Weight-Rounding Error in Deep Neural Networks

Jiří Šíma & Petra Vidnerová

ECML PKDD 2025

Efficient Processsing of Deep Neural Networks

(See: Sze,Chen,Yang,Emer: Efficient Processing of Deep Neural Networks, 2020)

- The **energy efficiency** of DNN implementations on **low-power, battery-operated hardware** (e.g. cellphones, smartwatches, smart glasses) becomes crucial
- **Possible solutions:** efficient hardware design and approximate computing

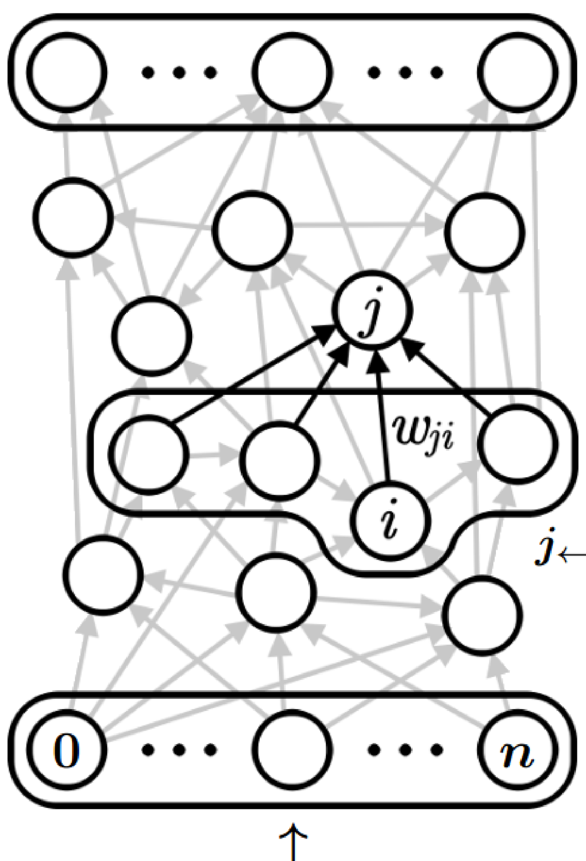
Approximate Computing in error-tolerant applications (e.g. image classification) save large amounts of energy with minimal accuracy loss by reducing:

- **model size** – pruning, compression, weight sharing, approximate multipliers
- **arithmetic precision** – fixed-point operations, reduction of weight bit-width, nonuniform quantization

The aim of this study: theoretical analysis of the effect of (post-training) weight rounding on DNN output to guarantee **maximum error bounds**

Formal Model of DNN

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

$$y_j = \begin{cases} x_j & \text{if } j \in X \\ R(\xi_j) & \text{if } j \in V \setminus (X \cup Y) \\ \xi_j & \text{if } j \in Y \end{cases}$$ output (state) of neuron $j \in V$

$R(\xi) = \mathrm{ReLU}(\xi) = \max(0, \xi)$ 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 $j \in V \setminus X$

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

$(x_1, \ldots, x_n) \in \mathbb{R}^n$ external input to $\mathcal{N}$

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

Regression Error of Approximated DNN

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

$\rightarrow$ regression error under L_1 norm for an external input $(x_1, \ldots, x_n) \in \mathbb{R}^n$

$$E(x_1, \ldots, x_n) = \sum_{j \in Y} |y_j - \tilde{y}_j| = \sum_{j \in Y} \left|\xi_j - \tilde{\xi}_j\right|$$

Weight Rounding — an important example of approximated $\widetilde{\mathcal{N}}$:

$$\widetilde{w_{ji}} = w_{ji} + \delta_{ji} \quad \text{for } j \in V \setminus X \ \& \ i \in j_\leftarrow$$

where $\delta_{ji} \in \mathbb{R}$ is a real rounding error of weight w_{ji}

Theorem: It is NP-hard to find the maximum error of approximated DNNs (for any approximation, not only weight rounding).

