Understanding the effect of sparsity on neural networks robustness

Lukas Timpl [* 1]   Rahim Entezari [* 1 2]   Hanie Sedghi [3]   Behnam Neyshabur [4]   Olga Saukh [1 2]

Abstract

This paper examines the impact of static sparsity on the robustness of a trained network to weight perturbations, data corruption, and adversarial examples. We show that, up to a certain sparsity achieved by increasing network width and depth while keeping the network capacity fixed, sparsified networks consistently match and often outperform their initially dense versions. Robustness and accuracy decline simultaneously for very high sparsity due to loose connectivity between network layers. Our findings show that a rapid robustness drop caused by network compression observed in the literature is due to a reduced network capacity rather than sparsity.

1. Introduction

Deep learning methods are increasingly used for solving complex tasks, yet little is known about the choice of the best architecture, the required model size, capacity, and the trade-offs involved. A common strategy is to train overparameterized models and compress them into smaller representations (Hoefler et al., 2021). This works remarkably well with an almost negligible drop in accuracy (Gale et al., 2019; Blalock et al., 2020), and is crucial to make use of these models in resource-constrained environments. Recent works, however, shows that test accuracy does not capture how model compression impacts the generalization properties of these models (Hooker et al., 2020).

Related literature refers to robustness as the network generalization ability to small shifts in the distribution that humans are usually robust to. There is a growing body of work studying methods for building robust models. Recent studies (Shankar et al., 2020; Recht et al., 2019) found that image classification models show a consistent accuracy drop when evaluated on ImageNet (Deng et al., 2009) and ImageNetV2 (Recht et al., 2019), while humans achieve the same accuracy. Another line of research aims at minimizing the worst case expected error over a set of probability distributions by applying distributionally robust optimization (Shafieezadeh-Abadeh et al., 2015; Duchi et al., 2020; Sagawa et al., 2020). A similar line of work focuses on finding models that have low performance drop on adversarial examples (Biggio & Roli, 2018; Madry et al., 2019).

A recent study by Hooker et al. (2020) shows that model compression, and to a smaller extent quantization, result in tremendous robustness degradation. At the same time, Golubeva et al. (2021) found that wider networks of the same capacity (same number of parameters) yield better performance. Model compression leads simultaneously to sparser and lower capacity networks, yet the contribution of both effects is mixed. Understanding the impact of these effects on model robustness in isolation is crucial when optimizing machine learning models for resource-constrained devices. This work evaluates the effect of model sparsification while keeping the network capacity, defined by the total number of parameters, fixed.

Contributions. We hypothesise that sparsity alone does not hurt model robustness when the network capacity is fixed and provide empirical evidence to support this hypothesis in a number of settings. We run our study on a range of network architectures (MLPs, VGG and ResNets), datasets (MNIST, CIFAR-10, CIFAR-100), robustness tests (weight perturbations, data corruptions, adversarial examples) and evaluate the overall and per class network performance. We observe that for randomly initialized models with a static sparsity pattern applied before or after training, network sparsification does not hurt or even improves robustness to a certain sparsity compared to a dense network of the same capacity. Robustness and accuracy decline simultaneously for very high sparsity due to loose connectivity between network layers. We show that our hypothesis holds when introducing sparsity by increasing network width and depth in separate experiments, applied before and after training. These findings show that a rapid robustness drop caused by network compression observed in the literature is due to a reduced network capacity rather than sparsity.

---

[*]Equal contribution [1]Institute of Technical Informatics, Graz University of Technology, Graz, Austria [2]Complexity Science Hub Vienna, Vienna, Austria [3]Google Research Brain team, USA [4]Blueshift, USA. Correspondence to: Rahim Entezari <entezari@tugraz.com>, Olga Saukh <saukh@tugraz.at>.

