof).

Remark 1. *Theorem 3.1 can lead to usual statistical risk bounds for deep nets by slightly modifying Algorithm 1 with a mirror averaging (see [46]). Then, generalization bounds relying on the Froebenius norm of the weights at each layer can be proposed and compared with the recent literature (see [55, 56, 57, 53]).*

Remark 2. *We can state the same result for more general prior involving different filter size inside each layer by considering:*

$$\pi(\mathbf{w}) = \sum_{k=1}^{L} \alpha_k \sum_{i=1}^{N_k} \beta_{ik} \sum_{j=1}^{F_{ki}} \gamma_{ijk} \pi_{ijk}(\mathbf{w}). \tag{11}$$

We omit this case for concision.

Remark 3. *In [1], we derive similar results in a more general case where the Gibbs measure in Algorithm 1 is replaced by a new optimization procedure based on a convex conjugate of Bregman divergences or optimal transport. It gives rise to more flexible penalties than in Theorem 3.1 and Theorem 3.2.*

3.2 Application to sparse XNOR-Nets

In this section we propose to extend the previous result to a binarized network. Following in particular [12] and [13], we propose to reduce drastically the real-valued operations involved in standard CNNs. Moreover we alleviate high quantization errors thanks to the introduction of sparse scaling factors in our bitwise approximations. Inspired by [11], we first keep the first convolutional layer and the last fully connected layer with sparse real-valued weights and activations. Then, scaling factors are learned by the learning procedure itself for intermediate layers, where sparsity is also enforced for these scaling parameters at each binary layer. In order to define properly the prior distribution, we now detail the approximation used for the intermediate layer. In the sequel, we denote by x_{l-1} the input of the l^{th}-layer, w_l^{bin} the binary set of weights involved in layer l and w_l^{scale} the scaling factors of layer l. Then, following [13], the output of a binarized layer l is given by the following approximation:

$$x_l = \left(\mathbf{w}_l^{\text{bin}} \bigoplus \text{sign} \circ \text{BNorm}\left(x_{l-1}\right) \right) \bigodot \mathbf{w}_l^{\text{scale}},$$

where $\bigoplus$ stands for the bitwise convolution, BNorm is the Batch Normalization, $\bigodot$ is the element-wise product between the tensor

$$\mathcal{B}_l := w_l^{\text{bin}} \bigoplus \text{sign} \circ \text{BNorm}\,(x_{l-1}) \in \{0,1\}^{h_1^{(l)} \times h_2^{(l)} \times h_3^{(l)}}$$

and the scaling tensor $w_l^{\text{scale}} = (w_{l,1}^{\text{scale}}, w_{l,2}^{\text{scale}}, w_{l,3}^{\text{scale}}) \in \mathbb{R}_+^{h_1^{(l)}+h_2^{(l)}+h_3^{(l)}}$ given by:

$$x_l(i,j,k) = \mathcal{B}_l(i,j,k) w_{l,1}^{\text{scale}}(i) w_{l,2}^{\text{scale}}(j) w_{l,3}^{\text{scale}}(k),\ \forall\, i = 1, \dots h_1^{(l)},\ j = 1, \dots h_2^{(l)},\ k = 1, \dots h_3^{(l)}. \tag{12}$$

Note that different choices of scaling tensor $\mathbf{w}_l^{\text{scale}}$ are discussed in [13]. It leads to the resulting binarized network:

$$g_{\mathbf{w}}(x) = \text{softmax}\left(w_L^{\text{real}} * x_{L-1} + b_L\right),$$

where x_{L-1} is given by:

$$\begin{aligned} x_{L-1} &= \left(w_{L-1}^{\text{bin}} \bigoplus \text{sign} \circ \text{BNorm}\,(x_{L-2})\right) \bigodot w_{L-1}^{\text{scale}} \\ &= \left(w_{L-1}^{\text{bin}} \bigoplus \text{sign} \circ \text{BNorm}\left(\mathbf{w}_{L-2}^{\text{bin}} \bigoplus \text{sign} \circ \text{BNorm}\,(x_{L-3}) \bigodot w_{L-2}^{\text{scale}}\right)\right) \bigodot w_{L-1}^{\text{scale}} \\ &= \left(w_{L-1}^{\text{bin}} \bigoplus \cdots \text{sign} \circ \text{BNorm}\left(w_1^{\text{real}} * x + b_1\right)\right) \bigodot w_{L-1}^{\text{scale}} \end{aligned}$$

Then we can state the following theorem.

Theorem 3.2. *Consider a prior π related with the particular architecture described above, based on the triplet of priors $(\pi_{\text{real}}, \pi_{\text{bin}}, \pi_{\text{scale}})$ as follows:*

$$\pi(\mathbf{w}) = \pi_{\text{real}}(\mathbf{w}^{\text{real}}) \pi_{\text{bin}}(\mathbf{w}^{\text{bin}}) \pi_{\text{scale}}(\mathbf{w}^{\text{scale}}). \tag{13}$$

Let us consider a sparsity prior π_{real} for the real-valued weights $\mathbf{w}^{\text{real}}$ as above:

$$\pi_{\text{real}}(\mathbf{w}^{\text{real}}) := \prod_{r=1}^{p_{\text{real}}} \frac{c_{\tau,R}}{(1 + |w_r^{\text{real}}|/\tau)^4} \mathbb{1}\left\{|w_r^{\text{real}}| \le R\right\},$$

where $c_{\tau,R} > 0$ is defined in Theorem 3.1. For $\mathbf{w}^{\text{bin}}$ we consider π_{bin} as a product of independent Rademacher distribution defined as:

$$\mathcal{R}(\{w_r^{\text{bin}} = 1\}) = \mathcal{R}(\{w_r^{\text{bin}} = -1\}) = \frac{1}{2},\ \forall r = 1, \dots, p_{\text{bin}}.$$

Then, $\mathbf{w}^{\text{scale}}$ *contains the* p_{scale} *scaling factors with associated prior:*

$$\pi_{\text{scale}}(\mathbf{w}^{\text{scale}}) := \prod_{r=1}^{p_{\text{scale}}} \frac{c'_{\tau,R}}{(1+w_r^{\text{scale}}/\tau)^4} \mathbb{1}\left\{0 \leq w_r^{\text{scale}} \leq R\right\},$$

where $c'_{\tau,R} = 3/\tau \left(1 - \frac{1}{(1+R/\tau)^3}\right)^{-1}$.

