LOST IN PRUNING: THE EFFECTS OF PRUNING NEURAL NETWORKS BEYOND TEST ACCURACY

Lucas Liebenwein [1] **Cenk Baykal** [1] **Brandon Carter** [1] **David Gifford** [1] **Daniela Rus** [1]

ABSTRACT

Neural network pruning is a popular technique used to reduce the inference costs of modern, potentially overparameterized, networks. Starting from a pre-trained network, the process is as follows: remove redundant parameters, retrain, and repeat while maintaining the same test accuracy. The result is a model that is a fraction of the size of the original with comparable predictive performance (test accuracy). Here, we reassess and evaluate whether the use of test accuracy alone in the terminating condition is sufficient to ensure that the resulting model performs well across a wide spectrum of "harder" metrics such as generalization to out-of-distribution data and resilience to noise. Across evaluations on varying architectures and data sets, we find that pruned networks effectively approximate the unpruned model, however, the prune ratio at which pruned networks achieve commensurate performance varies significantly across tasks. These results call into question the extent of *genuine* overparameterization in deep learning and raise concerns about the practicability of deploying pruned networks, specifically in the context of safety-critical systems, unless they are widely evaluated beyond test accuracy to reliably predict their performance. Our code is available at `https://github.com/lucaslie/torchprune`.

1 INTRODUCTION

Deep neural networks (Russakovsky et al., 2015; You et al., 2019) tend to contain millions or billions of parameters, necessitating costly computational resources in order to train and deploy them in practice, and motivating the need to develop compact networks with fewer parameters (Han et al., 2015; Liebenwein et al., 2020). Such a reduction in parameter count alleviates the computational burden on training and inference, making it easier to deploy high-capacity models to small devices and use them in resource-constrained environments (Baykal et al., 2019a; Frankle & Carbin, 2019; Renda et al., 2020).

To this end, network pruning is a popular technique used to obtain compact networks that maintain the accuracy of the original network with orders of magnitude of fewer parameters (Blalock et al., 2020; Gale et al., 2019). Typical pruning algorithms either proceed by gradually pruning the network during training (Gale et al., 2019; He et al., 2018; Zhu & Gupta, 2017) or by pruning the network after training followed by a retraining period (Baykal et al., 2019b; Han et al., 2015; Liebenwein et al., 2020; Renda et al., 2020).

A prune pipeline with retraining usually consists of the following steps:

1. Prune weights ("unstructured pruning") or filters/neurons ("structured pruning") from the trained network according to some criterion of importance;
2. Retrain the resulting network to regain the full accuracy;
3. Iteratively repeat steps 1 & 2 to further reduce the size.

These approaches are both simple and network-agnostic, and have been shown to yield state-of-the-art pruning results for both unstructured (Renda et al., 2020) and structured pruning (Liebenwein et al., 2020). For example, these techniques enable pruning of 89% of the weights (Renda et al., 2020) or 84% of the filters (Liebenwein et al., 2020) of a ResNet56 trained on CIFAR10.

While the ability to prune large portions of a network is not obvious at first sight, common wisdom attributes the apparent overparameterization (Arora et al., 2018; Baykal et al., 2019a; Du et al., 2019; Frankle & Carbin, 2019) of modern deep learning architectures as one of the key reasons why pruning such large portions of the network is possible without harming the performance of the network.

In other words, successfully pruning a network entails identifying and removing the redundant parameters. Moreover, starting from a pre-trained, overparameterized network with good performance (as opposed to a small, randomly initialized network) ensures that the pruned network maintains the same level of performance as its uncompressed counterpart.

In this paper, we revisit these common assumptions and

[1]Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology, Cambridge, MA, USA. Correspondence to: Lucas Liebenwein <lucasl@mit.edu>.

Proceedings of the 4^{th} MLSys Conference, San Jose, CA, USA, 2021.

rigorously assess how pruning using state-of-the-art prune-retrain techniques (Baykal et al., 2019b; Liebenwein et al., 2020; Renda et al., 2020) affects the function represented by a neural network, including the similarities and disparities exhibited by a pruned network with respect to its unpruned counterpart.

We formalize the notion of (functional) similarities between networks by introducing novel types of classification-based functional distance metrics. Using these metrics, we test the hypothesis that pruned models are (functionally) similar to their (unpruned) parent network and can be reliably distinguished from separately trained networks. We term the network's ability to be pruned for a particular task without performance decrease its *prune potential*, i.e., the maximal prune ratio for which the pruned network maintains its original performance, and test a network's prune potential under various tasks. The prune potential provides insights into the amount of overparameterization the network exhibits for a particular task and thus serves as a useful indicator of how much of the network can be safely pruned.

Our findings. We find that the pruned models are functionally similar to the uncompressed parent model, which enables us to distinguish the parent of a pruned network for a range of prune ratios. Despite the similarity between the pruned network and its parent, we observe that the prune potential of the network varies significantly for a large number of tasks. That is, a pruned model may be of similar predictive power as the original one when it comes to test accuracy, but may be much more brittle when faced with out of distribution data points. This raises concerns about deploying pruned models on the basis of accuracy alone, in particular for safety-critical applications such as autonomous driving (Schwarting et al., 2020), where unforeseen, out-of-distribution, or noisy data points commonly arise. Our insights, which hold even when considering robust training objectives, underscore the need to consider task-specific evaluation metrics during pruning, prior to the deployment of a pruned network to, e.g., safety-critical systems. These results also question the common assumption that there exists a significant amount of "redundant" parameters to begin with and provide a robust framework to measure the amount of genuine overparameterization in networks.

Guidelines. Based on these observations we formulate a set of easy-to-follow guidelines to pruning in practice:

1. Don't prune if unexpected shifts in the data distribution may occur during deployment.
2. Prune moderately if you have partial knowledge of the distribution shifts during training and pruning.
3. Prune to the full extent if you can account for all shifts in the data distribution during training and pruning.
4. Maximize the prune potential by explicitly considering data augmentation during retraining.

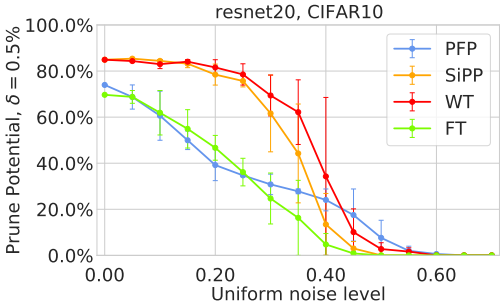


Figure 1: A network's ability to be pruned without loss of accuracy, i.e., its "prune potential", can be significantly affected by small changes in the input data distribution.

Contributions.

- We propose novel functional distance metrics for classification-based neural networks and investigate the functional similarities between the pruned network and its unpruned counterpart.
- We propose the notion of *prune potential*, i.e., the maximal prune ratio (model sparsity) at which the pruned network can achieve commensurate performance, as a quantifiable means to estimate the overparameterization of a network and show that it is significantly lower on challenging inference tasks.
- We provide a unified framework to establish task-specific guidelines that help practitioners assess the effects of pruning during the design and deployment of neural networks in practice.
- We conduct experiments across multiple data sets, architectures, and pruning methods showing that our observations hold across common pruning benchmarks and real-world scenarios.

2 Related Work

Pruning. Generally, pruning is categorized into unstructured (Han et al., 2015) and structured (He et al., 2018; Li et al., 2019) pruning approaches. While the former is beneficial for research, it provides little computational speed-ups compared to structured pruning (Luo & Wu, 2018). Moreover, pruning can be performed before (Lee et al., 2019; Tanaka et al., 2020; Wang et al., 2020), during (Kusupati et al., 2020; Yu et al., 2018; Zhu & Gupta, 2017), or after training (Baykal et al., 2019b; Liebenwein et al., 2020; Singh & Alistarh, 2020), and repeated iteratively (Renda et al., 2020). In this work, we focus on iterative pruning with retraining after training. While computationally more expensive than other approaches, this pruning pipeline produces state-of-the-art results, is usually architecture-agnostic, and requires little hyperparameter tuning, thus providing a simple and effective baseline for our experiments. A thorough overview of recent pruning approaches is, e.g., provided by Blalock et al. (2020); Gale et al. (2019); Liu et al. (2019).

Robustness. Our work builds on and extends previous work that investigates the robustness of pruned networks. The works of Gamboa et al. (2020); Guo et al. (2018); Wang et al. (2018); Ye et al. (2019); Zhao et al. (2018) have investigated the effect of adversarial input on pruned networks, however, the resulting evidence is inconclusive. While Gamboa et al. (2020); Guo et al. (2018) report that adversarial robustness may improve or remain the same for pruned networks, Wang et al. (2018); Ye et al. (2019); Zhao et al. (2018) report decreased robustness for pruned networks. Recently, the work of Hooker et al. (2019) investigated at the individual image level whether certain class accuracies are more affected than others. In contrast to prior work, we investigate both the functional similarities of pruned networks and the task-specific prune potential. Our work highlights the need to assess pruned networks across a wide variety of tasks to safely deploy networks due to the unpredictable nature of a network's prune potential.

Robust training and pruning. Recent works (Dhillon et al., 2018; Wijayanto et al., 2019) have investigated pipelines that incorporate pruning and robust training to obtain simultaneously sparse and robust networks. Gui et al. (2019); Sehwag et al. (2019) use magnitude-based pruning (Han et al., 2015) to train sparse, robust networks, while the work of Sehwag et al. (2020) incorporates the robustness objective into an optimization-based pruning procedure. Our work offers a complementary viewpoint to the findings of prior work in that we can indeed efficiently train pruned networks that are (adversarially) robust. However, for the first time we make the crucial observation that pruned networks are disproportionally more affected by distributional changes in the input regardless of the training procedure.

Generalization via pruning. Among others Arora et al. (2018); Baykal et al. (2019a); Nagarajan & Kolter (2019); Zhou et al. (2018) have demonstrated that (provable) pruning can facilitate tighter generalization bounds of networks. Pruning is hereby viewed as a form of noise injection into the network and by quantifying the (unpruned) network's ability to withstand random noise, they characterize the prunability of the network to establish generalization bounds. However, these works do not capture the generalization ability of pruned networks under distributional changes. Here, we show that as a result of pruning, the network's ability to withstand noise and other types of data corruption is diminished. In other words, the network's robustness is "traded" in exchange for compactness.

Implicit regularization via overparameterization. Our work is also related to the beneficial role of overparameterization in deep learning (Allen-Zhu et al., 2019; Zhang et al., 2016). Conventional wisdom (Du et al., 2019; Neyshabur et al., 2015; 2018; 2019) states that stochastic gradient methods used for training implicitly regularize the network, in effect ensuring it generalizes well despite the potential to severely overfit. Moreover, the works of Belkin et al. (2019); Nakkiran et al. (2020) note that the implicit regularization potential increases with the parameter count. Our work, for the first time, thoroughly establishes that pruned networks distinctly suffer more from small shifts in the input data distribution compared to unpruned networks – possibly due to the decreased *implicit* regularization potential as a result of the lower parameter count. Our findings concretely highlight that *explicit* regularization in the form of robust training can help regain some of the robustness properties that would otherwise be lost.

3 METHODOLOGY

3.1 Pruning Setup

For our experiments, we consider a variety of network architectures, data sets, and pruning methods as outlined below. Our pruning pipeline, see Algorithm 1, is based on iterative pruning and retraining following Renda et al. (2020). It is simple, network-agnostic, and widely used; hence we opted to choose it as representative pruning pipeline.

Data sets and network architectures. We consider CIFAR10 (Torralba et al., 2008), ImageNet (Russakovsky et al., 2015), and Pascal VOC segmentation data (Everingham et al., 2015) as data sets. We consider ResNets 18/56/110 (He et al., 2016), WRN16-8 (Zagoruyko & Komodakis, 2016), DenseNet22 (Huang et al., 2017), and VGG16 (Simonyan & Zisserman, 2015) on CIFAR10; ResNet18 and 101 (He et al., 2016) on ImageNet; and a DeeplabV3-ResNet50 (Chen et al., 2017) on VOC.

Training. For all networks, we apply the standard training parameters as indicated in the respective papers. We apply the linear scaling rule of Goyal et al. (2017) when training on multiple GPUs in parallel including warm-up. All hyperparameter settings with their numerical values are listed in Appendix B. All networks are trained once to completion before pruning (Line 2 of Algorithm 1).

Pruning. We consider multiple unstructured and structured pruning methods, where we prune individual weights and filters/neurons, respectively, see Table 1 for an overview. We perform pruning by updating a binary mask indicating whether the corresponding weight is active or pruned (Line 5 of Algorithm 1).

Unstructured pruning. The weight pruning approaches we consider follow a global pruning strategy: (1) globally sort the weights according to their relative importance, i.e., sensitivity, and (2) prune $r_{\text{prune}}\%$ of the weights with the lowest sensitivity. In particular, we study two methods to compute the sensitivity of weights, weight thresholding (Renda et al., 2020) and SiPP (Baykal et al., 2019b).

Type		Method	Data-informed	Sensitivity	Scope
Unstructured (Weights)	WT:	Weight Thresholding (Renda et al., 2020)	✗	$\lvert W_{ij}\rvert$	Global
	SiPP:	Sensitivity-informed Pruning (Baykal et al., 2019b)	✓	$\propto \lvert W_{ij}a_j(x)\rvert$	Global
Structured (Neurons/Filters)	FT:	Filter Thresholding (Renda et al., 2020)	✗	$\lVert W_{:j}\rVert_1$	Local
	PFP:	Provable Filter Pruning (Liebenwein et al., 2020)	✓	$\propto \lVert W_{:j}a(x)\rVert_\infty$	Local

Table 1: Overview of the pruning methods evaluated. Here, $a(x)$ denotes the activation of the corresponding layer with respect to a sample input x to the network.

Weight thresholding (WT) is a simple heuristic, originally introduced by Han et al. (2015) and re-purposed by Renda et al. (2020), that defines the sensitivity of a weight as the magnitude of the weight. SiPP, on the other hand, is a data-informed approach with provable guarantees to computing weight sensitivities (Baykal et al., 2019b). The approach uses a small batch of input points $\mathcal{S} \subseteq \mathcal{P}$, e.g., from the validation set, to evaluate the saliency of each network parameter. This is done by incorporating the corresponding (sample) activations, $a(x)$, $x \in \mathcal{S}$, along with the weight into the importance computation (see Table 1).

Algorithm 1 PRUNERETRAIN(n_{cycles}, r_{prune}, n_{train}, ρ_{train})

Input: n_{cycles}: number of prune-retrain cycles; r_{prune}: relative prune ratio; n_{train}: number of train epochs; ρ_{train}: training hyper-parameters
Output: c: pruning mask, θ: parameters of the pruned network

1: $\theta_0 \leftarrow$ RANDOMINIT()
2: $\theta \leftarrow$ TRAIN($\theta_0, n_{\text{train}}, \rho_{\text{train}}$)
3: $c \leftarrow 1^{|\theta_0|}$ ▷ binary mask for the parameters
4: **for** $i \in [n_{\text{cycles}}]$ **do**
5: $c \leftarrow$ PRUNE($c \odot \theta, r_{\text{prune}}$) ▷ Prune $r_{\text{prune}}\%$ of the remaining parameters.
6: $\theta \leftarrow$ TRAIN($c \odot \theta, n_{\text{train}}, \rho_{\text{train}}$)
7: **end for**
8: **return** c, θ

Structured pruning. The filter/neuron pruning approaches we consider follow a two-step strategy: (1) allocate a per-layer prune ratio satisfying the overall prune ratio and (2) prune the filters with lowest sensitivity in each layer. We study Filter Tresholding (FT) as used by Renda et al. (2020) and PFP (Liebenwein et al., 2020). FT, as originally introduced by He et al. (2018); Li et al. (2016) and used here analogous to Renda et al. (2020), uses the filter norm to evaluate its sensitivity. Layer allocation is performed manually and we deploy a uniform prune ratio across layers to avoid further hyperparameters. PFP (Liebenwein et al., 2020) is an extension of SiPP that evaluates filter sensitivity as the maximum sensitivity of the channel in the next layer (ℓ_∞-norm of the corresponding weight sensitivity). PFP uses the associated theoretical error guarantees to optimally allocate the layer-wise budget.

Retraining. We retrain the network with the exact training hyperparameters as is common (Baykal et al., 2019b; Liebenwein et al., 2020; Renda et al., 2020). Specifically, we re-use the same learning rate schedule and retrain for the same amount of epochs (Line 6 of Algorithm 1). After retraining, we iteratively repeat the pruning procedure to obtain even smaller networks (Lines 4-7 of Algorithm 1).

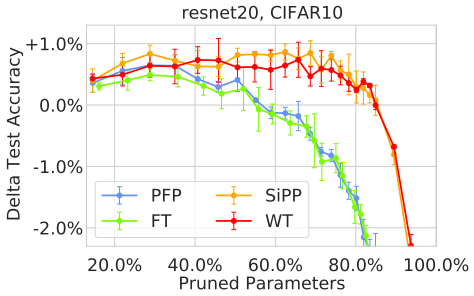


Figure 2: The accuracy of the generated pruned models for the evaluated pruning schemes for various target prune ratios using iterative fine-tuning.

Prune results. Figure 2 shows an exemplary test accuracy curve of a Resnet20 (CIFAR10) across different target prune ratios for iterative pruning. The remaining prune results are summarized in the supplementary material. We note while PRUNERETRAIN may be more computationally expensive than other pruning pipelines, it is network-agnostic and produces state-of-the-art pruning results (Renda et al., 2020).

3.2 Experiments Roadmap

Our observations stem from multiple experiments that can be clustered into one set of experiments pertaining to understanding the functional similarities (i.e., functional distance) of pruned networks and one set pertaining to the prune potential on a variety of image-based classification tasks. First, we compare subsets of pixels that are sufficiently informative for driving the network's decision. For this, we use the feature-wise (pixel-wise) selection mechanism of Carter et al. (2019; 2020). We also investigate how pruned networks behave under ℓ_∞-bounded random noise. Second, we assess the ability of pruned networks to generalize to out-of-distribution (o.o.d.) test data sets that contain random noise (ℓ_∞-bounded) and common corruptions (Hendrycks & Dietterich, 2019; Recht et al., 2018; 2019) including weather, contrast, and brightness changes. In all experiments, we compare the performance, i.e., the accuracy on the various test sets, of pruned networks with those of their unpruned counterparts as well as a separately trained unpruned network. Each experiment is repeated 3 times and we report

mean and standard deviation (error bars). In the main part of the paper, we focus on a subset of representative results to highlight the key findings. We refer the interested reader to the appendix for a complete exposition of our results.

4 Function Distance

Given a pruned model with commensurate test accuracy relative to the parent (uncompressed) network, can we conclude that the *function* represented by the pruned model is similar to the parent network for unforeseen data points? In this section, we investigate the extent to which the pruned and parent model are functionally similar under two distinct metrics: informative features and noise resilience. Our findings show that pruned networks are more functionally similar to their original network than a separately trained, unpruned network underscoring the intuition that pruned networks remain functionally similar to their unpruned counterpart.