(a) One layer MLP   (b) VGG11   (c) ResNet18
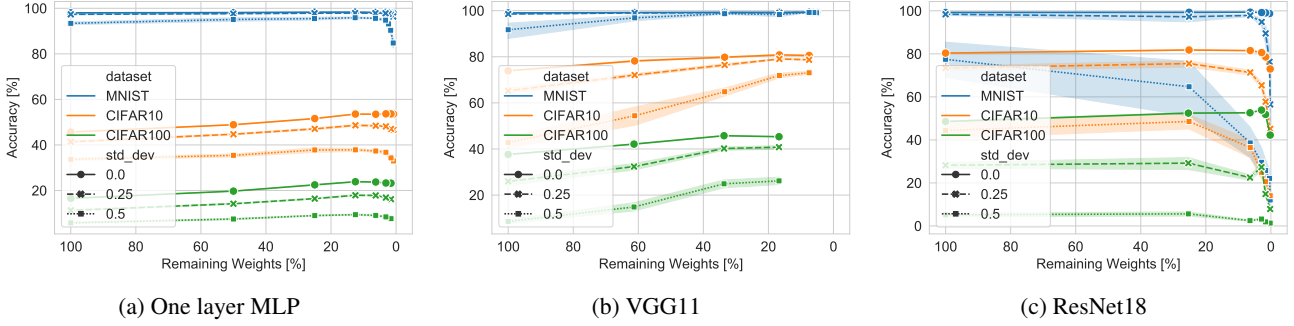
Figure 1. **Robustness to weight perturbations, sparsification by increasing width.** We add multiplicative Gaussian noise $z_i \sim \mathcal{N}(\mu, w_i^2 \sigma_i^2)$ to each weight and evaluate model performance. We observe that as we move towards higher sparsity levels, the performance first increases then decreases in extreme sparsity levels. We note that such increase is happening earlier for simpler tasks like MNIST. This performance improvement indicates a flatter loss landscape around the minima suggesting better generalization.

(a) One layer MLP   (b) VGG11   (c) ResNet18
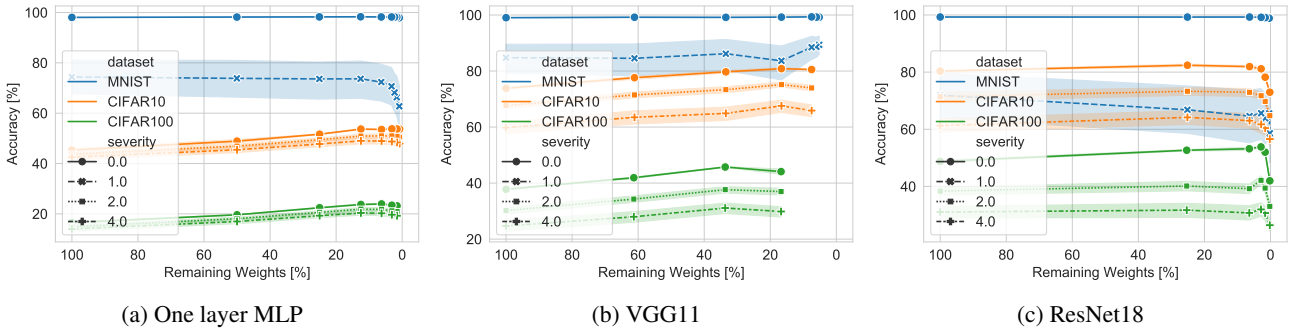
Figure 2. **Robustness to data corruption, sparsification by increasing width.** We evaluate the performance of the models on corrupted datasets MNIST-C, CIFAR10-C and CIFAR100-C. We observe that as we move towards higher sparsity levels, the performance first increases then decreases in extreme sparsity levels. We note that such increase is happening earlier for simpler tasks like MNIST.

2. Experimental Framework

We hypothesise that sparsity, while keeping the number of parameters fixed, does not hurt network robustness. We support our hypothesis by exhaustive tests covering multiple datasets, network architectures, model and data corruptions, sparsity levels, sparsification methods and schedules. The details are given below.

Datasets and architectures. The datasets used in the experiments include MINST (LeCun & Cortes, 2010), CIFAR-10 (Krizhevsky et al., 2009), and CIFAR-100) (Krizhevsky et al., 2009). We fix the number of weights in each network architecture (one layer MLP, VGG16 (Simonyan & Zisserman, 2015), ResNet18 (He et al., 2015)) throughout all experiments, by increasing the width or depth and introducing the proper corresponding sparsity. See *sparsification methods* for more details. We use one layer MLP with 2^7 hidden units, VGG with 11 layers, and ResNet18 as base architectures. We refer to these vanilla architectures as to 100 %-networks *before* sparsification. Note that for both ResNet and VGG our vanilla implementation uses the layer width of 16 as the base architecture, which is lower than

64 used in the original architecture. We use width to set the number of output channels for the first layer and use the same width ratios as the respective vanilla architectures for the following layers. All networks were trained using SGD with momentum 0.9. Details for each model family are provided in Section A.

Sparsification methods. Existing literature covers multiple ways to make use of sparsity during and after model training including static and dynamic sparsity (*e.g.*, β-Lasso (Neyshabur, 2020)), iterative hard thresholding (*e.g.*, Lottery Ticket Hypothesis with various pruning strategies (Frankle & Carbin, 2018; Renda et al., 2020)) and others. (Hoefler et al., 2021) provides a comprehensive survey on pruning strategies. Sparsification without changing the number of parameters was investigated in (Golubeva et al., 2021). In their study static sparsity showed the most prominent impact on network performance and is thus adopted in this work.

