

Generalized Accuracy of Approximated Deep Neural Networks

Jiří Šíma*
Institute of Computer Science
Czech Academy of Sciences
Prague, Czechia
sima@cs.cas.cz
*Corresponding author

Jérémy Cabessa
DAVID Laboratory
UVSQ – University Paris-Saclay
Versailles, France
jeremie.cabessa@uvsq.fr

Abstract—Deep neural networks (NNs) are computationally intensive and often approximated (e.g., via low-bit quantization or pruning) to reduce their energy consumption with minimal loss in accuracy on the dataset. However, such approximations can degrade generalization near data points, which standard accuracy does not capture. We introduce a novel measure, termed generalized accuracy, which also accounts for neighborhoods of data points given by the so-called correct convex polytopes, where both the original NN and its approximation (employing ReLU units) behave linearly and the original NN correctly classifies the inputs. The generalized accuracy is defined as the ratio of the total volume of subpolytopes within the correct polytopes for which the approximated NN predicts the correct class to the total volume of all such subpolytopes assigned to any class by the approximated NN. We approximate the nonzero volumes of subpolytopes using the mean width as a surrogate measure, which is estimated via linear programming based on the shortcut weights of the original and approximated NNs. To ensure the comparability of generalized accuracy across different approximations, the neighborhood of each data point has to be augmented into an equal-volume extended correct polytope (relaxing linearity of the approximated NN) using the Hit-and-Run Markov Chain Monte Carlo method. We validate this definition experimentally by showing that generalized accuracy converges as dataset size increases. We also demonstrate that, for fully-connected and convolutional neural networks trained on Fashion-MNIST, generalized accuracy decreases after reducing the weight bitwidth, indicating degraded generalization even when standard pointwise accuracy remains largely preserved.

Index Terms—deep neural networks, approximate computing, accuracy, convex polytope, mean width, data augmentation

I. INTRODUCTION

Deep neural networks (DNNs) are central to modern artificial intelligence, powering applications such as large language models, image and speech recognition, robotics, and intelligent assistants. Many embedded and edge devices now rely on DNNs to provide advanced functionalities, but their high computational demands translate into substantial energy consumption. This is particularly challenging for battery-powered

devices, including smartphones, smartwatches, augmented reality glasses, and voice-controlled systems.

To address this, two main strategies have emerged [1]. The first relies on specialized *hardware accelerators* optimized for DNN inference [2], [3], which share asymptotic energy complexity bounds [4]. The second leverages *approximate computing* techniques [5]–[10], which exploit the tolerance of many artificial intelligence tasks to minor inaccuracies. One approach reduces the number of operations and the model size through compression [11], pruning [12], or compact network architectures [13]. The other approach lowers the precision of arithmetic operations, including reduced bitwidth [14], nonuniform quantization [15], conversion from floating point to fixed point [16], weight sharing [17], or approximate multiplication circuits [18]. By applying these techniques, trained DNNs can be replaced with simplified approximations that significantly reduce energy consumption while maintaining acceptable task performance.

The error of an approximated DNN for classification is typically measured using accuracy on a dataset. However, the generalization capability can degrade even in the vicinity of data points without affecting this accuracy. In this paper, we therefore introduce a more robust measure of the error of approximated DNNs, called generalized accuracy, which also takes into account the neighborhoods of data points. For this purpose, in Section II, we first formalize a model of a feedforward neural network $\mathcal{N}$ with ReLU units (hereafter referred to as an NN or simply a network), which encompasses commonly used DNNs, including convolutional neural networks (CNNs) with pooling and output softmax layers. For an approximated network $\tilde{\mathcal{N}}$, we impose only minimal assumptions: it must have the same inputs as $\mathcal{N}$ and the same number of outputs assigned to the corresponding classes in the classification task. This means that $\tilde{\mathcal{N}}$ can be obtained by approximating a trained NN that generalizes over a given dataset, using any method, including those mentioned above, such as pruning or quantization.

The standard accuracy of $\tilde{\mathcal{N}}$ is measured on the dataset as the fraction of data points classified correctly. We extend this pointwise view to so-called correct (closed and bounded convex) polytopes around the data points. Within these poly-

This research was supported institutionally by RVO: 67985807 and by the Czech Science Foundation grant GA25-15490S. J. Cabessa was also partially supported by the research programme of Strategy AV21 “AI: Artificial Intelligence for Science and Society.” We thank Vít Kološ for his exceptional willingness to assist in running the computational experiments and the anonymous reviewer for valuable comments.

topes, due to the piecewise linearity of the ReLU activation function, both $\mathcal{N}$ and $\tilde{\mathcal{N}}$ behave linearly, and $\mathcal{N}$ predicts the correct class associated with the corresponding data point. In Section III, we thus define the *generalized accuracy* of the approximated network $\tilde{\mathcal{N}}$ as the ratio of the total volume of subpolytopes within the correct polytopes for which $\tilde{\mathcal{N}}$ also predicts the correct class, to the total volume of all such subpolytopes assigned to any class by $\tilde{\mathcal{N}}$.

The convex polytope is defined as the intersection of half-spaces bounded by hyperplanes whose coefficients are given by the so-called shortcut weights [19], expressing the linear dependence of the neurons in $\mathcal{N}$ and $\tilde{\mathcal{N}}$ on their inputs in the vicinity of the corresponding data point. These weights of virtual shortcut connections from any NN input can be computed efficiently via forward propagation of a one-hot vector corresponding to that input through $\mathcal{N}$ and $\tilde{\mathcal{N}}$, masking out neurons that produce zero output for that data point due to the ReLU activation function (Section II).

Nevertheless, computing directly the volume of a convex polytope is infeasible in high dimensions [20], [21]. Therefore, various approximate or surrogate methods are used to estimate the “size” of polytopes via other characteristics. To evaluate the generalized accuracy, we employ the *mean width* [26], which admits a proven asymptotic error bound [27], despite not being additive with respect to the polytope partition. The mean width of a convex polytope is defined as the integral of its width in all unit directions over the sphere of one dimension lower, divided by its surface area. We approximate the mean width by averaging over finitely many random directions [28], where the width of the polytope in each direction is computed by solving two linear programs (LPs).

However, the correct polytopes around data points may be small, especially in large NNs, thereby bringing the generalized accuracy closer to the pointwise accuracy. This is due to overly strong constraints on the linearity of the approximated $\tilde{\mathcal{N}}$. Even worse, the volumes of correct polytopes may differ substantially across approximated $\tilde{\mathcal{N}}$, making their generalized accuracies incomparable. To eliminate this issue, the neighborhood of each data point must be expanded to its equal-volume extended correct polytope by relaxing the linearity of $\tilde{\mathcal{N}}$. This can be achieved using the Hit-and-Run Markov Chain Monte Carlo method, which generates additional data points uniformly at random within this extended polytope, thereby covering it with its correct subpolytopes. These points are then used to evaluate the generalized accuracy, making it comparable across different approximated $\tilde{\mathcal{N}}$ and further strengthening the generalization guarantees of approximated NNs.