4.1 Methodology

Comparison of informative features. We compare features (pixels) that are informative for the decision-making of each model. Specifically, for a network $f_\theta(x)$ with parameters θ and input $x \in \mathbb{R}^n$ we want to find an input mask $m \in \{0,1\}^n$ such that $f_\theta(x) \approx f_\theta(m \odot x)$, i.e.,

$$m = \underset{\|m\|_0 \leq (1-B)n}{\operatorname{argmin}} \|f_\theta(x) - f_\theta(m \odot x)\| \qquad (1)$$

for some sparsity level B. To approximately solve equation 1 we use the greedy backward selection algorithm (`BackSelect`) of Carter et al. (2019; 2020). The procedure iteratively masks the least informative pixel (i.e., the pixel which if masked would reduce the confidence of the prediction of the correct label by the smallest amount) to obtain a sorting of the pixels in order of increasing importance. After sorting, we can remove the bottom $B\%$ of pixels.

Given two networks $f_\theta(\cdot)$ and $f_{\hat{\theta}}(\cdot)$, we can then measure the difference between the functions by switching up the respective input masks m and $\hat{m}$ to see the change in the output, i.e., $\|f_\theta(\hat{m} \odot x) - f_\theta(m \odot x)\|$ and vice versa. If one model can make a confident and correct prediction on the pixels that were informative to another model, the models may have similar decision-making strategies. We apply this strategy to identify subsets of informative pixels across a sample of 2000 CIFAR-10 test images. For each image, we compute the subset of informative pixels, i.e., the input mask m, for an unpruned network, five pruned networks (of increasing prune ratio) derived from that network, and a separate, unpruned network of the same type. We probe whether the informative pixels from one model are also informative to the other models for a sparsity level of 90%.

Noise similarities. We consider injecting l_∞-bounded, random noise into the test data and we compare the predicted

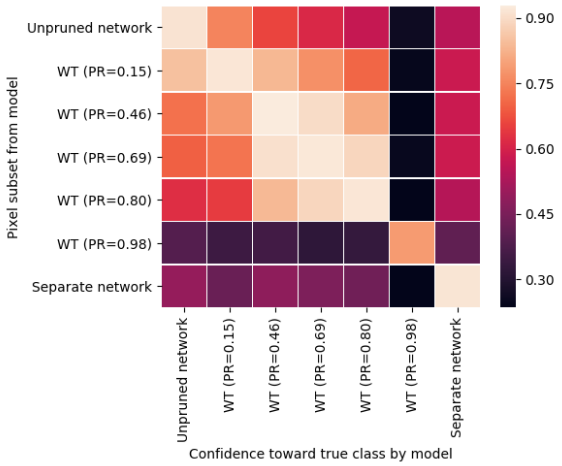
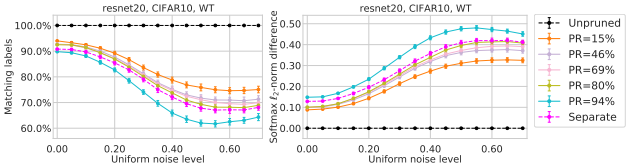


Figure 3: Confidence heatmaps on informative pixels from WT-pruned ResNet20s. Y-axis is the model used to generate the informative pixel subset toward the predicted class, x-axis describes the models evaluated with the subset, cells indicate mean confidence toward the true class.



(a) WT, Labels (b) WT, Norm

Figure 4: The functional similarities between WT-pruned ResNet20 models measured by considering (a) the percentage of matching predictions, (b) the difference in the softmax output after injecting noise into the input. Pruned networks are more similar.

labels between the pruned networks and their unpruned counterpart to investigate the behavior in local neighborhood of points. Specifically, for two networks $f_\theta(\cdot)$ and $f_{\hat{\theta}}(\cdot)$ with parameters θ and $\hat{\theta}$, respectively, we consider the expected number of matching label prediction and the expected norm difference of the output with noise ε, i.e.,

$$\mathbb{E}_{x' \sim \mathcal{D} + \mathcal{U}^n(-\varepsilon,\varepsilon)} \left[\operatorname{argmax} f_\theta(x') = \operatorname{argmax} f_{\hat{\theta}}(x') \right]$$

and

$$\mathbb{E}_{x' \sim \mathcal{D} + \mathcal{U}^n(-\varepsilon,\varepsilon)} \left\| f_\theta(x') - f_{\hat{\theta}}(x') \right\|_2,$$

respectively. We test the noise similarity of networks for a random subset of 1000 test images for 100 repetitions of random noise injection and average over the results.

4.2 Results

Comparison of informative features. Figure 3 shows heatmaps of mean confidence on masked images containing only the 10% most informative features as ordered by

`BackSelect` from an unpruned network (ResNet20 on CIFAR-10), five WT-pruned networks, and a separate, unpruned network. For each masked image containing only the informative features (found with respect to the predicted class), we evaluate the confidence toward the true class for all models to reveal whether such features are informative to the other models. We find features informative to the unpruned network suffice for confident predictions by the pruned networks derived from it, but do not suffice for prediction by the separate, unpruned network. We also find that informative features from pruned networks can be used for prediction by the original network, suggesting these models employ a similar decision-making process. In general, our results suggest weight-pruned networks maintain higher confidence on parent features than do filter-pruned networks (see Appendix C.1). For models pruned beyond commensurate accuracy (PR = 0.98 in Figure 3), the informative features are no longer predictive under any other model.

Noise similarities. In Figure 4a the percentage of matching label predictions of WT-pruned ResNet20 networks are shown with respect to their unpruned counterpart for multiple noise levels. We can conclude that the predictions of the pruned networks tend to correlate with the predictions of the unpruned parent – clearly more than the predictions from a separately trained, unpruned network. We also consider the overall difference in the ℓ_2-norm between pruned and unpruned networks of the softmax output where we observe similar trends, see Figure 4b. These results indicate that the decision boundaries of the pruned network tend to remain close to those of the unpruned network implying that during retraining properties of the original function are maintained. We note that the correlation decreases as we prune more corroborating our intuitive understanding of pruning.

5 Pruning under Distribution Changes

We highlighted that pruned networks behave functionally similarly, however, ultimately the performance is measured in terms of the loss or accuracy on previously unseen data. In this section, we investigate how pruned networks behave in the presence of shifts in the data distribution, including noise, weather, and other corruptions. While it is commonly known that out-of-distribution (o.o.d.) data can harm the performance of neural networks (Madry et al., 2018), we specifically investigate whether pruned network suffer *disproportionately more* from o.o.d. data compared to their parent network. Answering this question affirmatively has profound implications on the practical deployment of pruned networks, specifically for safety-critical systems.

To this end, we define a network's *prune potential* to be the maximal prune ratio for which the pruned network achieves similar loss (up to margin δ) compared to the unpruned one for data sampled from distribution $\mathcal{D}$.

Definition 1 (Prune Potential)**.** *Given a neural network $f_\theta(x)$ with parameters θ, input-label pair $(x, y) \sim \mathcal{D}$, and loss function $\ell(\cdot, \cdot)$ the* prune potential $P(\theta, \mathcal{D})$ *for some margin δ is given by*

$$P(\theta, \mathcal{D}) = \max_{c \in \{0,1\}^{|\theta|}} 1 - {\|c\|_0}/{\|\theta\|_0} \quad \textit{subject to} \quad \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\ell(y, f_{c\odot\hat{\theta}}(x)) - \ell(y, f_\theta(x))\right] \leq \delta, \tag{2}$$

where $\|\cdot\|_0$ denotes the number of nonzero elements, and c and $\hat{\theta}$ denote the prune mask and parameters, respectively, obtained from PruneRetrain *(Algorithm 1).*

The prune potential $\mathcal{P}(\theta, \mathcal{D})$ thus indicates how much of the network can be safely pruned with minimal additional loss incurred. In other words, it indicates to what degree the pruned network can *maintain* the performance of the parent network. As an additional benefit the prune potential may act as a robust measure to gauge the *overparameterization* of a network in the presence of distribution shifts.

Moreover, we define a network's *excess loss* to be the additional loss incurred under distributional changes of the input.

Definition 2 (Excess Loss)**.** *Given a neural network $f_\theta(\cdot)$ with parameters θ, training distribution $\mathcal{D}$ from which we can sample input-label pairs $(x, y) \sim \mathcal{D}$, test distribution $\mathcal{D}'$ from which we can also sample input-label pairs, and loss function $\ell(\cdot, \cdot)$, the* excess loss $e(\theta, \mathcal{D}')$ *is given by*

$$e(\theta, \mathcal{D}') = \mathop{\mathbb{E}}_{(x',y')\sim\mathcal{D}'} \ell(y', f_\theta(x')) - \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \ell(y, f_\theta(x)).$$

The excess loss hereby indicates the expected performance drop of the network for distribution changes, which we can evaluate for various unpruned and pruned parameter sets for a given network architecture to understand to what extend the excess loss varies.

5.1 Methodology

We choose test error (indicator loss function) to evaluate the prune potential and excess loss (excess error). We evaluate the constraint of (2) for a margin of $\delta = 0.5\%$.

We compare the prune potentials $p = P(\theta, \mathcal{D})$ and $p' = P(\theta, \mathcal{D}')$ for two distributions $\mathcal{D}$ and $\mathcal{D}'$ to assess whether pruning up to the prune potential p implies that we can also safely prune up to p for $\mathcal{D}'$. Specifically, the difference $p - p'$ in prune potential can indicate how much the prune potential varies and thus whether it is safe to prune the network up to its full potential p when the input is instead drawn from $\mathcal{D}'$. We note that in practice we may only have access to $\mathcal{D}$ but not $\mathcal{D}'$. Thus in order to safely prune a network up to some

prune ratio p it is crucial to understand to what degree the prune potential may vary for shifts in the distribution.

We also compare the excess error $e = e(\theta, \mathcal{D}')$ and $\hat{e} = e(c \odot \hat{\theta}, \mathcal{D}')$ for an unpruned and pruned network with parameters θ and $c \odot \hat{\theta}$, respectively. Note that the difference in excess error, $\hat{e} - e$, quantifies the additional error incurred by a pruned network under distribution changes compared to the additional error incurred by an unpruned network. Ideally, the difference $\hat{e} - e$ should be zero across all prune ratios, which would imply that the prune-accuracy trade-off for nominal data is indicative of the trade-off for o.o.d. data.

We evaluate the prune potential and excess error using nominal test data (train distribution $\mathcal{D}$) and o.o.d. test data (test distribution $\mathcal{D}'$). Specifically, we consider o.o.d. data with random noise following Section 4.1, and o.o.d. data corrupted using state-of-the-art corruption techniques, i.e., CIFAR10.1 (Recht et al., 2018), CIFAR10-C (Hendrycks & Dietterich, 2019) for CIFAR10; ObjectNet (Barbu et al., 2019), ImageNet-C (Hendrycks & Dietterich, 2019) for ImageNet; and VOC-C (Michaelis et al., 2019) for VOC. For CIFAR10-C, ImageNet-C, VOC-C we choose severity level 3 out of 5. The prune potential is evaluated separately for each corruption while the excess error is evaluated by averaging over all corruptions (test distribution).

5.2 Results

Noise. We evaluated the prune potential of a ResNet20 (CIFAR10) for various noise levels, the results of which are shown in Figure 1. Initially, the network exhibits high prune potential, similar to the prune potential on the original test data (noise level 0.0). However, as we increase the noise injected into the image the prune potential rapidly drops to 0%. As shown in Appendix D.1 most networks' prune potential based on noise exhibit similar properties. This is particularly discomforting as the noise does not deteriorate a human's ability to classify the images correctly as can be seen from Figure 5. These results highlight we may not be able to significantly prune networks if maintaining performance on slightly harder data is the goal.

Prune-accuracy curves for corruptions. Separately, we investigated the prune potential for image corruptions based on the CIFAR10-C, ImageNet-C. In Figures 6a and 6d we show the test accuracy of pruned networks across various target prune ratios for a subset of CIFAR10-C corruptions for a ResNet20 pruned with WT and FT, respectively. In particular, for some simpler corruptions, such as Jpeg, the prune-accuracy curves closely resembles the original CIFAR10 curve, while for metrics such as Speckle and Gauss the curve indicates a noticeable accuracy drop across all target prune ratios. Moreover, the prune-accuracy curve becomes more unpredictable and less stable as indicated by the significantly higher variance of the resulting accuracy.

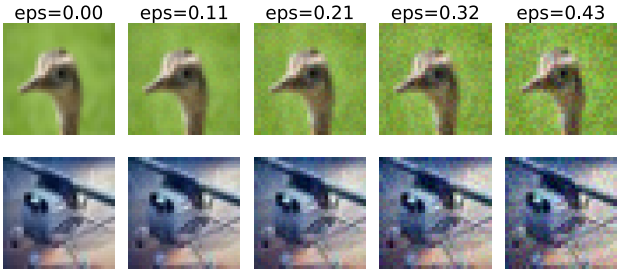


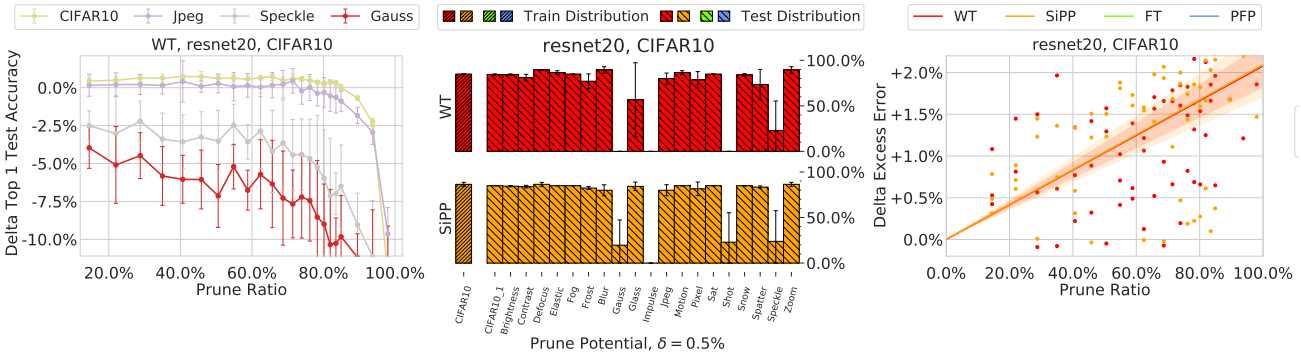
Figure 5: Example images from the CIFAR10 test dataset that were used in this study with various levels of noise injected. A human test subject can classify the images equally well despite the noise present.

We thus conclude that the achievable accuracy of the network depends on the pruning method and the target prune ratio, however, we observe an equally strong dependence on the chosen test metric. Consequently, this affects the prune potential of the network highlighting the sensible trade-off between generalization performance and prune potential.

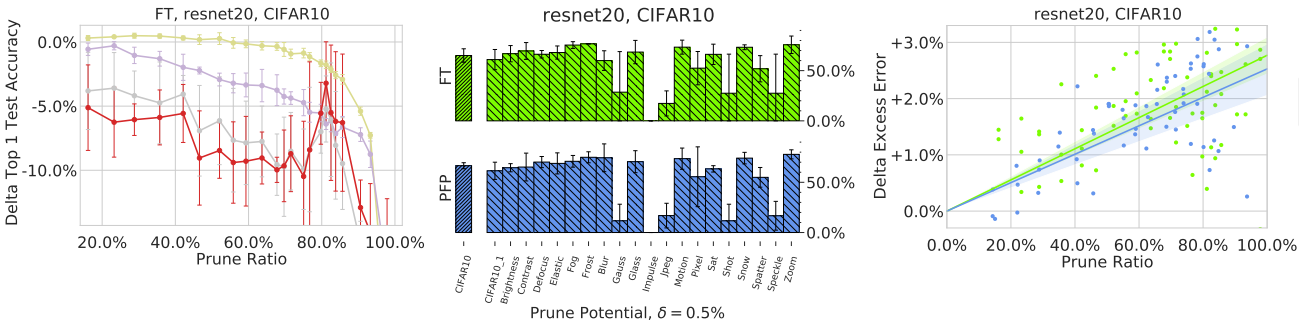
Prune potential for corruptions. For each corruption we then extracted the resulting prune potential from the prune-accuracy curves, see Figures 6b and 6e for weight pruning and filter pruning, respectively. In particular, for corruptions, such as Gauss, Impulse, or Shot, we observe that the network's prune potential hits (almost) 0% implying that any form of pruning may adversely affect the network's performance under such circumstances. We repeated the same experiment for a ResNet18 trained on ImageNet and tested on ImageNet-C, see Figure 7. Noticeably, the network exhibits significantly higher variance in the prune potential across different corruptions compared to the networks tested on CIFAR10. This effect is also more pronounced for filter pruning methods. In Appendix D.2, we provide results for additional networks for both CIFAR10 and ImageNet corroborating our findings presented here.

Choice of δ. We additionally investigate how the prune potential is affected by our choice of δ (see Appendix D.4). While the actual value of the prune potential is naturally affected by δ, we find that our observations of the resulting trends remain unaffected by our particular choice of δ. Hence, we simply choose $\delta = 0.5\%$ uniformly across all experiments reflecting the requirement that our pruned network should be close in accuracy to the parent network while allowing some slack to increase the prune potential.

Excess error. In contrast to the prune potential, the difference in excess error enables us to quantify across multiple prune ratios how much *additional error* is incurred by the pruned network on top of the unpruned network's excess error when tested on o.o.d. data. A non-zero difference thus indicates how the prune-accuracy curve changes under distribution changes. In other words, the difference in excess error quantifies the difference in error between the pruned



(a) Prune-test curve (WT) (b) Weight prune potential (c) Excess error (Weight)



(d) Prune-test curve (FT) (e) Filter prune potential (f) Excess error (Filter)

Figure 6: The prune potential for a ResNet20 on CIFAR10-C test datasets. We observe that depending on the type of corruption the network has significantly less prune potential than when measured w.r.t. the nominal CIFAR10 test accuracy.

and unpruned network *on top* of the difference in error that is incurred for nominal test data (train distribution).

We evaluated the difference in excess error between pruned and unpruned ResNet20s trained on CIFAR10 for various prune ratios (see Figures 6c, 6f). Note that by definition the excess error is 0% for a prune ratio of 0%. We observe that the difference in excess error can reach upwards of 2% and 3% for weight and filter pruning, respectively. Moreover, the higher the prune ratio the more variance we can observe indicating that the pruned network's behavior becomes less predictable overall. These observations strongly indicate that pruned networks suffer disproportionally more from o.o.d. data across a wide spectrum of prune ratios and that the additional performance drop on o.o.d. data is positively correlated with the prune ratio. In other words, while current pruning techniques achieve commensurate accuracy for high prune ratios on nominal test data, the same pruning techniques do not maintain commensurate accuracy for even small prune ratios on o.o.d. test data. Additional results are presented in Appendix D.5.

Measuring overparameterization. The results presented so far in this section highlight that the prune potential on nominal test data does not reliably indicate the overall performance of the pruned network. This may lead to novel insights into understanding the amount of overparameterization in deep networks. In Appendix D.6, we summarize the prune potential across all tested data distributions and networks as a way to gauge the amount of overparameterization of a network. A subset of the results are shown in Table 2. While some networks' prune potentials are significantly affected by changes in the distribution, other networks' prune potentials are virtually unaffected. Take for example the weight prune potential on nominal test data (training distribution) of a VGG16 and a WRN16-8, which is around 98% for both. However, when both networks are evaluated on o.o.d. test data (test distribution) they exhibit distinctly different behaviors. While the WRN16-8's prune potential remains fairly stable at around 95% (3% drop), the VGG16's prune potential falls to 80% (18% drop).