We sparsify a network while preserving its capacity by changing the network's width or depth. When sparsifying by increasing width, we leverage the approach introduced

(a) One layer MLP   (b) VGG11   (c) ResNet18
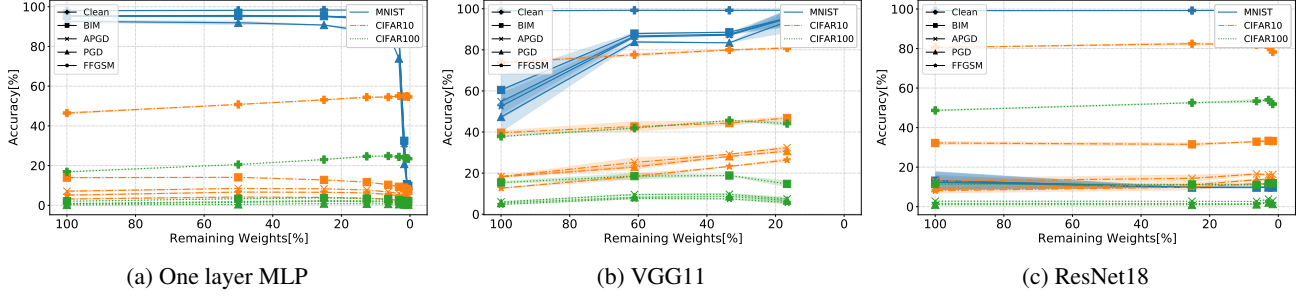
Figure 3. **Robustness to adversarial attacks. Sparsification by increasing width.** Robustness to all adversarial attacks (BIM (Kurakin et al., 2016), APGD (Croce & Hein, 2020), PGD (Madry et al., 2019), FFGSM (Goodfellow et al., 2014)) is improved as we have less remaining weights and decreases for extreme sparsity levels where overall network accuracy (clean) drops.

(a) One layer MLP   (b) VGG11
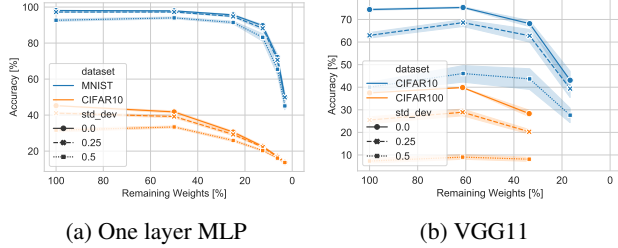
Figure 4. **Robustness to weight perturbations. Sparsification after training by increasing width.** We add multiplicative Gaussian noise $z_i \sim \mathcal{N}(\mu, w_i^2 \sigma_i^2)$ to each weight and evaluate performance on test data. As we move towards higher sparsity levels, the performance decreases in extreme sparsity levels.

(a) One layer MLP   (b) VGG11
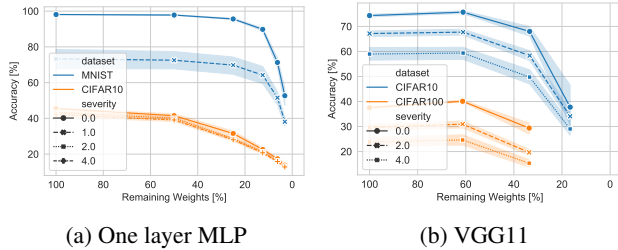
Figure 5. **Robustness to data corruption. Sparsification after training by increasing width.** We evaluate on the corrupted datasets MNIST-C, CIFAR10-C and CIFAR100-C. sompare to static sparsity at the prior to training, robustness degrades sooner.

in (Golubeva et al., 2021): every layer of the network is sparsified by removing weights at random in proportion to the layer size, using a static mask generated at initialization. This approach is referred to as *static sparsity*. We build on its publicly available implementation (Golubeva et al., 2021). Sparsifying by increasing network depth involves duplicating layers and then applying a static random mask to sparsify the weight tensors. When sparsifying by increasing depth, we consider MLP with 2^9 hidden units in each layer, and add layers of the same size. For VGG and ResNet

we build architecture families VGG11, VGG13, VGG16 and ResNet18, ResNet34, ResNet50 all enjoying the default width of 64.

Sparsification schedules. In addition to static sparsity applied prior to network training, we also investigate network pruning after training by removing a certain amount of weights with the lowest magnitude to match the required sparsity level. Note that no fine-tuning is applied.