In our previous work, we have measured the inaccuracy of approximated NNs as the maximum of linear error [19] or cross-entropy loss [29] within the correct convex polytope around the data point, maximized or averaged over the dataset, which represents only the worst case scenario. Moreover, in order to allow the use of linear programming, we actually employed an upper bound on the cross-entropy loss between the output categorical probability distributions of $\mathcal{N}$ and $\tilde{\mathcal{N}}$, which was further linearized. In fact, the cross-entropy

loss appears to be an unsuitable measure for classification tasks, since it grows without bound as the probability of any incorrect class approaches zero, thereby overshadowing the actual classification error.

In contrast, the proposed generalized accuracy represents a global measure of the performance of $\tilde{\mathcal{N}}$, because, unlike pointwise accuracy, it does not focus solely on individual data points or on the maximum errors within the polytopes, but considers the entire correct polytopes and their volumes. It thus takes into account the generalization capability of $\tilde{\mathcal{N}}$ in the vicinity of data points, which define the probabilistic distribution of the input space. This is the best achievable, since computing the maximum error of $\tilde{\mathcal{N}}$ with respect to the original $\mathcal{N}$ over the entire input space, regardless of the dataset, is an NP-hard problem [19].

Many standard pointwise evaluation metrics have been proposed in the literature, focusing solely on individual data points [30]–[33]. More recently, such metrics have been extended to account for local neighborhoods of inputs, which is related to our approach. By explicitly considering how model behavior varies across these regions, they go beyond traditional pointwise evaluation and provide richer insights into stability, reliability, and robustness.

For instance, *local calibration error* (LCE) evaluates confidence reliability across similar samples defined via kernel similarity in a pretrained feature space [34]. Robustness metrics, such as *certified adversarial robustness* via randomized smoothing, measure the fraction of points that remain correctly classified under bounded perturbations, providing certified region guarantees [35]. Subsequent work has refined these notions by addressing limitations of the average certified radius and improving evaluation over input neighborhoods [36]. Additionally, *CLEVER* (Cross-Lipschitz Extreme Value for nEtworK Robustness) provides an attack-agnostic robustness estimate based on extreme value theory, approximating the minimal perturbation needed to alter a classification [37]. However, unlike these approaches, which rely on norm-bounded perturbations or similarity-based neighborhoods, our generalized accuracy evaluates performance over relevant input regions, capturing the global probability distribution of the approximated NN rather than only local variations around individual data points.

In Section IV, we experimentally evaluate generalized accuracy for two NNs, a fully connected NN and a CNN, both trained on Fashion-MNIST with 32-bit weights and high standard accuracy, and approximated via uniform symmetric post-training b -bit weight quantization. Specifically, we validate its convergence with respect to dataset size and its sensitivity to weight bitwidth reduction. We demonstrate that the proposed definition of generalized accuracy provides stronger generalization guarantees than standard pointwise accuracy. Although evaluating the generalized accuracy may appear computationally demanding (e.g., solving hundreds of thousands of LPs), this computation represents a one-time offline cost that is substantially offset by energy savings during repeated deployments of the approximated NN compared to

the energy-intensive original NN. The proposed approach provides a more reliable assessment of approximation strategies and helps identify those that achieve the best trade-off between accuracy, performance, and generalization.

II. A FORMAL MODEL OF NNS AND SHORTCUT WEIGHTS

We present a formal framework for (artificial) feedforward neural networks (NNs, or simply networks) employing the ReLU (rectified linear unit) activation function. Our definition encompasses deep neural networks (DNNs) commonly used for classification tasks, including convolutional neural networks (CNNs).

The architecture of a network $\mathcal{N}$ is described by a connected, directed acyclic graph (V, E) with edges (connections) $E \subset V \times V$ and vertices V representing units, called neurons, each associated with a real-valued output (state) y_j for $j \in V$. Among these are n input neurons $X = \{1, \dots, n\} \subseteq V$, which serve solely to present an external n -dimensional real input $\mathbf{x} = (x_1, \dots, x_n)$ to $\mathcal{N}$ from a bounded input domain, assumed to be $[0, 1]^n$ without loss of generality [19]. Moreover, a set of at least two output neurons $Y \subseteq V' = V \setminus X$ produces the output of $\mathcal{N}$, where each $y_j \in [0, 1]$ for $j \in Y$ represents the probability that $\mathbf{x}$ belongs to class j , forming a categorical probability distribution over $|Y|$ classes. The neurons that are neither input nor output ones constitute the set of hidden neurons, $H = V \setminus (X \cup Y)$. For each neuron $j \in V$, we define $j_{\leftarrow} = \{i \in V \mid (i, j) \in E\}$ as the set of neurons that feed into j via edges, representing the inputs to j . We thus assume $j_{\leftarrow} = \emptyset$ for $j \in X$, and $j_{\leftarrow} \cap Y = \emptyset$ for all $j \in V$.

For any non-input neuron $j \in V' = H \cup Y$ and any of its input $i \in j_{\leftarrow}$, let $w_{ji} \in \mathbb{R}$ be a real-valued weight associated with the connection $(i, j) \in E$, whereas formally $w_{ji} = 0$ for $i \in V \setminus j_{\leftarrow}$. In addition, $w_{j0} \in \mathbb{R}$ denotes its real bias, which, as usual, can be viewed as a weight of an edge $(0, j) \in E$ leading from an additional formal input neuron $0 \in X$ to j , whose state is fixed to $y_0 = 1$ and $0 \in j_{\leftarrow}$ for every $j \in V'$ such that $w_{j0} \neq 0$. Furthermore, the excitation ξ_j of a non-input neuron $j \in V'$ is then defined as the weighted sum of the states of its inputs in $j_{\leftarrow}$:

$$\xi_j = \sum_{i \in j_{\leftarrow}} w_{ji} y_i, \quad (1)$$

assuming that these states y_i have already been determined for all neurons $i \in j_{\leftarrow}$.

When evaluating the network $\mathcal{N}$ on an external input $\mathbf{x} = (x_1, \dots, x_n)$, all neuron states are computed in a feedforward manner according to a topological ordering:

$$y_j = \begin{cases} x_j & \text{if } j \in X \\ R(\xi_j) & \text{if } j \in H \\ \frac{e^{\xi_j}}{\sum_{k \in Y} e^{\xi_k}} & \text{if } j \in Y \end{cases} \quad (2)$$

where $R: \mathbb{R} \rightarrow \mathbb{R}$ denotes the ReLU activation function:

$$R(\xi) = \max(0, \xi). \quad (3)$$

Note that the states y_j of output neurons $j \in Y$ normalize their excitations into a categorical probability distribution by using the softmax function.

The convolutional layers in CNNs are at times interlaced by max pooling layers whose units $j \in H$ implement the maximum of its inputs, $y_j = \max_{i \in j_{\leftarrow}} y_i$. Such a max pooling unit can be replaced in $\mathcal{N}$ by a subnetwork whose states are evaluated according to (1)–(3), since the maximum of two numbers $\max(x, y) = R(x - y) + y$ for $x, y \in \mathbb{R}$, can be used for evaluating the maximum of $|j_{\leftarrow}|$ inputs (e.g., the maxima of pairs is used to compute the maxima of fours, eights, sixteens, etc.) [38]. The alternative average pooling unit j that computes the average of its inputs, $y_j = \frac{1}{|j_{\leftarrow}|} \sum_{i \in j_{\leftarrow}} y_i$, can be viewed as a neuron $j \in H$ with the bias $w_{j0} = 0$ and weights $w_{ji} = 1/|j_{\leftarrow}|$ for $i \in j_{\leftarrow}$. Thus, we will hereafter assume without loss of generality that $\mathcal{N}$ does not contain pooling layers.