Overall, these results illustrate the fact that the prune potential may act as a robust measure of a network's genuine overparameterization in theory, and may also be helpful in informing the practitioner on the extent of pruning that should be conducted prior to deployment in practice.

6 Towards Robust Pruning

Our experiments raise the question whether the decreased performance of pruned networks is a limitation of our current pruning and training techniques or whether it is an

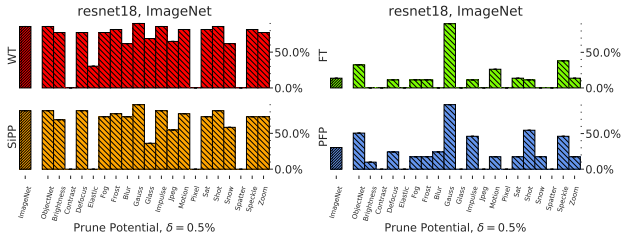


Figure 7: Prune potential of a ResNet18 (ImageNet).

Model	Method	Prune Potential (%)		
		Train Dist.	Test Dist.	Diff.
ResNet20	WT	84.9 ± 0.0	$\mathbf{66.7 \pm 3.3}$	-18.2
	FT	65.0 ± 6.7	$\mathbf{55.3 \pm 4.8}$	-9.7
VGG16	WT	98.0 ± 0.0	$\mathbf{80.9 \pm 2.2}$	-17.1
	FT	85.4 ± 2.4	$\mathbf{66.3 \pm 0.5}$	-19.1
WRN16-8	WT	98.0 ± 0.0	$\mathbf{95.7 \pm 0.7}$	-2.3
	FT	86.2 ± 1.3	$\mathbf{75.7 \pm 4.1}$	-10.5
ResNet18	WT	85.8	**63.6**	-22.2
	FT	13.7	**13.5**	-0.2

Table 2: The prune potential of various networks trained on CIFAR10 (upper part) and ImageNet (lower part) evaluated on the train and test distribution, which consist of nominal data and the average over all corruptions, respectively.

inherent limitation of pruned, i.e., smaller, networks themselves. To this end, we investigated whether training (and retraining) in a robust manner can benefit the pruned network and minimize the effects we observed previously.

6.1 Methodology

To test the hypothesis that we can regain some of the robustness properties of the unpruned network we repeated the experiments of Section 5 but during (re-)training we incorporated a randomly chosen fixed subset of nine corruptions from CIFAR10-C and ImageNet-C into the data augmentation pipeline. That is, every time we sample an image from the train set during (re-)training we choose an image corruption (or no corruption) uniformly at random to corrupt the image effectively altering the train distribution that the network sees. The remaining corruptions are not used during training and make up the new test distribution. Additional experimental details are provided in Appendix E.

6.2 Results

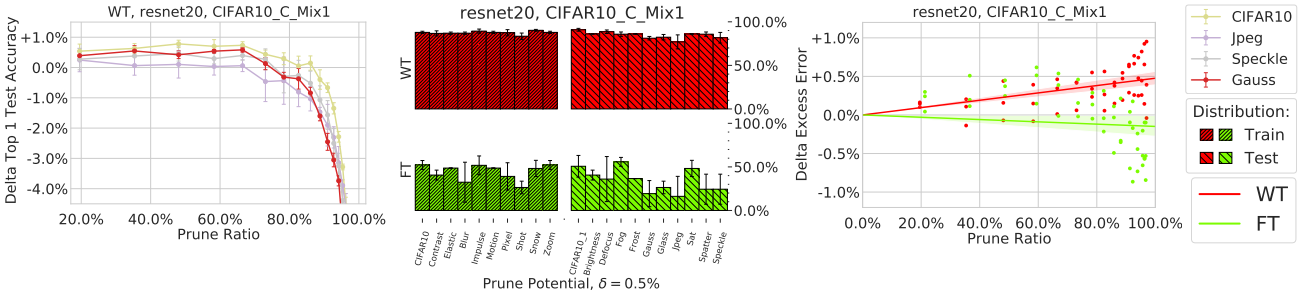
Prune-accuracy curves and prune potential. In Figure 8a we show the prune-accuracy curves for three corruptions from the test distribution for WT-pruned ResNet20s. Compared to the results in Figure 6a where we did not perform robust (re-)training we observe that the prune-accuracy curves are much more stable and that the prune-accuracy curve on nominal test data (CIFAR10) is more predictive of the others. However, the results on the evaluation of the prune potential as shown in Figure 8b for weight and filter pruning reveal that even in this setting the prune potential can be significantly lower (or exhibit high variation over multiple trials) for some of the corruptions from the test distribution. These observations further corroborate our findings but also highlight the beneficial effects of robust training in efficiently alleviating some of the short-comings (see Appendix E for a complete exposition of the results).

Excess error. Similar trends can also be observed when considering the difference in excess error as shown in Figure 8c. While we can reduce the correlation between prune ratio and excess error, we can not entirely eliminate it. Moreover, we can still observe high variations in the excess error confirming the sensible trade-off between generalization performance and prune potential.

Implicit regularization. In this section, we show that we can regain much of the prune potential even under distribution changes, at least when we can incorporate these additional data points into our training pipeline. For example, the weight prune potential for a ResNet20 for both the nominal and robust training scenario is around 85%. Consequently, we argue that both pruned and unpruned networks have sufficient capacity to represent the underlying distribution given that the training is performed using an appropriate optimization pipeline.

Specifically, previous work noted that overparameterized networks may benefit from implicit regularization when optimized with a stochastic optimizer and that more parameters amplify this effect. We can confirm these observations in the sense that pruned networks suffer disproportionally more from o.o.d. data than unpruned networks with more parameters. That is, unpruned network exhibit more implicit regularization through SGD leading to more robustness. However, we can regain some of the robustness properties by adding *explicit* regularization during training in the form of data augmentation. We can thus "trade" the implicit regularization potential which we lose by removing parameters for explicit regularization through data augmentation.

Choice of test distribution. While our results suggest that robust training indeed improves the generalization of pruned networks for o.o.d. data, we would like to emphasize that our conclusion intrinsically hinges upon on the choice of train and test distribution. While we did strictly separate the corruptions used during train and test time, these corruptions can be loosely categorized into four types, i.e. noise, blur, weather, digital, all of which are present in both the train and test distribution. Therefore, we suspect that for significantly different corruption models (or adversarial inputs) we may observe more significant trade-offs resembling the results of Section 5 where we performed nominal (re-)training. This is a consequence of requiring additional explicit regularization since the explicit regularization must be modeled.



(a) WT, prune-test curve (b) Prune potential (WT, FT) (c) Excess error (WT, FT)

Figure 8: The prune potential of a ResNet20 shown for corruptions that were included (train distribution) and excluded (test distribution) during training. The prune-accuracy curves in (a) are shown for corruptions from the test distribution.

7 Discussion

Weight vs filter pruning. We find that across all tested corruptions filter pruning methods are more error-prone and have lower prune potential compared to weight pruning. We conjecture this trend stems from the fact that structured pruning is overall a harder problem, implying that structurally pruned networks are less capable of maintaining the properties of the parent network when compared to those generated by weight-based pruning.

Genuine overparameterization. In light of our results we conjecture that while the high capacity of modern networks may not be strictly necessary to achieve high test accuracy – since pruned networks with commensurate accuracy exist –, the "excess" capacity of these networks may be beneficial to maintaining other crucial properties of the network, such as its ability to perform well on unforeseen or out-of-distribution data. This challenges the common wisdom that modern networks are overparameterized and, hence, contain redundant parameters that can be pruned in a straight-forward manner without "loss of performance." Unlike prior work that has predominantly pointed to the test accuracy as a gauge for overparameterization (and thus the ability to prune a network), we hypothesize that a more robust and accurate measure of *genuine* overparameterization is one that not only considers test accuracy, but also the minimum (or average) prune potential over a variety of tasks. Studying the prune potential is thus not only useful to study the ability to safely prune a network but also has the positive side-effect of establishing a robust measure of network overparameterization.

Implicit regularization. Our studies reveal that the amount of overparameterization is not only a function of the task and the network size but also a function of the *training procedure*. Specifically, we can prune the network more if we explicitly regularize the network during (re-)training thus increasing the "genuine" overparameterization of the network which implies a higher prune potential. However, with fewer parameters (due to pruning more) we trade in some of the implicit regularization potential from SGD. Since implicit regularization is not necessarily model-based we can only regain the robustness of the pruned network for known, i.e. modeled, distribution changes.

Generalization-aware pruning. Based on our results we formulate a set of guidelines for pruning in practice as shown in Section 1. We argue that in order to reliably and robustly deploy pruned networks especially in the context of safety-critical systems we should not only designate a *hold-out data set* (test set) but also a *hold-out data distribution* (test distribution). By assessing the performance of the pruned network on data from the train and test distribution, we can then quantify the effect of pruning in a way that can unearth some of the short-comings that are *lost in pruning* and are not apparent from a plain prune-accuracy curve on nominal test data. Following our framework, a practitioner will be able to more reliably assess whether the pruned network can be considered as performant (in a robust sense) as the unpruned network.

8 Conclusion

In this work, we have investigated the effects of the pruning process on the functional properties of the pruned network relative to its uncompressed counterpart. Our empirical results suggest that pruned models are functionally similar to the their uncompressed counterparts but that, despite this similarity, the prune potential of the network varies significantly on a task-dependent basis: the prune potential decreases significantly as the difficulty of the inference task increases. Our findings underscore the need to consider task-specific evaluation metrics beyond test accuracy prior to deploying a pruned network and provide novel insights into understanding the amount of network overparameterization in deep learning. We envision that our framework may invigorate further work towards rigorously understanding the inherent model size-performance trade-off and help practitioners in adequately designing and pruning network architectures in a task-specific manner.

ACKNOWLEDGEMENTS

This research was supported in part by the U.S. National Science Foundation (NSF) under Award 1723943, Office of Naval Research (ONR) Grant N00014-18-1-2830, and JP Morgan Chase. We thank them for their support. We would like to further thank Siddhartha Jain for the insightful discussions during the project conception phase and for helping with early drafts of the paper.

REFERENCES

Allen-Zhu, Z., Li, Y., and Liang, Y. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in Neural Information Processing Systems 32*, pp. 6158–6169. Curran Associates, Inc., 2019.

Arora, S., Ge, R., Neyshabur, B., and Zhang, Y. Stronger generalization bounds for deep nets via a compression approach. In *International Conference on Machine Learning*, pp. 254–263, 2018.

Barbu, A., Mayo, D., Alverio, J., Luo, W., Wang, C., Gutfreund, D., Tenenbaum, J., and Katz, B. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *Advances in Neural Information Processing Systems*, pp. 9448–9458, 2019.

Baykal, C., Liebenwein, L., Gilitschenski, I., Feldman, D., and Rus, D. Data-dependent coresets for compressing neural networks with applications to generalization bounds. In *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=HJfwJ2A5KX.

Baykal, C., Liebenwein, L., Gilitschenski, I., Feldman, D., and Rus, D. Sipping neural networks: Sensitivity-informed provable pruning of neural networks. *arXiv preprint arXiv:1910.05422*, 2019b.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Blalock, D., Gonzalez Ortiz, J. J., Frankle, J., and Guttag, J. What is the state of neural network pruning? In *Proceedings of Machine Learning and Systems 2020*, pp. 129–146. 2020.

Carter, B., Mueller, J., Jain, S., and Gifford, D. What made you do this? understanding black-box decisions with sufficient input subsets. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 567–576, 2019.

Carter, B., Jain, S., Mueller, J., and Gifford, D. Overinterpretation reveals image classification model pathologies. *arXiv preprint arXiv:2003.08907*, 2020.

Chen, L.-C., Papandreou, G., Schroff, F., and Adam, H. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

Dhillon, G. S., Azizzadenesheli, K., Bernstein, J. D., Kossaifi, J., Khanna, A., Lipton, Z. C., and Anandkumar, A. Stochastic activation pruning for robust adversarial defense. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1uR4GZRZ.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1eK3i09YQ.

Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

Gale, T., Elsen, E., and Hooker, S. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

Gamboa, N., Kudrolli, K., Dhoot, A., and Pedram, A. Campfire: Compressible, regularization-free, structured sparse training for hardware accelerators. *arXiv preprint arXiv:2001.03253*, 2020.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Gui, S., Wang, H. N., Yang, H., Yu, C., Wang, Z., and Liu, J. Model compression with adversarial robustness: A unified optimization framework. In *Advances in Neural Information Processing Systems*, pp. 1285–1296, 2019.

Guo, Y., Zhang, C., Zhang, C., and Chen, Y. Sparse dnns with improved adversarial robustness. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 242–251. Curran Associates, Inc., 2018.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2015. URL http://arxiv.org/abs/1510.00149.

Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S., and Malik, J. Semantic contours from inverse detectors. In *2011 International Conference on Computer Vision*, pp. 991–998. IEEE, 2011.

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.

He, Y., Kang, G., Dong, X., Fu, Y., and Yang, Y. Soft filter pruning for accelerating deep convolutional neural networks. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 2234–2240. AAAI Press, 2018.

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

Hooker, S., Courville, A., Dauphin, Y., and Frome, A. Selective brain damage: Measuring the disparate impact of model pruning. *arXiv preprint arXiv:1911.05248*, 2019.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Kusupati, A., Ramanujan, V., Somani, R., Wortsman, M., Jain, P., Kakade, S., and Farhadi, A. Soft threshold weight reparameterization for learnable sparsity. *arXiv preprint arXiv:2002.03231*, 2020.

Lee, N., Ajanthan, T., and Torr, P. SNIP: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1VZqjAcYX.

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

Li, Y., Gu, S., Gool, L. V., and Timofte, R. Learning filter basis for convolutional neural network compression. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5623–5632, 2019.

Liebenwein, L., Baykal, C., Lang, H., Feldman, D., and Rus, D. Provable filter pruning for efficient neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJxkOlSYDH.

Liu, Z., Sun, M., Zhou, T., Huang, G., and Darrell, T. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJlnB3C5Ym.

Luo, J.-H. and Wu, J. Autopruner: An end-to-end trainable filter pruning method for efficient deep model inference. *arXiv preprint arXiv:1805.08941*, 2018.

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*, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

Michaelis, C., Mitzkus, B., Geirhos, R., Rusak, E., Bringmann, O., Ecker, A. S., Bethge, M., and Brendel, W. Benchmarking robustness in object detection: Autonomous driving when winter is coming. *arXiv preprint arXiv:1907.07484*, 2019.

Nagarajan, V. and Kolter, Z. Deterministic PAC-bayesian generalization bounds for deep networks via generalizing noise-resilience. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Hygn2o0qKX.

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. Deep double descent: Where bigger models and more data hurt. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1g5sA4twr.

Neyshabur, B., Tomioka, R., and Srebro, N. In search of the real inductive bias: On the role of implicit regularization in deep learning. In *ICLR (Workshop)*, 2015. URL http://arxiv.org/abs/1412.6614.

Neyshabur, B., Li, Z., Bhojanapalli, S., LeCun, Y., and Srebro, N. Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*, 2018.

Neyshabur, B., Li, Z., Bhojanapalli, S., LeCun, Y., and Srebro, N. The role of over-parametrization in generalization of neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BygfghAcYX.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

Recht, B., Roelofs, R., Schmidt, L., and Shankar, V. Do cifar-10 classifiers generalize to cifar-10? *arXiv preprint arXiv:1806.00451*, 2018.

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

Renda, A., Frankle, J., and Carbin, M. Comparing fine-tuning and rewinding in neural network pruning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1gSj0NKvB.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Schwarting, W., Seyde, T., Gilitschenski, I., Liebenwein, L., Sander, R., Karaman, S., and Rus, D. Deep latent competition: Learning to race using visual control policies in latent space. In *Conference on Robot Learning*, Proceedings of Machine Learning Research. PMLR, 2020.

Sehwag, V., Wang, S., Mittal, P., and Jana, S. Towards compact and robust deep neural networks. *arXiv preprint arXiv:1906.06110*, 2019.

Sehwag, V., Wang, S., Mittal, P., and Jana, S. Hydra: Pruning adversarially robust neural networks. *arXiv preprint arXiv:2002.10509*, 2020.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

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

Singh, S. P. and Alistarh, D. Woodfisher: Efficient second-order approximations for model compression. *arXiv preprint arXiv:2004.14340*, 2020.

Tanaka, H., Kunin, D., Yamins, D. L., and Ganguli, S. Pruning neural networks without any data by iteratively conserving synaptic flow. *arXiv preprint arXiv:2006.05467*, 2020.

Torralba, A., Fergus, R., and Freeman, W. T. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970, 2008.

Wang, C., Zhang, G., and Grosse, R. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkgsACVKPH.

Wang, L., Ding, G. W., Huang, R., Cao, Y., and Lui, Y. C. Adversarial robustness of pruned neural networks. *ICLR Workshop submission*, 2018.

Wijayanto, A. W., Choong, J. J., Madhawa, K., and Murata, T. Towards robust compressed convolutional neural networks. In *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 1–8. IEEE, 2019.

Ye, S., Xu, K., Liu, S., Cheng, H., Lambrechts, J.-H., Zhang, H., Zhou, A., Ma, K., Wang, Y., and Lin, X. Adversarial robustness vs. model compression, or both. In *The IEEE International Conference on Computer Vision (ICCV)*, volume 2, 2019.

You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., and Hsieh, C.-J. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 1(5), 2019.

Yu, J., Yang, L., Xu, N., Yang, J., and Huang, T. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2016. URL https://openreview.net/forum?id=Sy8gdB9xx&.

Zhao, Y., Shumailov, I., Mullins, R., and Anderson, R. To compress or not to compress: Understanding the interactions between adversarial attacks and neural network compression. *arXiv preprint arXiv:1810.00208*, 2018.

Zhou, W., Veitch, V., Austern, M., Adams, R. P., and Orbanz, P. Compressibility and generalization in large-scale deep learning. *arXiv preprint arXiv:1804.05862*, 2018.

Zhu, M. and Gupta, S. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

BROADER IMPACT STATEMENT

In this paper, we study the impact of pruning when deploying neural networks in real-world conditions. This is of particular importance since the main motivation to prune neural networks lies within training neural networks that are simultaneously efficient and accurate. Henceforth, these networks can then be deployed in resource-constrained environments, such as robotics, to achieve tasks that could otherwise only be computed on large-scale compute infrastructure.

However, we show that pruned neural networks do not necessarily perform on par with their unpruned parent but rather that they exhibit significant performance decreases depending on slight variations within their assigned task. This raises concerns with regards to the ability to find effectively pruned architectures with current network pruning techniques. With the increasing amount of applications for deep learning including on small-scale devices, we have to vigilantly monitor and consider the effects, brittleness, and potential biasedness of small-scale neural networks at an even higher degree than for regular deep networks.

A OVERVIEW OF THE APPENDIX

In the main part of the paper, we have focused on representative subsets of each experiment. In the following, we provide additional experimental details and present the complete set of conducted experiments. Specifically, the supplementary material contains the following sections:

- *Section B*: additional experimental details and hyperparameters for how we prune networks
- *Section C*: experiments pertaining to measuring the functional similarities between pruned and unpruned networks
- *Section D*: experiments pertaining to the prune potential and excess error of pruned networks
- *Section E*: experiments pertaining to the prune potential and excess error of pruned networks with robust (re-)training

In addition to the results presented in the main part of the paper, we present further experimental evidence for our claims across a variety of architectures and data sets.

