Supplementary Material: Local Competition and Stochasticity for Adversarial Robustness in Deep Learning

Konstantinos P. Panousis[†] **Sotirios Chatzis**[†] **Antonios Alexos**[‡] **Sergios Theodoridis**[§]

[†]Cyprus University of Technology, Limassol, Cyprus
[‡]University of California Irvine, CA, USA
[§]National and Kapodistrian University of Athens, Athens, Greece & Aalborg University, Denmark
k.panousis@cut.ac.cy

1 Experimental Setup

To allow for transparency and comparability to related recent literature, we adopt the experimental setup of (Verma and Swami, 2019). For development, we use Tensorflow (Abadi et al., 2016), enabling GPU support for faster computations. Additionally, in order to test the proposed model against adversarial attacks, we use the Cleverhans package (Papernot et al., 2016). All experiments are performed on a workstation comprising an i7-7820X CPU, 64 GB RAM and an NVidia Quadro P5000.

1.1 Training Setup

As mentioned in the main text, we use Stochastic Gradient Variational Bayes (SGVB) in order to train the model. Thus, we can employ off-the-shelf stochastic optimizers; we choose ADAM (Kingma and Welling, 2014) with its default settings.

For the evaluation of our proposed approach, we use the MNIST and CIFAR-10 datasets. For MNIST, we train the *Standard* and *Ensemble* models for a maximum of 100 epochs with a learning rate of $3e$-4. We follow an analogous procedure for the CIFAR dataset with a learning rate of $1e$-4. Similar to Verma and Swami (2019), for training we add zero-mean Gaussian noise with a standard deviation of 0.3 for MNIST and 0.032 for CIFAR-10.

As far as the loss function of the different considered networks is concerned, we employ: (i) the standard cross-entropy loss for Softmax, (ii) the binary cross-entropy for Logistic, and (iii) the hinge-loss for networks operating with Hadamard codes.

For all the considered posterior concrete relaxations employed during training, i.e. the posterior distribution of the latent winning indicators $\boldsymbol{\xi}_n,\ n = 1, \ldots, N$, as well as the posteriors over the latent variables Z, we use a *constant* temperature of 0.67, as suggested in Maddison et al. (2016). We have had no convergence issues with this selection, whatsoever.

1.2 Network Setup

As mentioned before, in order to allow for comparison to recent state-of-the-art approaches, we use the same experimental setup as in Verma and Swami (2019). Thus, the considered network architectures are the ones described in Table 1 of the main text, and implemented similar to the code of Verma and Swami (2019)[1].

Table 1: Standard Architecture for MNIST dataset

Layer Type	Parameters
Conv2D	kernels=64, kernel_size=(5,5), strides=(1,1)
Conv2D	kernels=64, kernel_size=(3,3), strides=(2,2)
Conv2D	kernels=64, kernel_size=(3,3), strides=(1,1)
Conv2D	kernels=64, kernel_size=(3,3), strides=(2,2)
Conv2D	kernels=64, kernel_size=(3,3), strides=(1,1)
Conv2D	kernels=64, kernel_size=(3,3), strides=(2,2)
Dense	units=128
Dense	units=64
Dense	units=64
Dense	units=10

Table 2: Standard Architecture for CIFAR-10 dataset

Layer Type	Parameters
Conv2D	kernels=32, kernel_size=(5,5), strides=(1,1)
Conv2D	kernels=32, kernel_size=(5,5), strides=(1,1)
Conv2D	kernels=32, kernel_size=(3,3), strides=(2,2)
Conv2D	kernels=64, kernel_size=(3,3), strides=(1,1)
Conv2D	kernels=64, kernel_size=(3,3), strides=(1,1)
Conv2D	kernels=64, kernel_size=(3,3), strides=(2,2)
Conv2D	kernels=128, kernel_size=(3,3), strides=(1,1)
Conv2D	kernels=128, kernel_size=(3,3), strides=(1,1)
Conv2D	kernels=128, kernel_size=(3,3), strides=(2,2)
Dense	units=128
Dense	units=64
Dense	units=64
Dense	units=10

1.2.1 Standard Network Architectures

Tables 1 and 2 present the convolutional and fully-connected layers of the Standard Architecture, for MNIST and CIFAR-10 datasets, respectively. Every Convolutional layer is followed by a Batch Normalization layer, except for the last one. All the conventional layers have been replaced by their respective LWTA and IBP-driven variants, introduced in the main text.

