Towards improving robustness of compressed CNNs

Jasper Hoffmann [*1] **Shashank Agnihotri** [*1] **Tonmoy Saikia** [1] **Thomas Brox** [1]

Abstract

High capacity CNN models trained on large datasets with strong data augmentation are known to improve robustness to distribution shifts. However, in resource constrained scenarios, such as embedded devices, it is not always feasible to deploy such large CNNs. Model compression techniques, such as distillation and pruning, help reduce model size, however their robustness trade-offs are not known. In this work, we evaluate several distillation and pruning techniques to better understand their influence on out-of-distribution performance. We find that knowledge distillation and pruning combined with data augmentation help transfer much of the robustness to smaller models.

1. Introduction

Robustness to perturbations such as random noise, blur, or weather conditions such as snow, fog, etc. is crucial for deployment in the real world. State-of-the-art methods that focus on improving the robustness of CNNs typically consider very large models. For instance, Xie et al. show strong out-of-distribution (OOD) performance on ImageNet-C (Hendrycks & Dietterich, 2019) using an Efficient-L2 architecture (Tan & Le, 2020) with ∼500M parameters.

In constrained environments, such as embedded devices, where both computational and energy resources are limited, these large models cannot be deployed. Several model compression techniques exist (Frankle & Carbin, 2019; Liu et al., 2017; Sehwag et al., 2020; Hinton et al., 2015; Sergey & Komodakis, 2017; Tian et al., 2020), which can help reduce the memory and computational footprint. However, their robustness trade-offs are currently unknown. Prior work has focused on compressed models that are more robust to adversarial attacks (Sehwag et al., 2020; Goldblum et al., 2020). However, it is known that adversarial robustness results usually do not generalize to other types of OOD robustness (Hendrycks et al., 2020a; Yin et al., 2020).

In this work, we investigate the out-of-distribution (OOD) performance of compressed CNNs. We benchmark popular pruning and knowledge distillation techniques and study their robustness trade-offs. As data augmentation approaches such as AugMix (Hendrycks et al., 2020b) and DeepAugment (Hendrycks et al., 2020a) lead to large robustness gains, we also consider pruning and distillation under the influence of data augmentation. Our key findings are as follows: **(1)** much of the OOD performance of a robust teacher model can be distilled to smaller student models. **(2)** Compared to compressed models via distillation, unstructured pruning can help preserve more robustness. **(3)** Distillation and pruning are complementary — combining them leads to the strongest small-scale models in terms of OOD performance. **(4)** Robustness aware pruning approaches, such as Hydra (Sehwag et al., 2020), originally used for adversarial robustness, can be adapted for general OOD robustness. This is beneficial, particularly at high compression ratios.

2. Background

In this section, we briefly describe the model compression and data augmentation approaches used in our study.

2.1. Model compression

2.1.1. Distillation

Knowledge distillation methods try to transfer knowledge from a larger *teacher* network into a smaller *student* network. Based on the study by Tian et al. (2020), we consider three well-performing distillation methods: Knowledge Distillation (KD), Attention Transfer (AT) and Contrastive Representation Distillation (CRD).

Knowledge Distillation (KD). (Hinton et al., 2015), is the original knowledge distillation method. It minimizes the KL Divergence between the teacher's and the student's output probability distributions. Unlike the standard cross-entropy loss, KD uses soft-targets that can better capture the correlations between different classes.

Attention Transfer (AT). (Sergey & Komodakis, 2017), uses the activations of the intermediate convolutional layers

[*]Equal contribution [1]University of Freiburg, Germany. Correspondence to: Tonmoy Saikia <saikiat@cs.uni-freiburg.de>.

Presented at the ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning.,

to calculate *attention maps*. For the distillation objective, the distances between the student's and teacher's attention maps are minimized.

Contrastive Representation Distillation (CRD). (Tian et al., 2020), uses a *contrastive loss* to make the teacher and student representations for the same samples (positive pairs) more similar while pushing apart the representations of different samples (negative pairs).

2.1.2. Pruning

Neural network pruning removes parameters from an existing network, reducing storage and computational requirements. Whereas *structured pruning* typically prunes complete channels, filters, or layers, *unstructured pruning* considers the individual weights of a network, leading to sparse models. Regarding the relationship of pruning and OOD robustness, Hooker et al. (2020) found that the OOD performance quickly deteriorates for high pruning ratios, however they do not compare different pruning approaches or use diverse data augmentation (Hendrycks et al., 2020a).

Magnitude Pruning. A very successful approach for pruning is to use the absolute value (magnitude) of a weight as a heuristic for the importance of the weight (Han et al., 2015). *Global weight* pruning (Blalock et al., 2020) ranks all the weights of a network based on their magnitude and then prunes the k lowest ranking weights. On the other side, L^1*-filter* pruning (Li et al., 2017) ranks the filters of one convolutional layer by their L^1-norm and then prunes the k lowest ranking filters. Thus, global weight pruning is unstructured and L^1-filter pruning is structured.

Hydra. Instead of a heuristic like the magnitude, *Hydra* (Sehwag et al., 2020) introduces a score for each weight that can be optimized via back-propagation. This is done in a special *mask selection phase* where the gradients of the weights are not used for updating the weights but the scores instead. Importantly, any loss can be used during this phase, which allowed Sehwag et al. (2020) to use losses aimed at adversarial robustness improving the robustness of pruned networks to adversarial attacks. Note, that Hydra is an unstructured pruning method.

2.2. Data augmentation

Currently, the most effective way of achieving OOD robust models is using data augmentation. Some of the data augmentation techniques used for improving robustness are AutoAugment (Cubuk et al., 2018), Cutmix (Yun et al., 2019), AugMix (Hendrycks et al., 2020b) and DeepAugment (Hendrycks et al., 2020a).

AugMix (AM) blends a clean image with augmented variants of the clean image with random blending factors. Additionally, it minimizes the Jensen-Shannon Divergence (JSD) divergence between logits of the clean and augmented samples. In DeepAugment (DA), augmented images are generated by passing them through image-to-image translation networks with the network weights randomly perturbed during the forward pass. The combination of Augmix and DeepAugment (AMDA) (Hendrycks et al., 2020a) results in state-of-the-art robust performance.

3. Experiments

In this section, we describe our experimental setup, our experiments and analyze our results.

3.1. Experimental setup

3.1.1. Datasets

ImageNet100. To speed up training, we used a 100 class subset of ImageNet. The class ids are included in Appendix G. The training and test splits are the same as ImageNet's.

ImageNet100-C. We used a subset of ImageNet-C (Hendrycks & Dietterich, 2019) to evaluate OOD performance on common corruptions. This subset contains the same classes as ImageNet100. Corrupted images exhibit corruption types like noise, blur, weather, and digital corruptions. Each corruption type has five severity levels.

ImageNet100-R. Similar to ImageNet100 and ImageNet100-C, we also evaluate robustness to non-corruption based distribution shifts on a 100 class subset of ImageNet-R (Hendrycks et al., 2020a).

3.1.2. Implementation details

Data-augmentation. We used DeepAugment combined with AugMix. We just refer to this combination as *AMDA*. For AMDA we also used the JSD consistency loss (see 2.2).

Architectures. As small models, we use ResNet18 (He et al., 2015), MobileNetV2 (Sandler et al., 2019), and MnasNet (Tan et al., 2019). As large models we use Resnet50 (He et al., 2015). For distillation, large models are used as teachers and small models as students (more details in next section). For pruning, we use the small models mostly for ablation studies.

3.1.3. Evaluation metrics.

We used classification errors to measure *i.i.d.* and OOD performance. The *i.i.d* performance is measured on the ImageNet100 test set. For ImageNet100-C, we report the mean error over all severities and corruption types (denoted as *mCE*).

For pruned models, we either report the number of parameters that are not zero or the *compression ratio*, which is

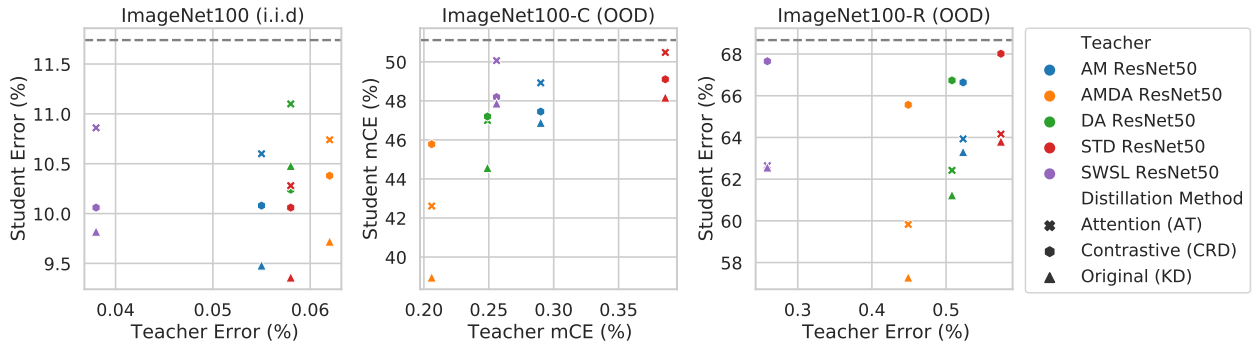


Figure 1. **Can robustness be distilled?** We show student vs teacher performance (error) for different distillation methods on ImageNet100, ImageNet100-C and ImageNet100-R (left to right). The student is a ResNet18 trained on clean images (dashed grey line without any distillation). The teachers are ResNet50s trained with clean images (STD), AugMix (AM), DeepAugment (DA), AugMix + DeepAugment (AMDA) and semi-supervised on an Instagram dataset (SWSL). Teacher and student robustness are correlated. The distillation technique matters.

defined by the original network size divided by the pruned network size.

3.2. Should we distill or prune?

We know that larger models trained with strong data augmentation typically exhibit superior OOD performance. In this section, we evaluate to what extent the robustness of larger models can be transferred to smaller models via model compression techniques.

3.2.1. Distillation

Distillation objectives are designed to mimic the teacher models. Therefore, it is natural to ask — **Can we distill robustness from a larger, more robust teacher model to a smaller student model?**

We considered different publicly available ResNet50 teacher models with varying levels of OOD robustness: **1)** Baseline (poor OOD performance): model trained on clean images from ImageNet (He et al., 2015). **2)** Robust on common corruptions: AugMix (AM) model (Hendrycks et al., 2020b), DeepAugment (DA) and AugMix + DeepAugment (AMDA) model from (Hendrycks et al., 2020a). **3)** Robust on non-corruption based shifts (ImageNet-R): ResNet50 (SWSL) trained in an semi-supervised manner on Instagram images (Yalniz et al., 2019).

We distilled each teacher into a smaller ResNet18 student model. For distillation we evaluated the three methods described in Section 2.1.1: KD, AT, and CRD. Initially, we did not use any data augmentation during distillation to clearly separate gains from different teacher models and distillation methods.

Yes, robustness can be distilled. The results in Figure 1 show that we can distill OOD robustness while only training on *i.i.d* samples (clean images). On ImageNet100-C the performance of the ResNet18 student does improve up to 12%, compared to its performance without distillation. Interestingly, when comparing the three distillation methods, the simpler knowledge distillation method, KD, results in the best OOD performance.

Table 1. Combining the distillation methods (KD, AT and CRD) with data augmentation. Standard: Baseline ResNet18 without distillation or data augmentation. Teacher: AMDA ResNet50. Student: ResNet18. All student variants use data augmentation (AMDA) during distillation.

Variant	Img100 Err.	Img100-C mCE	Img100-R Err.
Standard	11.9	50.7	68.9
AMDA	13.0	28.0	53.6
+ KD	**10.6**	26.0	**50.0**
+ AT	11.1	27.2	51.6
+ CRD	11.3	**25.4**	51.2

From Figure 1 we also see that distilling from the AMDA teacher performs best in terms of OOD robustness across all distillation methods and students. Similar results can also be seen for other student architectures (see appendix, Figure 3). Importantly, the improvements cannot be attributed to the increase in *i.i.d* accuracy, which according to Hendrycks et al. (2020a) is a good predictor for OOD performance. The students of the AMDA teacher have comparable or worse *i.i.d* performance compared to the other distilled models (see Figure 1). Also surprisingly, the OOD performance of the teacher does not determine how good a teacher is for distillation, see the SWSL teacher in Figure 1. This might hint at an inherent difference of the robust representation learned by a network with data augmentation, as compared to networks trained in a semi-supervised fashion.

Are robustness gains from distillation and data augmentation complementary? We also trained the ResNet18 student models with distillation and data augmentation (AMDA). The results are shown in Table 1.

We observe that using distillation is still beneficial when combined with data augmentation, however the gains are diminished. We also observe similar improvements for MobileNetV2 and MnasNet (see appendix, Table 3).

3.2.2. PRUNING

To test the impact of pruning on OOD performance we first considered both structured and unstructured pruning. However, we found that structured pruning methods, like L^1-filter pruning, lead to larger degradation of OOD performance compared to the unstructured counterparts (see Appendix B). Thus, we decided to focus more on unstructured pruning methods (Eg: *global weight*).

Pruning vs Distillation. As a first step, we were interested in comparing distilled and pruned models. On the one hand, we tried to *distill* the performance of a ResNet50 (AMDA) to a ResNet18 student model. On the other hand, we tried to *prune* a ResNet50 (AMDA) down to the size of the ResNet18 model with global weights pruning. In both cases, we used AMDA data augmentation. The results are shown in Table 2:

Table 2. Performance of an AMDA ResNet50 (R50) pruned down to the size of a ResNet18 (R18) using unstructured global weight pruning (denoted as "GW"). The pruned model is denoted as R50-pr.

Model	Variant	Params (M)	Img100 Err.(%)	Img100-C mCE(%)	Img100-R Err.(%)
R18	AMDA	11.2	13.0	28.0	53.6
R50 →R18	+KD	11.2	10.6	26.0	50.0
R50	AMDA	22.4	6.2	20.6	40.9
R50-pr	+ GW	11.1	7.3	20.8	45.0

Compared to the compressed models obtained via distillation, unstructured pruning can help preserve more robustness. This can probably be explained by bigger sparse models tending to perform better than smaller dense counterparts (Zhu & Gupta, 2017). We also found similar results when pruning down a ResNet18, which performed better than even an unpruned MobileNet or MnasNet (see Appendix D).

3.3. Robustness at high compression ratios

Hooker et al. highlighted that pruned models at high compression ratios can have very low OOD robustness, making them potentially useless in the real world. Yet, their analysis only considered one unstructured method and no data augmentation. In this section we investigate — **Can a better model compression vs OOD robustness trade-off be achieved?**

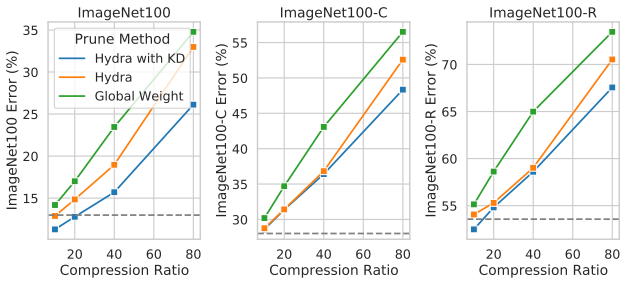


Figure 2. Hydra with and without KD distillation vs global weight pruning for different compression ratios on ImageNet100, ImageNet100-C and ImageNet100-R. The performance of the unpruned AMDA ResNet18 is marked with the dashed grey line.

We compared *global weight* pruning and *Hydra*, which was successfully used by (Sehwag et al., 2020) to achieve state-of-the-art results on adversarial robustness. To adopt the Hydra framework for our setting, instead of an adversarial robustness objective, we used the AMDA data augmentation and the JSD consistency loss (Hendrycks et al., 2020b). Specifically, we used AMDA for all three phases of the Hydra framework: **a)** pre-training, **b)** Hydra's mask selection phase **c)** fine-tuning phase. We analyze the benefits of using AMDA over not using AMDA during mask selection in Appendix E.

From Figure 2, we can see that both Hydra and global weight pruning can prune 90% (compression ratio of 10) of the parameters without sacrificing much or even improving in terms of OOD robustness. We can see that Hydra outperforms the strong global weight pruning baseline on all three domains. Hydra combined with KD leads to the best results, showing that distillation and pruning can be successfully combined to achieve even better model size vs robustness trade-offs. **Thus, OOD robustness at high-compression ratios can be improved.**

4. Conclusion

We found that distillation methods can help distill OOD robustness while training only on *i.i.d* samples. Also, we showed that unstructured pruning methods like Hydra can achieve very high compression ratios without significantly degrading OOD robustness. Strong data augmentation methods play an important role in improving robustness, even for smaller models. We hope our empirical results will serve as strong baselines and inspire future research in this direction.

References

Blalock, D., Ortiz, J. J. G., Frankle, J., and Guttag, J. What is the State of Neural Network Pruning? *arXiv:2003.03033 [cs, stat]*, March 2020. URL http://arxiv.org/abs/2003.03033. arXiv: 2003.03033.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

Falcon, W., Borovec, J., Wälchli, A., Eggert, N., Schock, J., Jordan, J., Skafte, N., Ir1dXD, Bereznyuk, V., Harris, E., Murrell, T., Yu, P., Præsius, S., Addair, T., Zhong, J., Lipin, D., Uchida, S., Bapat, S., Schröter, H., Dayma, B., Karnachev, A., Kulkarni, A., Komatsu, S., Martin.B, SCHIRATTI, J.-B., Mary, H., Byrne, D., Eyzaguirre, C., cinjon, and Bakhtin, A. PyTorchLightning/pytorch-lightning: 0.7.6 release, May 2020. URL https://zenodo.org/record/3828935#.YC45Lc9Khqs.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Pruning neural networks at initialization: Why are we missing the mark? *arXiv preprint arXiv:2009.08576*, 2020.

Goldblum, M., Fowl, L., Feizi, S., and Goldstein, T. Adversarially robust distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3996–4003, 2020.

Han, S., Pool, J., Tran, J., and Dally, W. J. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*, December 2015. URL http://arxiv.org/abs/1512.03385. arXiv: 1512.03385.

Hendrycks, D. and Dietterich, T. G. Benchmarking Neural Network Robustness to Common Corruptions and Surface Variations. *arXiv:1807.01697 [cs, stat]*, April 2019. URL http://arxiv.org/abs/1807.01697. arXiv: 1807.01697.

Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., Song, D., Steinhardt, J., and Gilmer, J. The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization. *arXiv:2006.16241 [cs, stat]*, August 2020a. URL http://arxiv.org/abs/2006.16241. arXiv: 2006.16241.

Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., and Lakshminarayanan, B. AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty. *arXiv:1912.02781 [cs, stat]*, February 2020b. URL http://arxiv.org/abs/1912.02781. arXiv: 1912.02781.

Hinton, G., Vinyals, O., and Dean, J. Distilling the Knowledge in a Neural Network. *arXiv:1503.02531 [cs, stat]*, March 2015. URL http://arxiv.org/abs/1503.02531. arXiv: 1503.02531.

Hooker, S., Courville, A., Clark, G., Dauphin, Y., and Frome, A. What Do Compressed Deep Neural Networks Forget? *arXiv:1911.05248 [cs, stat]*, July 2020. URL http://arxiv.org/abs/1911.05248. arXiv: 1911.05248.

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning Filters for Efficient ConvNets. *arXiv:1608.08710 [cs]*, March 2017. URL http://arxiv.org/abs/1608.08710. arXiv: 1608.08710.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. Learning efficient convolutional networks through network slimming, 2017.

Ramanujan, V., Wortsman, M., Kembhavi, A., Farhadi, A., and Rastegari, M. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11893–11902, 2020.

Renda, A., Frankle, J., and Carbin, M. Comparing Rewinding and Fine-tuning in Neural Network Pruning. *arXiv:2003.02389 [cs, stat]*, March 2020. URL http://arxiv.org/abs/2003.02389. arXiv: 2003.02389.

Saikia, T., Schmid, C., and Brox, T. Improving robustness against common corruptions with frequency biased models. *arXiv:2103.16241 [cs]*, March 2021. URL http://arxiv.org/abs/2103.16241. arXiv: 2103.16241.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *arXiv:1801.04381 [cs]*, March 2019. URL http://arxiv.org/abs/1801.04381. arXiv: 1801.04381.

Sehwag, V., Wang, S., Mittal, P., and Jana, S. HYDRA: Pruning Adversarially Robust Neural Networks. *arXiv:2002.10509 [cs, stat]*, November 2020. URL http://arxiv.org/abs/2002.10509. arXiv: 2002.10509.

Sergey, Z. and Komodakis, N. Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer. *arXiv:1612.03928 [cs]*, February 2017. URL http://arxiv.org/abs/1612.03928. arXiv: 1612.03928.

Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le, Q. V. MnasNet: Platform-Aware Neural Architecture Search for Mobile. *arXiv:1807.11626 [cs]*, May 2019. URL http://arxiv.org/abs/1807.11626. arXiv: 1807.11626.

Tian, Y., Krishnan, D., and Isola, P. Contrastive Representation Distillation. *arXiv:1910.10699 [cs, stat]*, January 2020. URL http://arxiv.org/abs/1910.10699. arXiv: 1910.10699.

Xie, Q., Luong, M.-T., Hovy, E., and Le, Q. V. Self-training with noisy student improves imagenet classification, 2020.

Yalniz, I. Z., Jégou, H., Chen, K., Paluri, M., and Mahajan, D. Billion-scale semi-supervised learning for image classification. *arXiv:1905.00546 [cs]*, May 2019. URL http://arxiv.org/abs/1905.00546. arXiv: 1905.00546.

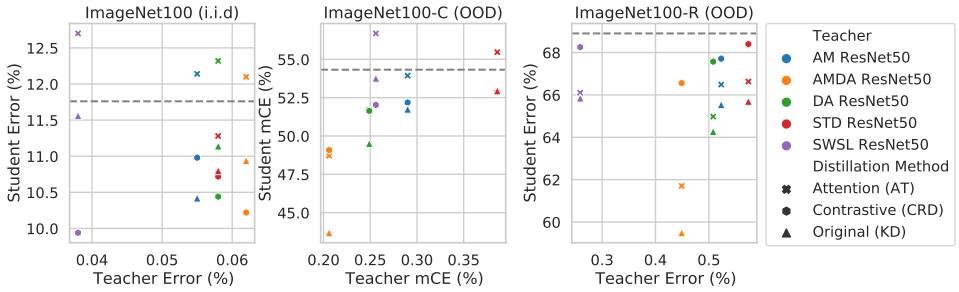
Yin, D., Lopes, R. G., Shlens, J., Cubuk, E. D., and Gilmer, J. A fourier perspective on model robustness in computer vision, 2020.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6023–6032, 2019.

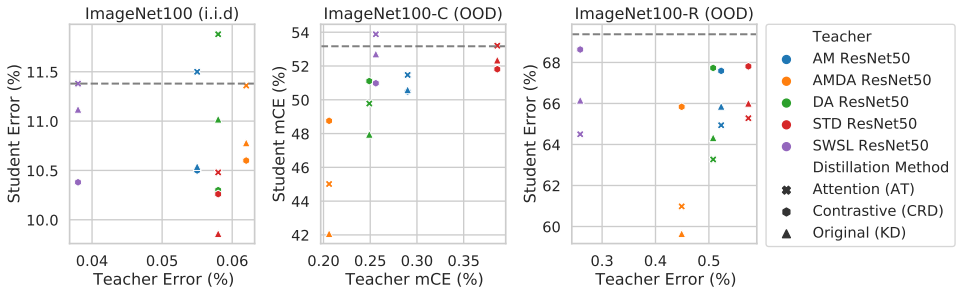
Zhu, M. and Gupta, S. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv:1710.01878 [cs, stat]*, November 2017. URL http://arxiv.org/abs/1710.01878. arXiv: 1710.01878.

A. Additional Distillation Results

In the following, we extended the analysis of Figure 1 by using also MnasNet and MobileNetV2 as students:



(a) Student: MobileNetV2



(b) Student: MnasNet

Figure 3. We compared the student error vs teacher error on ImageNet100, ImageNet100-C and ImageNet100-R. Note, that the teachers and distillation methods are the same as in Figure 1.

In Figure 3, we can see again that the AMDA ResNet50 teacher seems to be the best across all distillation methods. The Swsl ResNet50 is the best performing model on ImageNet100-R but its students do not perform better on ImageNe100-R. The attention based distillation method (AT) only consistently works well for ResNet18, but not for MobileNetV2 and MnasNet. This might be due to the limitation of the AT method requiring student and teacher architectures to be from the same architectural family (Tian et al., 2020).

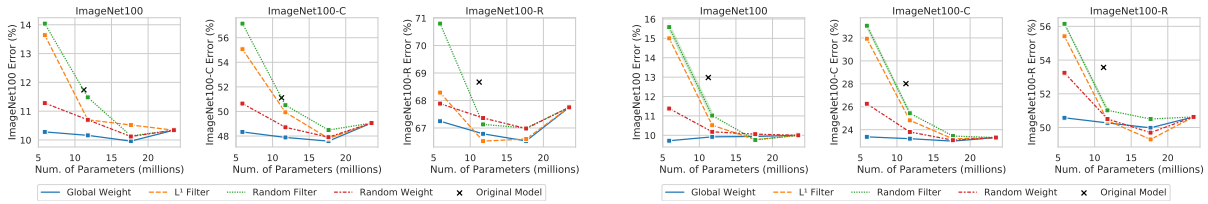
Combining distillation with data augmentation. Similar to Table 1 we also tested the combination of AMDA with distillation. Also, here distillation improved the results on all domains, but when compared to Figure 3 the OOD improvements are diminished.

Table 3. Combining the distillation methods (KD, AT and CRD) with data augmentation. Standard: Baseline model without distillation or data augmentation. Teacher: AMDA ResNet50. Students: MnasNet and MobileNetV2. All student variants use data augmentation (AMDA) during distillation

Model	Variant	Img100 Err.	Img100-C mCE	Img100-R Err.
MobileNetV2	Standard	12.3	54.7	69.2
	AMDA	13.2	30.4	56.4
	+ KD	**11.5**	30.0	**53.3**
	+ AT	13.3	31.5	53.9
	+ CRD	12.6	**29.7**	55.1
MnasNet	Standard	11.6	52.9	69.3
	AMDA	13.1	30.4	56.0
	+ KD	**11.4**	29.3	**53.2**
	+ AT	12.2	29.5	53.5
	+ CRD	12.0	**29.2**	55.0

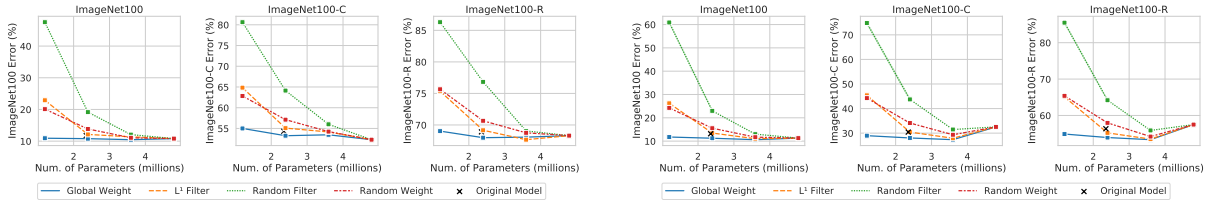
B. Additional Pruning Results

In this section, we provide an overview of the robustness vs model size trade-offs of different structured and unstructured pruning methods. We considered ResNet18, MnasNet and MobileNetV2 models, with *global weight* (unstructured) and L^1-*Filter* pruning (structured) pruning methods. As recommended by (Frankle et al., 2020), we also used some random baselines to compare to, namely *random weights* pruning (unstructured), which just randomly prunes weights, and *random filter* pruning (structured), which prunes whole filters at random. Additionally, we decided to scale up the models, in width, to twice their respective number of parameters. This allows us to then to prune the scaled-up model to the original size and potentially find a better robustness trade-off. As we see in Figure 4, this technique gives us reasonable results for ResNet18 but fails to work for MobileNetV2 and MnasNet.



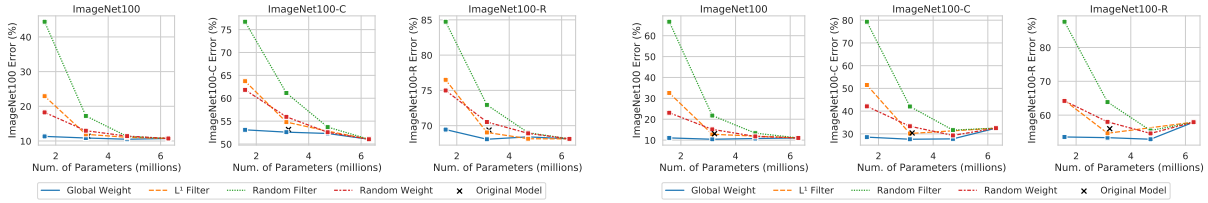
(a) ResNet18 without AMDA

(b) ResNet18 with AMDA



(c) MobileNetV2 without AMDA

(d) MobileNetV2 with AMDA



(e) MnasNet without AMDA

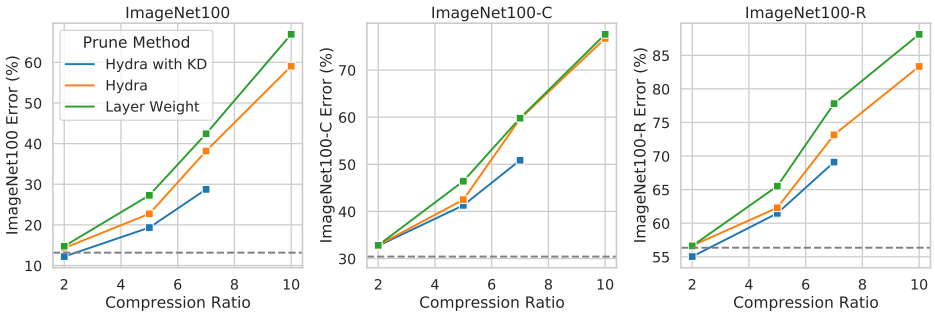
(f) MnasNet with AMDA

Figure 4. We report the error on ImageNet 100, ImageNet 100-C and ImageNet-R vs the number of parameters. In 4(b), 4(d) and 4(f) we used AMDA during training and fine-tuning, whereas in figures 4(a), 4(c) and 4(e) we did not. The performance of the respective unpruned models are marked with a 'X' in the figures.

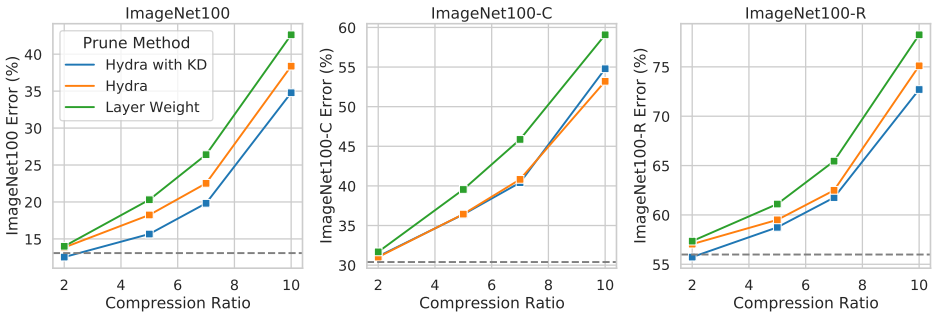
In Figure 4, we observe that the unstructured pruning method global weight can mostly preserve the performance on all three domains. On the other side, the performance of structured methods quickly deteriorates for higher pruning amounts. Furthermore, we can also see that for MobileNet and MnasNet, scaling the model up in size and then pruning it to the original size does not improve the performance for structured pruning. This might be because the models are already efficient and compact and were found via a structural neural architecture search (Tan et al., 2019).

C. Combining Hydra with KD

Here we present additional Hydra results for MobileNetV2 and MnasNet. Note that, for MobileNetV2 and MnasNet instead of using global weight pruning, thus comparing the magnitude of weights across different layers, we used *layer weight* pruning, which just prunes each layer the same amount. This was necessary because, at high pruning ratios, global weight pruning often prunes out multiple layers at the end of the network, which we needed to avoid.



(a) MobileNetV2



(b) MnasNet

Figure 5. We compare Hydra with and without KD distillation vs global weight pruning for different compression ratios on ImageNet100, ImageNet100-C and ImageNet100-R. The respective student networks are mentioned in the caption. The performance can be compared to the unpruned AMDA ResNet18 which is the grey dashed line.

Figure 5 strengthens our finding of section 3.3, that Hydra outperforms global weight pruning on clean, corruption, and rendition errors, and combining Hydra with KD during mask selection and fine-tuning helps us improve further on clean and rendition errors.

D. Achieving small-scale OOD Robust Networks.

Here we further explore our findings in section 3.2.2 by analysing the ImageNet100, ImageNet-C and ImageNet-R performances of a very sparse ResNet18 in comparison to MobileNetV2 and MnasNet. We observe that it is more beneficial to prune the bigger ResNet18 by a lot than pruning a MnasNet or MobileNetV2. The effects seems to be stronger for OOD robustenss than for *i.i.d*.

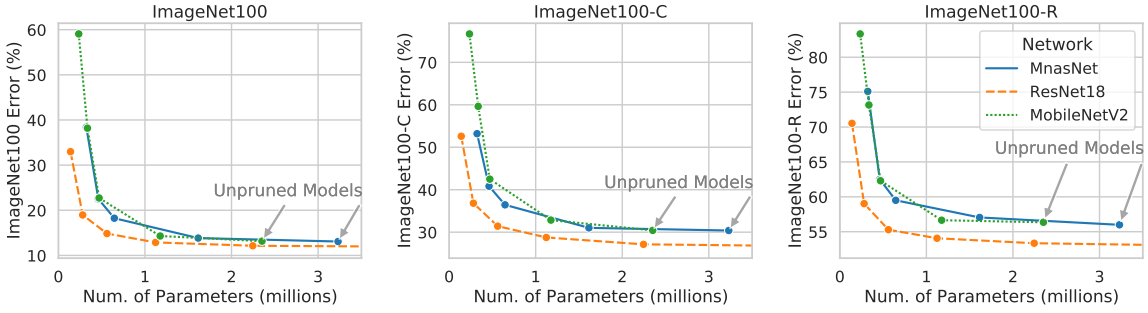


Figure 6. Performances of ResNet18, MobileNetV2 and MnasNet pruned with Hydra and trained with AMDA on ImageNet100, ImageNet100-C and ImageNet-R datasets after pruning ResNet18, MobileNetV2 and MnasNet trained with AMDA using Hydra. The unpruned MobileNetV2 and MnasNet performances have been annotated in the respective figures.

E. Robustness aware pruning?

As Hydra allows us to select a mask optimizing the AMDA loss, we also wanted to see the difference between choosing a pruning mask that does or does not take robustness into account. To measure the difference, we used AMDA for training and fine-tuning the pruned model, but switched between using and not using AMDA during the mask selection phase. In Table 4 we see that pruning with robustness in mind does make a difference for Hydra:

Table 4. Difference between using and not using AMDA during Hydra's mask selection phase. We can see a trade-off between *i.i.d* and OOD accuracy. A negative value means that the mask selection with AMDA is better than without AMDA.

Comp. Ratio	Δ Img100 Err. (%)	Δ Img100-C mCE (%)	Δ Img100-R Err. (%)
5	0.1	-1.1	-0.2
10	1.0	-1.5	-1.0
20	-0.3	-1.9	-1.6
40	0.4	-1.5	-1.2

F. Experimental Setup

In this section, we provide additional details of our experimental setup that might be help reproduce our results.

Implementation details. To make our implementation scale to ImageNet and make it more reusable we used PyTorchLightning (Falcon et al., 2020). For pruning, we used the standard pruning library of PyTorch and for CRD loss we integrated the original implementation of Tian et al. (2020) in our codebase. For the Hydra experiments, we integrated the code from Ramanujan et al. (2020).

Fine-Tuning Regime. In general, we always used the same ImageNet multistep learning rate schedule. Thus, also for fine-tuning our pruned models and for Hydra's mask selection phase. Note that, for one-shot pruning this is the recommended default by Renda et al. (2020) and is called learning rate rewinding.

Teachers. For the experiments with knowledge distillation, we used different publicly available ResNet50s as our teachers. The AugMix (AM) version from (Hendrycks et al., 2020b), the DeepAugment (DA) and AugMix + DeepAugment (AMDA) version are from (Hendrycks et al., 2020a) and the semi-supervised trained version from (Yalniz et al., 2019). Note, that all of them were trained on the complete ImageNet dataset.

Hyperparameters. For most of the hyperparameters for KD, CRD, and AT, we used the default values mentioned in Tian et al. (2020). We changed the values for a few hyperparameters: for the mixture coefficient of KD, we used 0.5 for most experiments with AMDA whereas without AMDA we used 0.9. For scaling up the JSD loss of AugMix, we used the default value in Hendrycks et al. (2020b), which is 12.0, but used sometimes 6.0 when combining with distillation.

G. ImageNet100 - Class Ids

The class ids for the ImageNet-100 dataset that we use for our experiments are the same as in Saikia et al. (2021). Note, that ImageNet-R already only has 200 classes, so the subset must be chosen accordingly:

n01443537, n01484850, n01494475, n01498041, n01514859, n01518878, n01531178, n01534433, n01614925, n01616318, n01630670, n01632777, n01644373, n01677366, n01694178, n01748264, n01770393, n01774750, n01784675, n01806143, n01820546, n01833805, n01843383, n01847000, n01855672, n01860187, n01882714, n01910747, n01944390, n01983481, n01986214, n02007558, n02009912, n02051845, n02056570, n02066245, n02071294, n02077923, n02085620, n02086240, n02088094, n02088238, n02088364, n02088466, n02091032, n02091134, n02092339, n02094433, n02096585, n02097298, n02098286, n02099601, n02099712, n02102318, n02106030, n02106166, n02106550, n02106662, n02108089, n02108915, n02109525, n02110185, n02110341, n02110958, n02112018, n02112137, n02113023, n02113624, n02113799, n02114367, n02117135, n02119022, n02123045, n02128385, n02128757, n02129165, n02129604, n02130308, n02134084, n02138441, n02165456, n02190166, n02206856, n02219486, n02226429, n02233338, n02236044, n02268443, n02279972, n02317335, n02325366, n02346627, n02356798, n02363005, n02364673, n02391049, n02395406, n02398521, n02410509, n02423022