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)

$\rightarrow$ within a subset $\Xi \subseteq [0,1]^n$ of the input space, the excitation is a linear function of the input-neuron states:

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

where W_{ji} are the coefficients of the linear function, referred to as the shortcut weights (bias) from input neurons $i \in X$ to neuron $j \in V \setminus X$

for input $(x_1, \ldots, x_n) \in [0,1]^n$, its neighborhood Ξ_S is defined so that ξ_j are linear for all $j \in V \setminus X$ with fixed shortcut weights, where

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

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

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

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

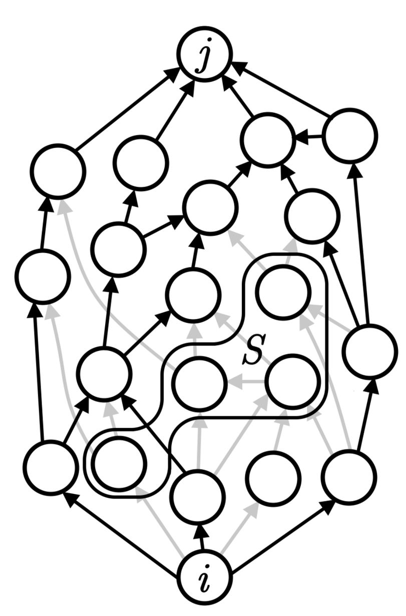
$$0 \leq y_i \leq 1 \quad \text{for } i \in X$$

Calculating shortcut weights

The shortcut weight W_{ji} represents the cumulative influence from input neuron $i \in X$ to neuron $j \in V \setminus X$, corresponding to the product of weights along all connecting unsaturated paths in $\mathcal{N}$:

$$W_{ji} = \sum_{\substack{\text{paths } i = j_0, j_1, \ldots, j_m = j \text{ in } (V,E) \\ j_1, \ldots, j_{m-1} \notin S}} \prod_{\ell=1}^{m} w_{j_\ell j_{\ell-1}}$$

The shortcut weights can be calculated efficiently via feed-forward propagation.



Estimating the Maximum Error of Approximated DNNs

- Approximating the maximum or average error using data points from the training or test set T

$$E_T = \max_{(x_1,\ldots,x_n) \in T} E(x_1, \ldots, x_n), \quad \overline{E_T} = \frac{1}{|T|} \sum_{(x_1,\ldots,x_n) \in T} E(x_1, \ldots, x_n)$$

- Refining the error estimate using the maximum over the convex polytope $\Xi_{S(x_1,\ldots,x_n)}$ surrounding the data point $(x_1, \ldots, x_n) \in T$:

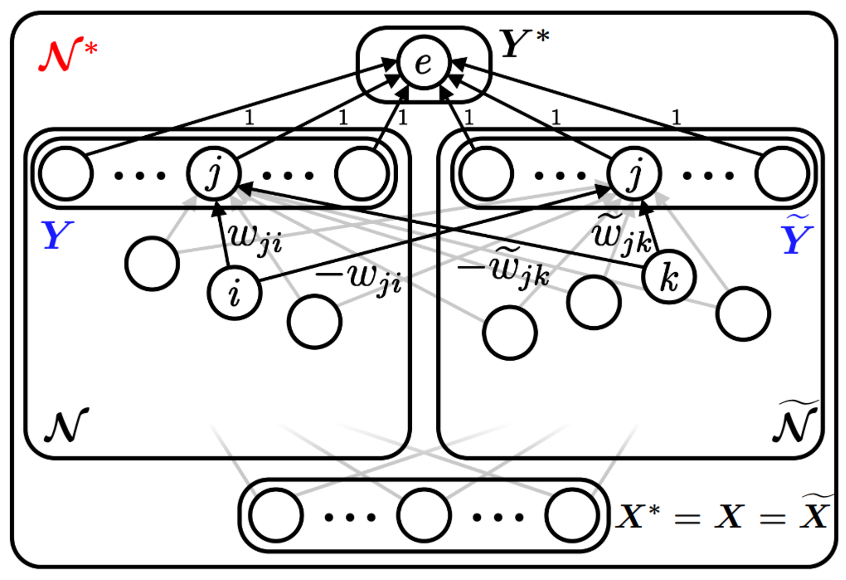
$$E_{\Xi_{S(T)}} = \max_{(x_1,\ldots,x_n) \in T} E_{\Xi_{S(x_1,\ldots,x_n)}}, \quad \overline{E_{\Xi_{S(T)}}} = \frac{1}{|T|} \sum_{(x_1,\ldots,x_n) \in T} E_{\Xi_{S(x_1,\ldots,x_n)}}$$