Let $\{(x_t, y_t)\}_{t=1}^T$ *a deterministic sequence and* $\{\pi_t\}_{t=1}^T$ *the sequence of distribution defined in Algorithm 1 with prior* π *defined in* (13) *with standard deviation* $\tau > 0$. *Then, for some* $R > 0$, *by denoting* $\mathcal{W}(R) := \{\mathbf{w} = (\mathbf{w}^{\text{real}}, \mathbf{w}^{\text{bin}}, \mathbf{w}^{\text{scale}}) : \forall w \in \mathbf{w}^{\text{real}} \cup \mathbf{w}^{\text{scale}}, |w| \leq R\}$, *we have for any* $\lambda > 0$*:*

$$\sum_{t=1}^T \mathbb{E}_{\pi_t} \mathcal{H}(y_t, g_{\mathbf{w}_t}(x_t)) \leq \inf_{\mathbf{w} \in \mathcal{W}(R)} \left\{ \sum_{t=1}^T \mathcal{H}\left(y_t, g_{\mathbf{w}}(x_t)\right) + \frac{\text{pen}_{\text{XNOR}}(\mathbf{w})}{\lambda} \right\} + \text{res}_{\lambda,\tau}(T),$$

where:

$$\text{pen}_{\text{XNOR}}(\mathbf{w}) = \sum_{\mathbf{w} \in \{\mathbf{w}^{real}, \mathbf{w}^{scale}\}} \|\mathbf{w}\|_0 \log\left(1 + \frac{\|\mathbf{w}\|_1}{\tau \|\mathbf{w}\|_0}\right) + p_{\text{bin}} \log 2,$$

and $\text{res}_{\lambda,\tau}(T)$ *is a residual term of order* $\sqrt{T}$ *for particular choices of hyper-parameter* (λ, τ) *in Algorithm 1 and prior* (13).

Remark 4. *Theorem 3.2 allows to bound the cumulative loss of the PAC-Bayesian version of XNOR-nets presented in [13]. Interestingly, the RHS in Theorem 3.2 depends on the sparsity of the set of real-valued parameters of the XNOR-nets. It gives some insights to construct sparse versions of classical binarized networks where continuous weights are used to approximate real-valued convolutions.*

Remark 5. *However, this sparsity-induced penalty scales linearly with the dimension of the binary weights. As in [58], this unconstrained bound strongly depends on the network size, and can be trivial when the number of parameters exceeds the sample size. This dependence is not considered here where we focus on a sparse version of XNOR-Nets in its real-valued weights.*

4 Proofs

4.1 Proof of Regret bounds

4.1.1 Proof of Theorem 3.1

The proof is based on the online PAC-Bayesian bound stated in [48] as follows:

Theorem 4.1. *[48, Theorem 4.6] Let (x_t, y_t),$t = 1, \dots, T$ a deterministic sequence of data and $(\pi_t)_{t=1}^T$ a sequence of distributions based on Algorithm 1. Then we have the following bound:*

$$\sum_{t=1}^{T} \mathbb{E}_{\pi_t} \ell(y_t, \hat{g}_t(x_t)) \leq \min_{\rho \in \mathcal{P}(\mathcal{W})} \left\{ \mathbb{E}_{\mathbf{w}\sim\rho} \sum_{t=1}^{T} \bar{\ell}(y_t, g_{\mathbf{w}}(x_t)) + \frac{\mathcal{K}(\rho, \pi)}{\lambda} \right\},$$

where $\bar{\ell}(y_t, g_{\mathbf{w}}(x_t) = \ell(y_t, g_{\mathbf{w}}(x_t) + \frac{\lambda}{2}(\ell(y_t, g(x_t)) - \ell(y_t, \hat{g}_t(x_t))^2$.

In what follows, we use this bound with a particular measure ρ in the RHS. Let $\mathbf{w}^* \in \mathcal{W} = \bigcup_{k=1}^{L} \bigcup_{i=1}^{N} \bigcup_{j=1}^{F} \mathbb{W}^{k(i+1)j}$ a specific set of weights in architecture $\mathbf{a}^*$. Without loss of generality, we denote by L the number of layers of $\mathbf{w}^*$. Then, for a fixed parameter $\tau > 0$, consider $\pi_{\mathbf{w}^*}$ the translation of the sparsity prior π_{ijL} defined in Theorem 3.1 as follows:

$$\pi_{\mathbf{w}^*}(d\mathbf{w}) = \frac{d\pi_{ijL}}{d\mathbf{w}}(\mathbf{w} - \mathbf{w}^*)d\mathbf{w}. \tag{14}$$

Lemma 4.2 below is useful to control the first term in the RHS of Theorem 4.1, whereas Lemma 4.6 below controls the Kullback-Leibler divergence $\mathcal{K}(\pi_{\mathbf{w}^*}, \pi)$.

Lemma 4.2. *Let $\mathbf{w}^* = (w_l^*, b_l^*)_{l=1}^k \in \mathcal{W}$ for some $k \in \{2, \dots, L\}$. Then we have for any parameter $\tau > 0$:*

$$\mathbb{E}_{\pi_{\mathbf{w}^*}} \sum_{t=1}^{T} \mathcal{H}\left(y_t, g_{\mathbf{w}}(x_t)\right) \leq \sum_{t=1}^{T} \mathcal{H}\left(y_t, g_{\mathbf{w}^*}(x_t)\right) + 2\,\mathrm{pen}_1(\mathbf{w}^*, \tau) \sum_{t=1}^{T} \|x_t\| + 2T\,\mathrm{pen}_2(\mathbf{w}^*, \tau),$$

where

$$\mathrm{pen}_1(\mathbf{w}^*, \tau) = \prod_{l=1}^{k} (\|w_l^*\| + C_{n_l,\tau}^1) - \prod_{l=1}^{k} \|w_l^*\|,$$

$$\mathrm{pen}_2(\mathbf{w}^*, \tau) = \sum_{l=1}^{k-1} \left(\prod_{l'=l+2}^{k} \|w_{l'}^*\| \right) \left[C_{n_{l+1},\tau}^1 (\|ReLU(b_l^*)\| + C_{n_l,\tau}^2) + C_{n_{l+1},\tau}^2 \right]$$

and $C_{l,\tau}^{1,2}$ are constants depending on the distribution $\pi_{\mathbf{w}^*}$ in its layer l.

Proof. Observe that it suffices to obtain the estimate at each time t. Then, for each $(x, y) \in \mathcal{X} \times \mathcal{Y}$, we need to proove the following bound:

$$\mathbb{E}_{\pi_{\mathbf{w}^*}} \mathcal{H}(y, g_{\mathbf{w}}(x)) \leq \mathcal{H}(y, g_{\mathbf{w}^*}(x)) + 2\,\mathrm{pen}_1(\mathbf{w}^*, \tau)\|x\| + 2\,\mathrm{pen}_2(\mathbf{w}^*, \tau).$$

Let $x_0 = x$ the input data and $k = L$ without loss of generality. From the choice of $\pi_{\mathbf{w}^*}$ in (14), we denote the output of layer $1 \leq l \leq L-1$ by

$$x_l = \mathrm{ReLU}((w_l + w_l^*) * x_{l-1} + b_l + b_l^*),$$

where $\mathbf{w} = (w_l)_{l=1}^L$ has law π. Moreover, we denote in the sequel $\mathbb{E}^l := \mathbb{E}_{(w_1,\dots,w_l)\sim\pi}$ the expectation with respect to the distribution π in the first l layers.