Furthermore, we introduce a concept of so-called shortcut weights. For any external input $\mathbf{x} = (x_1, \dots, x_n) \in [0, 1]^n$ to $\mathcal{N}$, we define a subset of hidden neurons,

$$A = A(\mathbf{x}) = \{j \in H \mid \xi_j > 0\} \quad (4)$$

whose states are *activated* with $y_j > 0$ according to (3). Let $A \subset H$ be a fixed set of activated units. Then the excitation ξ_j of any neuron $j \in V'$ is a linear function of the input neurons' states $y_1, \dots, y_n$, representing an external input for $\mathcal{N}$ (i.e., $y_i = x_i$ for $i \in X \setminus \{0\}$) that satisfies (4). The reason is that, in this case, only the linear parts of the ReLU function are used, namely, either $y_j = \xi_j$ if $j \in A$ or $y_j = 0$ if $j \in H \setminus A$, according to (3). Hence, for each neuron $j \in V'$, we can introduce its real-valued *shortcut weights* $W_{ji} \in \mathbb{R}$ for input neurons $i \in X$ (including its shortcut bias W_{j0}), which are coefficients of this linear function,

$$\xi_j = \sum_{i \in X} W_{ji} y_i \quad (5)$$

valid for any external input that satisfies (4). If condition (4) holds, then it can be rewritten using the shortcut weights as

$$\sum_{i \in X} W_{ji} y_i \begin{cases} > 0 & \text{if } j \in A \\ \leq 0 & \text{if } j \in H \setminus A, \end{cases} \quad (6)$$

according to (5).

For any unit $j \in V'$, its shortcut weight W_{ji} for an input neuron $i \in X$ can be expanded from (1)–(3) as the sum of products of weights $w_{j_1, j_0}, w_{j_2, j_1}, \dots, w_{j_p, j_{p-1}}$ over all paths $i = j_0, j_1, \dots, j_p = j$ in the graph (V, E) that lead from i to j through activated hidden units $j_1, \dots, j_{p-1} \in A$ in $\mathcal{N}$ (i.e., $(j_{\ell-1}, j_\ell) \in E$ for every $\ell \in \{1, \dots, p\}$),

$$W_{ji} = \sum_{\substack{\text{path } i=j_0, j_1, \dots, j_p=j \text{ in } (V, E) \\ j_1, \dots, j_{p-1} \in A}} \prod_{\ell=1}^p w_{j_\ell, j_{\ell-1}}. \quad (7)$$

The shortcut weight W_{ji} can thus be efficiently computed by forward propagation of the one-hot vector corresponding to input neuron i through $\mathcal{N}$, restricted to activated neurons in A .

In particular, for any non-input unit $j \in V'$, we calculate its shortcut weights as

$$W_{ji} = w_{ji} + \sum_{k \in j \leftarrow \cap A} w_{jk} W_{ki} \quad \text{for } i \in X, \quad (8)$$

provided that the shortcut weights W_{ki} have already been computed for all neurons $k \in j \leftarrow$ and $i \in X$.

III. GENERALIZED ACCURACY

Assume that $\mathcal{N}$ is a trained NN that perfectly generalizes to the dataset $T \subseteq [0, 1]^n$ for a given classification task. We thus assume that the standard *accuracy*, defined as the ratio of correctly classified data points in T by $\mathcal{N}$, satisfies $\alpha_T(\mathcal{N}) = 1$. We specify the conditions under which a network $\tilde{\mathcal{N}}$ can formally be considered an approximation of $\mathcal{N}$. Attributes of the approximated network $\tilde{\mathcal{N}}$, such as its sets of neurons, weights, excitations, and states, are denoted with a tilde. We impose natural minimal requirements on $\tilde{\mathcal{N}}$, namely that it shares the same input neurons $\tilde{X} = X$ and the same associated output classes as $\mathcal{N}$, which means the same number of output neurons $|\tilde{Y}| = |Y|$, since $\mathcal{N}$ and $\tilde{\mathcal{N}}$ perform the same classification task.

For each data point $\mathbf{x} = (x_1, \dots, x_n) \in T$ belonging to the class $c \in Y$, which is correctly classified by $\mathcal{N}$, we define the so-called *correct polytope* $\Xi_{\mathbf{x}} \subseteq [0, 1]^n$ as the intersection of half-spaces given by the following inequalities in the real-valued variables $y_1, \dots, y_n$, using the shortcut weights (5) of $\mathcal{N}$ and $\tilde{\mathcal{N}}$:

$$\xi_c - \xi_j = W_{c0} - W_{j0} + \sum_{i=1}^n (W_{ci} - W_{ji}) y_i \geq 0 \quad (9)$$

for all $j \in Y \setminus \{c\}$

$$\xi_j = W_{j0} + \sum_{i=1}^n W_{ji} y_i \begin{cases} \geq 0 & \text{if } j \in A(\mathbf{x}) \\ \leq 0 & \text{if } j \notin A(\mathbf{x}) \end{cases} \quad (10)$$

for all $j \in H$

$$\tilde{\xi}_j = \tilde{W}_{j0} + \sum_{i=1}^n \tilde{W}_{ji} y_i \begin{cases} \geq 0 & \text{if } j \in \tilde{A}(\mathbf{x}) \\ \leq 0 & \text{if } j \notin \tilde{A}(\mathbf{x}) \end{cases} \quad (11)$$

for all $j \in \tilde{H}$

$$0 \leq y_j \leq 1 \quad \text{for } j \in \{1, \dots, n\}, \quad (12)$$

which is a closed and bounded convex subset of the input space $[0, 1]^n$. In contrast to (6), zero excitations are allowed for activated neurons $j \in A$ to ensure the polytope is closed.

Note that $\mathcal{N}$ and $\tilde{\mathcal{N}}$ behave linearly on $\Xi_{\mathbf{x}}$, meaning that ξ_j for $j \in Y$ and $\tilde{\xi}_j$ for $j \in \tilde{Y}$ are linear in $\mathbf{y} = (y_1, \dots, y_n) \in \Xi_{\mathbf{x}}$ under (10) and (11), respectively, as follows from (4)–(6). In addition, $\mathbf{x} \in \Xi_{\mathbf{x}}$ according to (9) since $\mathcal{N}$ classifies $\mathbf{x}$ correctly as class $c = \arg \max_{j \in Y} y_j$ with the highest probability y_c , thereby ensuring that $\xi_c \geq \xi_j$ for all $j \in Y \setminus \{c\}$, by the softmax function employed in (2). In fact, the constraints (9) imply that all inputs $\mathbf{y} \in \Xi_{\mathbf{x}}$ are assigned by $\mathcal{N}$ to this correct class c , and hence $\Xi_{\mathbf{x}}$ is referred to as the correct polytope.