We provide the code to reproduce our experiments at `https://github.com/lucaslie/torchprune`.

B PRUNING RESULTS

Our experimental evaluations are based on a variety of neural network architectures including ResNets (He et al., 2016), VGGs (Simonyan & Zisserman, 2014), DenseNets (Huang et al., 2017), and WideResNets (Zagoruyko & Komodakis, 2016) trained on CIFAR10 (Torralba et al., 2008) and ImageNet (Russakovsky et al., 2015). We also conduct experiments on a DeeplabV3 (Chen et al., 2017) with a ResNet50 backbone trained on the Pascal VOC 2011 segmentation data set (Everingham et al., 2015). In the following section we outline the experimental details of the experiments on which we base our observations. All networks were trained and evaluated on a compute cluster with NVIDIA RTX 2080Ti and NVIDIA Titan RTX, and the experiments were implemented in PyTorch (Paszke et al., 2017). For each trained network, we summarize the hyperparameters and the resulting prune results on the nominal test data.

B.1 Experimental Setup for CIFAR10

All hyperparameters for training, retraining, and pruning are outlined in Table 3. For training CIFAR10 networks we used the training hyperparameters outlined in the respective original papers, i.e., as described by He et al. (2016), Simonyan & Zisserman (2014), Huang et al. (2017), and Zagoruyko & Komodakis (2016) for ResNets, VGGs, DenseNets, and WideResNets, respectively. For retraining, we did not change the hyperparameters and repurposed the training hyperparameters following the approaches of Liebenwein et al. (2020); Renda et al. (2020). We added a warmup period in the beginning where we linearly scale up the learning rate from 0 to the nominal learning rate. Iterative pruning is conducted by repeatedly removing the same ratio of parameters (denoted by α in Table 3). The prune parameter γ describes the failure probability of the (provable) randomized pruning algorithms SiPP and PFP. We refer the reader to the respective papers for more details, see the papers by Baykal et al. (2019b) and (Liebenwein et al., 2020) for SiPP and PFP, respectively.

B.2 Pruning Performance on CIFAR10

Below we provide the results regarding the achievable test accuracy of pruned networks across multiple target prune ratios. Figure 9 indices the results for various networks trained on CIFAR10 using an iterative schedule to prune them. In Table 4,

		VGG16	Resnet20/56/110	DenseNet22	WRN-16-8
Train	test error	7.19	8.6/7.19/6.43	10.10	4.81
	loss	cross-entropy	cross-entropy	cross-entropy	cross-entropy
	optimizer	SGD	SGD	SGD	SGD
	epochs	300	182	300	200
	warm-up	5	5	5	5
	batch size	256	128	64	128
	LR	0.05	0.1	0.1	0.1
	LR decay	0.5@{30, ...}	0.1@{91, 136}	0.1@{150, 225}	0.2@{60, ...}
	momentum	0.9	0.9	0.9	0.9
	Nesterov	✗	✗	✓	✓
	weight decay	5.0e-4	1.0e-4	1.0e-4	5.0e-4
Prune	γ	1.0e-16	1.0e-16	1.0e-16	1.0e-16
	α	0.85	0.85	0.85	0.85

Table 3: We report the hyperparameters used during training, pruning, and retraining for various convolutional architectures on CIFAR-10. LR hereby denotes the learning rate and LR decay denotes the learning rate decay that we deploy after a certain number of epochs. During retraining we used the same hyperparameters. $\{30, \ldots\}$ denotes that the learning rate is decayed every 30 epochs.

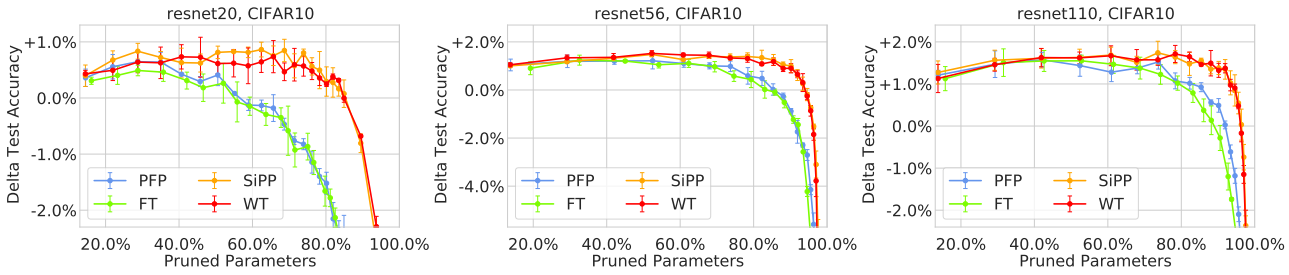
Model	Orig.	WT			SiPP			FT			PFP		
	Err.	Err.	PR	FR	Err.	PR	FR	Err.	PR	FR	Err.	PR	FR
Resnet20	8.60	-0.02	**84.92**	81.04	-0.08	**84.92**	78.78	-0.33	52.06	24.98	-0.10	44.9	**31.27**
Resnet56	7.19	-0.53	92.91	94.31	-0.30	**93.30**	93.90	-0.38	82.71	65.50	-0.11	84.31	**73.65**
Resnet110	6.73	-0.10	**95.76**	96.73	-0.42	95.36	95.88	-0.25	86.71	72.07	-0.25	90.27	**82.42**
VGG16	7.19	-1.01	97.87	91.24	-0.82	**98.00**	88.45	-0.38	61.39	56.17	-0.15	90.30	**72.03**
DenseNet22	10.10	-0.09	71.38	76.81	-0.20	**73.16**	76.60	+0.21	43.55	42.95	-0.04	46.18	**51.86**
WRN16-8	4.81	-0.18	95.22	92.89	-0.21	**95.30**	92.03	+0.13	76.76	71.03	-0.08	78.79	**74.51**

Table 4: Overview of the pruning performance of each algorithm for various CNN architectures evaluated on the CIFAR data set. For each algorithm and network architecture, the table reports the prune ratio (PR, %) and the ratio of flop reduction (FR, %) of pruned models when achieving test accuracy within $\delta = 0.5\%$ of the original network's test accuracy (or the closest result when the desired test accuracy was not achieved for the range of tested PRs). The top values for the error and either PR (for weight-based) or FR (for filter-based algorithms) are bolded, respectively.

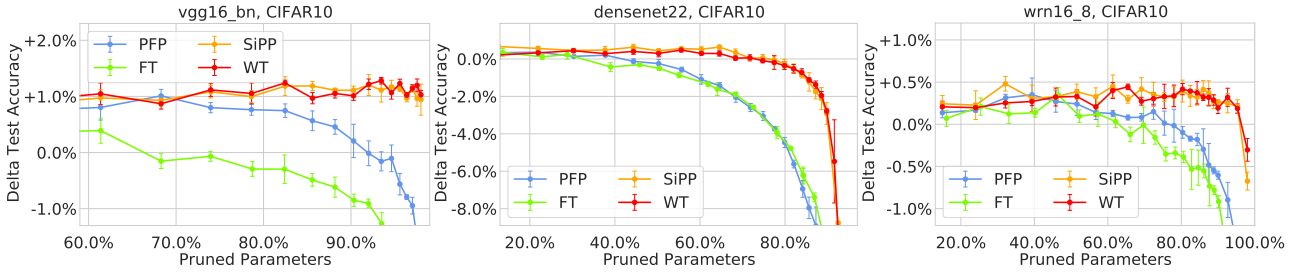
we indicate the maximal prune ratio (PR) and the maximal ratio of reduced flops (FR) for which the network achieves commensurate accuracy (within 0.5% of the original accuracy). We note that the performance of our pruned networks is competitive with state-of-the-art pruning results (Baykal et al., 2019b; Han et al., 2015; Liebenwein et al., 2020; Renda et al., 2020). For ResNet20 for example, we are able to prune the network to 85% sparsity while maintaining the original test error (-0.02% test error), see Table 4.

B.3 Experimental Setup on ImageNet

The hyperparameters for the ImageNet pruning experiments are summarized in Table 5. We consider pruned convolutional neural networks derived from Resnet18 and Resnet101. As in the case of the CIFAR10 experiments, we re-purpose the training schedule from the original ResNet paper (He et al., 2016) for both training and retraining. For multi-gpu training we use the linear scaling rule of (Goyal et al., 2017) to scale up the learning rate and we use learning rate warm, where we linearly scale up the learning rate from 0 to the nominal learning rate.



(a) Resnet20 (b) Resnet56 (c) Resnet110



(d) VGG16 (e) DenseNet22 (f) WRN16-8

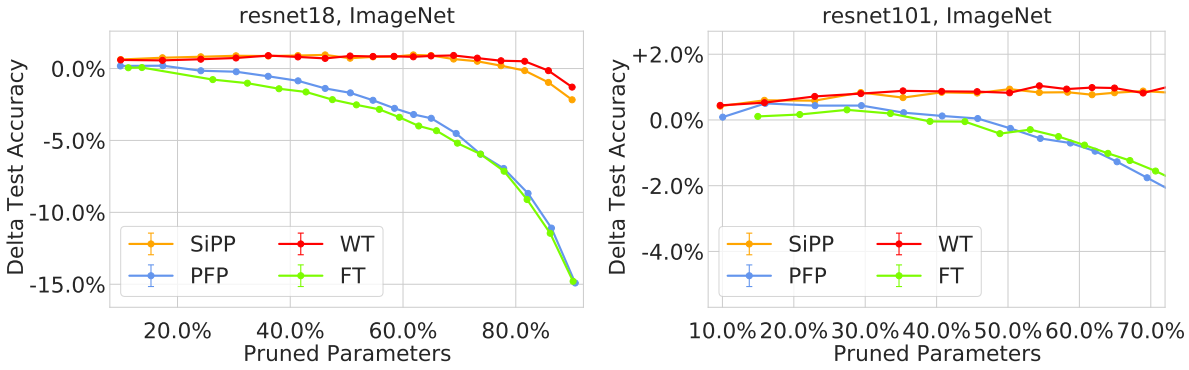
Figure 9: The difference in test accuracy to the uncompressed network for the generated pruned models trained on CIFAR10 for the evaluated pruning schemes for various target prune ratios.

		ResNet18/101
Train	top-1 test error	30.26/22.63
	top-5 test error	10.93/6.45
	loss	cross-entropy
	optimizer	SGD
	epochs	90
	warm-up	5
	batch size	256
	LR	0.1
	LR decay	0.1@{30, 60, 80}
	momentum	0.9
	Nesterov	✗
	weight decay	1.0e-4
Prune	γ	1.0e-16
	α	0.90

Table 5: We report the hyperparameters used during training, pruning, and retraining for various convolutional architectures on ImageNet. LR hereby denotes the learning rate and LR decay denotes the learning rate decay that we deploy after a certain number of epochs.

B.4 Pruning Performance on ImageNet

The results of our pruning experiments are summarized in Figure 10 and Table 6. Given the computationally expensive nature of ImageNet experiments, we stopped the experiments once the pruned network did not achieve commensurate accuracy anymore (instead of going to extreme prune ratios where the performance decays further). Specifically, we show the achievable test accuracy on nominal ImageNet data for various target prune ratios in Figure 10. In Table 6 we additionally report the maximal prune ratio (PR) and ratio of removed flops (FR) for which the pruned network achieves commensurate accuracy (i.e. within 0.5% of the unpruned network's accuracy). Just as in the case of CIFAR10, our results are competitive with those reported in state-of-the-art papers (Han et al., 2015; Liebenwein et al., 2020; Renda et al., 2020).



(a) Resnet18 (b) Resnet101

Figure 10: The accuracy of the generated pruned models trained on ImageNet for the evaluated pruning schemes for various target prune ratios.

Model	Orig.	WT			SiPP			FT			PFP		
	Err.	Err.	PR	FR	Err.	PR	FR	Err.	PR	FR	Err.	PR	FR
ResNet18	30.24	+0.15	**85.79**	77.14	+0.15	81.58	78.36	-0.07	13.69	10.52	+0.22	30.42	**15.74**
ResNet101	22.63	-0.71	**81.56**	83.31	-0.59	**81.56**	81.85	+0.29	53.11	**53.28**	+0.26	50.33	44.64

Table 6: Overview of the pruning performance of each algorithm for various CNN architectures trained and evaluated on the ImageNet data set. For each algorithm and network architecture, the table reports the prune ratio (PR, %) and the ratio of flop reduction (FR, %) of pruned models when achieving test accuracy within $\delta = 0.5\%$ of the original network's test accuracy (or the closest result when the desired test accuracy was not achieved for the range of tested PRs). The top values for the error and either PR (for weight-based) or FR (for filter-based algorithms) are bolded, respectively.

B.5 Experimental Setup for Pascal VOC

In addition to CIFAR and ImageNet, we also consider the segmentation task from Pascal VOC 2011 (Everingham et al., 2015). We augment the nominal data training data using the extra labels as provided by Hariharan et al. (2011). As network architecture we consider a DeeplabV3 (Chen et al., 2017) with ResNet50 backbone pre-trained on ImageNet. During training we use the following data augmentation pipeline: (1) randomly resize (256x256 to 1024x1024) and crop to 513x513; (2) random horizontal flip; (3) channel-wise normalization. During inference, we resize to 513x513 exactly before the normalization (3) is applied. We report both intersection-over-union (IoU) and Top1 test error for each of the pruned and unpruned networks. The experimental hyperparameters are summarized in Table 7.

B.6 Pruning Performance on VOC

The results of our pruning experiments are summarized in Figure 11 and Table 8. Specifically, we show the achievable test accuracy on nominal VOC data for various target prune ratios in Figure 11. In Table 8 we report the maximal prune ratio (PR) and ratio of removed flops (FR) for which the pruned network achieves commensurate accuracy (i.e. within 0.5% of the unpruned network's accuracy).

		DeeplabV3-ResNet50
Train	IoU test error (%)	34.78
	top-1 test error (%)	7.94
	Loss	cross-entropy
	Optimizer	SGD
	Epochs	45
	Warm-up	0
	Batch size	32
	LR	0.02
	LR decay	(1 - "step"/"total steps")$^{0.9}$
	Momentum	0.9
	Nesterov	✗
	Weight decay	1.0e-4
Prune	γ	1.0e-16
	α	0.80

Table 7: We report the hyperparameters used during training, pruning, and retraining for various architectures on Pascal VOC 2011. LR hereby denotes the learning rate and LR decay denotes the learning rate decay. Note that the learning rate is polynomially decayed after each step.

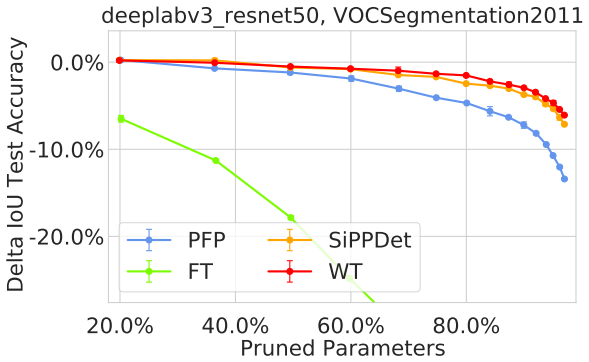


Figure 11: The accuracy of the generated pruned models trained on VOC for the evaluated pruning schemes and various target prune ratios for a DeeplabV3-ResNet50 architecture.

Model	Orig.	WT			SiPP			FT			PFP		
	Err.	Err.	PR	FR	Err.	PR	FR	Err.	PR	FR	Err.	PR	FR
DeeplabV3	34.78	+0.47	**58.87**	58.65	+0.29	42.98	42.70	+0.00	0.00	0.00	-0.25	20.16	**19.14**

Table 8: Overview of the pruning performance of each algorithm for DeeplabV3 trained and evaluated on Pascal VOC segmentation data. For each algorithm, the table reports the prune ratio (PR, %) and the ratio of flop reduction (FR, %) of pruned models when achieving IoU test accuracy within $\delta = 0.5\%$ of the original network's test accuracy (or the closest result when the desired test accuracy was not achieved for the range of tested PRs). The top values for the error and either PR (for weight-based) or FR (for filter-based algorithms) are bolded, respectively.

C Additional Results for Function Distance of Pruned Networks

In the following, we provide additional empirical evidence for the results presented in Section 4 of the main paper. We consider additional CIFAR networks for comparing informative input features and comparing matching predictions when injecting random noise as described in Section 4.

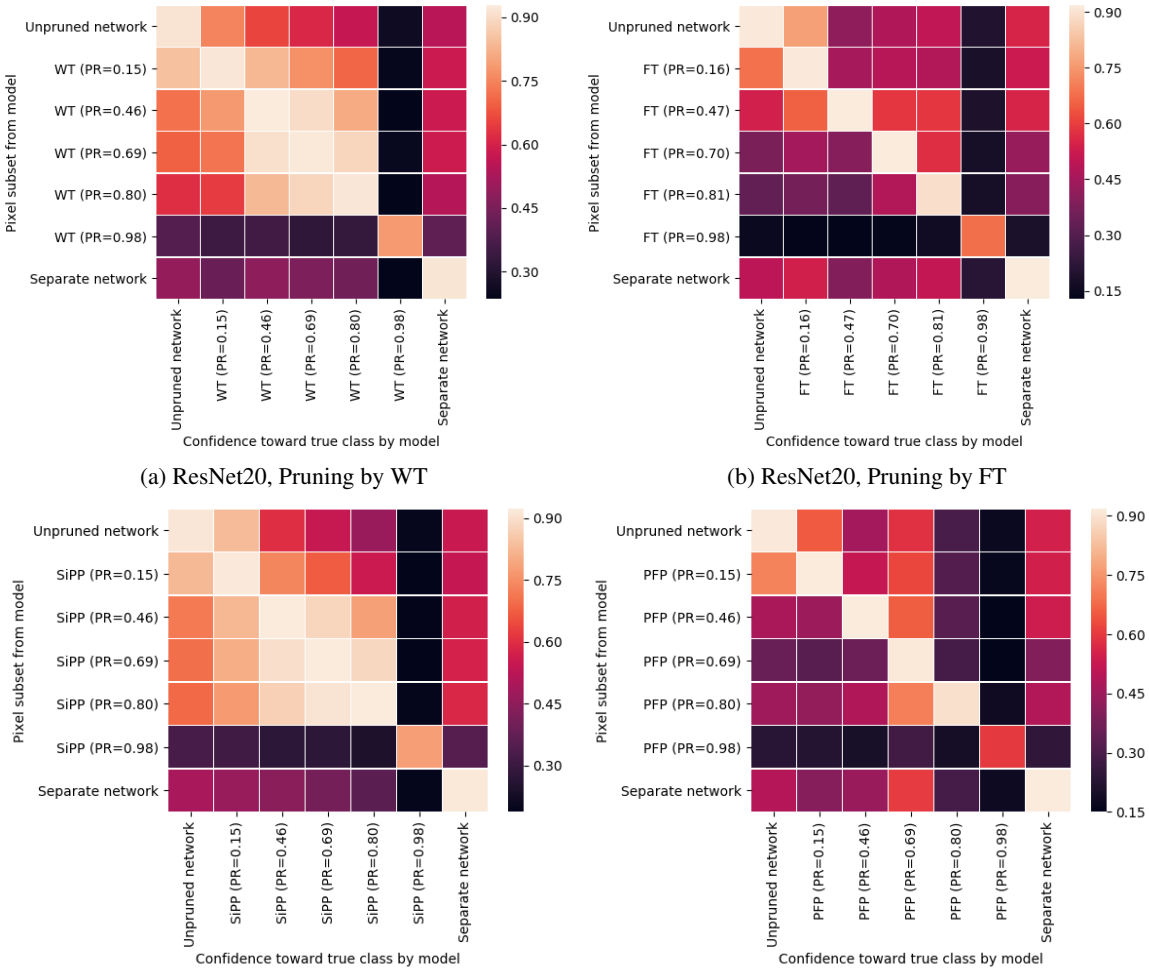
C.1 Comparison of Informative Features

C.1.1 Informative Features Based on Nominal Test Data

Figure 12 includes results on comparison of informative features for models pruned by other pruning algorithms on ResNet20 (see Section 4.2 and Figure 3). Figure 13 shows results for VGG16. The informative features were computed from a random subset of CIFAR10 test data.