We start by unpacking the interaction of the cross entropy loss with the softmax activation function in the final layer. For the weights (w_l, b_l) in layer l, $w_{l,i}$ and $b_l(i)$ denote their i^{th} row and i^{th} entry, respectively. Under the previous notations, the final output is:

$$g_{\mathbf{w}^*}(x_0) = \mathrm{softmax}(x_L) = \left(\frac{e^{x_L(1)}}{C_L}, \cdots, \frac{e^{x_L(K)}}{C_L}\right)^T,$$

where $x_L = w_L * x_{L-1} + b_L$ and $C_L = \sum_{i=1}^K e^{x_L(i)}$. Without loss of generality, suppose that $y_t = e_1 = (1, 0, ..., 0)$. Then, by definition of the cross entropy:

$$\begin{aligned}\mathbb{E}_{\mathbf{w}\sim\pi_{\mathbf{w}^*}} \mathcal{H}(y, g(x)) &= -\mathbb{E}_{\mathbf{w}\sim\pi_{\mathbf{w}^*}} \log \frac{e^{x_L(1)}}{C_L} \\ &= -\mathbb{E}_{\mathbf{w}\sim\pi_{\mathbf{w}^*}} x_L(1) + \mathbb{E}_{\mathbf{w}\sim\pi_{\mathbf{w}^*}} \log C_L.\end{aligned}$$

We bound the two above terms separately. For the first term, by symmetry of the distribution $\pi_{\mathbf{w}^*}$, one has:

$$\begin{aligned}-\mathbb{E}_{\mathbf{w}\sim\pi_{\mathbf{w}^*}} x_{L,1} = &= -\mathbb{E}_{\mathbf{w}\sim\pi_{\mathbf{w}^*}}((w_{L,1}^* + w_{L,1}) * x_{L-1} + b_L(1)^* + b_L(1)) \\ &= -(w_{L,1}^* x_{L-1}^* + b_L(1)^*) - \mathbb{E}^L(w_{L,1} x_{L-1} + b_L(1)) - w_{L,1}^* \mathbb{E}^{L-1}(x_{L-1} - x_{L-1}^*) \\ &\leq -x_{L,1}^* + \|W_L^*\| \mathbb{E}^{L-1} \|x_{L-1} - x_{L-1}^*\|.\end{aligned}$$

On the other hand, for any $i = 1, ..., K$, using the same computations:

$$\begin{aligned}
x_L(i) &= (w^*_{L,i} + w_{L,i}) * x_{L-1} + b_L(i)^* + b_L(i) \\
&= (w^*_{L,i} x^*_{L-1} + b_L(i)^*) + w^*_{L,i}(x_{L-1} - x^*_{L-1}) + W_{L,i} x_{L-1} + b_{L,i} \\
&\leq x^*_{L,i} + \|w^*_{L,i}\| \|x_{L-1} - x^*_{L-1}\| + \mathrm{ReLU}(W_{L,i} x_{L-1}) + \mathrm{ReLU}(b_L(i)) \\
&\leq x_L(i)^* + \|w^*_L\| \|x_{L-1} - x^*_{L-1}\| + \|\mathrm{ReLU}(w_L x_{L-1})\| + \|\mathrm{ReLU}(b_L)\|,
\end{aligned}$$

where we adopt the notation that for a vector $x \in \mathbb{R}^k$, $\mathrm{ReLU}(x) = (\mathrm{ReLU}(x_1), ..., \mathrm{ReLU}(x_k))^T$. Then,

$$\begin{aligned}
&\mathbb{E}_{\mathbf{w} \sim \pi_{\mathbf{w}^*}} \log C_L \\
&= \mathbb{E}_{\mathbf{w} \sim \pi_{\mathbf{w}^*}} \log \sum_{i=1}^{K} e^{x_L(i)} \\
&\leq \mathbb{E}_{\mathbf{w} \sim \pi} \log \sum_{i=1}^{K} e^{x_L(i)^*} \exp\left(\|w^*_L\| \|x_{L-1} - x^*_{L-1}\| + \|\mathrm{ReLU}(W_L x_{L-1})\| + \|\mathrm{ReLU}(b_L)\| \right) \\
&= \log \sum_{i=1}^{K} e^{x^*_L(i)} + \mathbb{E}_{\mathbf{w} \sim \pi} \left\{ \|w^*_L\| \|x_{L-1} - x^*_{L-1}\| + \left\| \mathrm{ReLU}\left(w_L \frac{x_{L-1}}{\|x_{L-1}\|} \right) \right\| \|x_{L-1}\| + \|\mathrm{ReLU}(b_L)\| \right\} \\
&\leq \log C_L + \|w^*_L\| \mathbb{E}^{L-1} \|x_{L-1} - x^*_{L-1}\| + C^1_{L,\tau} \mathbb{E}^{L-1} \|x_{L-1}\| + C^2_{L,\tau},
\end{aligned}$$

where

$$\begin{aligned}
C^1_{l,\tau} &= \max_{\|x\|=1} \mathbb{E}^l \|\mathrm{ReLU}(w_l x)\| \\
C^2_{l,\tau} &= \mathbb{E}^l \|\mathrm{ReLU}(b_l)\|
\end{aligned}$$

In summary:

$$\begin{aligned}
\mathbb{E}_{\mathbf{w} \sim \pi_{\mathbf{w}^*}} \mathcal{H}(y, g_{\mathbf{w}}(x)) &\leq \mathcal{H}(y, g_{\mathbf{w}^*}(x)) + C^1_{L,\tau} \mathbb{E}^{L-1} \|x_{L-1}\| + 2\|W^*_L\| \mathbb{E}^{L-1} \|x_{L-1} - x^*_{L-1}\| + C^2_{L,\tau} \\
&\quad + \mathbb{E}^{L-1} \|\mathrm{ReLU}\left(w_L x_{L-1}\right)\| + \mathbb{E}^{L-1} \|\mathrm{ReLU}\left(b_L\right)\| \\
&\leq \mathcal{H}(y, g_{\mathbf{w}^*}(x)) + 2\left[C^1_{L,\tau} \mathbb{E}^{L-1} \|x_{L-1}\| + \|W^*_L\| \mathbb{E}^{L-1} \|x_{L-1} - x^*_{L-1}\| + C^2_{L,\tau} \right].
\end{aligned}$$