We further partition the correct polytope $\Xi_{\mathbf{x}} = \bigcup_{k \in Y} \tilde{\Xi}_{\mathbf{x}}^k$ to the convex subpolytopes $\tilde{\Xi}_{\mathbf{x}}^k$ where $\tilde{\mathcal{N}}$ infers the class $k \in Y$, which extends (9)–(12) by the following conditions:

$$\tilde{\xi}_k - \tilde{\xi}_j = \tilde{W}_{k0} - \tilde{W}_{j0} + \sum_{i=1}^n (\tilde{W}_{ki} - \tilde{W}_{ji}) y_i \geq 0 \quad (13)$$

for all $j \in \tilde{Y} \setminus \{k\}$.

Then, the performance of the approximated $\tilde{\mathcal{N}}$ compared to $\mathcal{N}$ could, in principle, be measured by the averaged ratio

$$\frac{1}{|T|} \sum_{\mathbf{x} \in T} \frac{\text{vol}(\tilde{\Xi}_{\mathbf{x}})}{\text{vol}(\Xi_{\mathbf{x}})} \quad (14)$$

of the volume of the subpolytope $\tilde{\Xi}_{\mathbf{x}} = \tilde{\Xi}_{\mathbf{x}}^c$, where $\tilde{\mathcal{N}}$ predicts the correct class c , to the volume of the correct polytope $\Xi_{\mathbf{x}}$, over all data points $\mathbf{x} \in T$.

However, in very high-dimensional settings (e.g., $n \sim 1000$), the exact computation of a convex polytope's volume becomes computationally intractable. In fact, this task is known to be #P-hard [20], [21] and, moreover, it cannot be approximated within any polynomial factor by deterministic polynomial-time algorithms [22], [23]. On the other hand, randomized polynomial-time methods achieving approximate solutions do exist [24], [25]. Nevertheless, their practical applicability is typically limited to moderate dimensions, roughly in the hundreds. As a result, one commonly resorts to alternative surrogate measures or approximation techniques to characterize the “size” of such sets.

As an alternative surrogate for estimating the volume of a bounded convex polytope $\Xi \subseteq [0, 1]^n$, for which a provable asymptotic error bound is available [27], one may use the notion of *mean width* [26]. It is defined as

$$\bar{d}(\Xi) = \frac{1}{\omega_{n-1}} \int_{\mathcal{S}_{n-1}} d(\Xi, \mathbf{u}) d\mathbf{u} \quad (15)$$

where $\mathcal{S}_{n-1}$ is the unit sphere in $\mathbb{R}^n$ with surface measure ω_{n-1} . The quantity

$$d(\Xi, \mathbf{u}) = \max_{\mathbf{y} \in \Xi} \sum_{i=1}^n u_i y_i - \min_{\mathbf{y} \in \Xi} \sum_{i=1}^n u_i y_i \quad (16)$$

represents the width of Ξ in the direction of a unit vector $\mathbf{u} = (u_1, \dots, u_n) \in \mathbb{R}^n$. This width can be computed by solving two linear programs (LPs) that determine the maximum and minimum of the objective function $\sum_{i=1}^n u_i y_i$ over the convex polytope Ξ . This polytope Ξ is defined by the constraints (9)–(12) for the correct polytope $\Xi_{\mathbf{x}}$ and is specified by (9)–(13) for its subpolytopes $\tilde{\Xi}_{\mathbf{x}}^k$ where $k \in Y$.

According to the law of large numbers, the mean width (15) can be estimated by taking an average over N directions $\mathbf{u}_r = (u_{r1}, \dots, u_{rn}) \in \mathbb{R}^n$, where each $\mathbf{u}_r$ is drawn independently and uniformly from the unit sphere $\mathcal{S}_{n-1}$ for $r \in \{1, \dots, N\}$:

$$\bar{d}(\Xi) \sim \tilde{d}(\Xi) = \frac{1}{N} \sum_{r=1}^N d(\Xi, \mathbf{u}_r). \quad (17)$$

In practice, this procedure can be implemented [28] by first sampling random vectors $\mathbf{u}'_r = (u'_{r1}, \dots, u'_{rn}) \in \mathbb{R}^n$ whose components are independent standard normal random variables (with zero mean and unit variance). The corresponding unit directions $\mathbf{u}_r$ uniformly distributed on $\mathcal{S}_{n-1}$ are then obtained via normalization of these Gaussian vectors $\mathbf{u}'_r$ for $r \in \{1, \dots, N\}$:

$$u_{ri} = \frac{u'_{ri}}{\sqrt{\sum_{\ell=1}^n (u'_{r\ell})^2}} \quad \text{for } i \in \{1, \dots, n\}. \quad (18)$$

Nevertheless, one cannot simply replace the volume in (14) by the estimated mean width, which is not additive with respect to a polytope partition, i.e., in general $\sum_{k \in Y} d(\tilde{\Xi}_x^k) \neq d(\Xi_x)$. Instead, we use the normalized mean width ratio

$$\frac{\tilde{d}(\tilde{\Xi}_x)}{\sum_{k \in Y} \tilde{d}(\tilde{\Xi}_x^k)}. \quad (19)$$

Moreover, the mean width $d(\tilde{\Xi}_x^k)$ of a subpolytope $\tilde{\Xi}_x^k$ may be positive even when its volume $\text{vol}(\tilde{\Xi}_x^k)$ is zero. For example, this frequently occurs when $\tilde{\Xi}_x^k \subseteq \Xi_x$ lies on the boundary of Ξ_x and thus does not have the full dimension n of the input space, significantly distorting the resulting ratio. We therefore use a modified estimated mean width that assigns zero to zero-volume polytopes having zero probability:

$$\tilde{d}_0(\Xi) = \begin{cases} \tilde{d}(\Xi) & \text{if } \text{vol}(\Xi) > 0 \\ 0 & \text{if } \text{vol}(\Xi) = 0. \end{cases} \quad (20)$$

The condition in (20) that the polytope Ξ has zero volume can be evaluated using the Chebyshev radius $\varrho(\Xi)$ of the largest n -dimensional ball contained entirely in Ξ , which satisfies $\varrho(\Xi) = 0$ iff $\text{vol}(\Xi) = 0$. In practice, the Chebyshev radius can be computed as the maximum value of ϱ in the LP over the $n+1$ variables $\mathbf{y} = (y_1, \dots, y_n)$ and $\varrho \geq 0$, subject to the constraints $\mathbf{a}_j \mathbf{y} + \|\mathbf{a}_j\| \varrho \leq b_j$ for each inequality $\mathbf{a}_j \mathbf{y} \leq b_j$ defining the polytope Ξ , where $\mathbf{a}_j \in \mathbb{R}^n$ and $b_j \in \mathbb{R}$. For $\Xi = \tilde{\Xi}_x^k$, these are the inequalities (9)–(13). For example, the inequalities (13) yield the constraints $\sum_{i=1}^n (\tilde{W}_{ji} - \tilde{W}_{ki}) y_i + \varrho \sqrt{\sum_{i=1}^n (\tilde{W}_{ji} - \tilde{W}_{ki})^2} \leq \tilde{W}_{k0} - \tilde{W}_{j0}$ for all $j \in \tilde{Y} \setminus \{k\}$.