Robustness measures. We evaluate the impact of sparsity on model performance with respect to weight perturbations (von Oswald et al., 2021), data corruptions (Hendrycks & Dietterich, 2019) and natural adversarial examples (Hendrycks et al., 2021).

Model perturbation. Similarly to (von Oswald et al., 2021), we perturb model weights by applying Gaussian noise $z_i \sim \mathcal{N}(\mu, w_i^2 \sigma_i^2)$ in proportion to the magnitude of each weight $w_i, i \in L$, and then measure the difference in the loss $\delta \mathcal{L} = \mathbb{E}_z[\mathcal{L}(w_i + z) - \mathcal{L}(w_i)]$. Accuracy drop due to model perturbation is related to the flatness of the loss landscape around the obtained optimum. Robustness to weight perturbation could also represents a proxy for quantization error. This error is introduced in neural network compression by weight quantization in the literature (Novac et al., 2021).

Corrupted data. We apply numerous algorithmically generated corruptions, similar to the ones evaluated in (Hooker et al., 2020) (*e.g.*, blur, contrast, pixelation) to all datasets used in this paper. This allows us investigating how sensitive the sparsified models are to data corruptions of different severity which humans are oblivious to. Our corrupted datasets are MNIST-C (Mu & Gilmer, 2019), CIFAR10-C and CIFAR100-C (Hendrycks & Dietterich, 2019).

Natural adversarial examples. We use Torchattacks (Kim, 2020) to generate a diverse range of adversarial attacks for different combination of mentioned architectures and datasets. This include FGSM (Goodfellow et al., 2014), BIM (Kurakin et al., 2016), APGD (Croce & Hein, 2020), and PGD (Madry et al., 2019).

When applying sparsity, we evaluate both the overall model performance and its performance on the most sensitive class. We follow the methodology introduced in (Hooker et al., 2020) and evaluate the change to class level recall compared to the overall model accuracy. The obtained results are presented below.

3. Results

Perturbed model weights. We first investigate the networks that were sparsified while growing the width to keep their capacity fixed. Figure 1 shows that as we move towards higher sparsity levels, the test performance first increases then decreases in extreme sparsity levels. We note that such increase is happening earlier for simpler tasks like MNIST. We observe that sparse configurations are indeed in flatter regions of weight space as $\delta\mathcal{L}$ increases more slowly with δz_i. This suggests better robustness and generalization around the minima (Pittorino et al., 2020; Jiang et al., 2019). Each point in this plot shows the mean over five networks trained from different initializations. When sparsification is applied while increasing network depth, the maximum accuracy and robustness are achieved for smaller depth values in all experiments. Note that keeping a network connected while increasing its depth, in contrast to width, becomes difficult with higher sparsity. The results are summarized in Section B.1. The outcome across all experiments consistently suggests that sparsification alone does not undermine network robustness to weight perturbations as long as sufficient network connectivity is maintained.

Corrupted data. Figure 2 evaluates the performance of the models on corrupted datasets MNIST-C, CIFAR10-C and CIFAR100-C. We observe that as we move towards higher sparsity levels, the test performance first increases then decreases in extreme sparsity levels. We note that such increase is happening earlier for simpler tasks like MNIST. Each point in Figure 2 is mean performance over three trained networks. For each network we randomly sample 1000 examples from a dataset and add five noise samples in each run. On CIFAR10-C and CIFAR100-C our evaluation considers corruption severity of two and four as classified by (Hendrycks & Dietterich, 2019). Detailed results for specific corruption types can be found in Section B.2. The results for the achieved performance of networks sparsified by increasing depth are also shown in Section B.1. We note that VGG networks experience convergence issues as the network sparsity approaches 10% due to lacking connectivity between layers. This is not the case for MLP and ResNet which also converge for lower percentage of remaining weights. We attribute these differences to the power of skip connections in ResNet and low overall tested network depths (1,2,4 and 8) for MLP.

Sensitive classes. Similarly to (Hooker et al., 2020), sen-

sitive classes are considered those with the lowest recall. For each sparsity level, we train five models, evaluate them on the test data and report the minimum recall among all classes. (Hooker et al., 2020) shows that there are some particular examples in each class that a pruned network forgets easily. However, we observe that as the networks get wider (or deeper) and sparser the minimum recall does not decrease. Sparsification does not disproportionally affect sensitive classes, which may not be noticeable by just looking at the overall accuracy. This is due to the fact that the capacity of the networks is fixed. The results are shown in Figure 8 in Section B.1.