Finally, by inducting with Lemma 4.3 and Lemma 4.4 below, we have:

$$
\begin{aligned}
&C^1_{L,\tau}\mathbb{E}^{L-1}\|x_{L-1}\| + \|w^*_L\|\mathbb{E}^{L-1}\|x_{L-1} - x^*_{L-1}\| + C^2_{L,\tau} \\
&\leq \sum_{l=1}^{L-1}\left(\prod_{k=l+2}^{L}\|w^*_k\|\right)(C^1_{l+1,\tau}\mathbb{E}\|x_l\| + C^2_{l+1,\tau}) \\
&\leq \left[\prod_{l=1}^{L}(\|w^*_l\| + C^1_{l,\tau}) - \prod_{l=1}^{L}\|w^*_l\|\right]\|x_0\| \\
&\quad + \sum_{l=1}^{L-1}\left(\prod_{k=l+2}^{L}\|w^*_k\|\right)\left(C^1_{l+1,\tau}(\|\mathrm{ReLU}(b^*_l)\| + C^2_{l,\tau}) + C^2_{l+1,\tau}\right) \\
&= \mathrm{pen}_1(\mathbf{w}^*,\tau)\|x_0\| + \mathrm{pen}_2(\mathbf{w}^*,\tau).
\end{aligned}
$$

□

Lemma 4.3. *For $l = 1, ..., L-1$.*

$$
\mathbb{E}^l\|x_l - x^*_l\| \leq \|w^*_l\|\mathbb{E}^{l-1}\|x_{l-1} - x^*_{l-1}\| + C^1_{l,\tau}\mathbb{E}\|x_{l-1}\| + C^2_{l,\tau}. \tag{15}
$$

Proof. We use the following facts:

- $\mathrm{ReLU}(a+b) \leq \mathrm{ReLU}(a) + \mathrm{ReLU}(b)$
- $\|\mathrm{ReLU}(a)\| \leq \|a\|$
- $\mathrm{ReLU}(a) = \mathrm{ReLU}\left(\frac{a}{\|a\|}\right)\|a\|$.

Then, we have:

$$
\begin{aligned}
&\mathbb{E}^l\|x_l - x^*_l\| \\
&= \mathbb{E}^l\|\mathrm{ReLU}((w^*_l + w_l)x_{l-1} + b^*_l + b_l) - \mathrm{ReLU}(w^*_l x^*_{l-1} + b^*_l)\| \\
&\leq \mathbb{E}^l\|\mathrm{ReLU}(w^*_l(x_{l-1} - x^*_{l-1}))\| + \mathbb{E}^l\|\mathrm{ReLU}(w_l x_{l-1})\| + \mathbb{E}^l\|\mathrm{ReLU}(b_l)\| \\
&\leq \|w^*_l\|\mathbb{E}^{l-1}\|x_{l-1} - x^*_{l-1}\| + \mathbb{E}^l\left\|\mathrm{ReLU}\left(w_l\frac{x_{l-1}}{\|x_{l-1}\|}\right)\|x_{l-1}\|\right\| + C^2_{l,\tau} \\
&\leq \|w^*_l\|\mathbb{E}^{l-1}\|x_{l-1} - x^*_{l-1}\| + \mathbb{E}^{l-1}\left[\mathbb{E}_{w_l\sim\pi^1_{A,\tau,l}}\left\|\mathrm{ReLU}\left(w_l\frac{x_{l-1}}{\|x_{l-1}\|}\right)\right\|\|x_{l-1}\|\right] + C^2_{l,\tau} \\
&\leq \|w^*_l\|\mathbb{E}^{l-1}\|x_{l-1} - x^*_{l-1}\| + C^1_{l,\tau}\mathbb{E}^{l-1}\|x_{l-1}\| + C^2_{l,\tau}.
\end{aligned}
$$

□

Lemma 4.4. *For $l = 1, ..., L-1$.*

$$\mathbb{E}^l\|x_l\| \leq \|w_l^*\|\mathbb{E}^{l-1}\|x_{l-1}\| + C_{l,\tau}^1\mathbb{E}\|x_{l-1}\| + C_{l,\tau}^2 + \|ReLU(b_l^*)\|. \tag{16}$$

Proof. From a similar computation as in the proof of the previous lemma, one gets:

$$\begin{aligned}
\mathbb{E}^l\|x_l\| &= \mathbb{E}^l\|\mathrm{ReLU}((w_l^* + w_l)x_{l-1} + b_l^* + b_l)\| \\
&\leq \mathbb{E}^l\|\mathrm{ReLU}(w_l^* x_{l-1})\| + \|\mathrm{ReLU}(b_l^*)\| + \mathbb{E}^l\|\mathrm{ReLU}(w_l x_{l-1})\| + \mathbb{E}^l\|\mathrm{ReLU}(b_l)\| \\
&\leq \|w_l^*\|\mathbb{E}^{l-1}\|x_{l-1}\| + \|\mathrm{ReLU}(b_l^*)\| + +C_{n_l,\tau}^1\mathbb{E}^{l-1}\|x_{l-1}\| + C_{n_l,\tau}^2,
\end{aligned}$$

□

Lemma 4.5. *For any $l \in \{1, \ldots, L\}$, the constants $C_{l,\tau}^1$ and $C_{l,\tau}^2$ are linear in τ as follows:*

$$C_{l,\tau}^1 = \tau C_l^1, \quad C_{l,\tau}^2 = \tau C_l^2,$$

where $C_l^1 := C_{l,1}^1$ and $C_l^2 := C_{l,\tau}^2$.

Proof. The proof follows from working with the sparsity prior directly and substitution.

Let $x \in \mathbb{R}^{n_{l-1}}$ and $\|x\| = 1$. We use the substitution $W_1 \mapsto W_l/\tau$. Then,

$$\begin{aligned}
\mathbb{E}_{W_l\sim\pi_{A,\tau,l}^1}\|\mathrm{ReLU}(W_l x)\| &= \int \left(\sum_{i=1}^{n_l}\mathrm{ReLU}(W_{l,i}x)^2\right)^{1/2}\left(\prod_{i=1}^{n_l}\prod_{j=1}^{n_{l-1}}\frac{(3/\tau)\,dW_{l;ij}}{2(1+|W_{l;ij}|/\tau)^4}\right) \\
&= \tau\int \left(\sum_{i=1}^{n_l}\mathrm{ReLU}\left(\frac{W_{l,i}}{\tau}x\right)^2\right)^{1/2}\left(\prod_{i=1}^{n_l}\prod_{j=1}^{n_{l-1}}\frac{(3/\tau)\,dW_{l;ij}}{2(1+|W_{l;ij}|/\tau)^4}\right) \\
&= \tau\int \left(\sum_{i=1}^{n_l}\mathrm{ReLU}\left(W_{l,i}x\right)^2\right)^{1/2}\left(\prod_{i=1}^{n_l}\prod_{j=1}^{n_{l-1}}\frac{3\,dW_{l;ij}}{2(1+|W_{l;ij}|)^4}\right) \\
&= \tau\mathbb{E}_{W_l\sim\pi_{A,1,l}^1}\|\mathrm{ReLU}(W_l x)\|.
\end{aligned}$$