1.2.2 Ensemble Network Architectures

Tables 3 and 4 depict the Ensemble architectures for the MNIST and CIFAR-10 datasets, respectively. On each table, after the first double lines, the network splits into four branches. Every branch comprises convolutional and densely connected layers. Finally, a dense layer with linear activation outputs the logits. Every Convolutional layer before the double lines is once again followed by a Batch Normalization layer, except for the last one. The Convolutional layers and the Dense layers in between do not include a Batch Normalization layer. Again, all conventional definitions of the layers have been replaced by their respective LWTA and IBP-based variants proposed in this work.

2 Effect of Block Size, U

Since in our ablation study, in Section 4.3.1 of the main text, we considered blocks with $U = 2$ competing units, we repeat here similar experiments considering blocks of $U = 4$ competitors. The obtained results are presented

[1]https://github.com/Gunjan108/robust-ecoc

Table 3: Ensemble Architecture for MNIST dataset. The layers between the two liners consist the Ensemble Networks and are repeated 4 times.

Layer Type	Parameters
Conv2D	kernels=32, kernel_size=(5,5), strides=(1,1)
Conv2D	kernels=32, kernel_size=(5,5), strides=(1,1)
Conv2D	kernels=32, kernel_size=(3,3), strides=(2,2)
Conv2D	kernels=64, kernel_size=(3,3), strides=(1,1)
Conv2D	kernels=64, kernel_size=(3,3), strides=(1,1)
Conv2D	kernels=64, kernel_size=(3,3), strides=(2,2)
Conv2D	kernels=128, kernel_size=(3,3), strides=(1,1)
Conv2D	kernels=128, kernel_size=(3,3), strides=(1,1)
Conv2D	kernels=128, kernel_size=(3,3), strides=(2,2)
Conv2D	kernels=4, kernel_size=(5,5), strides=(2,2)
Conv2D	kernels=4, kernel_size=(3,3), strides=(2,2)
Conv2D	kernels=4, kernel_size=(3,3), strides=(2,2)
Conv2D	kernels=4, kernel_size=(2,2), strides=(1,1)
Conv2D	kernels=4, kernel_size=(2,2), strides=(1,1)
Dense	units=16
Dense	units=8
Dense	units=4
Dense	units=2
Dense	units=1

Table 4: Ensemble Architecture for CIFAR-10 dataset. The layers between the two liners consist the Ensemble Networks and are repeated 4 times.

Layer Type	Parameters
Conv2D	kernels=32, kernel_size=(5,5), strides=(1,1)
Conv2D	kernels=32, kernel_size=(5,5), strides=(1,1)
Conv2D	kernels=32, kernel_size=(3,3), strides=(2,2)
Conv2D	kernels=64, kernel_size=(3,3), strides=(1,1)
Conv2D	kernels=64, kernel_size=(3,3), strides=(1,1)
Conv2D	kernels=64, kernel_size=(3,3), strides=(2,2)
Conv2D	kernels=128, kernel_size=(3,3), strides=(1,1)
Conv2D	kernels=128, kernel_size=(3,3), strides=(1,1)
Conv2D	kernels=128, kernel_size=(3,3), strides=(2,2)
Conv2D	kernels=16, kernel_size=(5,5), strides=(2,2)
Conv2D	kernels=16, kernel_size=(3,3), strides=(2,2)
Conv2D	kernels=16, kernel_size=(3,3), strides=(2,2)
Conv2D	kernels=16, kernel_size=(2,2), strides=(1,1)
Conv2D	kernels=16, kernel_size=(2,2), strides=(1,1)
Dense	units=16
Dense	units=8
Dense	units=4
Dense	units=2
Dense	units=1

in Tables 5 and 6. For the MNIST dataset, we observe in Table 5 that only the Softmax and Tanh16 networks produce high accuracy scores in the majority of the attacks. In the case of CIFAR-10, we observe that all networks yield accuracy much inferior to the case of $U = 2$.

Table 5: Accuracy scores with $U = 4$ competing units on MNIST.

