Supplementary Materials for "Training Adversarially Robust Sparse Networks via Bayesian Connectivity Sampling"

Ozan Özdenizci [1 2] **Robert Legenstein** [1]

A. Details on the Experimental Settings

A.1. Datasets and Preprocessing

We used CIFAR-10, CIFAR-100 and SVHN datasets for our experiments. CIFAR-10 and CIFAR-100 datasets both consist of $50,000$ training and $10,000$ test images (Krizhevsky, 2009). SVHN dataset consists of $73,257$ training and $26,032$ test samples (Netzer et al., 2011).

We follow the conventional data augmentation scheme for training the models (He et al., 2016). This data augmentation scheme involves shifting images randomly to left or right (with a maximum shift range proportion of 10% with respect to the total image width and height) followed by cropping the image back to its original dimensionality, and a random horizontal flip. We normalize all images into $[0, 1]$ pixel values. At test time we evaluate the images using only the original 32×32 single view. All clean and robust accuracies are evaluated using the complete test sets.

A.2. Optimization Hyper-parameter Choices

We trained all models using a momentum stochastic gradient descent (SGD) optimizer with decoupled weight decay regularization (Loshchilov & Hutter, 2019) for 200 epochs with a batch size of 128. While training models with RST on CIFAR-10, since 500K additional pseudo-labeled data samples alongside original training images are exploited, we used batches of size 256 while keeping the total number of iterations the same, similar to (Carmon et al., 2019). Momentum parameter for the optimizer was set to 0.9 and the initial learning rate was set to 0.1. We used synchronous piecewise constant decay learning rate and weight decay schedulers. Both learning rate and weight decay factors were divided by 10 at the 100th and 150th epochs.

Due to the variability in their respective robust regularization schemes, the initial weight decay factor was determined differently for particular robust training objectives. Specifically we chose the initial weight decay factors as 0.001 for standard AT and mixed-batch AT, 0.0005 for TRADES, MART and RST, and 0.0001 for natural training. We determined these initial weight decay factors from a possible set of options: $\{0.005, 0.001, 0.0005, 0.0001\}$ based on clean/robust accuracies. Especially for models trained with RST on CIFAR-10, and TRADES on SVHN, we tended to choose the weight decay factor that yielded similar clean and robust accuracy values in fully-connected dense models with respect to the dense models reported in (Sehwag et al., 2020). Only for WideResNet-28-4 models at 99% sparsity that were trained with TRADES, we used a higher initial weight decay factor of 0.001 which performed better.

We set the noise scaling factor to $\sigma = 10^{-6}$. We did not observe a significant difference by varying this parameter, except at very low values which led to comparably low performance (see Section B.1). All code and software were implemented with Tensorflow 2.3.0 (Abadi et al., 2016), and experiments were performed using GPU hardware of types NVIDIA GeForce GTX 2080Ti, NVIDIA Quadro P6000 and NVIDIA Tesla P100.

A.3. Initialization of Network Connectivities

We perform robust training for all networks starting from a sparse initialization by dynamically changing the connectivity. In order to promote a faster robust training warm-up at early epochs and eventually yield faster convergence across 200 epochs, we use an unbalanced sparse connectivity initialization scheme for network weights. Consistently staying within the boundaries of the global sparsity constraint for all networks, we initialize the parameters of the input convolutional blocks and output classification dense layers of the networks with a higher sparse connectivity then all remaining intermediate layers. More specifically, we initialize (assign a non-zero connection weight $\boldsymbol{w}_k$ by setting $\boldsymbol{\theta}_k > 0$) approximately 60% of connections in the layers of the first convolutional block and output dense layer, 20% of connections in the layers of the second convolutional/residual blocks (for WideResNet models this was chosen to be 10% due to the larger extent of the second block), and $p_c/2$ of connections in all remaining interme-

[1]Graz University of Technology, Institute of Theoretical Computer Science, Graz, Austria [2]Silicon Austria Labs, TU Graz - SAL Dependable Embedded Systems Lab, Graz, Austria. Correspondence to: Ozan Özdenizci <ozan.ozdenizci@igi.tugraz.at>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

Table 1. Number and proportions of active connections (non-zero parameters) at each convolutional block after robust training of VGG-16 models with RST on CIFAR-10. Models with 90% and 99% sparsity are illustrated.