where

$$E_{\Xi_{S(x_1,\ldots,x_n)}} = \max_{(y_1,\ldots,y_n) \in \Xi_{S(x_1,\ldots,x_n)}} E(y_1, \ldots, y_n)$$

AppMax Method for Computing $E_{\Xi_{S^*(x_1,\ldots,x_n)}} = \max_{(y_1,\ldots,y_n) \in \Xi_{S^*(x_1,\ldots,x_n)}} E(y_1, \ldots, y_n)$

- Construct $\mathcal{N}^*$ from $\mathcal{N}$ & $\widetilde{\mathcal{N}}$:



$$\xi_j^* = \xi_j - \tilde{\xi}_j \quad \text{for } j \in Y$$
$$= \sum_{i \in j_\leftarrow} w_{ji} y_i - \sum_{i \in j_\leftarrow} \widetilde{w_{ji}}\, \tilde{y}_i$$

$$\xi_j^* = \tilde{\xi}_j - \xi_j \quad \text{for } j \in \widetilde{Y}$$
$$= \sum_{k \in j_\leftarrow} \widetilde{w_{jk}}\, y_k - \sum_{k \in j_\leftarrow} w_{jk} \tilde{y}_k$$

$$y_e^* = \xi_e^* = \sum_{j \in Y} y_j^* + \sum_{j \in \widetilde{Y}} y_j^* = \sum_{j \in Y} R\left(\xi_j^*\right) + \sum_{j \in \widetilde{Y}} R\left(\xi_j^*\right)$$
$$= \sum_{j \in Y} \left(R\left(\xi_j - \tilde{\xi}_j\right) + R\left(\tilde{\xi}_j - \xi_j\right) \right) = \sum_{j \in Y} \left|\xi_j - \tilde{\xi}_j\right| = E(x_1, \ldots, x_n)$$

- Determine the saturated neurons $S^* = S^*(x_1, \ldots, x_n)$
- Compute the shortcut weights W_{ji}^* of $\mathcal{N}^*$ for all $j \in V^* \setminus X^*$ and $i \in X^*$
- Solve the linear program to find the input-neuron states $y_1, \ldots, y_n$ that

$$\text{maximize} \quad y_e^* = \sum_{i \in X} W_{ei}^* y_i \quad \rightarrow \quad E_{\Xi_{S^*(x_1,\ldots,x_n)}}$$

$$\text{subject to} \quad \xi_j^* = \sum_{i \in X} W_{ji}^* y_i \begin{cases} \leq 0 & \text{if } j \in S^* \\ \geq 0 & \text{if } j \notin S^* \end{cases} \quad \text{for } j \in V^* \setminus (X^* \cup Y^*)$$
$$0 \leq y_i \leq 1 \quad \text{for } i \in X^*$$