Thus, the *generalized accuracy* $\gamma_T(\tilde{\mathcal{N}})$ of the approximated network $\tilde{\mathcal{N}}$ can be defined as the weighted average over all data points $\mathbf{x} \in T$ of the normalized mean width ratio (19). This ratio serves as a surrogate for the volume ratio in (14) using (20), with weights given by the sum of the mean widths of all the subpolytopes in the respective partition normalized over T . This reduces to the ratio of the aggregated mean width of the subpolytopes $\tilde{\Xi}_x$ to that of all subpolytopes $\tilde{\Xi}_x^k$ of Ξ_x across the dataset T :

$$\begin{aligned} \gamma_T(\tilde{\mathcal{N}}) &= \sum_{\mathbf{x} \in T} \frac{\sum_{k \in Y} \tilde{d}_0(\tilde{\Xi}_x^k)}{\sum_{\mathbf{y} \in T} \sum_{k \in Y} \tilde{d}_0(\tilde{\Xi}_y^k)} \frac{\tilde{d}_0(\tilde{\Xi}_x)}{\sum_{k \in Y} \tilde{d}_0(\tilde{\Xi}_x^k)} \\ &= \frac{\sum_{\mathbf{x} \in T} \tilde{d}_0(\tilde{\Xi}_x)}{\sum_{\mathbf{x} \in T, k \in Y} \tilde{d}_0(\tilde{\Xi}_x^k)}. \end{aligned} \quad (21)$$

Clearly, $0 \leq \gamma(\tilde{\mathcal{N}}) \leq 1$, where $\gamma(\tilde{\mathcal{N}}) = 1$ represents perfect approximation. Overall, generalized accuracy seems to provide the best feasible measure of agreement between $\tilde{\mathcal{N}}$ and the correctly generalizing network $\mathcal{N}$ over the dataset T , which approximates the input distribution, since evaluating the error over the entire input space is NP-hard [19].

However, generalized accuracy (21) depends on the size of the correct polytopes, which are constrained by the linear behavior of the approximated NN according to (11) and may therefore vary across different $\tilde{\mathcal{N}}$, making the value $\gamma_T(\tilde{\mathcal{N}})$ not directly comparable across different $\tilde{\mathcal{N}}$. This issue can be addressed by introducing a so-called *extended* correct polytope $\bar{\Xi}_x \supseteq \Xi_x$, in which only $\mathcal{N}$ (i.e., not necessarily $\tilde{\mathcal{N}}$) behaves linearly, as defined by constraints (9), (10), and (12), while relaxing constraint (11) in the definition of Ξ_x . Hence, $\bar{\Xi}_x$ is determined solely by the original $\mathcal{N}$ and is therefore identical for any approximated $\tilde{\mathcal{N}}$, which partitions $\bar{\Xi}_x$ into its own subpolytopes where it behaves linearly. These subpolytopes, which collectively cover $\bar{\Xi}_x$, are then considered as the correct polytopes for evaluating generalized accuracy (21). The constant total volume of $\bar{\Xi}_x$ ensures comparability of generalized accuracy across different $\tilde{\mathcal{N}}$, while the normalization (19) accounts for partition-dependent differences, despite approximating the volume ratios in (14) by ratios of non-additive mean widths in (21). In practice, we generate these correct polytopes by means of *data augmentation*, where additional data points are added to the dataset T within $\bar{\Xi}_x$. To this end, we can employ the standard *Hit-and-Run Markov Chain Monte Carlo* (MCMC) method (with no burn-in), which generates points uniformly within a given bounded convex polytope [39].

In particular, starting from each data point $\mathbf{x}_0 = \mathbf{x} \in T$, we construct a sequence of P points $\mathbf{x}_r = (x_{r1}, \dots, x_{rn}) \in \bar{\Xi}_x$, which defines the set $T'_x = \{\mathbf{x}_1, \dots, \mathbf{x}_P\}$, where P is sufficiently large, using the recursive formula

$$\mathbf{x}_r = \mathbf{x}_{r-1} + \lambda_r \mathbf{u}_r \quad \text{for } r \in \{1, \dots, P\}. \quad (22)$$

Here, $\mathbf{u}_r = (u_{r1}, \dots, u_{rn}) \in \mathbb{R}^n$ is a random direction drawn independently and uniformly from the unit sphere $\mathcal{S}_{n-1}$ according to (18), and λ_r is a real number chosen uniformly at random from the interval:

$$\lambda_r \in [\max M_r^-, \min M_r^+] \quad (23)$$

which is defined using the finite sets $M_r^- = \{\mu \in M_r \mid \mu \leq 0\}$ and $M_r^+ = \{\mu \in M_r \mid \mu \geq 0\}$ where $M_r = M_{r1} \cup M_{r2} \cup M_{r3}$ corresponds to constraints (9), (10), and (12), respectively, so that $\mathbf{x}_r \in \bar{\Xi}_x$. Namely, by plugging (22) into (9), (10) with $\mathbf{x}_{r-1} \in \bar{\Xi}_x$, and into (12), we obtain

$$M_{r1} = \left\{ \frac{\xi_c(\mathbf{x}_{r-1}) - \xi_j(\mathbf{x}_{r-1})}{\xi'_j(\mathbf{u}_r) - \xi'_c(\mathbf{u}_r)} \mid \xi'_j(\mathbf{u}_r) \neq \xi'_c(\mathbf{u}_r), j \in Y \setminus \{c\} \right\} \quad (24)$$

$$M_{r2} = \left\{ \frac{-\xi_j(\mathbf{x}_{r-1})}{\xi'_j(\mathbf{u}_r)} \mid \xi'_j(\mathbf{u}_r) \neq 0, j \in H \right\} \quad (25)$$

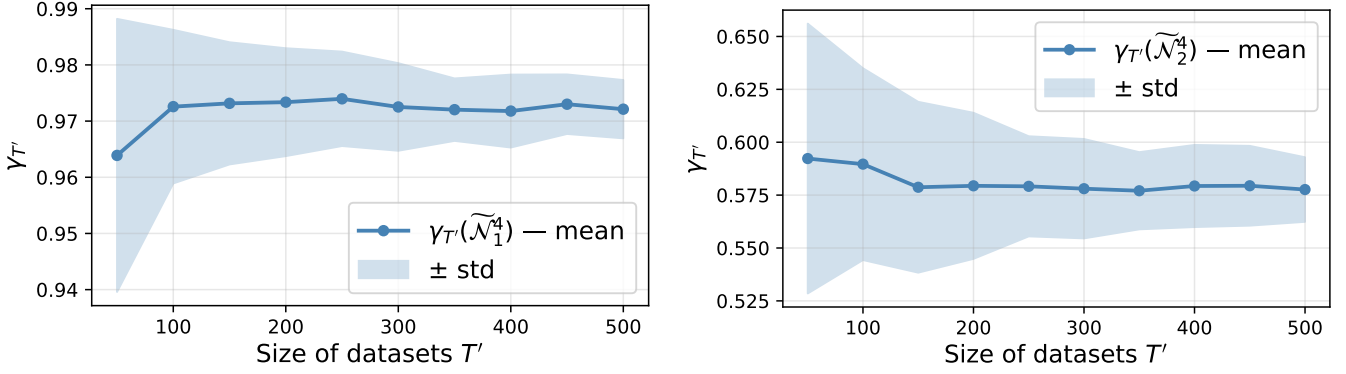


Fig. 1. Estimates of the generalized accuracy $\gamma_{T'}$ of the 4-bit quantized $\tilde{\mathcal{N}}_1^4$ (left) and $\tilde{\mathcal{N}}_2^4$ (right) for increasing sizes of the datasets T' .