The homogeneity of order 1 in τ is preserved when taking the max over all

$\|x\| = 1$.

$$\begin{aligned} C^1_{l,\tau} &= \max_{\|x\|=1} \mathbb{E}_{W_l \sim \pi^1_{A,\tau,l}} \|\mathrm{ReLU}(W_l x)\| \\ &= \tau \max_{\|x\|=1} \mathbb{E}_{W_l \sim \pi^1_{A,1,l}} \|\mathrm{ReLU}(W_l x)\| \\ &= \tau C^1_l. \end{aligned}$$

The result for $C^2_{l,\tau}$ follows from a similar computation. □

We are now on time to bound the quadratic term in Theorem 4.1. Using the previous computations, by noting that $\mathbf{w}^* \in \mathcal{W}(R)$ for some positive $R > 0$, and since π has a bounded support, we have for any $(x, y) \in \mathcal{X} \times \mathcal{Y}$:

$$\begin{aligned} \left(\mathcal{H}(y, g_{\mathbf{w}}(x)) - \mathcal{H}(y, g_{\mathbf{w}'}(x))\right)^2 &\leq \min\left(1, \max\left(\mathcal{H}(y, g_{\mathbf{w}}(x)), \mathcal{H}(y, g_{\mathbf{w}'}(x))\right)^2\right) \\ &= \min\left(\left(1, \left(x^+_L(y) - \log C^+_L\right)^2\right)\right) \\ &\leq \min\left(\left(\left\langle x_{L-1}, w_{L,y}\right\rangle + b_{L,y} - \log \sum_{k=1}^{K} e^{x_{L,k}}\right)^2\right) \\ &\leq \left(\|\mathrm{ReLU}(w_{L-1} * x_{L-2} + b_{L-1})\| \|w_L\| + 2R\right)^2 \\ &\leq \left(\|w_{L-1} * x_{L-2} + b_{L-1}\| \|w_L\| + 2R\right)^2 \\ &\quad \vdots \\ &\leq C(R). \end{aligned}$$

We then get the result by using Lemma 4.2 and the following lemma.

Lemma 4.6. *Let $\mathbf{w}^* \in \mathcal{W}$ and $\rho_{\mathbf{w}^*}$ given by* (14)*. Then we have:*

$$\mathcal{K}(\rho_{\mathbf{w}^*}, \pi) \leq 4\|\mathbf{w}^*\|_0 \log\left(1 + \frac{\|\mathbf{w}^*\|_1}{\tau \|\mathbf{w}^*\|_0}\right) - \log(\alpha_k \beta_i \gamma_j).$$

Let $\mathbf{w}^* \in \mathcal{W}$. Then we have denoting $\rho = \rho_{\mathbf{w}^*}$:

$$\begin{aligned}
\mathcal{K}(\rho, \pi) = \int \log \frac{\rho}{\pi} d\rho &= \int \log \frac{\rho}{\pi_{ijk}} \frac{\pi_{ijk}}{\pi} d\rho, \\
&= \mathcal{K}(\rho, \pi_{ijk}) + \int_{\mathbb{R}^{k(i+1)j}} \log \frac{\pi_{ijk}}{\alpha_k \beta_i \gamma_j \pi_{ijk}} d\rho, \\
&= \sum_{u=1}^{p} \int \log \frac{(1+|w_u|/\tau)^4}{(1+|w_u - w_u^*|/\tau)^4} \Bigg) \rho(\mathbf{w})\, d\mathbf{w} - \log(\alpha_k \beta_i \gamma_j), \\
&= 4 \sum_{u=1}^{p} \log \left(1 + \frac{|w_u|}{\tau}\right) - \log(\alpha_k \beta_i \gamma_j), \\
&\leq 4 \|\mathbf{w}\|_0 \log \left(1 + \frac{\|\mathbf{w}\|_1}{\tau \|\mathbf{w}\|_0}\right) - \log(\alpha_k \beta_i \gamma_j),
\end{aligned}$$

where we use Jensen inequality to the concave function $x \mapsto \log(1+x)$ in last line.

4.1.2 Proof of Theorem 3.2

Let $\mathbf{w}_0 = (\mathbf{w}_0^{\text{real}}, \mathbf{w}_0^{\text{bin}}, \mathbf{w}_0^{\text{scale}}) \in \mathcal{W}$ a particular set of weights of the particular XNORNets++ architectura defined in Section 3. In what follows, with a slight abuse of notations, for any ℓ, we denote $\mathbf{x}_\ell(\mathbf{w}_0)$ the output of the ℓ-th layer with particular weights $\mathbf{w}_0$ until it does not depend on weights of layers k, for $k > \ell$. We also denote by $\mathbf{x}_\ell(\mathbf{w}_{0,l})$ the output of the ℓ-th layer with particular weights $\mathbf{w}_{0,l}$ at layer l. Moreover, given the prior π introduced in (13), we introduce a particular measure ρ_0 based on $\mathbf{w}_0$ as follows:

$$\rho_0(\mathbf{w}) := \pi_{\text{real}}(\mathbf{w}_0^{\text{real}} - \mathbf{w}^{\text{real}}) \delta_{\mathbf{w}_0^{\text{bin}}}(\mathbf{w}^{\text{bin}}) \pi^0_{\text{scale}}(\mathbf{w}^{\text{scale}}),$$

where $\delta_{\mathbf{w}}(\cdot)$ stands for the dirac distribution at weight $\mathbf{w}$ and π^0_{scale} is based on the prior π_{scale} and the set of weights $\mathbf{w}_0$ as follows:

$$\pi^0_{\text{scale}}(\mathbf{w}^{\text{scale}}) = \Pi_{r=1}^{p_{\text{scale}}} \frac{c''_{\tau,R,w_r^0}}{(1+|w_r - w_r^0|/\tau)^4} \mathbb{1}\left\{0 \leq w_r \leq R\right\},$$

where $c''_{\tau,R,w_r^0} = 3/\tau \left(2 - \frac{1}{(1+w_r^0/\tau)^3} - \frac{1}{(1+(R-w_r^0)/\tau)^3}\right)^{-1}$.