Network	Params	Benign	PGD	CW	BSA	Rand
Softmax	327,380	0.9613	0.905	0.97	0.95	0.187
Logistic	327,380	0.538	0.48	0.46	0.41	0.407
Tanh16	328,352	0.9432	0.895	0.93	0.94	0.43
LogisticEns10	205,150	0.1369	0.123	0.08	0.17	0.53

Table 6: Accuracy scores with $U = 4$ competing units on CIFAR-10.

Network	Params	Benign	PGD	CW	BSA	Rand
Softmax	772,628	0.5532	0.506	0.51	0.59	0.025
Logistic	772,628	0.4241	0.381	0.32	0.41	0.177
Tanh16	773,600	0.5097	0.46	0.55	0.6	0.368
LogisticEns10	1,197,998	0.4662	0.447	0.43	0.51	0.255

3 Change of the output logit values

Similar to the illustrations of Fig. 3 in the main text, we here revisit how the classifier output logits change in the context of an adversarial attack to our model. Thus, we again consider the Softmax network trained on the MNIST and CIFAR-10 datasets. In Fig. 1, we present the change in the output logit values of our model trained on MNIST for three randomly selected examples from the MNIST test set, under a PGD attack. The corresponding results from three randomly selected test examples considering the CIFAR-10 dataset are depicted in Fig.1. In both cases, we observe that our approach exhibits inconsistent and varying changes of the logit values, obstructing the attacker from distorting the dominant class. In contrast, the ReLU-based counterpart exhibits a smooth change, allowing the attacker to successfully attack the model.

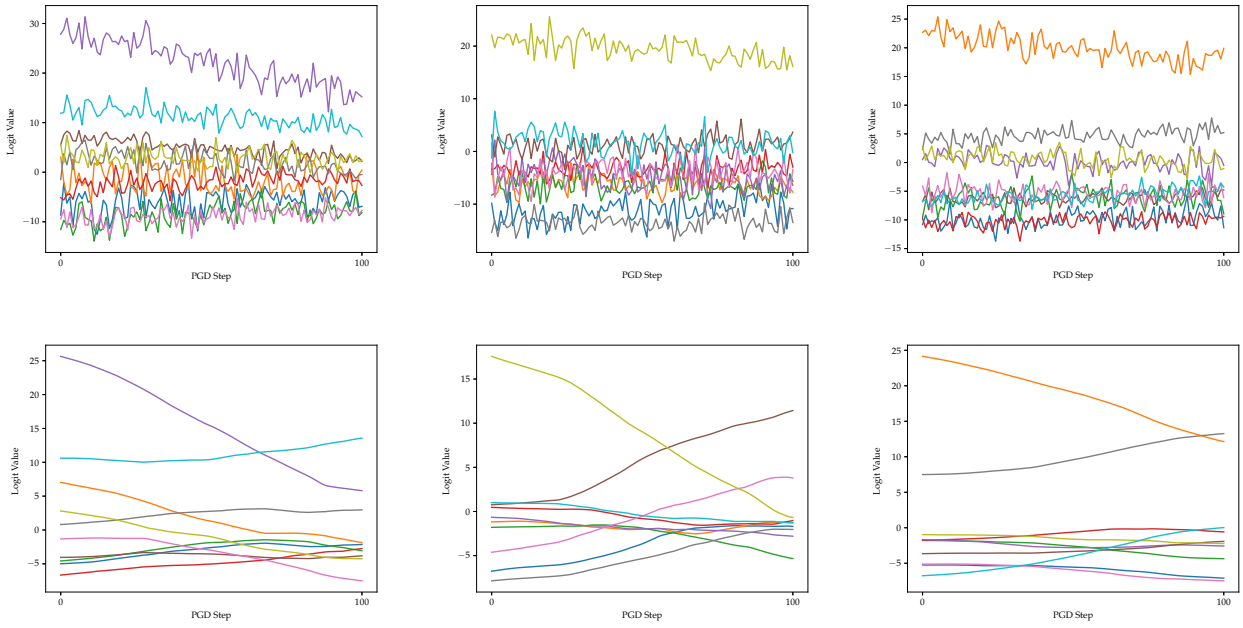

Figure 1: Change of the output logit values under a PGD attack (MNIST dataset), for our proposed approach (top row), and the ReLU-based counterpart (bottom row).