$$M_{r3} = \left\{ \frac{-x_{r-1,j}}{u_{rj}}, \frac{1-x_{r-1,j}}{u_{rj}} \mid u_{rj} \neq 0, j \in \{1, \dots, n\} \right\} \quad (26)$$

where

$$\xi_j(\mathbf{x}_{r-1}) = W_{j0} + \sum_{i=1}^n W_{ji} x_{r-1,i} \quad \text{and} \quad (27)$$

$$\xi'_j(\mathbf{u}_r) = \sum_{i=1}^n W_{ji} u_{ri} \quad \text{for } j \in H \cup Y. \quad (28)$$

We define the equivalence relation $\sim$ on the set T'_x as $\mathbf{x}_r \sim \mathbf{x}_s$ iff $\tilde{A}(\mathbf{x}_r) = \tilde{A}(\mathbf{x}_s)$, for $\mathbf{x}_r, \mathbf{x}_s \in T'_x$, which induces identical correct polytopes $\Xi_{\mathbf{x}_r} = \Xi_{\mathbf{x}_s} \subseteq \tilde{\Xi}_{\mathbf{x}}$. Let $T_x \subseteq T'_x$ be a set of representatives of the equivalence classes of T'_x under $\sim$. Then, we can augment the original dataset T as $T_P = T \cup \bigcup_{\mathbf{x} \in T} T_x$ where parameter P is the number of points randomly generated by MCMC method. Clearly, $T_0 = T$, and $T^* = \lim_{P \rightarrow \infty} T_P$ is a finite dataset, since the index of the equivalence relation $\sim$, determined by $\tilde{A}(\mathbf{x}) \subseteq \tilde{H}$, is bounded by $2^{|\tilde{H}|}$. The MCMC method also guarantees that, for each $\mathbf{x} \in T$, the extended polytope $\tilde{\Xi}_{\mathbf{x}}$ is covered by its correct subpolytopes $\Xi_{\mathbf{y}}$ for some $\mathbf{y} \in T^*$. The revisited generalized accuracy $\gamma_{T^*}(\tilde{\mathcal{N}})$ of the approximated $\tilde{\mathcal{N}}$ is thus defined by (21) evaluated over the augmented dataset T^* , and is estimated by $\gamma_{T_P}(\tilde{\mathcal{N}})$ for sufficiently large P , e.g., when the estimate stabilizes within a prescribed tolerance.

IV. EXPERIMENTS

We evaluated our new definition of generalized accuracy (21), introduced in Section III, on two NNs, both trained on the Fashion-MNIST dataset [40], which consists of 28×28 grayscale images of clothing items categorized into 10 classes. The computational experiments were conducted in Python using the PyTorch [41] and SciPy [42] libraries for NN implementation and linear programming (HiGHS solver), respectively. The source code is publicly available [43]. The program was run on the Jean Zay supercomputer (IDRIS/CNRS, France). Each data point was processed as an independent Slurm array task allocated to a dedicated CPU node (Intel

Xeon Gold 6248, 40 physical CPU cores at 2.5 GHz, 192 GB RAM), with tens to hundreds of tasks executing concurrently (400–4,000 CPU cores) depending on queue availability.

The first, $\mathcal{N}_1$, is a fully connected NN with five layers of sizes 784–1024–512–256–128–10, including $n = 784 = 28^2$ input units representing image pixels and $m = 10$ output neurons corresponding to the classes. The second is a CNN composed of two convolutional layers with 10 and 20 kernels of size 3×3 (stride 1, padding 1), each followed by a 2×2 max pooling layer (stride 2), then flattened to 980 units, and ending with two fully connected layers with 96 and 10 neurons, respectively. This CNN is further transformed to a network $\mathcal{N}_2$ with ten layers of sizes 784–7840–7840–3920–1960–3920–3920–1960–980–96–10 by implementing the max pooling layers using ReLU neurons (see Section II).

Both NNs were originally trained on the Fashion-MNIST dataset with 60,000 training and 10,000 validation data points, using 32-bit floating-point weights. The first achieves an accuracy of 94.47% after 50 training epochs (with dropout of 0.3), while the second achieves 98.83% after 100 epochs. For evaluating generalized accuracy in the subsequent experiments, we retained only the randomly sampled dataset T , consisting of 1,000 out of 56,334 training data points correctly classified by both $\mathcal{N}_1$ and $\mathcal{N}_2$ with $\alpha_T(\mathcal{N}_1) = \alpha_T(\mathcal{N}_2) = 1$. The approximated networks $\tilde{\mathcal{N}}_1^b$ and $\tilde{\mathcal{N}}_2^b$ for $b \in \{4, 6, 8, 10, 12, 16\}$ were obtained from $\mathcal{N}_1$ and $\mathcal{N}_2$, respectively, via uniform symmetric post-training b -bit quantization applied to each weight:

$$\tilde{w}_{ji}^b = \Delta_\ell \cdot \min \left(\max \left(\text{round} \left(\frac{w_{ji}}{\Delta_\ell} \right), -2^{b-1} \right), 2^{b-1} - 1 \right)$$

for all $j \in V_\ell$ and $i \in j_\leftarrow$, where $\Delta_\ell = \max_{j \in V_\ell, i \in j_\leftarrow} \frac{|w_{ji}|}{2^{b-1}-1}$ is the scale parameter associated with layer $V_\ell \subset V$.

First, we verified the correctness of the definition of generalized accuracy by assessing whether it converges to a stable value as the number of data points increases. We randomly sampled subsets $T' \subset T$, with sample sizes ranging from 50 to 500, repeating each sampling 100 times. For each T' , the generalized accuracy $\gamma_{T'}(\tilde{\mathcal{N}}_1^4)$ and $\gamma_{T'}(\tilde{\mathcal{N}}_2^4)$ was evaluated for the 4-bit quantized $\tilde{\mathcal{N}}_1^4$ and $\tilde{\mathcal{N}}_2^4$, according to (21), where the mean width was estimated over $N = 200$ random directions in (17).

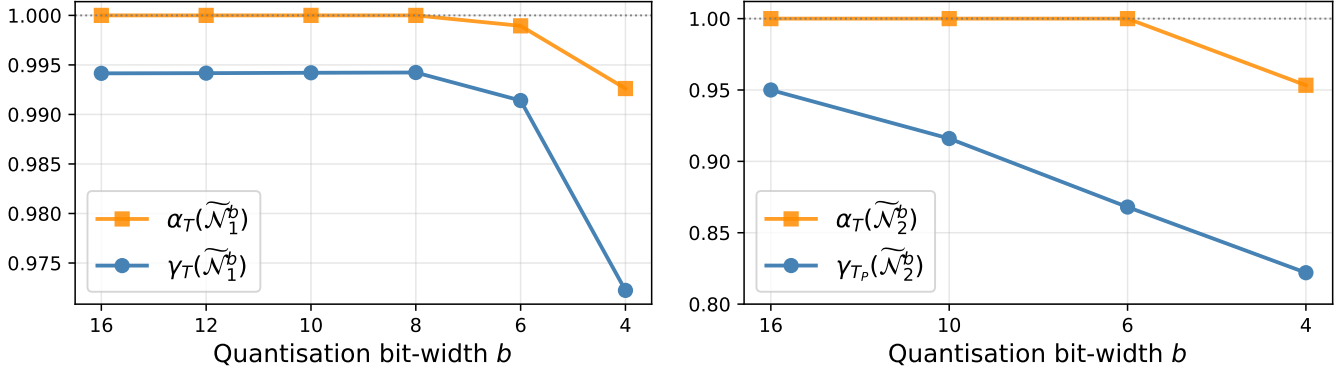


Fig. 2. The generalized accuracy γ_T compared with the standard accuracy α_T for $\tilde{\mathcal{N}}_1^b$ (left) and $\tilde{\mathcal{N}}_2^b$ (right), respectively, for decreasing weight bitwidth b .