		ConvBlock-1	*ConvBlock-2*	*ConvBlock-3*	*ConvBlock-4*	*ConvBlock-5*	*Dense Layers*
VGG-16 CIFAR-10	# parameters	38,592	221,184	1,474,560	5,898,240	7,077,888	592,384
	Ours (10%)	9,180 (23.78%)	36,231 (16.38%)	154,074 (10.45%)	547,296 (9.28%)	692,846 (9.79%)	90,657 (15.30%)
	Ours (1%)	4,722 (12.24%)	8,667 (3.92%)	17,481 (1.19%)	53,307 (0.90%)	64,351 (0.90%)	4,500 (0.76%)

Table 2. Number and proportion of active connections (non-zero parameters) at each convolutional residual block after robust training of ResNet-18 models with TRADES on CIFAR-100. Models with 90% and 99% sparsity are illustrated.

		ConvBlock-1	*ResBlock-1*	*ResBlock-2*	*ResBlock-3*	*ResBlock-4*	*Dense Layers*
ResNet-18 CIFAR-100	# parameters	1,728	147,456	524,288	2,097,152	8,388,608	51,200
	Ours (10%)	371 (21.47%)	27,581 (18.70%)	73,792 (14.07%)	194,611 (9.28%)	808,796 (9.64%)	15,892 (31.04%)
	Ours (1%)	240 (13.89%)	5,367 (3.64%)	7,402 (1.41%)	16,949 (0.81%)	74,600 (0.89%)	7,546 (14.74%)

diate layers of the network, with p_c being the pre-defined global connectivity constraint (e.g., $p_c = 1\%$ for 99% sparse models). Consistently with these proportions, we determine which connections to be initialized at each layer randomly by uniform sampling. In all experiments we make sure that the global connectivity becomes strictly bounded with p_c following this initialization scheme.

Number of parameters in the fully connected models and distributions of parameters across convolutional/residual blocks of the sparse models after training are illustrated in Tables 1, 2 and 3. For simpler visualization purposes in these tables, we present the number of parameters that relate to subsequent convolution operations in groups of convolutional/residual blocks, consistently with the definitions of these architectures (Simonyan & Zisserman, 2015; He et al., 2016; Zagoruyko & Komodakis, 2016). However our models indeed have varying connectivity levels for each convolution/dense kernel within these blocks, and considers each layer individually (cf. our code repository and model checkpoints for details). Note that the parameters related to batch normalization layers are ignored in these tables. Our implementations of VGG-16 and WideResNet-28-4 architectures are specifically compared to be identical in terms of the precise number of parameters per layer as in (Sehwag et al., 2020) to enable a fair comparison of sparsity and robustness. In all models that we adversarially trained via Bayesian connectivity sampling, resulting per-layer sparsity distributions were significantly different then the connectivities at initialization. In Tables 1, 2 and 3, we illustrate these resulting differences as we let the robust training process guide the sparse connectivity search.

B. Additional Experimental Analyses

B.1. Impact of the Noise Scaling Factor in Bayesian Connectivity Sampling

Our approach relies on a posterior sampling principle, which necessitates a sufficient amount of noise added to the negative log-posterior objective such that the trajectory of sampled network parameters converge to sampling connectivities from the posterior with a high probability. Here we examine the impact of the noise scaling factor choice for connectivity sampling. Figure 1 depicts clean and robust test accuracies of models trained with various choices of the noise scaling factor σ, illustrated for a VGG-16 at 99% sparsity trained on CIFAR-10 via TRADES loss and mixed-batch AT. Our investigations consistently indicated that a sufficient amount of noise is important for convergence towards sampling posterior network configurations with better benign and robust accuracies under given sparsity constraints (cf. Figure 1 drop in performances at $\sigma = 10^{-9}$). However we also observed that the precise value of the noise scaling factor did not influence the final model performance to a significant extent except the number of converged connections at each layer. Hence we determined $\sigma = 10^{-6}$ in all our further experimentations.

B.2. Impact of Decoupled Weight Decay Regularization

We investigate the use of decoupled weight decay regularization to promote Bayesian connectivity sampling. (Loshchilov & Hutter, 2019) studied the inequivalence of adding an l_2-regularization term to the loss function versus

Table 3. Number and proportion of active connections (non-zero parameters) at each convolutional residual block after robust training of WideResNet-28-4 models with TRADES on SVHN. Models with 90% and 99% sparsity are illustrated.

		ConvBlock-1	*NetworkBlock-1*	*NetworkBlock-2*	*NetworkBlock-3*	*Dense Layers*
WideResNet-28-4 SVHN	# parameters	432	268,288	1,114,112	4,456,448	2,560
	Ours (10%)	71 (16.44%)	37,762 (14.07%)	124,866 (11.21%)	420,827 (9.44%)	658 (25.70%)
	Ours (1%)	28 (6.48%)	7,371 (2.75%)	14,264 (1.28%)	36,585 (0.82%)	170 (6.64%)

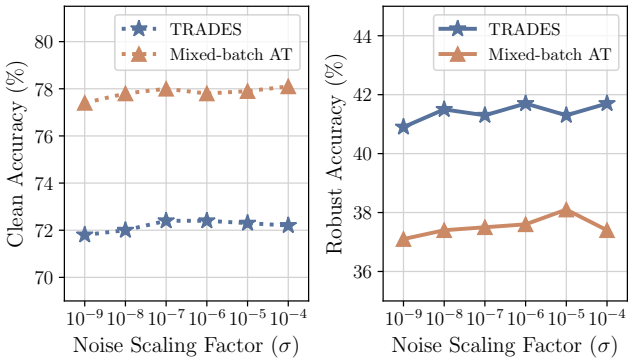

Figure 1. Illustration of the impact of varying the noise scale factor for a VGG-16 trained on CIFAR-10 with 99% sparsity. Each marker on plot (a) represents a different model trained with its respective robust training objective (TRADES or mixed-batch AT) and noise scaling factor σ. Markers on plot (b) represents the robust accuracies of the models in part (a) under PGD^{50} attacks.

specifically decaying the weights of network parameters, for traditional adaptive gradient update methods such as momentum SGD. Decoupled weight decay was shown to yield stronger regularization of variables and better generalization. We follow their lead to promote novel robust connectivity sampling through regularization by pushing the network weights towards zero during optimization. Here, we compare this choice with respect to a standard sparsity-promoting l_1-regularization approach (Bellec et al., 2018; Rakin et al., 2019), including a respective parameter search for the l_1-penalty weight constant, while keeping our momentum SGD optimization scheme same. Note that this also alters the training approach from a pipeline that one would use to train a fully-connected counterpart of this network. Table 4 illustrates our comparisons against models trained via l_1-regularization. We show these comparisons with VGG-16 models trained on CIFAR-10, and ResNet-18 models trained on CIFAR-100, both with TRADES loss under a 90% sparsity constraint. Overall, we observe that an l_1-regularization choice yields significantly lower accuracies with the state-of-the-art robust training objective TRADES, especially for the CIFAR-100 dataset.

Table 4. Clean/robust accuracy (%) comparisons with models trained via l_1-regularization. All models (VGG-16 for CIFAR-10, ResNet-18 for CIFAR-100) are trained with TRADES loss under a 90% sparsity constraint. Robust accuracies are evaluated by PGD^{50} attacks. λ_r denotes the l_1-regularization weight.

		CIFAR-10	CIFAR-100
l_1-Regularization	$\lambda_r = 10^{-3}$	70.9/40.4	48.5/23.2
	$\lambda_r = 10^{-4}$	77.9/45.2	49.8/23.4
	$\lambda_r = 10^{-5}$	75.5/41.6	48.6/22.0
Decoupled Weight Decay	**Ours**	**78.2/45.7**	**55.2/27.2**

B.3. Varying the Number of Iterations in PGD Attacks

We discuss our choice of the white box adversarial robustness evaluation scheme with PGD^{50} attacks (i.e., PGD attacks with 50 iterations and 10 restarts). While adversarial training with different robust objectives is harnessing examples obtained via PGD with 10 iterations and a step size of $2/255$ for $\epsilon = 8/255$, we evaluate all models on adversarial examples crafted with a stronger attack PGD^{50}. To present and justify our explorations for this choice, we illustrate in Table 5 the robustness as number of PGD iterations vary for CIFAR-10 experiments with VGG-16 trained via RST, and SVHN experiments with WideResNet-28-4 trained via TRADES loss. Consistently with all our experimental analyses, we evaluate clean and robust accuracies on the complete test sets. All attacks in Table 5 are evaluated with an $\epsilon = 8/255$ and 10 restarts. PGD step sizes were determined by the $2.5 * \epsilon / \#steps$ rule of thumb (Madry et al., 2018). We did not observe major robust accuracy drops with increasing PGD steps. Due to the overall robust accuracy stability, we evaluated the main results with PGD^{50} in our assessments to quantify white box adversarial robustness.

B.4. Assessing Scalability with Increasing Category and Feature Dimensionality

We performed additional experiments to assess the scalability of our method to adapt to larger datasets. Our ResNet-18 models trained on CIFAR-100 with TRADES previously

Table 5. Robust accuracies as the number of PGD attack iterations vary. Attacks were performed with 10 random restarts, perturbation strength was $\epsilon = 8/255$, and step sizes were determined by $2.5 * \epsilon/\#steps$. CIFAR-10 experiments are shown with VGG-16 trained via RST, and SVHN with WideResNet-28-4 trained via TRADES.

	#steps	Standard	90% Sparsity	99% Sparsity
CIFAR-10	10	52.31	49.98	42.41
	20	52.14	49.77	42.29
	50	52.09	49.64	42.25
	250	52.04	49.64	42.24
	500	52.01	49.64	42.17
SVHN	10	57.42	56.42	53.24
	20	56.93	55.88	52.93
	50	56.59	55.58	52.74
	100	56.48	55.47	52.70
	250	56.35	55.38	52.62

yielded clean/robust accuracies (%) of 55.2/27.2 at 90% and 47.7/22.2 at 99% sparsity. We additionally evaluated HYDRA (Sehwag et al., 2020) with a similar training setup based on the authors' implementation and obtained inferior models with accuracies of 50.1/25.0 at 90% and 44.4/20.1 at 99% sparsity for this 100-class classification problem.

To illustrate the time cost of our algorithm, we timed end-to-end robust training of a ResNet-18 at 99% sparsity with TRADES on CIFAR-100, which necessitated approximately 26 hours on a single NVIDIA Tesla P100. Note that at this stage we also did not use any sparse-matrix optimized implementations for these experiments, which would reduce wall-clock time drastically. Obtaining a sparse model with HYDRA however necessitates several stages: robust dense training, accumulating importance scores for all network weights, one-shot pruning and weight fine-tuning. Using the authors' implementation (Sehwag et al., 2020), obtaining a 99% sparse ResNet-18 on CIFAR-100 took approximately 32 hours on two NVIDIA Tesla P100s.

Furthermore, we also performed experiments on the Tiny-ImageNet dataset which consists of 200 ImageNet classes with images at 64×64 resolution, to test our algorithm's scalability to increased category- and feature-dimensionalities. We trained ResNet-34 models with the TRADES loss (training with 10 PGD iterations and a step size of $2/255$ for $\epsilon = 8/255$, TRADES $\beta = 6$) using Tiny-ImageNet. A standard dense ResNet-34 model yielded accuracies of 35.0/10.5, while our approach can obtain 90% sparse models with 33.8/11.8 and 99% sparse ResNet-34 models with 30.6/9.5 clean/robust accuracies, showing scalability.

References

Abadi, M. et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation*, pp. 265–283, 2016.

Bellec, G., Kappel, D., Maass, W., and Legenstein, R. Deep rewiring: Training very sparse deep networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Carmon, Y., Raghunathan, A., Schmidt, L., Duchi, J. C., and Liang, P. S. Unlabeled data improves adversarial robustness. In *Advances in Neural Information Processing Systems*, pp. 11192–11203, 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Krizhevsky, A. Learning multiple layers of features from tiny images. *Technical Report, University of Toronto*, 2009.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning, 2011.

Rakin, A. S., He, Z., Yang, L., Wang, Y., Wang, L., and Fan, D. Robust sparse regularization: Simultaneously optimizing neural network robustness and compactness. *arXiv preprint arXiv:1905.13074*, 2019.

Sehwag, V., Wang, S., Mittal, P., and Jana, S. HYDRA: Pruning adversarially robust neural networks. *Advances in Neural Information Processing Systems*, 7, 2020.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *British Machine Vision Conference*, 2016.