With a slight abuse of notation, we also denote by $\mathbb{E}_{\mathbf{w}_1^l \sim \rho_0}$ the expectation with respect to the first l layers of ρ_0 whereas $\mathbb{E}_{\mathbf{w}_l \sim \rho_0}$ denotes the conditoinal expectation of layer l given layers $1, \ldots, l-1$. To prove Theorem 3.2, we follow

the same line as the proof of Theorem 3.1 and adapt the computations to the particular XNOR Nets defined in Section 3.2. The first step is to apply Theorem 4.1 to distribution ρ_0 defined above to get:

$$\sum_{t=1}^{T} \mathbb{E}_{\pi_t} \mathcal{H}\left(y_t, g_{\mathbf{w}_t}(x_t)\right) \leq \mathbb{E}_{\mathbf{w}\sim\rho_0} \sum_{t=1}^{T} \mathcal{H}\left(y_t, g_{\mathbf{w}}(x_t)\right) + \mathbb{E}_{\mathbf{w}\sim\rho_0} \mathbb{E}_{\pi_0^{T-1}} \sum_{t=1}^{T} \delta_\lambda(x_t, y_t, \mathbf{w}, \hat{\mathbf{w}}_{t-1}) + \frac{\mathcal{K}(\rho_0, \pi)}{\lambda}.$$

To study the first term in the upper bound, we perform the same computations as above. For a given (x, y), we have:

$$\mathbb{E}_{\mathbf{w}\sim\rho_0} \mathcal{H}\left(y, g_{\mathbf{w}}(x)\right) = -\mathbb{E}_{\mathbf{w}\sim\rho_0} \left(\langle w_{L,y}^{\text{real}}, \mathbf{x}_{L-1}\rangle + b_{L,y}\right) + \mathbb{E}_{\mathbf{w}\sim\rho_0} \log(C_L) \tag{17}$$

$$= -\left(\langle w_{0,L,y}^{\text{real}}, \mathbb{E}_{\mathbf{w}\sim\rho_0^{L-1}} \mathbf{x}_{L-1}\rangle + b_{0,L,y}\right) + \mathbb{E}_{\mathbf{w}\sim\rho_0} \log(C_L). \tag{18}$$

Then we need the following lemma:

Lemma 4.7. *Let $\mathbf{w}_0 \in \mathcal{W}$ and ρ_0 a shifted version of prior π defined in* (13) *with mean $\mathbf{w}_0$. Then we have $\forall l \in \{2, \ldots, L-1\}$, if $\operatorname{Var} w^{\text{scale}} = \frac{c_r}{T}\text{Id}$:*

$$\left|\mathbb{E}_{\mathbf{w}_1^l\sim\rho_0} x_l(u) - \mathbb{E}_{\mathbf{w}_1^{l-1}\sim\rho_0} x_l(\mathbf{w}_{0,l})(u)\right| \leq \frac{c}{\sqrt{T}} + \frac{c'}{T}, \; \forall u = 1, \ldots, h_1^{(l)} h_2^{(l)} h_3^{(l)},$$

where $c, c' > 0$ are absolute constant depending on R and the dimension of the network.

Proof : Let $l \in \{2, L-1\}$.The proof uses a standard decomposition of the expectation and uses Tchebychev inequality. Let us denote $\mathbf{w}_l^{\text{scale}} = (\alpha, \beta, \gamma) \in \mathbb{R}_+^{a+b+c}$ the triplet of scaling factors according to [13], where a, b, c are the dimensions of the output tensor at layer l defined in (12). Consider the event $\Omega_{\alpha,\beta,\gamma} := \{\forall k = 1, \ldots a, |\alpha_k - \alpha_{0,k}| \leq \delta_1\} \bigcap \{\forall i = 1, \ldots b, |\beta_i - \beta_{0,i}| \leq \delta_2\} \bigcap \{\forall j = 1, \ldots, c, |\gamma_j - \gamma_{0,j}| \leq \delta_3\}$. Then we have by definition of x_l in (12), and noting $x = \text{sign} \circ \text{BN}(x_{L-2}) \in \{-1, +1\}^{abc}$:

$$\mathbb{E}_{\mathbf{w}_1^l\sim\rho_0} x_l = \mathbb{E}_{\mathbf{w}_1^l\sim\rho_0} \left((\mathbf{w}_l^{\text{bin}} \oplus x) \odot \mathbf{w}_l^{\text{scale}}\right)$$

$$= \mathbb{E}_{\mathbf{w}_1^l\sim\rho_0} \left((\mathbf{w}_l^{\text{bin}} \oplus x) \odot \mathbf{w}_l^{\text{scale}}\right) \left[\mathbb{1}(\Omega_{\alpha,\beta,\gamma}) + \mathbb{1}(\bar{\Omega}_{\alpha,\beta,\gamma})\right]$$

Then, for any $u \in \left\{1, \dots, h_1^{(l)} h_2^{(l)} h_3^{(l)}\right\}$, there exists a binary filter $w_l^r \in \{0,1\}^{s_1 s_2 s_3}$, $r \in \{1, \dots, N\}$, and integers i, j, k such that:

$$\begin{aligned}
\mathbb{E}_{\mathbf{w}_1^l \sim \rho_0} x_l(u) &= \mathbb{E}_{\mathbf{w}_1^l \sim \rho_0} \left(\sum_{f_1=1}^{s_1} \sum_{f_2=1}^{s_2} \sum_{f_3=1}^{s_3} \mathbf{w}_l^r(f_1, f_2, f_3) x(f_1, f_2, f_3) \alpha_k \beta_i \gamma_j \right) \\
&\quad \left[\mathbb{1}(\Omega_{\alpha,\beta,\gamma}^{ijk}) + \mathbb{1}(\bar{\Omega}_{\alpha,\beta,\gamma}^{ijk}) \right] \\
&= \mathbb{E}_{\mathbf{w}_1^{l-1} \sim \rho_0} \left(\sum_{f_1,f_2,f_3} w^r(f_1, f_2, f_3) x(f_1, f_2, f_3) (\alpha_{0,k} + \delta_1)(\beta_{0,i} + \delta_2)(\gamma_{0,j} + \delta_3) \right) \\
&\quad + \mathbb{P}\left(\bar{\Omega}_{\alpha,\beta,\gamma}^{ijk} \right) \\
&\leq \mathbb{E}_{\mathbf{w}_1^{l-1} \sim \rho_0} x_{L-1}(\mathbf{w}_{0,l})(u) + \frac{c}{\sqrt{T}} + \mathbb{P}\left(|X - \mu| \geq \delta_i | \right) \\
&\leq \mathbb{E}_{\mathbf{w}_1^{l-1} \sim \rho_0} x_{L-1}(\mathbf{w}_{0,l})(u) + \frac{c}{\sqrt{T}} + \frac{c'}{T},
\end{aligned}$$

where last two lines follows from Tchebychev by choosing $\delta_i \sim 1/\sqrt{T}$, the assumption on the variance of $\mathbf{w}^{\text{scale}}$ and the boundness of $\mathbf{w}^{\text{scale}}$. It leads to the RHS upper bound. For the RLS, we have using the same notations as above:

$$\begin{aligned}
\mathbb{E}_{\mathbf{w}_1^l \sim \rho_0} x_l(u) &= \mathbb{E}_{\mathbf{w}_1^{l-1} \sim \rho_0} \left(\sum_{f_1,f_2,f_3} w^r(f_1, f_2, f_3) x(f_1, f_2, f_3) (\alpha_{0,k} + \delta_1)(\beta_{0,i} + \delta_2)(\gamma_{0,j} + \delta_3) \right) \\
&\quad + \mathbb{P}\left(\bar{\Omega}_{\alpha,\beta,\gamma}^{ijk} \right) \\
&\geq \mathbb{E}_{\mathbf{w}_1^{l-1} \sim \rho_0} x_{L-1}(\mathbf{w}_{0,l})(u) - \frac{c}{\sqrt{T}} - \frac{c'}{T},
\end{aligned}$$

where we use in last line the decomposition as above and noting that on $\Omega_{\alpha,\beta,\gamma}^{ijk}$, we have $\alpha = \alpha_0 + \epsilon$ with $\epsilon > -\delta$, and on the complementary, $\epsilon \geq -\|\mathbf{w}^{\text{scale}}\|_\infty$.

Now we need to control the second term in the previous inequality. As in the proof of Theorem 3.1, we have, for any $i = 1, ..., K$, under the shifted distribution,

$$\begin{aligned}
\mathbf{x}_L(i) &= (\mathbf{w}_{0,L}^{\text{real}}(i) + \mathbf{w}_L|(i) * \mathbf{x}_{L-1} + b_{0,L}(i) + b_L(i) \\
&= \mathbf{w}_{0,L}^{\text{real}}(i) * \mathbf{x}_{L-1}(\mathbf{w}_0) + b_{0,L}(i) + \mathbf{w}_{0,L}^{\text{real}}(i) * (\mathbf{x}_{L-1} - \mathbf{x}_{L-1}(\mathbf{w}_0)) + \mathbf{w}_L(i) * \mathbf{x}_{L-1} + b_L(i) \\
&= \mathbf{x}_L(\mathbf{w}_0)(i) + \mathbf{w}_{0,L}^{\text{real}}(i) * (\mathbf{x}_{L-1} - \mathbf{x}_{L-1}(\mathbf{w}_0)) + \mathbf{w}_L(i) * \mathbf{x}_{L-1} + b_L(i).
\end{aligned}$$

Then we have:

$$\begin{aligned}\mathbb{E}_{\mathbf{w}\sim\rho_0}\log(C_L) &= \mathbb{E}_{\mathbf{w}\sim\rho_0}\log\sum_{i=1}^{K} e^{\mathbf{x}_L(\mathbf{w}_0)|_i}\exp\left(\mathbf{w}_{0,L}^{\text{real}}|_i * (\mathbf{x}_{L-1}-\mathbf{x}_{L-1}(\mathbf{w}_0)) + \mathbf{w}_L|_i * \mathbf{x}_{L-1} + b_L|_i\right)\\ &\leq \log C_L(\mathbf{w}_0) + \mathbb{E}_{\mathbf{w}\sim\rho_0}\max_{i=1,\dots K}\left\{\mathbf{w}_{0,L}^{\text{real}}|_i * (\mathbf{x}_{L-1}-\mathbf{x}_{L-1}(\mathbf{w}_0)) + \mathbf{w}_L|_i * \mathbf{x}_{L-1} + b_L|_i\right\}\\ &\leq \log C_L(\mathbf{w}_0) + \frac{c_3}{\sqrt{T}},\end{aligned}$$

where we use Lemma 4.7 in the last inequality. Then, applying $L-3$ times Lemma 4.7 for discrete layers and the same computations as in the proof Theorem 3.1 for the first and last layer, we have finally, gathering with the last inequality:

$$\begin{aligned}\sum_{t=1}^{T}\mathbb{E}_{\pi_t}\mathcal{H}\left(y_t, g_{\mathbf{w}_t}(x_t)\right) &\leq \sum_{t=1}^{T}\mathcal{H}\left(y_t, g_{\mathbf{w}_0}(x_t)\right) + c\sqrt{T}\\ &\quad + \mathbb{E}_{\pi_0^{T-1}}\sum_{t=1}^{T}\frac{\lambda}{2}\mathbb{E}_{\mathbf{w}\sim\rho_0}(\mathcal{H}(y_t, g_{\mathbf{w}}(x_t)) - \mathcal{H}(y_t, \hat{g}_t(x_t))^2 + \frac{\mathcal{K}(\rho_0,\pi)}{\lambda}\\ &\leq \sum_{t=1}^{T}\mathcal{H}\left(y_t, g_{\mathbf{w}_0}(x_t)\right) + c_1\sqrt{T} + c_2\lambda T\frac{\mathcal{K}(\rho_0,\pi)}{\lambda},\end{aligned}$$

where $c_1 > 0$ is a positive constant depending on the previous computations. Moreover, using the boundness assumption on $\mathbf{w}_0$ and the same computations as in Theorem 3.1, we arrive at:

$$\sum_{t=1}^{T}\mathbb{E}_{\pi_t}\mathcal{H}\left(y_t, g_{\mathbf{w}_t}(x_t)\right) \leq \sum_{t=1}^{T}\mathcal{H}\left(y_t, g_{\mathbf{w}_0}(x_t)\right) + c_1\sqrt{T} + c_2\lambda T + \frac{\mathcal{K}(\rho_0,\pi)}{\lambda},$$

where $c_2 := c_2(R) > 0$. Finally, by the definition of ρ_0 and the sparsity prior π, we can bound the Kullback-Leibler divergence as follows:

$$\begin{aligned}K(\rho_0,\pi) &= \int\log\left(\frac{\rho_0(\mathbf{w})}{\pi(\mathbf{w})}\right)\rho_0(\mathbf{w})\,d\mathbf{w}\\ &= \sum_{r=1}^{p_{\text{real}}}\int\log\frac{(1+|w_r^{\text{real}}|/\tau)^4}{(1+|w_r^{\text{real}}-w_{0,r}^{\text{real}}|/\tau)^4}\rho_0(\mathbf{w})\,d\mathbf{w} + \sum_{r=1}^{p_{\text{bin}}}\int\log\left(2\delta_{w_{0,r}}(w_r)\right)\rho_0(\mathbf{w})d\mathbf{w}\\ &\quad + \sum_{r=1}^{p_{\text{scale}}}\int\log\frac{(1+|w_r^{\text{scale}}/\tau)^4}{(1+|w_r^{\text{scale}}-w_{0,r}^{\text{scale}}|/\tau)^4}\rho_0(\mathbf{w})\,d\mathbf{w}\\ &\leq 4\sum_{r=1}^{p_{\text{real}}}\log\left(1+\frac{|w_{0,r}^{\text{real}}|}{\tau}\right) + p_{\text{bin}}\log 2 + 4\sum_{r=1}^{p_{\text{scale}}}\log\left(1+\frac{w_{0,r}^{\text{scale}}}{\tau}\right)\\ &\leq 4\|\mathbf{w}_0^{\text{real}}\|_0\log\left(1+\frac{\|\mathbf{w}_0^{\text{real}}\|_1}{\tau\|\mathbf{w}_0^{\text{real}}\|_0}\right) + 4\|\mathbf{w}_0^{\text{scale}}\|_0\log\left(1+\frac{\|\mathbf{w}_0^{\text{scale}}\|_1}{\tau\|\mathbf{w}_0^{\text{scale}}\|_0}\right) + p_{\text{bin}}\log 2\\ &:= \text{pen}_{\text{XNOR}}(\mathbf{w}_0).\end{aligned}$$