The results of this experiment required solving altogether more than 1.6 million linear programs (LPs), each with 5,426 (for $\mathcal{N}_1$) or 66,458 (for $\mathcal{N}_2$) constraints (9)–(13) in 784 variables, to estimate the mean width of, on average, two nonempty subpolytope $\tilde{\Xi}_x^k$ for each of 1,000 data points x in T .

The averaged results shown in Fig. 1 indicate that it is not necessary to use the entire dataset to obtain reliable estimates of generalized accuracy, as the improvement obtained by using more than $|T'| = 150$ sampled data points is only marginal and $N = 100$ random directions also appear to be sufficient for robust mean width approximations. This significantly reduces time required to determine generalized accuracy, e.g., from solving more than 200 thousand LPs for $|T'| = 500$ and $N = 200$, to only about 30 thousand for $|T'| = 150$ and $N = 100$. More importantly, this computational cost is incurred only once, offline, while the resulting generalized accuracy assessment supports the selection of higher-quality approximated NNs, whose repeated deployment results in significant energy savings compared to the original NN.

In the next experiment, we demonstrate the effect of reducing the weight bitwidth b on the generalized accuracy γ_T compared to the standard pointwise accuracy α_T for the approximated $\tilde{\mathcal{N}}_1^b$ for $b \in \{4, 6, 8, 10, 12, 16\}$ and $\tilde{\mathcal{N}}_2^b$ for $b \in \{4, 6, 10, 16\}$. For the smaller network $\tilde{\mathcal{N}}_1^b$ the original dataset T was sufficient to achieve comparable values of $\gamma_T(\tilde{\mathcal{N}}_1^b)$ across different b , since, in this case, the correct-to-extended polytope mean width ratio $\tilde{d}(\Xi_x)/\tilde{d}(\Xi_x)$ was close to 1 for all $x \in T$. The computation took about 5 minutes of wall-clock time per data point on 40 cores of a single node. For $|T| = 1,000$ data points and $N = 200$ random directions, evaluating $\alpha_T(\tilde{\mathcal{N}}_1^b)$ consumed on average roughly 510 CPU-hours per approximated network $\tilde{\mathcal{N}}_1^b$. Fig. 2 illustrates that $\gamma_T(\tilde{\mathcal{N}}_1^b)$ decreases with decreasing b , and that this trend is more pronounced than for $\alpha_T(\tilde{\mathcal{N}}_1^b)$ at small values of b .

On the other hand, the larger network $\tilde{\mathcal{N}}_2^b$ required data augmentation to obtain a comparable evaluation of $\gamma_T(\tilde{\mathcal{N}}_2^b)$, since the ratio $\tilde{d}(\Xi_x)/\tilde{d}(\Xi_x)$ was around 0.5 for $b = 4$. In this case, we sampled a dataset T of size $|T| = 150$, which should be sufficient based on the previous experiment (cf. Fig. 1). For each such data point $x = x_0 \in T$, we

generated a sufficient number of P points $x_r \in T'_x$ within the extended polytope $\tilde{\Xi}_x$ using the MCMC method (22)–(28), so that at most five additional nonequivalent representatives $y \in T_x \subseteq T'_x$, corresponding to pairwise disjoint correct subpolytopes $\Xi_y \subseteq \tilde{\Xi}_x$, were obtained. These subpolytopes together with Ξ_x , approximately cover $\tilde{\Xi}_x$. In fact, fewer than five additional representatives were found except for $b = 4$, where five sufficed for the generalized accuracy to stabilize with only minor variation. This resulted in augmented datasets T_P of sizes $|T_P| = 159, 158, 206, 900$ for $b = 16, 10, 6, 4$, respectively, over which generalized accuracy $\gamma_{T_P}(\tilde{\mathcal{N}}_2^b)$ was evaluated according to (21).¹ For $N = 100$ random directions, computing $\alpha_T(\tilde{\mathcal{N}}_2^b)$ required an average of 8800 CPU-hours per approximated network $\tilde{\mathcal{N}}_2^b$, corresponding to 35 minutes per data point on 40 cores of a single node. Fig. 2 shows that the values of $\gamma_{T_P}(\tilde{\mathcal{N}}_2^b)$ are well below those of $\alpha_T(\tilde{\mathcal{N}}_2^b)$, revealing reduced generalization in the approximated NNs, similarly to $\tilde{\mathcal{N}}_1^b$. Clearly, the generalized accuracy represents a finer measure of accuracy, as it also takes into account the neighborhoods of data points and more closely reflects the generalization capabilities of the approximated NNs.

V. CONCLUSION

In this work, we introduced generalized accuracy, a measure that extends standard pointwise accuracy by evaluating the behavior of approximated NNs over neighborhoods of data points. These neighborhoods are defined as convex polytopes where both the original and approximated NNs are locally linear, and their evaluation is based on a surrogate geometric quantity, the mean width, estimated via randomized directions and linear programming. To ensure enhanced comparability of

¹During the computation of generalized accuracy, the LP used to compute the Chebyshev radius in (20) encountered numerical difficulties with the HiGHS solver, even though the LPs used to estimate the mean widths of the subpolytopes $\tilde{\Xi}_x^k$ in (21) were solved efficiently and reliably with high numerical precision. We therefore used an ad hoc heuristic to test whether the Chebyshev radius is zero, exploiting the fact that a polytope and its subpolytope must coincide if they have the same mean width [26], thereby proving that the remaining subpolytopes have zero volume. In the final version, we plan to use a more numerically robust LP solver, such as the Gurobi optimizer [44].

generalized accuracy across different approximated NNs, data augmentation using the MCMC method can be employed.

Experimental results on fully connected and convolutional NNs trained on Fashion-MNIST show that generalized accuracy is more sensitive to NN approximation than standard accuracy, particularly under weight quantization. While point-wise accuracy remains largely stable, generalized accuracy reveals a clearer degradation in local consistency and robustness as the weight bitwidth is reduced. These findings indicate that the proposed measure provides a more informative characterization of generalization under approximation and can help assess different efficiency-accuracy tradeoffs in NN design. Although determining generalized accuracy is computationally demanding, this cost is incurred only once offline and can be amortized through the energy savings achieved by repeatedly deploying a high-quality approximated NN. Further research in this direction could facilitate the development of techniques for identifying DNN components suitable for approximation or removal, aiming to reduce energy consumption while maintaining explicit and reliable error guarantees.