C.1.2 Informative Features Based on Out-of-distribution Test Data

We repeat the experiment with the informative features being computed from a random subset of CIFAR10-C test data (any corruption). Figure 14 includes results on comparison of informative features, c.f. Section 4.2, for models pruned by WT and FT on ResNet20. Figure 15 shows results for VGG16. We note that even when tested with out-of-distribution test data we observe similar trends, i.e., pruned networks in general are more similar in the functional sense to their parent network than a separately trained, unpruned network.



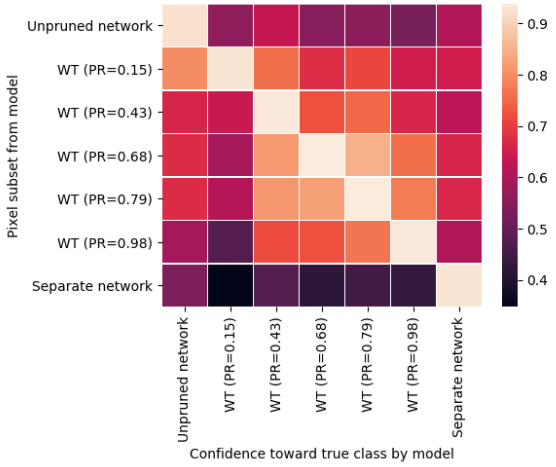
(a) ResNet20, Pruning by WT

(b) ResNet20, Pruning by FT

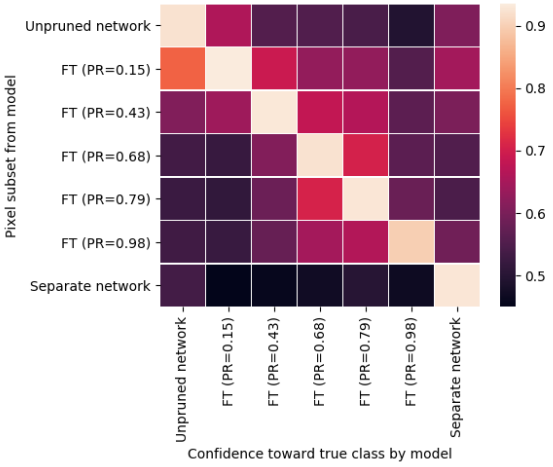
(c) ResNet20, Pruning by SiPP

(d) ResNet20, Pruning by PFP

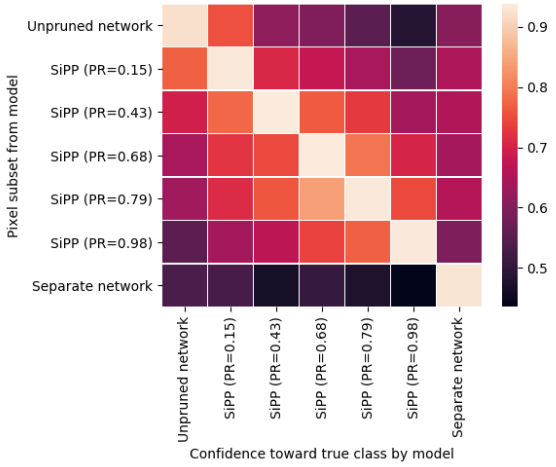
Figure 12: Heatmap of confidences on informative pixels from pruned ResNet20 models. Y-axis is the model used to generate 10% pixel subsets of 2000 sampled CIFAR-10 test images, x-axis describes the models evaluated with each 10% pixel subset, cells indicate mean confidence towards true class of the model from the x-axis on tested data from y-axis. Pruning by (a) Weight Thresholding (WT), (b) Filter Thresholding (FT), (c) SiPP, (d) Provable Filter Pruning (PFP).



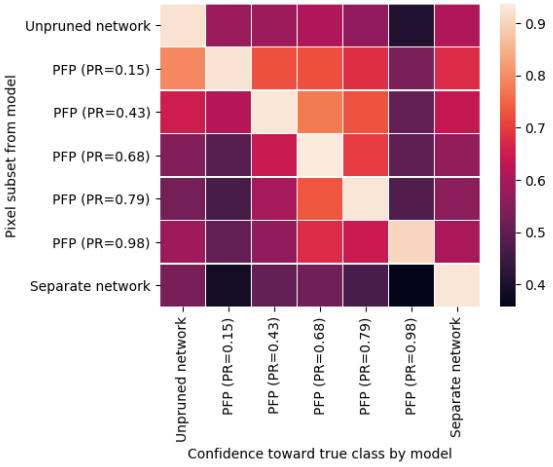
(a) VGG16, Pruning by WT



(b) VGG16, Pruning by FT

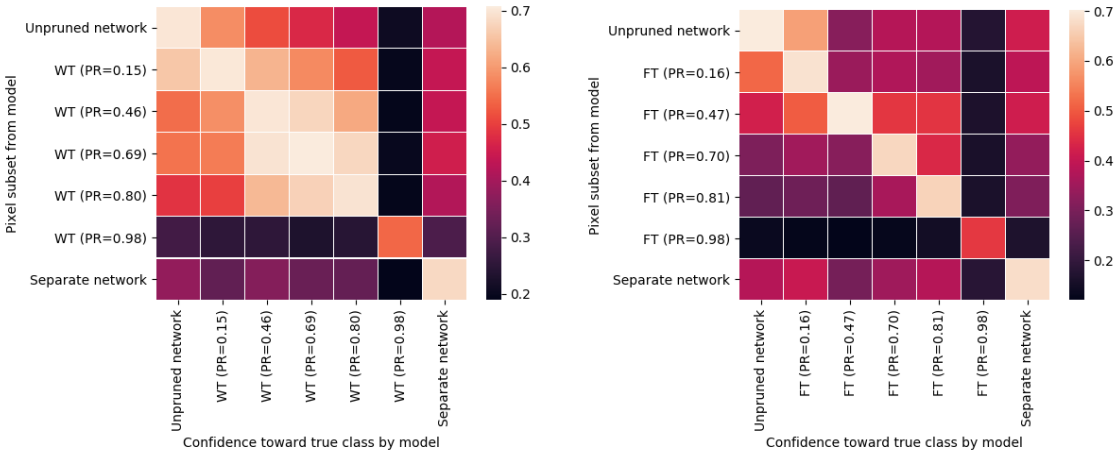


(c) VGG16, Pruning by SiPP



(d) VGG16, Pruning by PFP

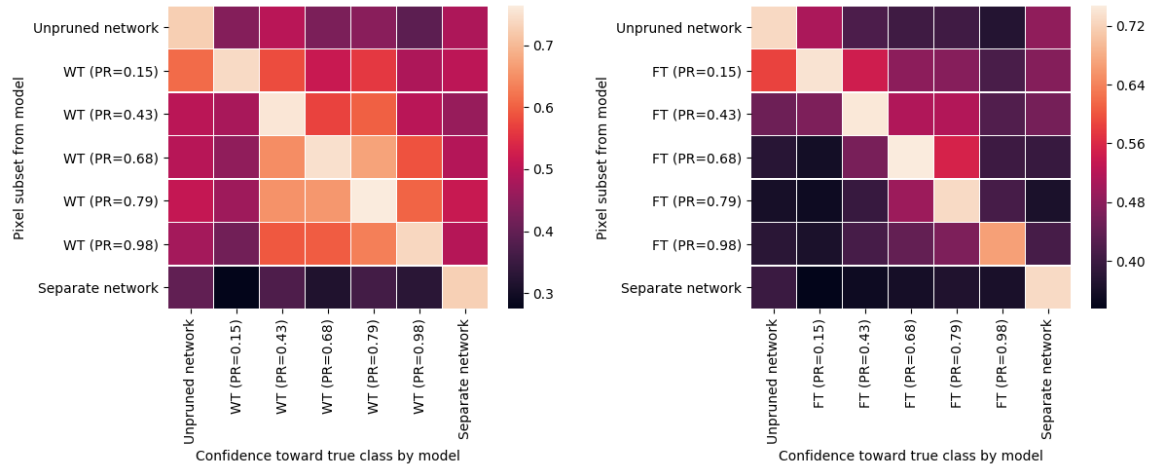
Figure 13: Heatmap of confidences on informative pixels from pruned VGG16 models. Y-axis is the model used to generate 10% pixel subsets of 2000 sampled CIFAR-10 test images, x-axis describes the models evaluated with each 10% pixel subset, cells indicate mean confidence towards true class of the model from the x-axis on tested data from y-axis. Pruning by (a) Weight Thresholding (WT), (b) Filter Thresholding (FT), (c) SiPP, (d) Provable Filter Pruning (PFP).



(a) ResNet20, Pruning by WT

(b) ResNet20, Pruning by FT

Figure 14: Heatmap of confidences on informative pixels from pruned ResNet20 models. Y-axis is the model used to generate 10% pixel subsets of 2000 randomly sampled CIFAR10-C corrupted test images, x-axis describes the models evaluated with each 10% pixel subset, cells indicate mean confidence towards true class of the model from the x-axis on tested data from y-axis. Pruning by (a) Weight Thresholding (WT), (b) Filter Thresholding (FT).



(a) VGG16, Pruning by WT

(b) VGG16, Pruning by FT

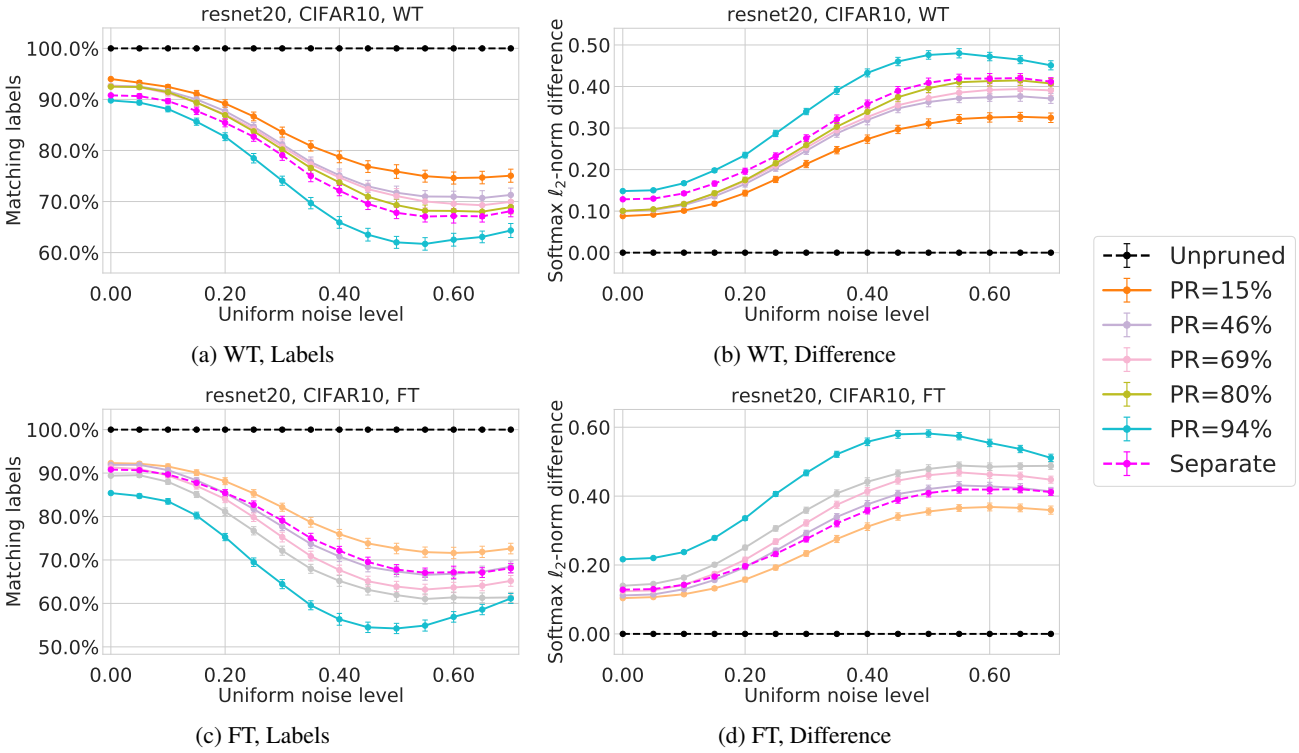
Figure 15: Heatmap of confidences on informative pixels from pruned VGG16 models. Y-axis is the model used to generate 10% pixel subsets of 2000 randomly sampled CIFAR10-C corrupted test images, x-axis describes the models evaluated with each 10% pixel subset, cells indicate mean confidence towards true class of the model from the x-axis on tested data from y-axis. Pruning by (a) Weight Thresholding (WT), (b) Filter Thresholding (FT).

C.2 Noise Similarities

We consider noise properties of networks when feeding perturbed data into the network. In particular, we are interested in understanding the similarities of the output of a pruned network and its unpruned parent as described in Section 4 of the main paper. To this end we consider two metrics: (i) percentage of matching predictions (labels) of pruned networks w.r.t. their unpruned parent for various target prune ratios and (ii) the norm-based difference between pruned networks and their unpruned parent. Each result also includes comparisons to a separately trained network of the same architecture with a different random initialization to highlight the functional similarities between unpruned and pruned networks. Overall, we find that pruned networks functionally approximate their pruned parent more closely than a separately trained network. We provide additional empirical evidence for this conclusion below.

C.2.1 Results for WT and FT on Additional Networks

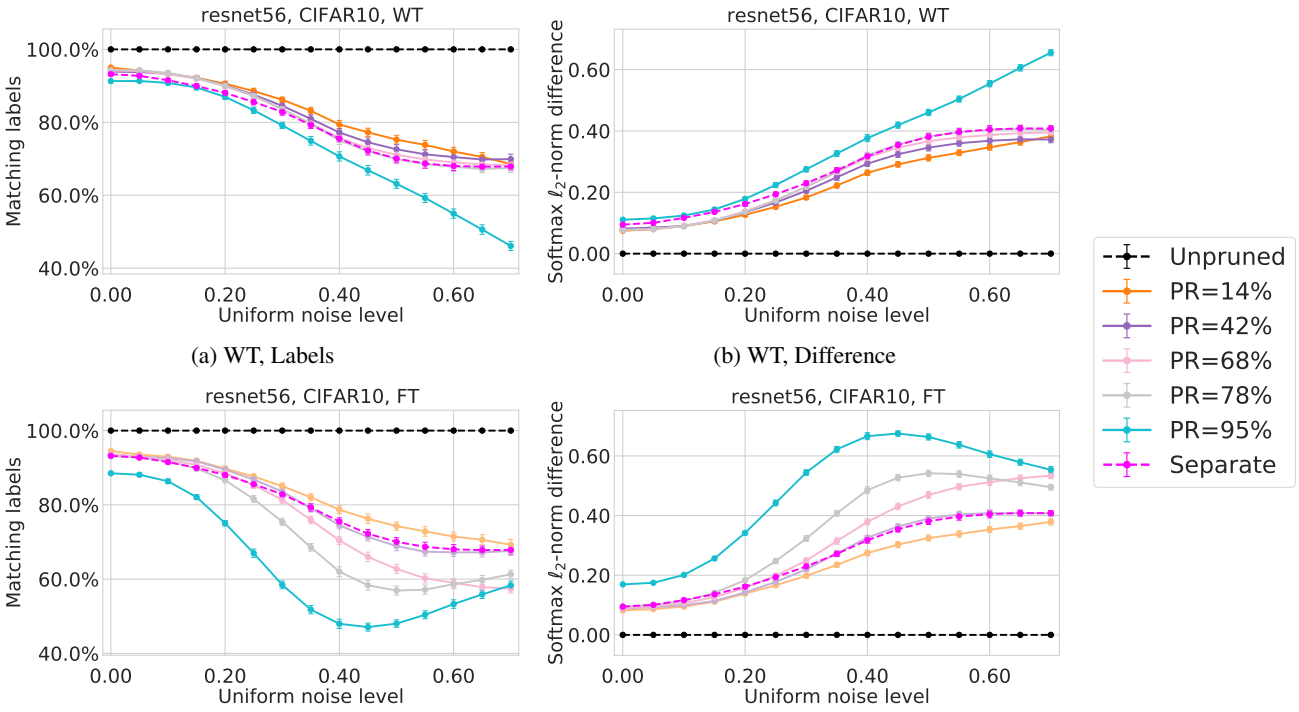
We consider the functional similarities between pruned networks and their unpruned parent for the neural network architectures ResNet20, ResNet56, ResNet110, VGG16, DenseNet22, and WideResNet16-8 trained on CIFAR10 as shown in Figures 16, 17, 18, 19, 20, and 21, respectively. All networks shown here were retrained using the same iterative prune schedule. As apparent from the respective figures, the functional similarities are consistent across architectures for the same pruning strategies.



(a) WT, Labels (b) WT, Difference

(c) FT, Labels (d) FT, Difference

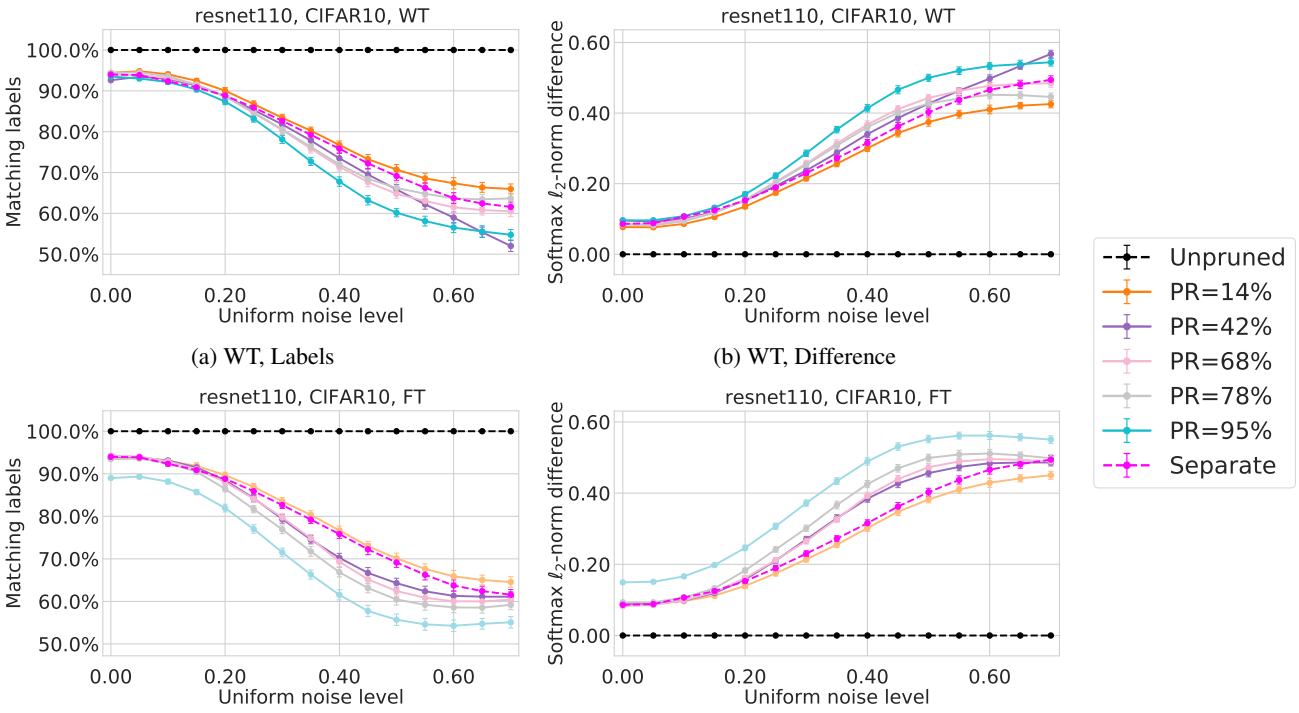
Figure 16: The functional similarities between pruned **ResNet20** models and their unpruned parent. We consider the difference in the output after injecting various amounts of noise into the input, see (a), (b) and (c), (d) for networks weight-pruned with WT and filter-pruned with FT, respectively. The differences between a separately trained network and the unpruned parent is also shown. The plots depict the difference measured as the percentage of matching predictions and as norm-based difference in the output after applying softmax, see (a), (c) and (b), (d), respectively.



(a) WT, Labels (b) WT, Difference

(c) FT, Labels (d) FT, Difference

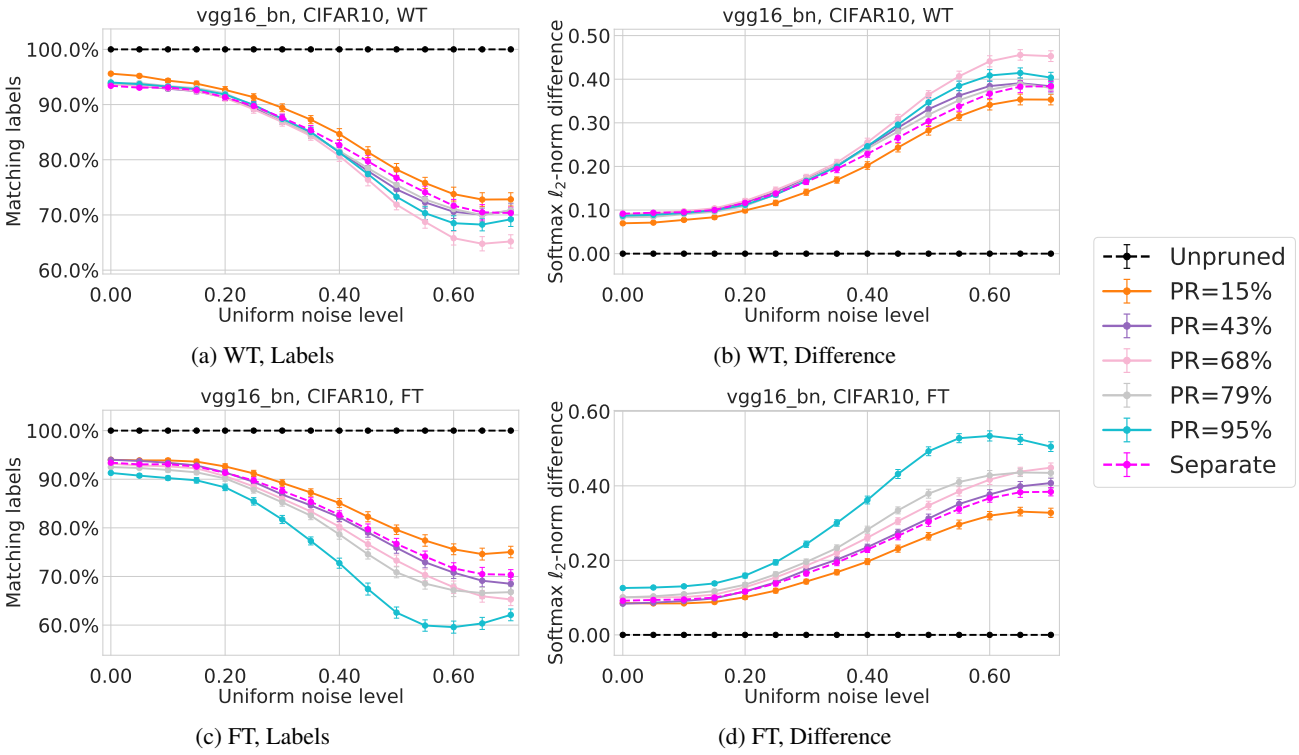
Figure 17: The functional similarities between pruned **ResNet56** models and their unpruned parent.



(a) WT, Labels (b) WT, Difference

(c) FT, Labels (d) FT, Difference

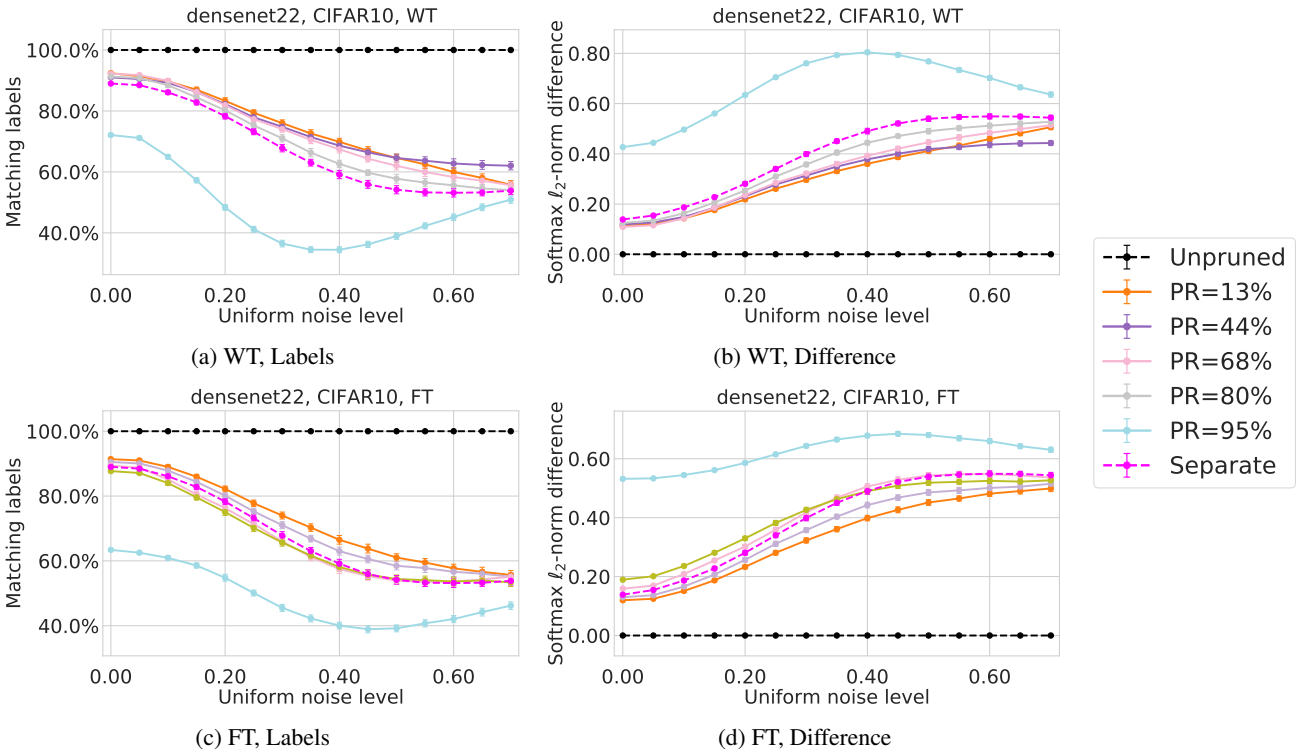
Figure 18: The functional similarities between pruned **ResNet110** models and their unpruned parent.



(a) WT, Labels (b) WT, Difference

(c) FT, Labels (d) FT, Difference

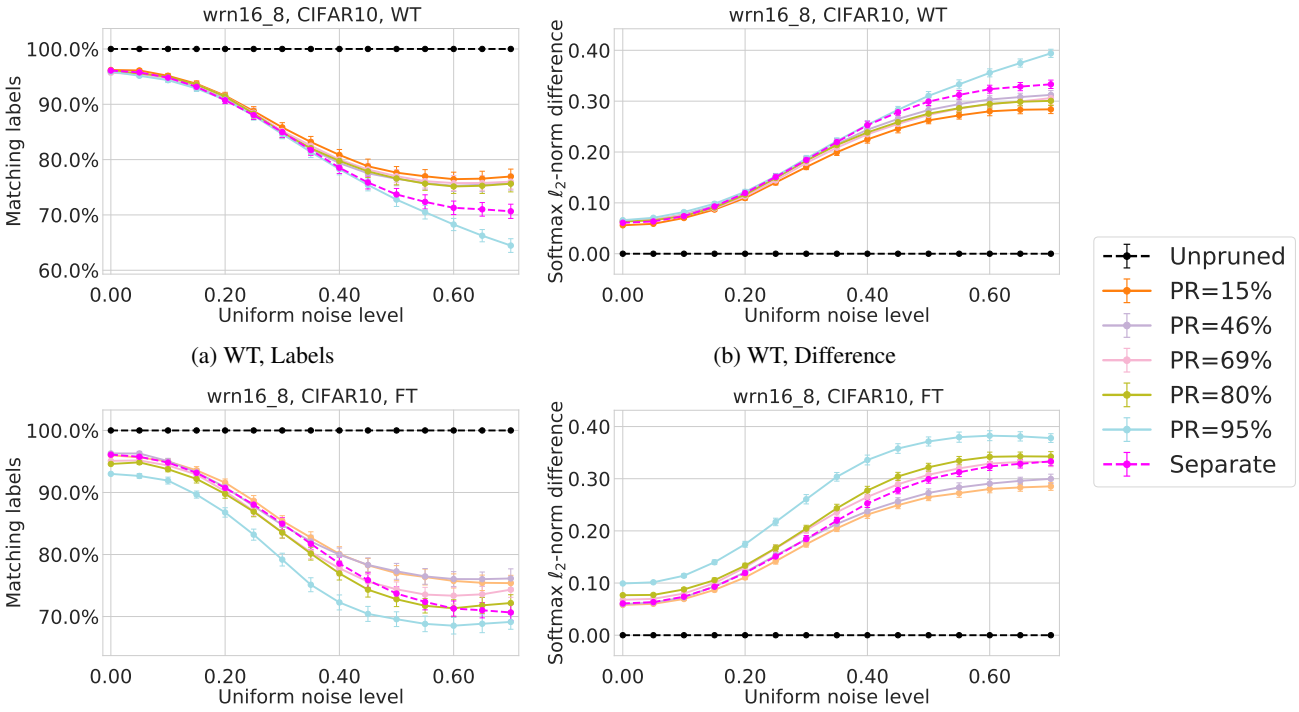
Figure 19: The functional similarities between pruned **VGG16** models and their unpruned parent.



(a) WT, Labels (b) WT, Difference

(c) FT, Labels (d) FT, Difference

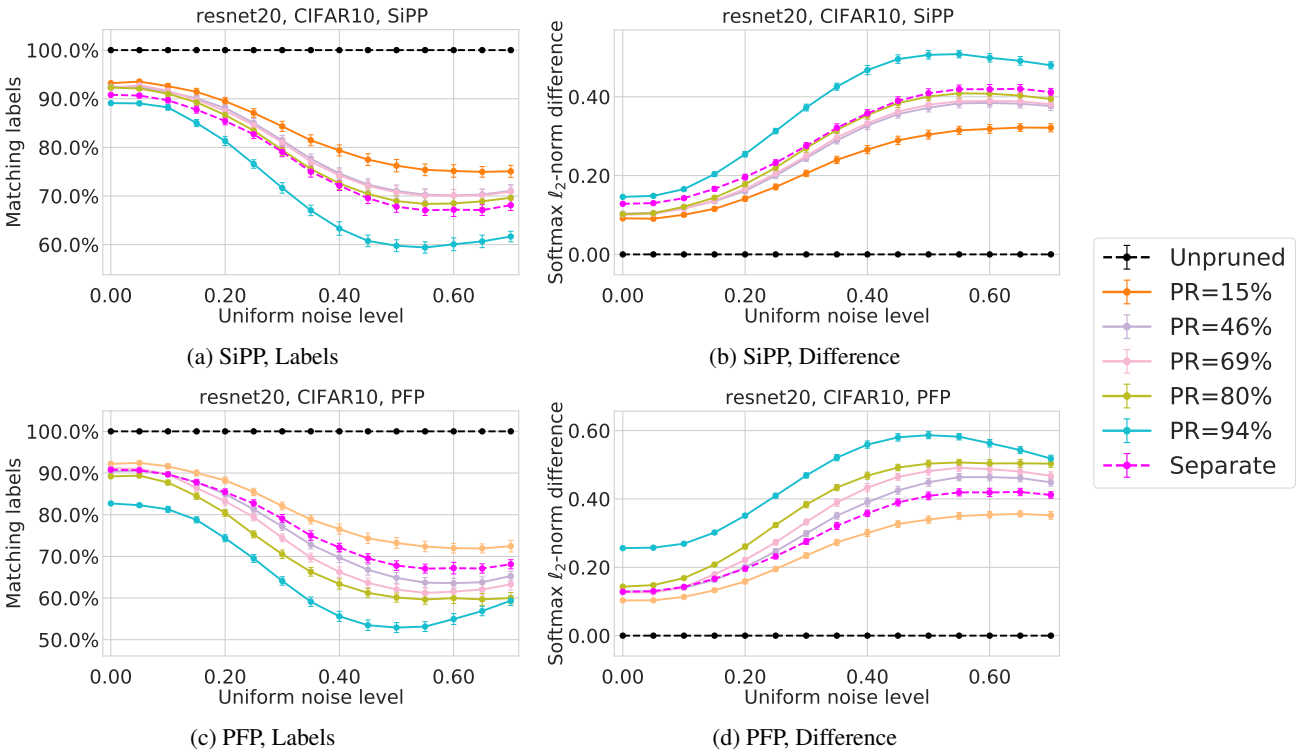
Figure 20: The functional similarities between pruned **DenseNet22** models and their unpruned parent.



(a) WT, Labels (b) WT, Difference

(c) FT, Labels (d) FT, Difference

Figure 21: The functional similarities between pruned **WRN16-8** models and their unpruned parent.



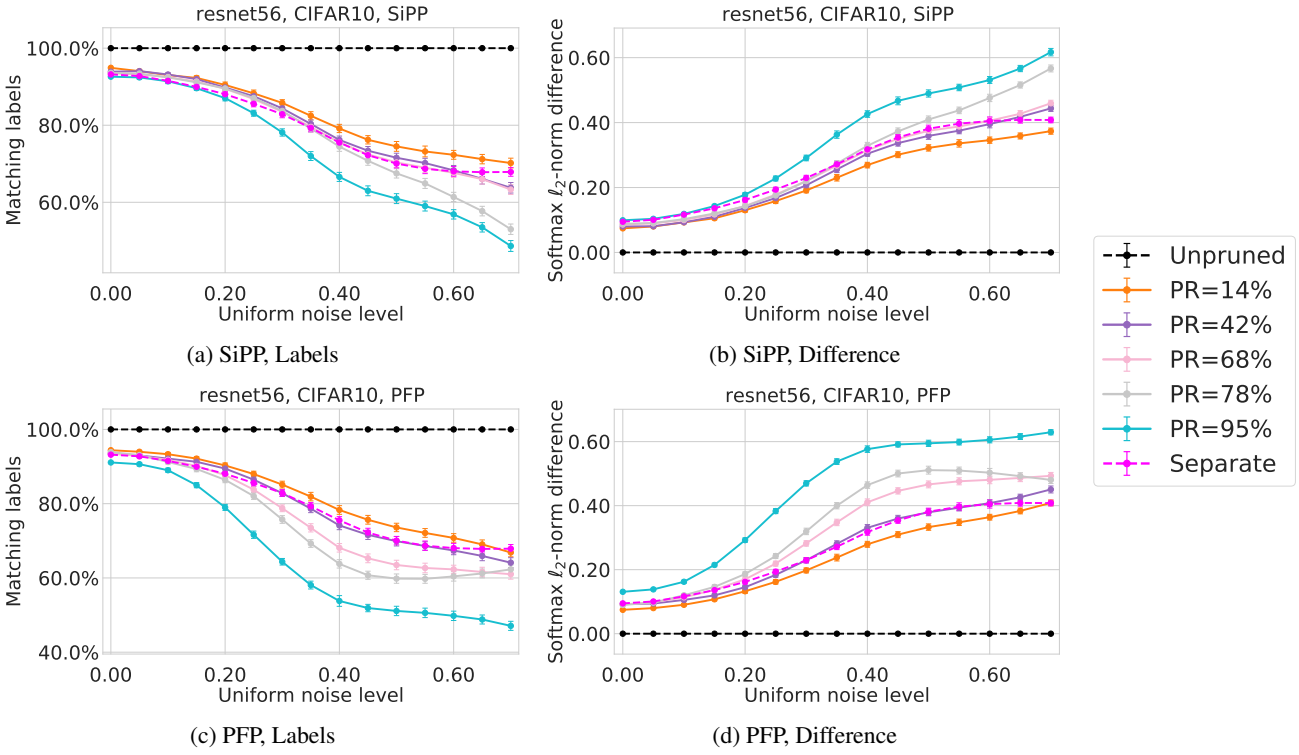
(a) SiPP, Labels (b) SiPP, Difference

(c) PFP, Labels (d) PFP, Difference

Figure 22: The functional similarities between pruned **ResNet20** models and their unpruned parent.

C.2.2 Results for Additional Pruning Methods (SiPP and PFP)

In the following we compare the functional similarities using the alternative weight and filter pruning methods SiPP (Baykal et al., 2019b) and PFP (Liebenwein et al., 2020), respectively. The methods are described in more detail in Section 3 of the main paper. Below we present results for ResNet20, ResNet56, ResNet110, VGG16, DenseNet22, and WRN16-8, see Figures 22, 23, 24, 25, 26, and 27 respectively. We note that the conclusions with regards to the functional similarities remain in essence unaltered for alternative pruning methods. Networks were trained with the same iterative prune-retrain schedule as in the previous subsection.