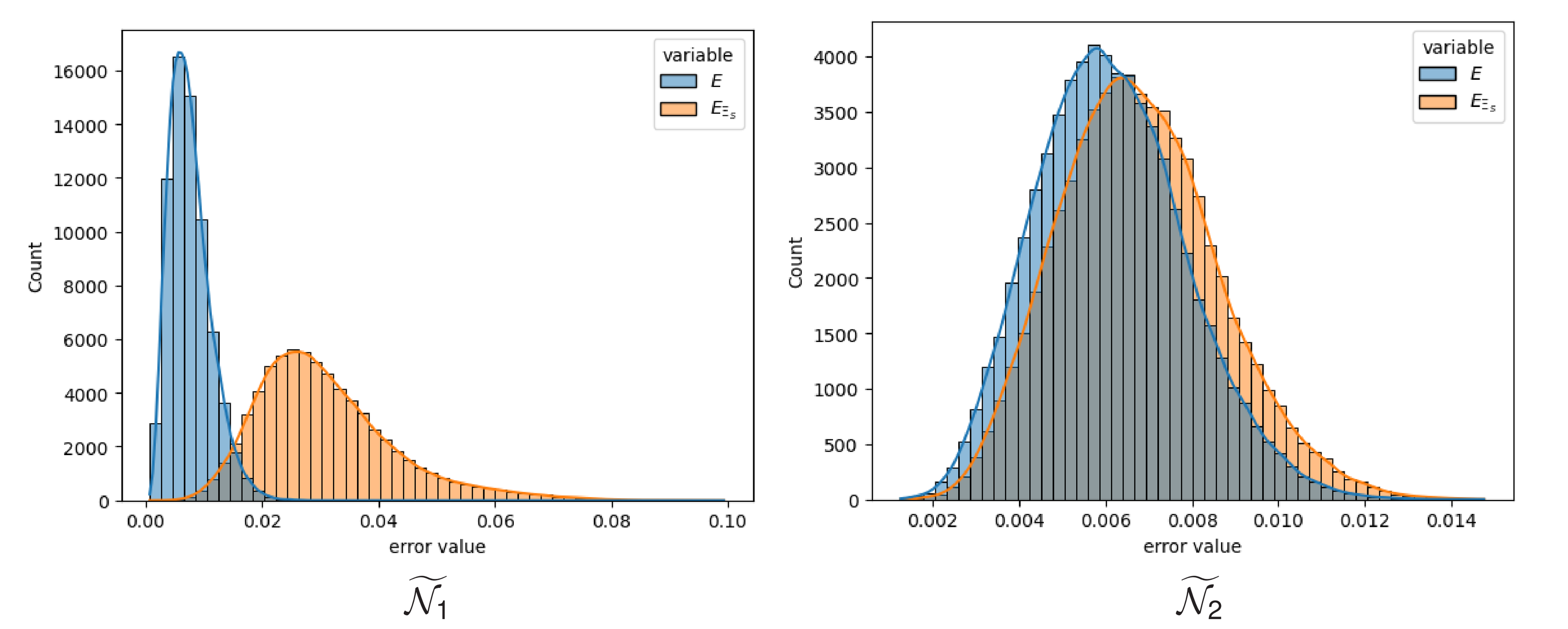
Experiments

- MNIST dataset, PyTorch & SciPy(linprog)
- **Source Code:** `https://github.com/PetraVidnerova/RoundingErrorEstimation`
- **DNNs:** trained on MNIST with 32-bit weights
  1. fully connected NN $\mathcal{N}_1$: 3 FC layers 784–2000–1000–10
  2. convolutional NN $\mathcal{N}_2$: 2 convolutional layers with 32 and 64 3x3-kernels (stride 1, padding 1), 1 max pooling layer with 64 2x2-kernels (stride 2), 2 FC layers (1024–10)
  $\rightarrow$ 8 FC layers 784–25088–50176–50176–25088–25088–12544–1024–10

	E_T	$E_{\Xi_{S(T)}}$	$\overline{E_T}$	$\overline{E_{\Xi_{S(T)}}}$
$\widetilde{\mathcal{N}}_1$	0.032854	0.099374	0.007629	0.030884
$\widetilde{\mathcal{N}}_2$	0.013466	0.014763	0.006127	0.006777

- weights rounded to **16 bits**
- T with 70,000 data points

Error Histograms: E at data points in T **vs.** E_{Ξ_S} over convex polytopes Ξ_S surrounding points in T



Reducing the Sample Size for AppMax

Error estimates E_{T_s} and $E_{\Xi_{S(T_s)}}$ for random samples $T_s \subset T$ of increasing size (50–60,000), averaged over 100 trials: