Cross-Entropy Loss of Approximated Deep Neural Networks

Jiří Šíma
sima@cs.cas.cz

joint work with **Petra Vidnerová**
petra@cs.cas.cz

Institute of Computer Science
Czech Academy of Sciences, Prague, Czechia

Efficient Processing of Deep Neural Networks (DNNs)

(Sze, Chen, Yang, Emer, Morgan & Claypool Publishers, 2020)

The energy efficiency of DNNs on low-power, battery-operated embedded hardware (e.g., cellphones, smartwatches, augmented reality glasses) is a critical challenge.

$\longrightarrow$ **reducing the energy cost of DNNs:**

1. Specialized **Hardware Accelerators** for DNN inference, based on GPUs, FPGAs, in-memory computing, etc.

2. Approximate Computing in error-tolerant applications (e.g. image classification) save large amounts of energy with minimal accuracy loss by reducing:

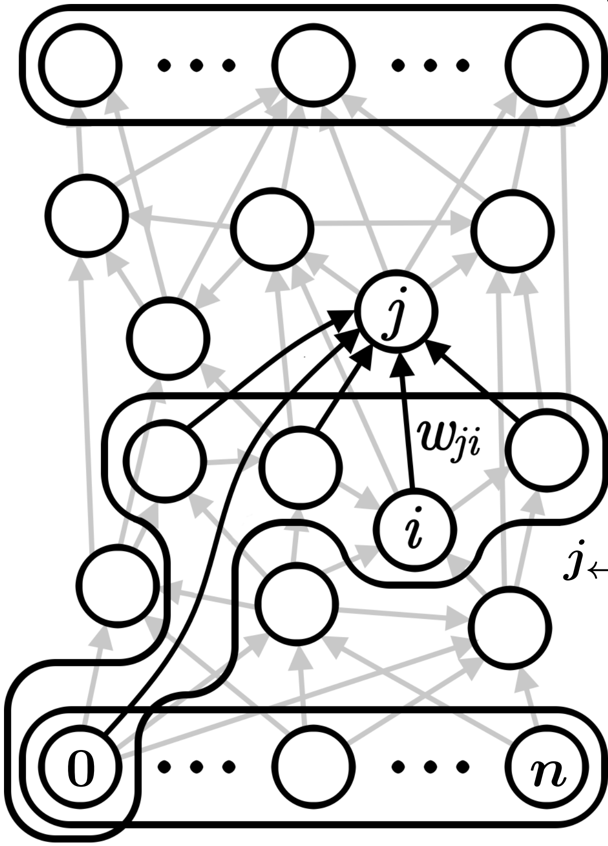
- Model Size: pruning, compression, weight sharing, approximate multipliers
- Arithmetic Precision: fixed-point operations, reduced weight bit-width, nonuniform quantization

The aim of this study: Estimate the maximum cross-entropy loss of approximated classification DNNs.

Formal Model of DNNs

The architecture of a DNN $\mathcal{N}$ is a connected directed acyclic graph (V, E) composed of neurons, where edges $(i, j) \in E \subset V \times V$ are labeled with weights $w_{ji} \in \mathbb{R}$.

$Y \subseteq V$ output neurons



$$y_j = \begin{cases} x_j & \text{if } j \in X \\ R(\xi_j) & \text{if } j \in V \setminus (X \cup Y) \\ \frac{e^{\xi_j}}{\sum_{k \in Y} e^{\xi_k}} & \text{if } j \in Y \text{ sofmax function} \end{cases}$$

output (state) of neuron $j \in V$

$R(\xi) = \text{ReLU}(\xi) = \max(0, \xi)$ activation function

$\xi_j = \sum_{i \in j_\leftarrow} w_{ji} y_i$ excitation of $j \in V \setminus X$

$j_\leftarrow = \{i \in V \mid (i, j) \in E\}$ inputs to neuron $j \in V \setminus X$

$V \setminus (X \cup Y)$ hidden neurons

$X = \{0, 1, \ldots, n\} \subseteq V$ input neurons including formal $0 \in X$ ($y_0 \equiv 1$) for biases w_{j0} for $j \in V \setminus X$

$\uparrow$

$(x_1, \ldots, x_n) \in [0, 1]^n$ external input to $\mathcal{N}$

w.l.o.g., excluding (max) pooling layers ($\max(y_1, y_2) = R(y_1 - y_2) + y_2$)

1. Tool: Shortcut Weights

The excitation ξ_j of any neuron $j \in V \setminus X$ is a continuous piecewise linear function of the external input (due to ReLU is piecewise linear)

$$\to \quad \xi_j = \sum_{i \in X} W_{ji}\, y_i \quad \text{for } (y_1, \ldots, y_n) \in \Xi$$

within a subset $\Xi \subset [0,1]^n$ of the input space, where W_{ji} are referred to as the shortcut weights (bias) from input neurons $i \in X$ to neuron $j \in V \setminus X$.

For an input $(x_1, \ldots, x_n) \in [0,1]^n$, let $\Xi_S \subset [0,1]^n$ be its neighborhood within which ξ_j are linear for all $j \in V \setminus X$ under fixed shortcut weights, where

$$S = S(x_1, \ldots, x_n) = \{j \in V \setminus (X \cup Y) \mid \xi_j < 0\}$$

denotes the set of hidden neurons saturated at zero output, $y_j = R(\xi_j) = 0$.

$\to$ Ξ_S is a convex polytope—an intersection of finitely many half-spaces:

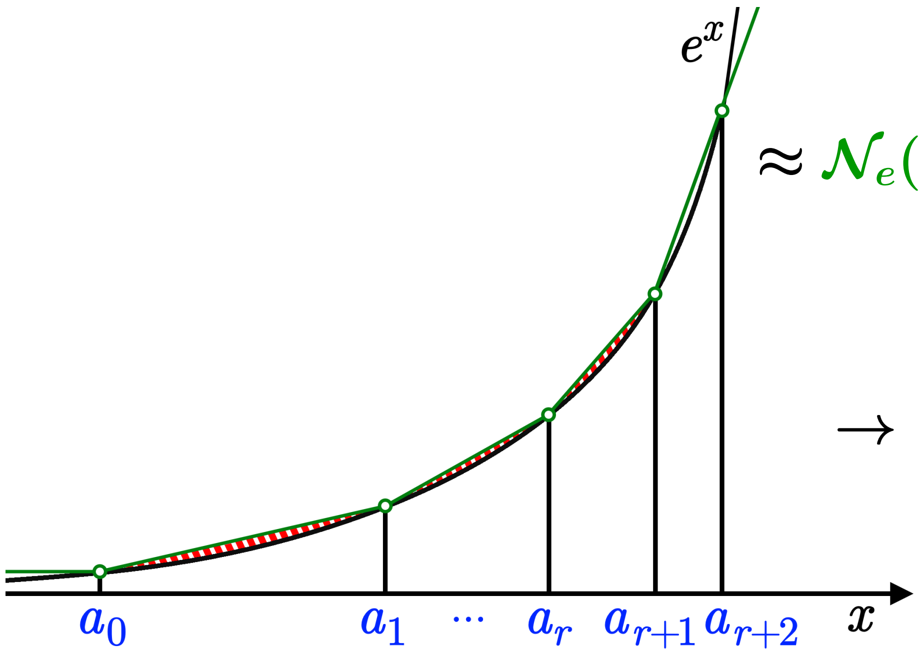
$$\xi_j = \sum_{i \in X} W_{ji}\, y_i \begin{cases} < 0 & \text{if } j \in S \\ \geq 0 & \text{if } j \notin S \end{cases} \quad \text{for } j \in V \setminus (X \cup Y)$$

$$0 \leq y_i \leq 1 \quad \text{for } i \in X$$

Efficient Computation of the shortcut weights via feedforward propagation (skipping S) for $j \in V \setminus X$:

$$W_{ji} = \sum_{k \in j_{\leftarrow} \setminus S} w_{jk}\, W_{ki} \ \text{ for all } i \in X \quad \left(W_{ki} = \begin{cases} 1 & \text{if } k = i \\ 0 & \text{otherwise} \end{cases} \ \text{ for } k, i \in X \right)$$

2. Tool: Continuous Piecewise Linear Interpolation of e^x



$$\approx \mathcal{N}_e(x) = \begin{cases} e^{a_0} & \text{if } x \le a_0 \\ m_i x + b_i & \text{if } a_i < x \le a_{i+1} \\ m_{r+1}x + b_{r+1} & \text{if } x > a_{r+2} \end{cases}$$

$\to$

- $e^x \le \mathcal{N}_e(x)$ for $x \in (-\infty, a_{r+2}]$
- $e^{a_i} = \mathcal{N}_e(a_i)$ for $i \in \{0, \ldots, r+2\}$

Theorem. *There are* r *unique points* $a_1, \ldots, a_r$ *inside* $[a_0, a_{r+1}]$ *that minimize*

$$\sum_{i=0}^{r} \int_{a_i}^{a_{i+1}} (m_i x + b_i - e^x)\, dx \qquad \left(\to \quad e^{a_i} = \frac{e^{a_{i+1}} - e^{a_{i-1}}}{a_{i+1} - a_{i-1}}\right).$$

Evaluating the linear interpolation using ReLU networks:

$$\mathcal{N}_e(x) = e^{a_0} + \sum_{i=0}^{r+1} R\left(m_i x + b_i - e^{a_i}\right) - \sum_{i=0}^{r} R\left(m_i x + b_i - e^{a_{i+1}}\right)$$

Cross-Entropy Loss of Approximated Classification DNNs

$\widetilde{\mathcal{N}}$ is an approximated DNN of $\mathcal{N}$, sharing the same input neurons ($\widetilde{X} = X$) and the same number of output neurons ($|\widetilde{Y}| = |Y|$) (tilde denotes parameters of $\widetilde{\mathcal{N}}$)

e.g., the weights of $\widetilde{\mathcal{N}}$ are rounded to a given number of binary digits in their floating-point representations

The classification error of $\widetilde{\mathcal{N}}$ for an input $(x_1, \dots, x_n) \in [0,1]^n$, measured by the cross-entropy loss between the softmax categorical probability distributions of $\mathcal{N}$ and $\widetilde{\mathcal{N}}$:

$$L(x_1, \dots, x_n) = -\sum_{k \in Y} y_k \ln \widetilde{y_k}$$

(upper bounds the Kullback-Leibler divergence of $\mathcal{N}$ from $\widetilde{\mathcal{N}}$)

Theorem. *It is NP-hard to find the maximum of the cross-entropy loss L over the input space $[0,1]^n$.*

Upper-Bounding L Using ReLU Networks $\widetilde{\mathcal{N}_{ck}}$ & $\mathcal{N}_c$

within the convex polytope Ξ_S around a data point $(x_1, \ldots, x_n) \in T$ of category $c \in Y$ from a test/training set $T \subset [0,1]^n$, where $S = S(x_1, \ldots, x_n)$, restricted to inputs in Ξ_S that are classified by $\mathcal{N}$ into the category c with probability at least p (e.g., $p = 0.8$):

$$y_c \geq p \tag{1}$$

$$\overset{\text{softmax}}{\longleftarrow} \sum_{j \in Y \setminus \{c\}} e^{\xi_j - \xi_c} \overset{\xi_j - \xi_c \leq 0 \leq a_{r+2}}{\leq} \sum_{j \in Y \setminus \{c\}} \mathcal{N}_e\left(\xi_j - \xi_c\right) = \mathcal{N}_c(x_1, \ldots, x_n) \leq \frac{1}{p} - 1$$

$$L(x_1, \ldots, x_n) = \sum_{k \in Y} y_k \ln \frac{1}{\widetilde{y_k}} \leq y_c \ln \frac{1}{\widetilde{y_c}} + (1 - y_c) \max_{k \in \widetilde{Y} \setminus \{c\}} \ln \frac{1}{\widetilde{y_k}}$$

$$\overset{\text{softmax}}{=} \max_{k \in \widetilde{Y} \setminus \{c\}} \ln \sum_{j \in \widetilde{Y}} e^{\widetilde{\xi_j} - \widetilde{\xi_k} + y_c\left(\widetilde{\xi_k} - \widetilde{\xi_c}\right)} \overset{(1)}{\leq} \ln \max_{k \in \widetilde{Y} \setminus \{c\}} \sum_{j \in \widetilde{Y}} e^{\widetilde{\xi_j} - \widetilde{\xi_k} + R\left(\widetilde{\xi_k} - \widetilde{\xi_c}\right) - pR\left(\widetilde{\xi_c} - \widetilde{\xi_k}\right)}$$

$$\overset{\substack{\widetilde{\xi_j} - \widetilde{\xi_k} + R\left(\widetilde{\xi_k} - \widetilde{\xi_c}\right) \\ -pR\left(\widetilde{\xi_c} - \widetilde{\xi_k}\right) \leq a_{r+2}}}{\leq} \ln \max_{k \in \widetilde{Y} \setminus \{c\}} \sum_{j \in \widetilde{Y}} \mathcal{N}_e\left(\widetilde{\xi_j} - \widetilde{\xi_k} + R\left(\widetilde{\xi_k} - \widetilde{\xi_c}\right) - pR\left(\widetilde{\xi_c} - \widetilde{\xi_k}\right)\right)$$

$$= \ln \max_{k \in \widetilde{Y} \setminus \{c\}} \widetilde{\mathcal{N}_{ck}}(x_1, \ldots, x_n)$$