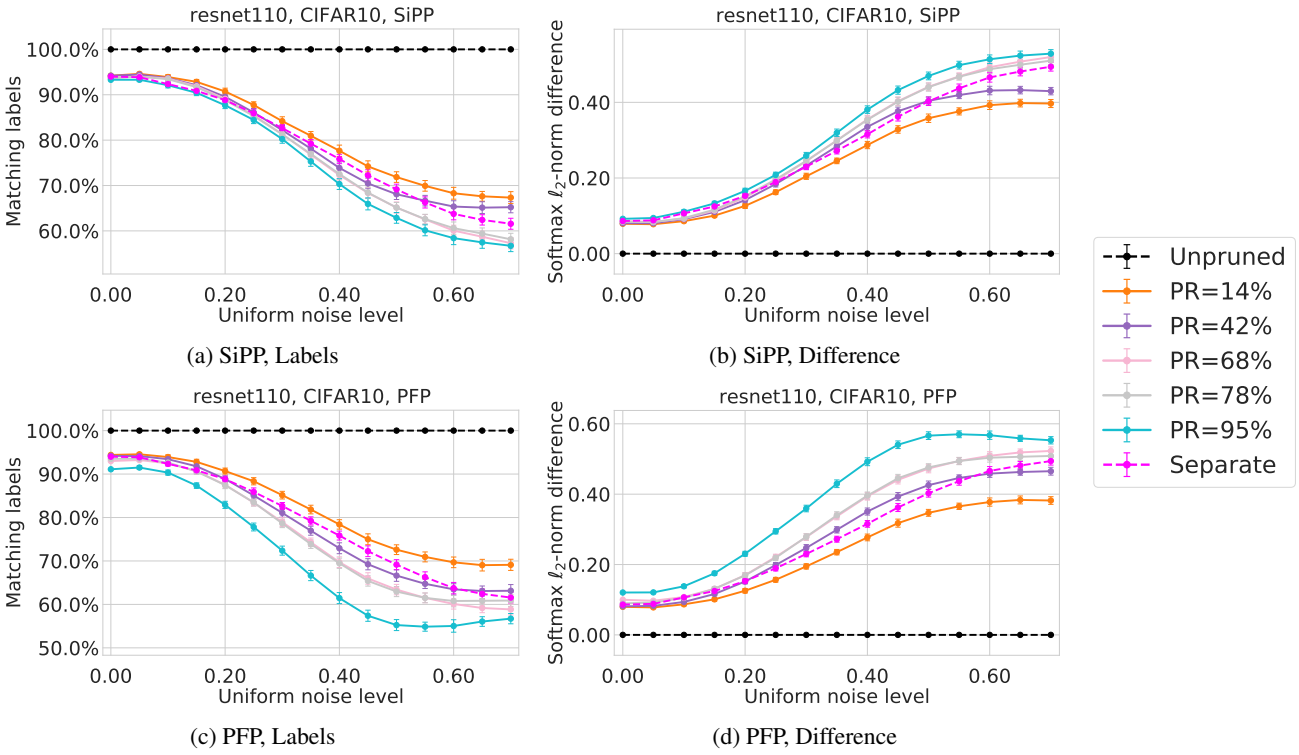
(a) SiPP, Labels

(b) SiPP, Difference

(c) PFP, Labels

(d) PFP, Difference

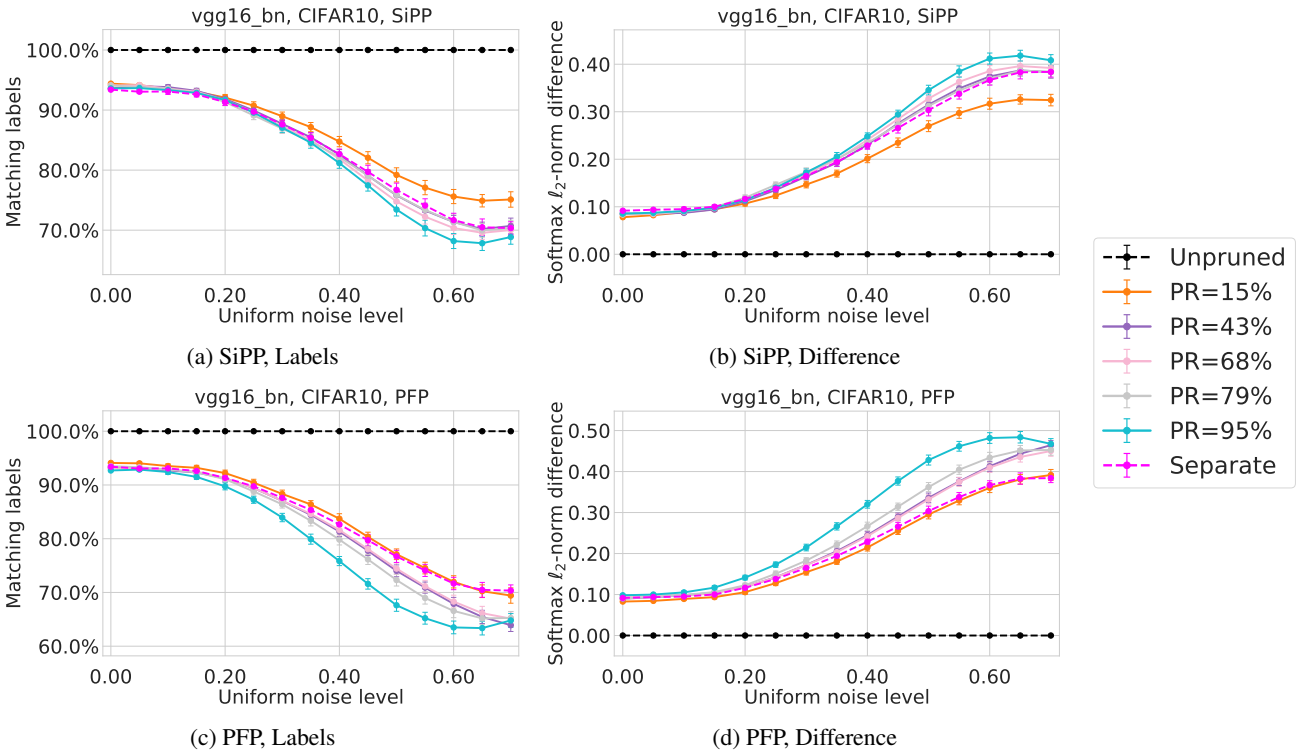
Figure 23: The functional similarities between pruned **ResNet56** models and their unpruned parent.



(a) SiPP, Labels (b) SiPP, Difference

(c) PFP, Labels (d) PFP, Difference

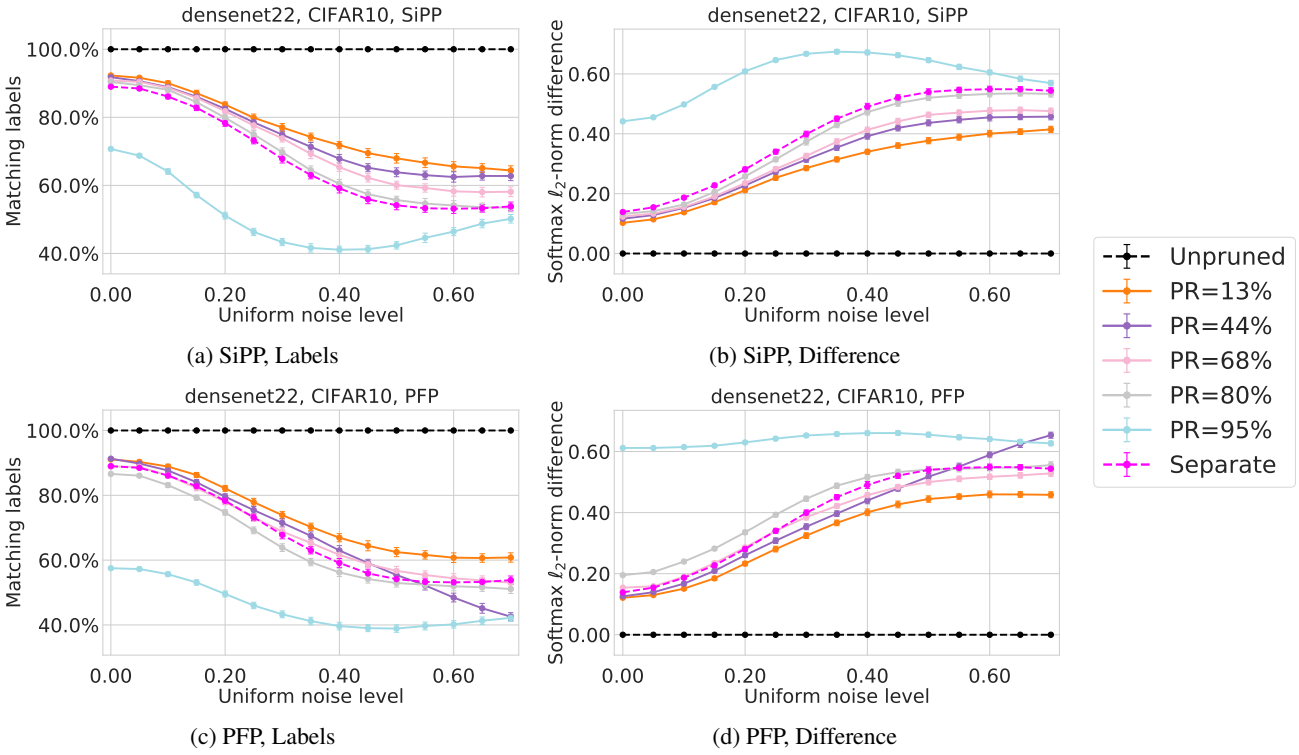
Figure 24: The functional similarities between pruned **ResNet110** models and their unpruned parent.



(a) SiPP, Labels (b) SiPP, Difference

(c) PFP, Labels (d) PFP, Difference

Figure 25: The functional similarities between pruned **VGG16** models and their unpruned parent.



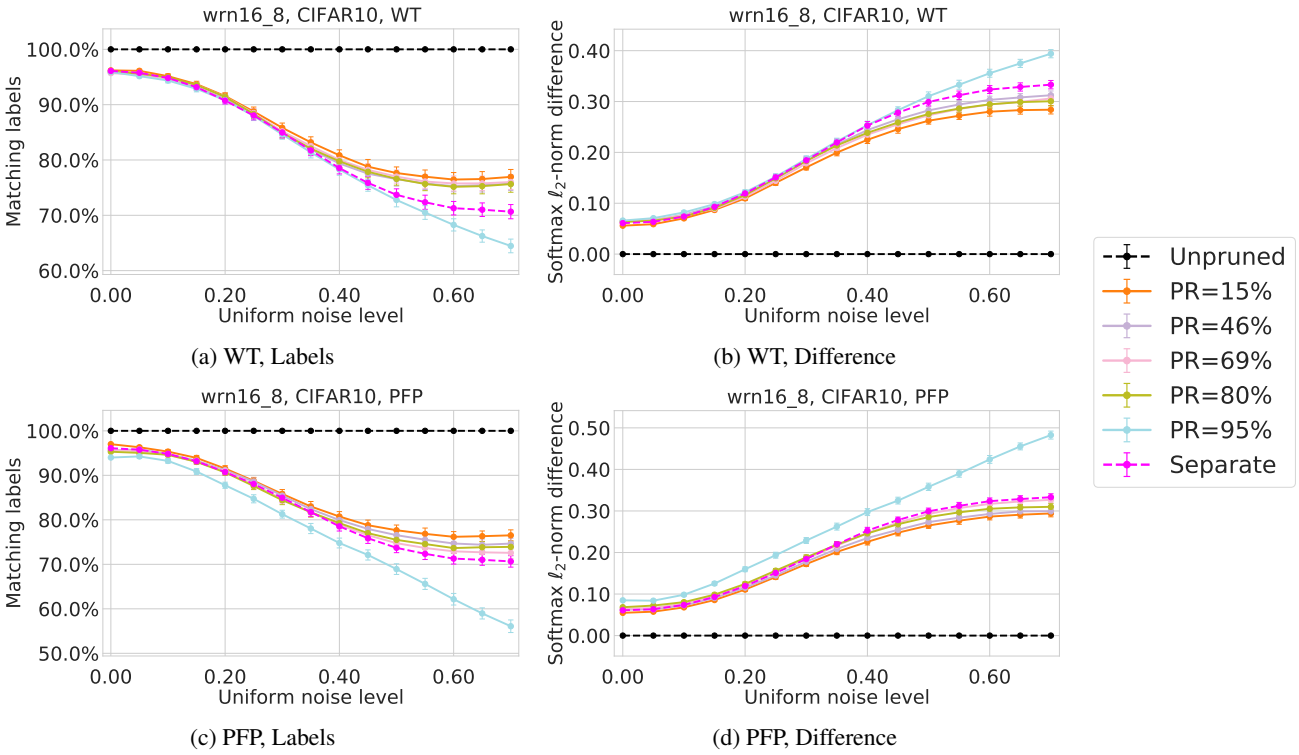
(a) SiPP, Labels

(b) SiPP, Difference

(c) PFP, Labels

(d) PFP, Difference

Figure 26: The functional similarities between pruned **DenseNet22** models and their unpruned parent.



(a) WT, Labels

(b) WT, Difference

(c) PFP, Labels

(d) PFP, Difference

Figure 27: The functional similarities between pruned **WRN16-8** models and their unpruned parent.

D ADDITIONAL RESULTS FOR PRUNE POTENTIAL

We present additional experimental evaluation for the results presented in Section 5 of the main paper. As in Section 5, we consider the *prune potential* for pruned networks when testing the network under noisy input and under corrupted images.

D.1 Additional Results for Prune Potential Based on Noise

Here, we inject randomly sampled, bounded uniform noise into the normalized input and investigate the prune potential under various noise levels. Overall we find that the prune potential significantly decreases as more noise is injected into the input providing further evidence of the results presented in the main paper.

We consider the prune potential for the neural network architectures ResNet20, ResNet56, ResNet110, VGG16, DenseNet22, and WideResNet16-8 trained on CIFAR10 as shown in Figure 28. All networks were iteratively pruned and retrained. Overall, we can observe that the prune potential decreases with higher levels of noise and that filter-pruned networks tend to have lower prune potential than weight-pruned networks for any given target prune ratio. However, we note that depending on the architecture the specific prune potential may be less affected. Specifically, we note that the prune potential of WideResNets (Figure 28, bottom right) seems to be entirely unaffected by noise across all tested prune methods. In contrast to the other networks, WideResNets are wider and less deep, which may provide a possible explanation for the observed behaviors. Due to the wide spatial dimensions of each network, the noise may be better absorbed since it may spread across the width of the layer. Moreover, the reduced depth may help in avoiding positive amplification of the noise as depth increases. From this observation, we may conclude that WideResNets are indeed overparameterized and henceforth the prune potential remains unaffected for slight perturbations in the input.

D.2 Additional Results for Prune Potential Based on Corruptions

Following Section 5 we present additional results pertaining to the prune potential of networks when generalizing to out-of-distribution test data of various kinds. For CIFAR10, we consider the o.o.d. data sets by Hendrycks & Dietterich (2019) (CIFAR10-C) which are publicly available. Additionally, we also compare to CIFAR10.1 by Recht et al. (2018), which is an alternative in-distribution test data set for CIFAR10. For ImageNet, we consider the ImageNet versions of the CIFAR10-C data sets, denoted by ImageNet-C (Hendrycks & Dietterich, 2019). Additionally, we compare to ObjectNet (Barbu et al., 2019), an o.o.d. data set that exhibits large variations over the context and the pose of an object instead of image corruptions. For VOC, we consider the same set of corruptions, denoted by VOC-C, based on the generalization of the CIFAR10-C corruptions to any image data set by Michaelis et al. (2019). For CIFAR10-C, ImageNet-C, and VOC-C, we evaluate the prune potential for severity level 3 out of 5 (Hendrycks & Dietterich, 2019; Michaelis et al., 2019). In accordance with the results presented in Section 5 we find that the prune potential can vary substantially depending on the task.

We consider the out-of-distribution prune potential for different CIFAR10-C data sets (severity level 3) for the network architectures ResNet20, ResNet56, ResNet110, VGG16, DenseNet22, and WideResNet16-8, see Figures 29, 30, 31, 32, 33, and 34, respectively. Each network was weight-pruned and filter-pruned with WT and SiPP, and FT and PFP, respectively. We note that in general the prune potential varies across prune methods, network architectures, and task. Moreover, some of the networks seem to cope better with out-of-distribution data than other networks (e.g. WideResNet16-8 as seen from Figure 34). However, across all experiments the prune potential varies significantly and it seems difficult to predict clear trends highlighting the sensitivity of the prune potential w.r.t. out-of-distribution test data.

D.3 Additional Results for Prune Potential Based on Corruptions on ImageNet and VOC

Finally, we consider the prune potential for out-of-distribution test data on a ResNet18 (ImageNet), ResNet101 (ImageNet), and DeeplabV3 (VOC), see Figures 35, 36, and 37, respectively. We note that the prune potential in this case is equally sensitive to the test task emphasizing that our observations scale to larger networks and data sets as well instead of being confined to small-scale data sets such as CIFAR10. Moreover, considering the expansive nature of ImageNet experiments we did not prune the network to very extreme prune ratios but instead stopped at around 80%-90%. In light of these observations, we conjecture that the nominal prune ratio is even higher, which would result in an even larger overall gap in prune potential between in-distribution and out-of-distribution test data.