Adversarial attacks. Figure 3 shows the robustness of sparsified networks when applying adversarial attacks to perturb test data. We observe a consistent trend for robustness to all adversarial attacks (BIM (Kurakin et al., 2016), APGD (Croce & Hein, 2020), PGD (Madry et al., 2019), FGSM (Goodfellow et al., 2014)). Similar to perturbed model weights and corrupted data, as we have less remaining weights, test performance for adversarial examples is first improved and then decreases for extreme sparsity levels where the overall (clean) network accuracy drops. Dense VGG networks trained on MNIST show the highest accuracy decline in the presence of all attacks, while sparsification helps to improve adversarial robustness.

Post-training sparsification. Figure 4 depicts the results for post-training sparsification for MLP and VGG architectures challenged with perturbed model weights. The results indicate a similar trend to the experiments with static sparsity applied at initialization. For VGG we observe a slight improvement followed by an accuracy drop. However, the performance does deteriorate sooner than with static sparsity. For MLP the results show stagnating accuracy and a slight drop in performance on CIFAR-10. We attribute this to the simplicity of our sparsification method and a relatively low number of weights in the one layer MLP. Similar results are obtained on corrupted datasets visualized in Figure 5.

4. Conclusion

In this work we hypothesise that sparsity, while keeping the number of parameters fixed, does not hurt network robustness. We provide experimental evidence to support this claim based on several standard architectures, datasets, sparsification methods and measures of robustness. Our observation is that network sparsification often helps to improve robustness compared to a dense model, yet the benefits decline together with the overall model accuracy for high sparsity levels. This is due to the increasingly loose connectivity between layers which complicates optimization. Since network capacity rather than sparsity causes accuracy and robustness drop of compressed models, designing pruning methods that treat network capacity and sparsity separately

can lead to better compressed models. In addition, our work emphasizes the need for training procedures that better support sparse operations, which would allow for a faster and more memory efficient training of sparse networks.

References

Biggio, B. and Roli, F. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, Dec 2018. ISSN 0031-3203. doi: 10.1016/j.patcog.2018.07.023. URL http://dx.doi.org/10.1016/j.patcog.2018.07.023.

Blalock, D., Ortiz, J. J. G., Frankle, J., and Guttag, J. What is the state of neural network pruning?, 2020.

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, pp. 2206–2216. PMLR, 2020.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Duchi, J., Hashimoto, T., and Namkoong, H. Distributionally robust losses for latent covariate mixtures, 2020.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Gale, T., Elsen, E., and Hooker, S. The state of sparsity in deep neural networks, 2019.

Golubeva, A., Neyshabur, B., and Gur-Ari, G. Are wider nets better given the same number of parameters? https://github.com/google-research/wide-sparse-nets, 2021.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition, 2015.

Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., and Song, D. Natural adversarial examples, 2021.

Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., and Peste, A. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks, 2021.

Hooker, S., Courville, A., Clark, G., Dauphin, Y., and Frome, A. What do compressed deep neural networks forget?, 2020.

Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D., and Bengio, S. Fantastic generalization measures and where to find them, 2019.

Kim, H. Torchattacks: A pytorch repository for adversarial attacks. *arXiv preprint arXiv:2010.01950*, 2020.

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-100 and cifar-10 (canadian institute for advanced research), 2009. URL http://www.cs.toronto.edu/~kriz/cifar.html. MIT License.

Kurakin, A., Goodfellow, I., Bengio, S., et al. Adversarial examples in the physical world, 2016.

LeCun, Y. and Cortes, C. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010. URL http://yann.lecun.com/exdb/mnist/. Creative Commons Attribution-Share Alike 3.0.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks, 2019.

Mu, N. and Gilmer, J. Mnist-c: A robustness benchmark for computer vision. *arXiv preprint arXiv:1906.02337*, 2019.

Neyshabur, B. Towards learning convolutions from scratch. *arXiv preprint arXiv:2007.13657*, 2020.

Novac, P.-E., Hacene, G. B., Pegatoquet, A., Miramond, B., and Gripon, V. Quantization and deployment of deep neural networks on microcontrollers. *Sensors*, 21(9): 2984, 2021.

Pittorino, F., Lucibello, C., Feinauer, C., Malatesta, E. M., Perugini, G., Baldassi, C., Negri, M., Demyanenko, E., and Zecchina, R. Entropic gradient descent algorithms and wide flat minima. *arXiv preprint arXiv:2006.07897*, 2020.

Recht, B., Roelofs, R., Schmidt, L., and Shankar, V. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pp. 5389–5400. PMLR, 2019.

Renda, A., Frankle, J., and Carbin, M. Comparing rewinding and fine-tuning in neural network pruning. *arXiv preprint arXiv:2003.02389*, 2020.

Ritchie, S., Slone, A., and Ramasesh, V. Caliban: Docker-based job manager for reproducible workflows. *Journal of Open Source Software*, 5(53):2403, 2020.

Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization, 2020.

Shafieezadeh-Abadeh, S., Esfahani, P. M., and Kuhn, D. Distributionally robust logistic regression, 2015.

Shankar, V., Roelofs, R., Mania, H., Fang, A., Recht, B., and Schmidt, L. Evaluating machine accuracy on imagenet. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8634–8644. PMLR, 2020. URL http://proceedings.mlr.press/v119/shankar20c.html.

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

von Oswald, J., Kobayashi, S., Sacramento, J., Meulemans, A., Henning, C., and Grewe, B. F. Neural networks with late-phase weights, 2021.

A. Implementation Details

We used Caliban (Ritchie et al., 2020) to manage all experiments in a reproducible environment in Google Cloud's AI Platform. Each point in plots show the mean value taken over at least five different runs.

A.1. Training for MLP

- Dataset: MNIST (LeCun & Cortes, 2010), CIFAR-10 (Krizhevsky et al., 2009) and CIFAR-100 (Krizhevsky et al., 2009)

- Network: MLP

- Width experiments: single hidden layer, hidden neurons 2^n where n $\in$ 7..15.

- Depth experiments: 2^9 neurons per layer, we add additional layers with the same number of neurons.

- Hyper parameters:
    - LR = fixed 0.01
    - stopping criteria = 300 epochs or loss (CE) < 0.01
    - Momentum = 0.9

A.2. Training for CNN architecture family

- Dataset: MNIST (LeCun & Cortes, 2010), CIFAR-10 (Krizhevsky et al., 2009) and CIFAR-100 (Krizhevsky et al., 2009)

- Network: VGG (Simonyan & Zisserman, 2015), ResNet (He et al., 2015))

- Width experiments: we use VGG11 and ResNet18 changing the width of the first layer and adapting the following layers according to the ratios in the vanilla version of the networks.

- Depth experiments: we change the architecture: VGG11, VGG13, VGG16 and ResNet18, ResNet34, ResNet50 while keeping the default width of the vanilla networks.

- Hyper parameters:
    - LR = Cosine Annealing (Loshchilov & Hutter, 2016) with initial LR=0.01
    - stopping criteria = 300 epochs or loss < 0.1
    - Momentum = 0.9

B. Additional Plots

B.1. Sparsification by increasing network depth

In addition to increasing network width before applying random static sparsification, we also test sparsification by increasing network depth. In this case, we duplicate layers while keeping their width unchanged. In contrast to wider networks, high sparsity levels severely impairs connectivity between layers of deeper networks which make these more difficult to train and converge. VGG experience convergence issues when sparsifying by increasing depth when sparsity approaches 10 %. ResNet appears less susceptible to this issue due to the presence of skip connections, whereas MLPs are more robust due to a considerably lower depth in our settings (1,2,4 and 8). Our experiments with deeper MLP networks and high sparsity also reveal convergence issues similarly to VGG. We run experiments for increasing depth for all architectures and datasets up to a point where the overall network accuracy starts to drop and as long as network training converges.

Figure 6 and Figure 7 show the results for weight perturbations and corrupted data of varying severity. The results are similar and show that robustness improves as long as network accuracy without intervention remains steady.

Figure 8 shows the results for the lowest recall among all classes. Network accuracy on the most sensitive classes does not decline with sparsity. The test is conducted without weight perturbation or data corruption.

B.2. Detailed results for corrupted data

The plots presented in Figure 9, Figure 10 and Figure 11 provide details on the impact of individual corruption methods on network performance. We show details for MLP, VGG and ResNet architectures trained on MNIST, CIFAR-10 and CIFAR-100. Sparsification is achieved by increasing network width. Although the effect of data corruptions on model performance varies widely, it can be observed that in all cases a sparser network matches the accuracy of the vanilla 100 % network. This observation holds up to high sparsity levels where the overall model performance declines.

(a) MLP

(b) ResNet
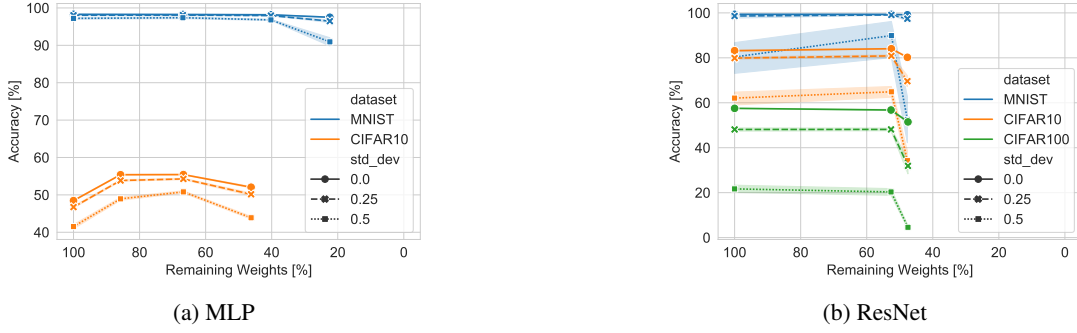
Figure 6. **Robustness to weight perturbations. Sparsification by increasing network depth** We add multiplicative Gaussian noise $z_i \sim \mathcal{N}(\mu, w_i^2 \sigma_i^2)$ to each weight and evaluate model performance. There is a sweetspot corresponding to optimal sparsity. With higher depth and sparsity network connectivity declines leading to simultaneous accuracy and robustness drop.

(a) MLP

(b) ResNet
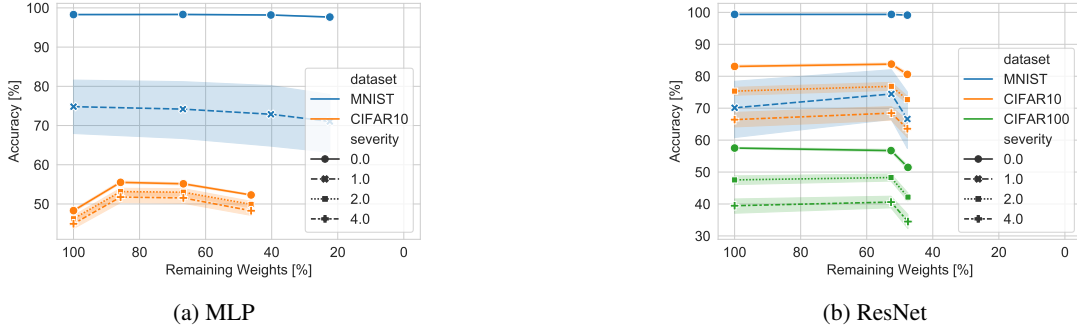
Figure 7. **Robustness to corrupted data. Sparsification by increasing network depth** Corrupted datasets MNIST-C, CIFAR10-C and CIFAR100-C. There is a sweetspot corresponding to optimal sparsity. With higher depth and sparsity network connectivity declines leading to simultaneous accuracy and robustness drop.


(a) Sparsification by increasing network width

(b) Sparsification by increasing network depth
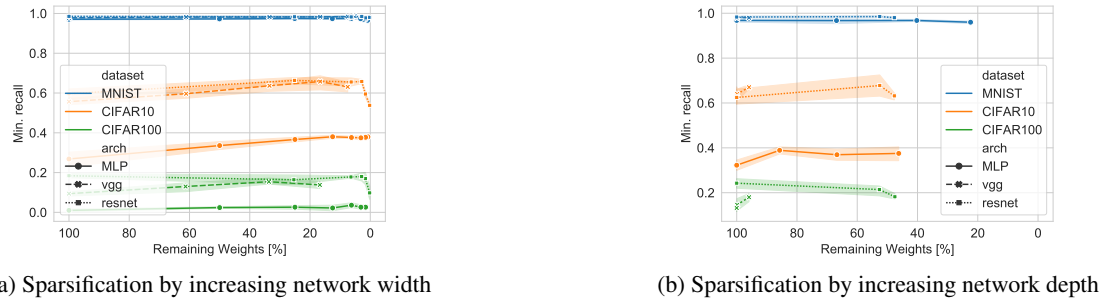
Figure 8. **Minimum recall among all classes.** Network accuracy on the most sensitive classes does not decline with sparsity. Tested without weight / data corruptions. Sparsification by increasing network width (left) and depth (right).