References

[1] Andrew Chee and Sébastien Loustau. Learning with bot - bregman and optimal transport divergences. *https://hal.archives-ouvertes.fr/hal-03262687*, 2021.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[4] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

[5] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[6] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[7] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[8] Iasonas Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6129–6138, 2017.

[9] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[10] Wei Tang, Gang Hua, and Liang Wang. How to train a compact binary neural network with high accuracy? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 1, 2017.

[11] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[12] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

[13] Adrian Bulat and Georgios Tzimiropoulos. Xnor-net++: Improved binary neural networks. *arXiv preprint arXiv:1909.13863*, 2019.

[14] Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

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

[16] Wonyong Sung, Sungho Shin, and Kyuyeon Hwang. Resiliency of deep neural networks under quantization. *arXiv preprint arXiv:1511.06488*, 2015.

[17] Renzo Andri, Lukas Cavigelli, Davide Rossi, and Luca Benini. Yodann: An architecture for ultralow power binary-weight cnn acceleration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(1):48–60, 2017.

[18] Ahmad Shawahna, Sadiq M Sait, and Aiman El-Maleh. Fpga-based accelerators of deep learning networks for learning and classification: A review. *IEEE Access*, 7:7823–7859, 2018.

[19] Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *Proceedings of*

the 1st International Conference on Neural Information Processing Systems, pages 107–115, 1988.

[20] Zichao Yang, Marcin Moczulski, Misha Denil, Nando De Freitas, Alex Smola, Le Song, and Ziyu Wang. Deep fried convnets. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1476–1483, 2015.

[21] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

[22] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*, 2020.

[23] Pau de Jorge, Amartya Sanyal, Harkirat S Behl, Philip HS Torr, Gregory Rogez, and Puneet K Dokania. Progressive skeletonization: Trimming more fat from a network at initialization. *arXiv preprint arXiv:2006.09081*, 2020.

[24] Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep rewiring: Training very sparse deep networks. In *International Conference on Learning Representations*, 2018.

[25] Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *arXiv preprint arXiv:1908.00709*, 2019.

[26] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019.

[27] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[28] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[29] Tom Veniat and Ludovic Denoyer. Learning time/memory-efficient deep architectures with budgeted super networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3492–3500, 2018.

[30] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, et al. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. *arXiv preprint arXiv:2004.05565*, 2020.

[31] Adrià Ruiz and Jakob Verbeek. Distilled hierarchical neural ensembles with adaptive inference cost. *arXiv preprint arXiv:2003.01474*, 2020.

[32] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.

[33] Titouan Parcollet and Mirco Ravanelli. The Energy and Carbon Footprint of Training End-to-End Speech Recognizers. working paper or preprint, April 2021.

[34] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.

[35] Peter Henderson, Jie-Ru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. Towards the systematic reporting of the energy and carbon footprints of machine learning. *ArXiv*, abs/2002.05651, 2020.

[36] Loïc Lannelongue, Jason Grealey, and Michael Inouye. Green algorithms: Quantifying the carbon emissions of computation. *Advanced science*, 05 2021.

[37] Lasse Anthony, Benjamin Kanding, and Raghavendra Selvan. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. page arXiv preprint https://arxiv.org/abs/2007.03051, 07 2020.

[38] David A. Patterson, Joseph Gonzalez, Quoc V. Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R. So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *CoRR*, abs/2104.10350, 2021.

[39] Qingqing Cao, Aruna Balasubramanian, and Niranjan Balasubramanian. Towards accurate and reliable energy measurement of NLP models. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 141–148, Online, November 2020. Association for Computational Linguistics.

[40] Crefeda Faviola Rodrigues, Graham Riley, and Mikel Luján. Synergy: An energy measurement and prediction framework for convolutional neural networks on jetson tx1. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, pages 375–382. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2018.

[41] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[42] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

[43] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

[44] David A McAllester. Some pac-bayesian theorems. *Machine Learning*, 37(3):355–363, 1999.

[45] David A McAllester. Pac-bayesian stochastic model selection. *Machine Learning*, 51(1):5–21, 2003.

[46] Arnak S Dalalyan, Alexandre B Tsybakov, et al. Mirror averaging with sparsity priors. *Bernoulli*, 18(3):914–944, 2012.

[47] Anatoli Juditsky, Philippe Rigollet, Alexandre B Tsybakov, et al. Learning by mirror averaging. *The Annals of Statistics*, 36(5):2183–2206, 2008.

[48] Jean-Yves Audibert et al. Fast learning rates in statistical inference through aggregation. *The Annals of Statistics*, 37(4):1591–1646, 2009.

[49] Andrew R Barron. Are bayes rules consistent in information? In *Open Problems in Communication and Computation*, pages 85–91. Springer, 1987.

[50] Sébastien Gerchinovitz. Sparsity regret bounds for individual sequences in online linear regression. *Journal of Machine Learning Research*, 14(Mar):729–769, 2013.

[51] Le Li, Benjamin Guedj, Sébastien Loustau, et al. A quasi-bayesian perspective to online clustering. *Electronic journal of statistics*, 12(2):3071–3113, 2018.

[52] Arnak S Dalalyan and Alexandre B Tsybakov. Sparse regression learning by aggregation and langevin monte-carlo. *Journal of Computer and System Sciences*, 78(5):1423–1443, 2012.

[53] Johannes Schmidt-Hieber. Statistical guarantees for regularized neural networks. *Taheri, Masha and Xie, Eang and Lederer, Johannes*, 48(4):1875–1897, 2020.

[54] Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Bichen Wu, Zijian He, Zhen Wei, Kan Chen, Yuandong Tian, Matthew Yu, Peter Vajda, and Joseph E. Gonzalez. Fbnetv3: Joint architecture-recipe search using predictor pretraining. *arXiv preprint arXiv:2006.02049*, 2021.

[55] Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. In Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet, editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 297–299. PMLR, 06–09 Jul 2018.

[56] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[57] Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[58] Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.