REFERENCES

- [1] V. Sze, Y. Chen, T. Yang, and J. S. Emer, *Efficient Processing of Deep Neural Networks*, ser. Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, 2020.
- [2] N. P. Jouppi, C. Young, N. Patil, and D. A. Patterson, "A domain-specific architecture for deep neural networks," *Commun ACM*, vol. 61, no. 9, pp. 50–59, 2018.
- [3] C. Silvano *et al.*, "A survey on deep learning hardware accelerators for heterogeneous HPC platforms," *CoRR*, arXiv:2306.15552 [cs.AR], 2023.
- [4] J. Šíma, P. Vidnerová, and V. Mrázek, "Energy complexity of convolutional neural networks," *Neural Comput*, vol. 36, no. 8, pp. 1601–1625, 2024.
- [5] G. Armeniakos, G. Zervakis, D. Soudris, and J. Henkel, "Hardware approximate techniques for deep neural network accelerators: A survey," *ACM Comput Surv*, vol. 55, no. 4, p. 83, 2023.
- [6] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proc IEEE*, vol. 108, no. 4, pp. 485–532, 2020.
- [7] Z. Li, H. Li, and L. Meng, "Model compression for deep neural networks: A survey," *Computers*, vol. 12, no. 3, p. 60, 2023.
- [8] Z. Lyu *et al.*, "A survey of model compression strategies for object detection," *Multimedia Tools Appl*, vol. 83, no. 16, pp. 48 165–48 236, 2024.
- [9] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput Surv*, vol. 48, no. 4, p. 62, 2016.
- [10] Y. Tang *et al.*, "A survey on transformer compression," *CoRR*, arXiv:2402.05964 [cs.LG], 2024.
- [11] Y. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J Solid State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.
- [12] T. Yang, Y. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *Proc. IEEE CVPR 2017*, 2017, pp. 6071–6079.
- [13] Y. Kim *et al.*, "Compression of deep convolutional neural networks for fast and low power mobile applications," in *Proc. ICLR 2016*, 2016.
- [14] B. Moons and M. Verhelst, "An energy-efficient precision-scalable ConvNet processor in 40-nm CMOS," *IEEE J Solid State Circuits*, vol. 52, no. 4, pp. 903–914, 2017.
- [15] E. H. Lee, D. Miyashita, E. Chai, B. Murmann, and S. S. Wong, "LogNet: Energy-efficient neural networks using logarithmic computation," in *Proc. IEEE ICASSP 2017*, 2017, pp. 5900–5904.
- [16] P. Wang and J. Cheng, "Fixed-point factorized networks," in *Proc. IEEE CVPR 2017*, 2017, pp. 3966–3974.
- [17] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and Huffman coding," in *Proc. ICLR 2016*, 2016.
- [18] M. S. Ansari *et al.*, "Improving the accuracy and hardware efficiency of neural networks using approximate multipliers," *IEEE Trans Very Large Scale Integr Syst*, vol. 28, no. 2, pp. 317–328, 2020.
- [19] J. Šíma and P. Vidnerová, "Weight-rounding error in deep neural networks," in *Proc. ECML PKDD 2025, Part IV*, ser. LNAI, vol. 16016. Springer, 2026, pp. 398–416.
- [20] L. G. Khachiyan, "The problem of calculating the volume of a polyhedron is enumerably hard," *Russ Math Surv*, vol. 44, no. 3, pp. 199–200, 1989.
- [21] M. E. Dyer and A. M. Frieze, "On the complexity of computing the volume of a polyhedron," *SIAM J Comput*, vol. 17, no. 5, pp. 967–974, 1988.
- [22] G. Elekes, "A geometric inequality and the complexity of computing volume," *Discrete Comput Geom*, vol. 1, no. 4, pp. 289–292, 1986.
- [23] I. Bárány and Z. Füredi, "Computing the volume is difficult," *Discrete Comput Geom*, vol. 2, no. 4, pp. 319–326, 1987.
- [24] M. E. Dyer, A. M. Frieze, and R. Kannan, "A random polynomial-time algorithm for approximating the volume of convex bodies," *J ACM*, vol. 38, no. 1, pp. 1–17, 1991.
- [25] L. Lovász and S. Vempala, "Simulated annealing in convex bodies and an $o(n^4)$ volume algorithm," *J Comput Syst Sci*, vol. 72, no. 2, pp. 392–417, 2006.
- [26] R. Schneider, *Convex Bodies: The Brunn-Minkowski Theory*, 2nd ed., ser. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2013.
- [27] K. J. Böröczky, F. Fodor, and D. Hug, "The mean width of random polytopes circumscribed around a convex body," *J London Math Soc*, vol. 81, no. 2, pp. 499–523, 2010.
- [28] M. E. Muller, "A note on a method for generating points uniformly on n -dimensional spheres," *Commun ACM*, vol. 2, no. 4, pp. 19–20, 1959.
- [29] J. Šíma and P. Vidnerová, "Cross-entropy loss of approximated deep neural networks," in *Proc. ICONIP 2025, Part I*, ser. LNCS, vol. 16309. Springer, 2026, pp. 460–475.
- [30] L. Ferrer, "Analysis and comparison of classification metrics," *CoRR*, arXiv:2209.05355 [cs.LG], 2022.
- [31] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: an overview," *CoRR*, arXiv:2008.05756 [stat.ML], 2020.
- [32] J. R. Terven, D. Córdova-Esparza, J. Romero-González, A. Ramirez-Pedraza, and E. A. Chavez-Urbíola, "A comprehensive survey of loss functions and metrics in deep learning," *Artif Intell Rev*, vol. 58, no. 7, p. 195, 2025.
- [33] Ž. Vujović, "Classification model evaluation metrics," *Int J Adv Comput Sci Appl*, vol. 12, no. 6, pp. 599–606, 2021.
- [34] R. Luo *et al.*, "Local calibration: Metrics and recalibration," in *Proc. UAI 2022*, ser. PMLR, vol. 180, 2022, pp. 1286–1295.
- [35] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *Proc. ICML 2019*, ser. PMLR, vol. 97, 2019, pp. 1310–1320.
- [36] C. Sun, Y. Mao, M. N. Mueller, and M. Vechev, "Average certified radius is a poor metric for randomized smoothing," in *Proc. ICML 2025*, ser. PMLR, vol. 267, 2025, pp. 57 651–57 666.
- [37] T. Weng *et al.*, "Evaluating the robustness of neural networks: An extreme value theory approach," in *Proc. ICLR 2018*, 2018. [Online]. Available: <https://openreview.net/forum?id=BkUHIMZ0b>
- [38] J. Šíma and J. Cabessa, "The power of max pooling layer," in *Proc. ICANN 2025, Part I*, ser. LNCS, vol. 16068. Springer, 2026, pp. 221–233.
- [39] R. L. Smith, "Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions," *Oper Res*, vol. 32, no. 6, pp. 1296–1308, 1984.
- [40] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," *CoRR*, arXiv:1708.07747 [cs.LG], 2017.
- [41] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," *CoRR*, arXiv:1912.01703 [cs.LG], 2019.
- [42] P. Virtanen *et al.*, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, no. 1, pp. 261–272, 2020.
- [43] J. Cabessa, "Source code for experiments," 2026. [Online]. Available: <https://github.com/JeremCab/ErrorVolumePolytopes>
- [44] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2026. [Online]. Available: <https://www.gurobi.com>