(a) MNIST-C
(b) CIFAR10-C
(c) CIFAR100-C
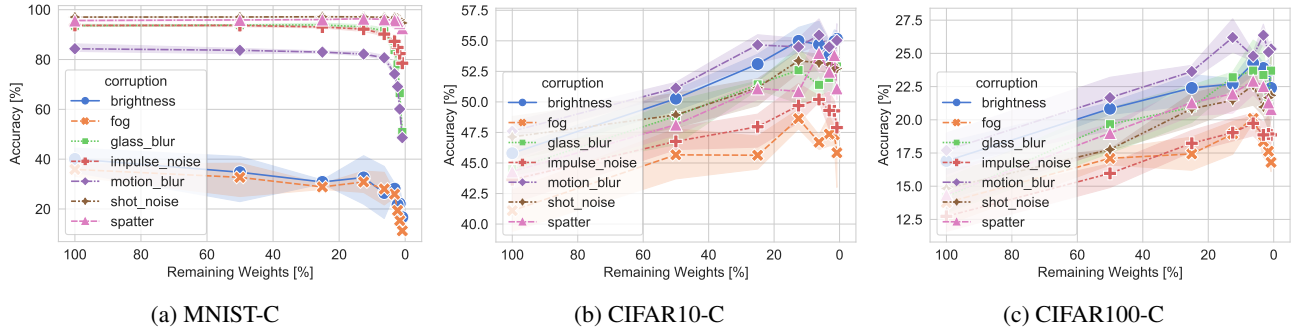
Figure 9. **One layer MLP performance on selected corruption types.** For CIFAR10-C and CIFAR100-C we observe a clear trend across all corruption types, which suggests that the sparser networks with increased width are more robust. We note that for simpler task MNIST-C such increase in the performance is happening earlier in sparsity levels.

(a) MNIST-C
(b) CIFAR10-C
(c) CIFAR100-C
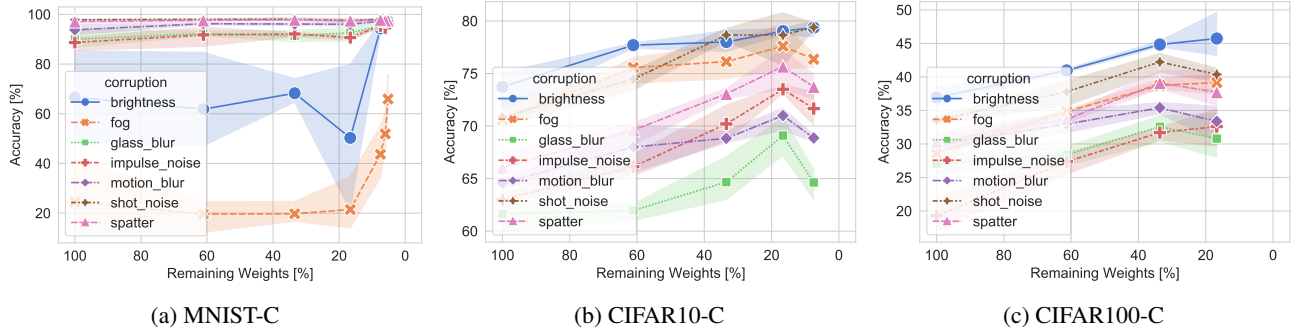
Figure 10. **VGG11 performance on selected corruption types.** The results show a clear upwards trend across different corruption types which indicates, that the networks get more robust as the sparsity and width increase.

(a) MNIST-C
(b) CIFAR10-C
(c) CIFAR100-C
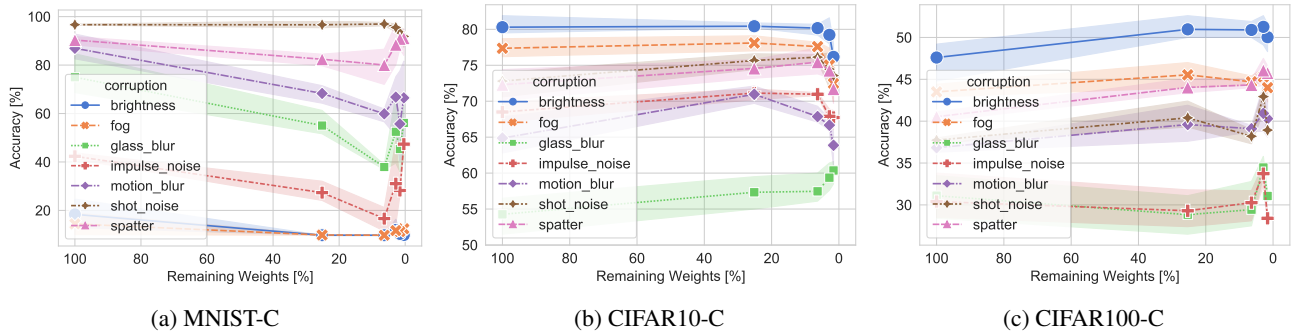
Figure 11. **ResNet18 performance on selected corruption types.** We observe a upwards trend across corruption types for CIFAR10-C and CIFAR100-C, models with higher width and higher sparsity perform better on corrupted data. We note that the increase in the performance for simpler task MNIST-C happens sooner.