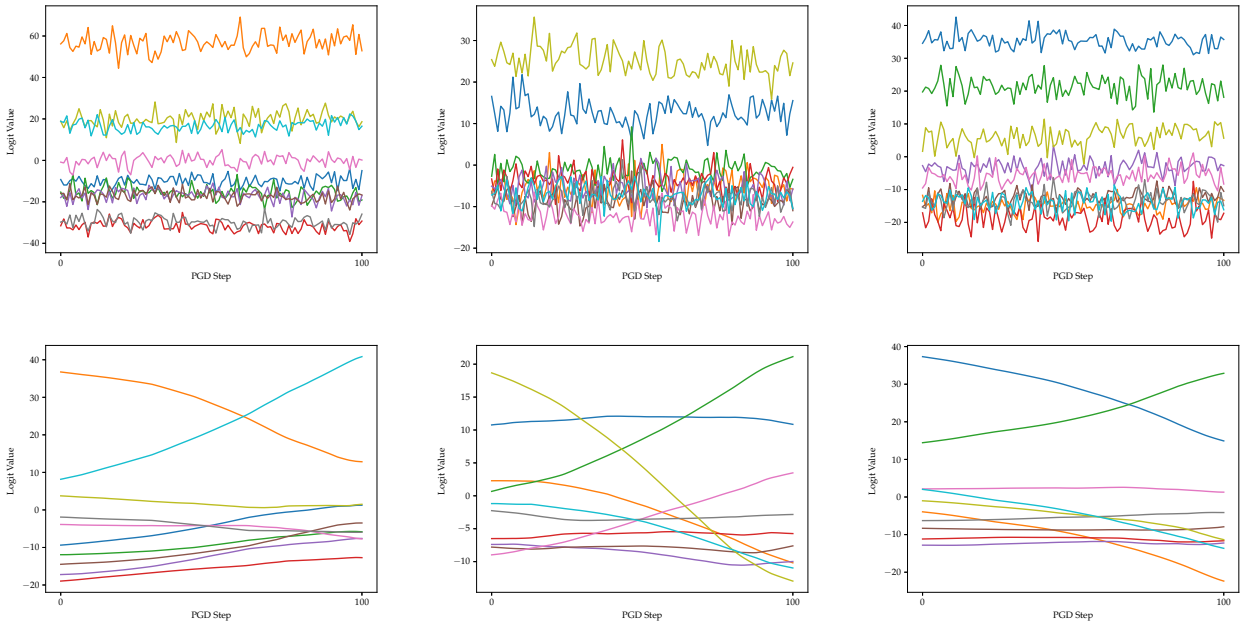

Figure 2: Change of the output logit values under a PGD attack (CIFAR-10 dataset), for our proposed approach (top row) and the ReLU-based counterpart (bottom row).

4 Complexity

Finally, Table 7 depicts the inference times for the Softmax network trained on MNIST, for various types of attacks. We compare the Softmax network employing the conventional definitions of the convolutional and feedforward layers (Verma and Swami, 2019) to an implementation employing our LWTA and IBP-based variant, proposed in the main text. It is characteristic that, on the CW attack, our approach is **4.35 times faster**, while for the Random Noise attack our approach is **3 times faster**.

Table 7: Inference times (in seconds) for various attacks to the Softmax network; MNIST dataset.

Model	Benign	PGD	CW	BSA	Rand
Softmax with $U = 2$	7.969	108.219	767.292	823.25	1.370
Softmax (Verma and Swami, 2019)	2.218	48.294	3322.743	3581.75	3.703

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In Bengio, Y. and LeCun, Y., editors, *ICLR*.

Maddison, C. J., Mnih, A., and Teh, Y. W. (2016). The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

Papernot, N., Carlini, N., Goodfellow, I., Feinman, R., Faghri, F., Matyasko, A., Hambardzumyan, K., Juang, Y.-L., Kurakin, A., Sheatsley, R., et al. (2016). cleverhans v2. 0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*.

Verma, G. and Swami, A. (2019). Error correcting output codes improve probability estimation and adversarial robustness of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 8643–8653.