AppMax Method

Input: DNN $\mathcal{N}$, its approximation $\widetilde{\mathcal{N}}$, $(x_1,\ldots,x_n)\in T$ of category $c\in Y$

Output: an upper bound on $\max\limits_{(y_1,\ldots,y_n)\in\bigcup_{k\in\widetilde{Y}\setminus\{c\}}\Xi_k^*} L(y_1,\ldots,y_n)$

Algorithm: For each $k\in\widetilde{Y}\setminus\{c\}$ do

- Compose $\mathcal{N}^*$ from $\widetilde{\mathcal{N}_{ck}}$ & $\mathcal{N}_c$.
- Determine the saturated neurons $S^*=S^*(x_1,\ldots,x_n)$ in $\mathcal{N}^*$.
- Compute the shortcut weights W_{ji}^* of $\mathcal{N}^*$ for all $j\in V^*\setminus X^*$ and $i\in X^*$.
- Solve the linear program (LP) to find $(y_1,\ldots,y_n)$ that

maximize $\widetilde{\mathcal{N}_{ck}}(y_1,\ldots,y_n)$ $(\to\sigma_k)$ over the polytope $\Xi_k^*\subseteq\overline{\Xi}_{S^*}$

defined by $\xi_j^*=\sum\limits_{i\in X}W_{ji}^*y_i\begin{cases}\leq 0 & \text{if } j\in S^*\\ \geq 0 & \text{if } j\notin S^*\end{cases}$ for $j\in V^*\setminus(X^*\cup Y^*)$,

$$\mathcal{N}_c(x_1,\ldots,x_n)\leq\frac{1}{p}-1 \qquad \left(\text{where } e^{a_0}\leq\frac{1}{p}-1\leq e^{a_{r+2}}\right),$$

$$\widetilde{\xi}_j-\widetilde{\xi}_k+R\left(\widetilde{\xi}_k-\widetilde{\xi}_c\right)-pR\left(\widetilde{\xi}_c-\widetilde{\xi}_k\right)\leq a_{r+2}\,,\quad (y_1,\ldots,y_n)\in[0,1]^n.$$

and output $\ln\max_{k\in\widetilde{Y}\setminus\{c\}}\sigma_k$

Experiments

- **DNN:** $\mathcal{N}$ with 3 fully connected layers (784–64–32–10) and 32-bit weights, trained on the MNIST dataset; approximated as $\widetilde{\mathcal{N}}$ with rounded 4 bit weights
- **Software Libraries:** PyTorch (deep learning), SciPy (linear programming)
- **Source Code:** publicly available at `https://github.com/PetraVidnerova/ClassificationRoundingErrors`
- **Test Set:** T with 1135 data points of class $c = 1$, on which $\mathcal{N}$ and $\widetilde{\mathcal{N}}$ achieve 100% and 99.91% accuracy, respectively
- **Linear Interp. of e^x:** $r = 14$ optimal points & $a_0 = -5$, $a_{r+1} = 5$, $a_{r+2} = 20$

Number of "misclassified polytopes" around data points in T, including so-called misclassified inputs with the maximum estimated upper bound of cross-entropy loss, which are classified correctly as $c = 1$ by $\mathcal{N}$ with probability at least p, but misclassified as $\kappa \neq c$ by $\widetilde{\mathcal{N}}$:

p	0.30	0.40	0.50	0.60	0.70	0.80	0.90	0.95	0.99
misclassified	134	71	30	11	8	0	0	0	0

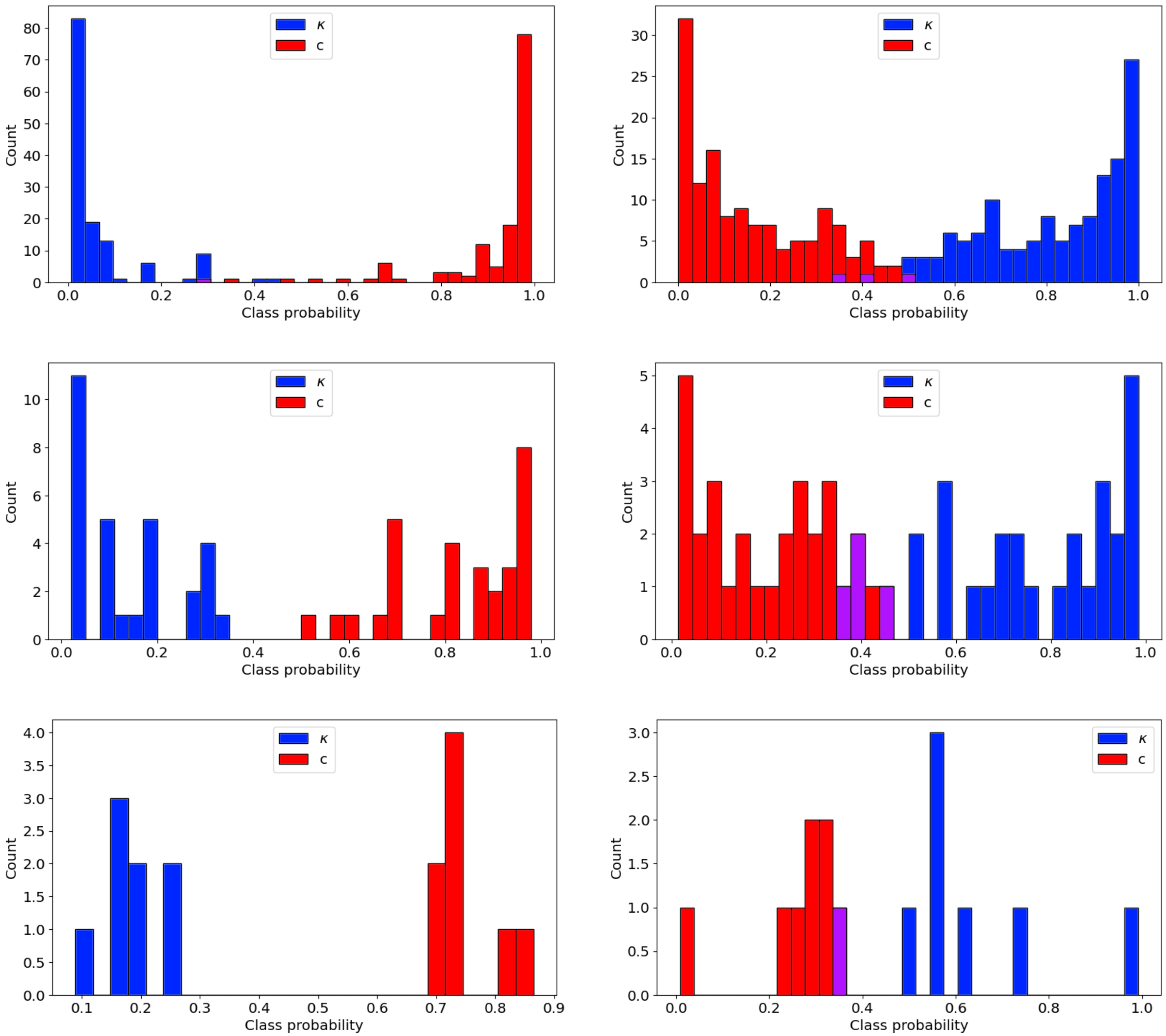
Histograms of probabilities y_c and y_κ over the misclassified inputs:

$\mathcal{N}$ $\widetilde{\mathcal{N}}$

$p = 0.3$

$p = 0.5$

$p = 0.7$



Future Research Directions

- Broaden AppMax evaluation to other datasets (e.g., CIFAR-100, ImageNet).
- The most suitable error variant (KL divergence, cross-entropy loss, accuracy) and its optimal upper bound.
- AppMax for classification DNNs with softmax, via nonlinear Karush-Kuhn-Tucker optimization.
- Approximate global error by estimating the probabilities of convex polytopes from their volumes, measured using the average mean width (evaluated by LP).
- Identify DNN components that can be neglected (e.g., specific weights to be rounded) under explicitly bounded output error.