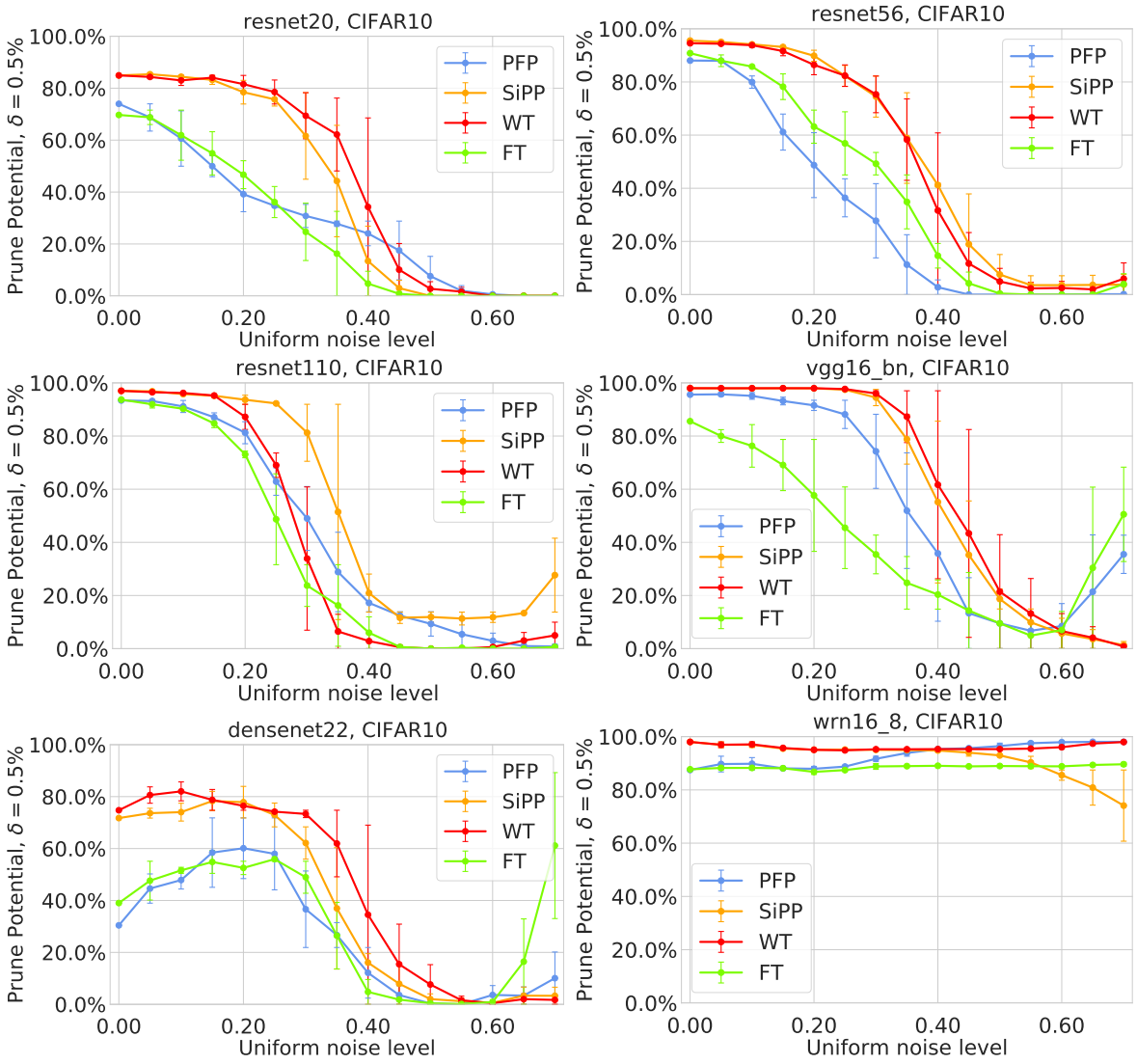
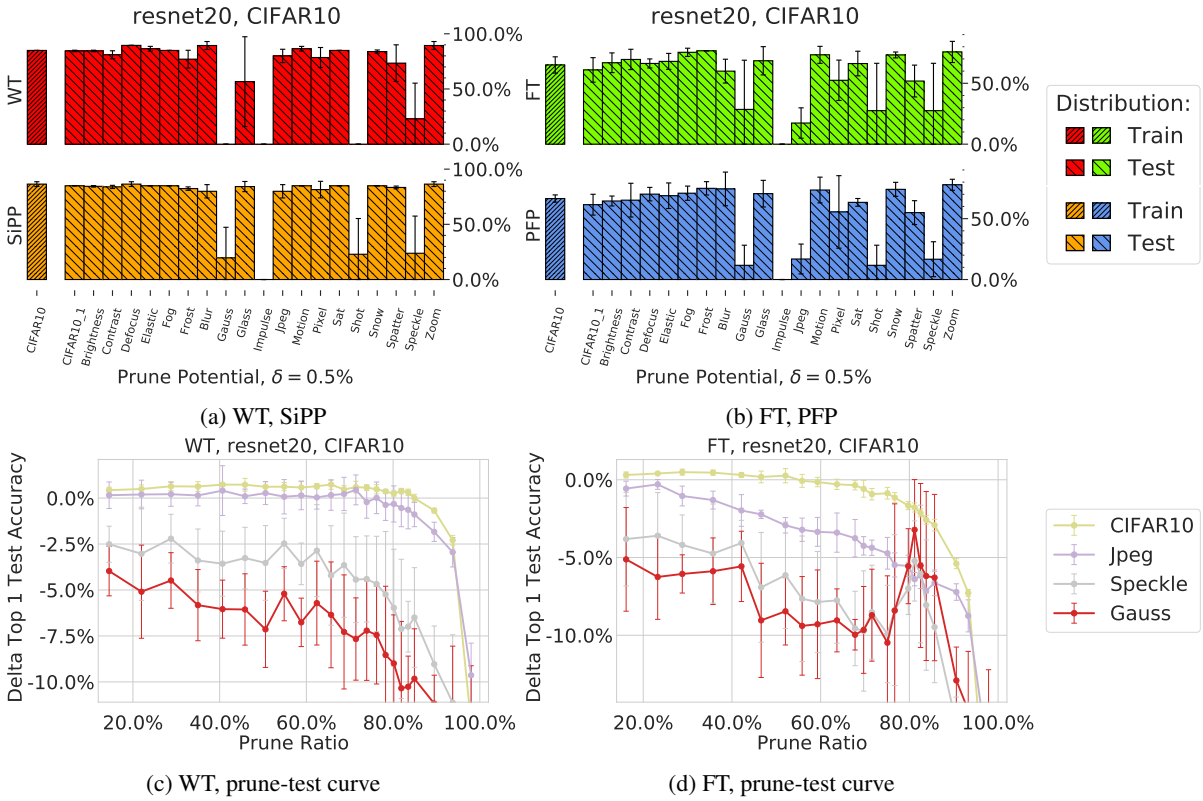


Figure 28: The prune potential (%) achievable over various levels of noise injected into the input.



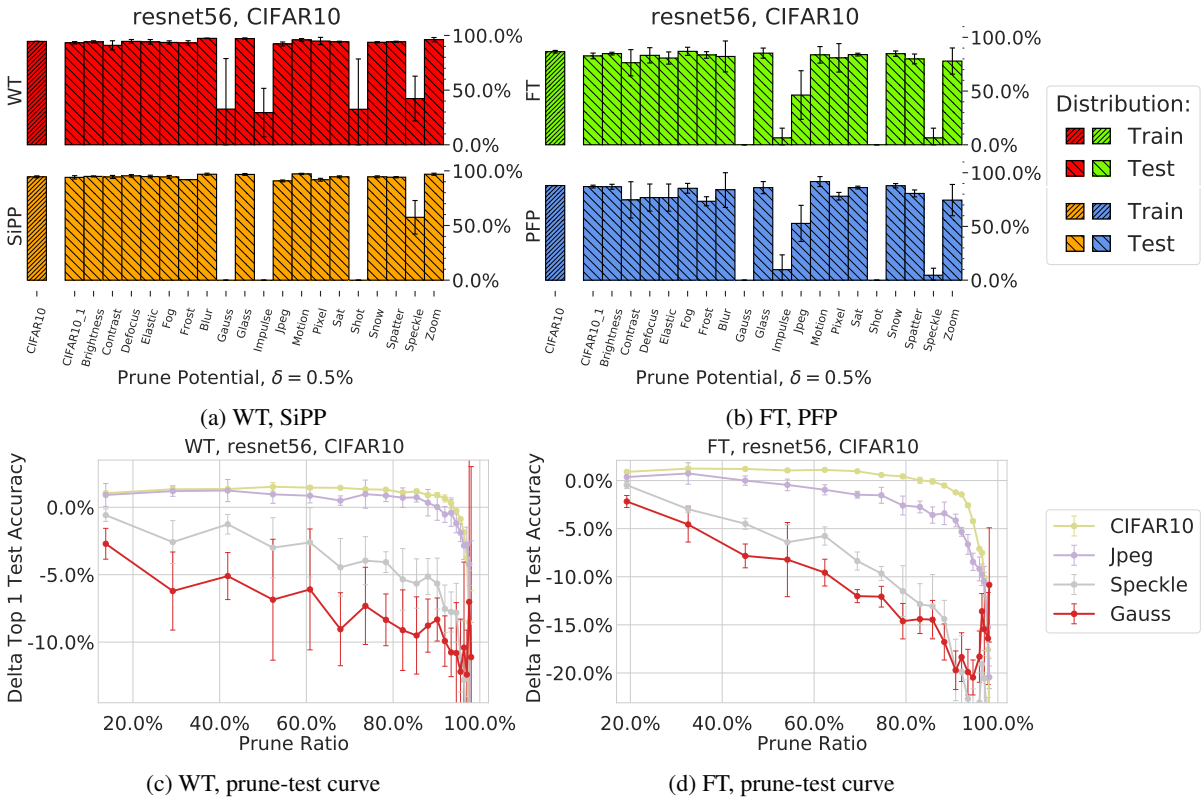
(a) WT, SiPP

(b) FT, PFP

(c) WT, prune-test curve

(d) FT, prune-test curve

Figure 29: The prune potential of a **ResNet20** achievable for **CIFAR10** out-of-distribution data sets.



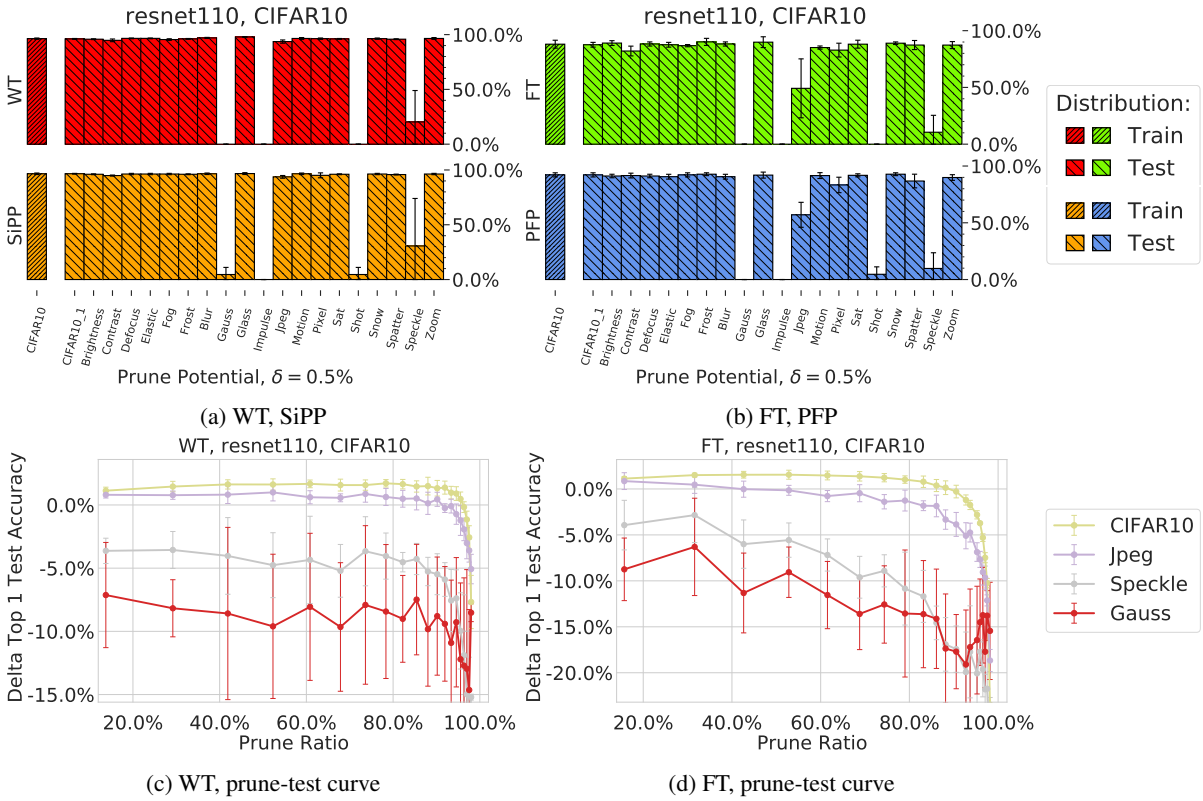
(a) WT, SiPP

(b) FT, PFP

(c) WT, prune-test curve

(d) FT, prune-test curve

Figure 30: The prune potential of a **ResNet56** achievable for **CIFAR10** out-of-distribution data sets.



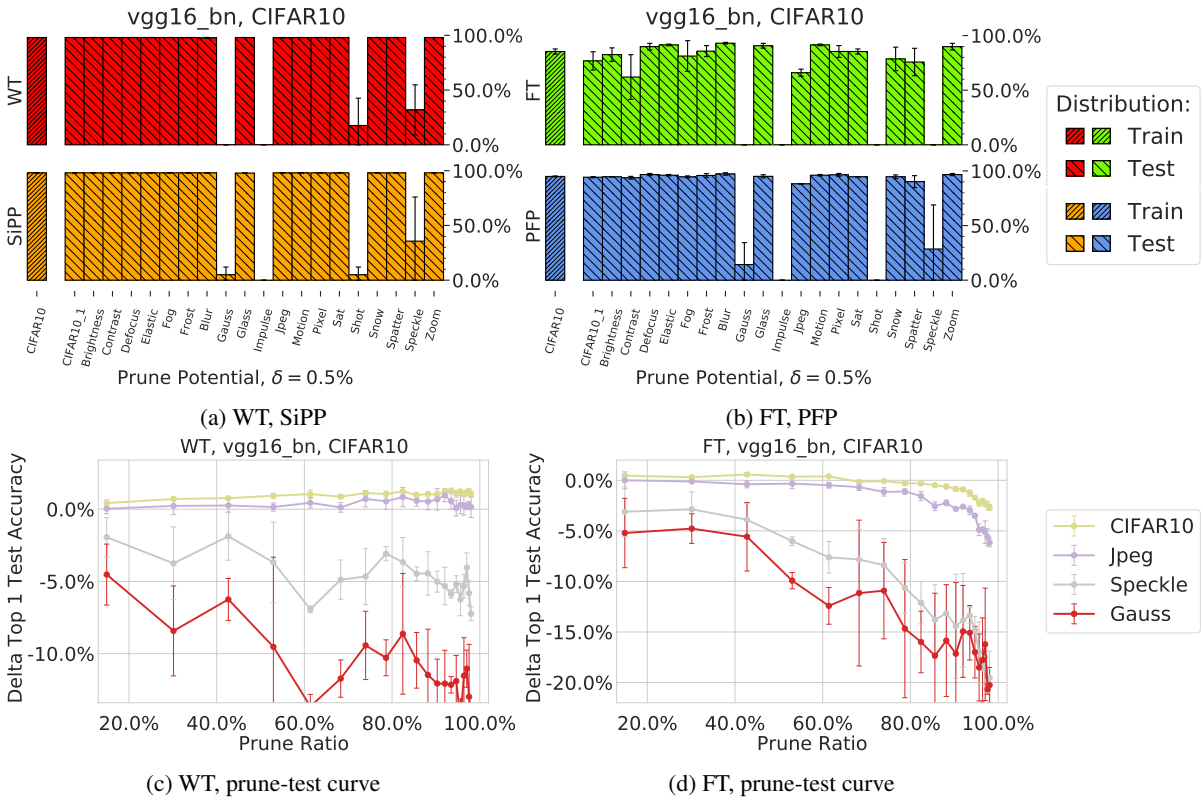
(a) WT, SiPP

(b) FT, PFP

(c) WT, prune-test curve

(d) FT, prune-test curve

Figure 31: The prune potential of a **ResNet110** achievable for **CIFAR10** out-of-distribution data sets.



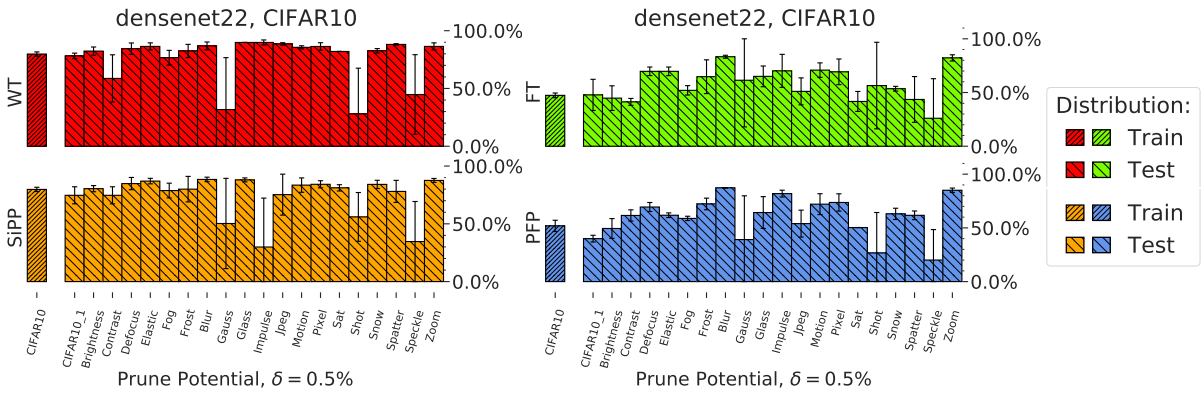
(a) WT, SiPP

(b) FT, PFP

(c) WT, prune-test curve

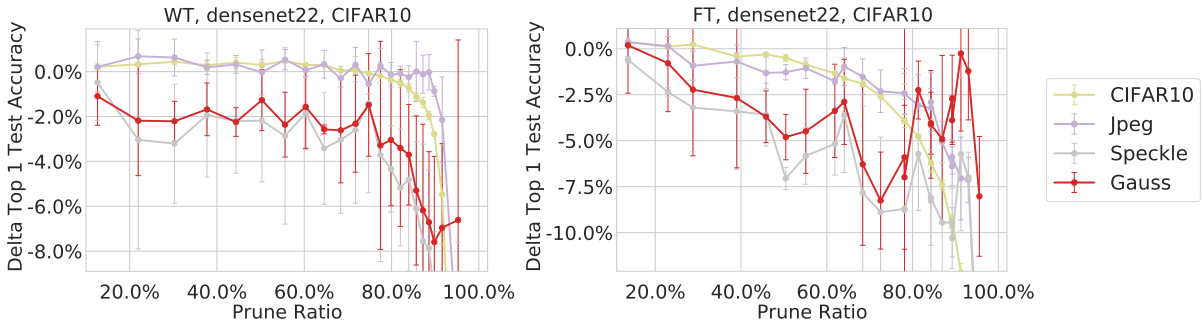
(d) FT, prune-test curve

Figure 32: The prune potential of a **VGG16** achievable for **CIFAR10** out-of-distribution data sets.



(a) WT, SiPP

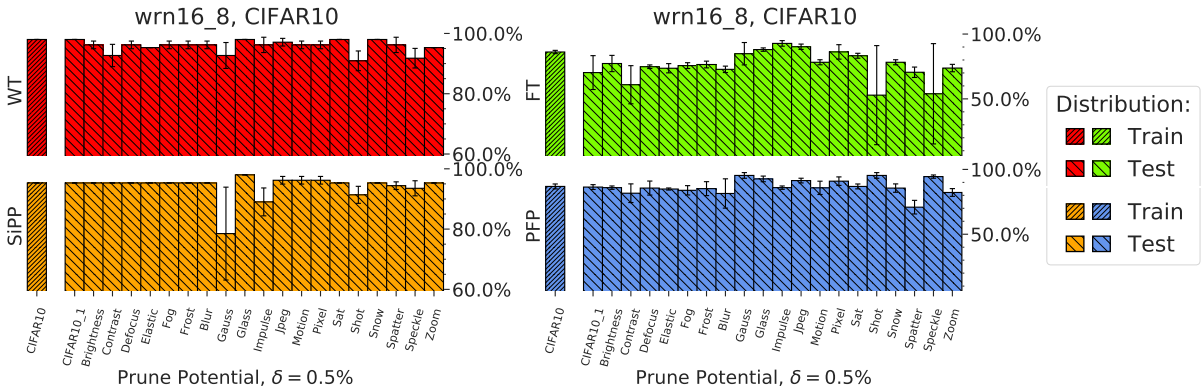
(b) FT, PFP



(c) WT, prune-test curve

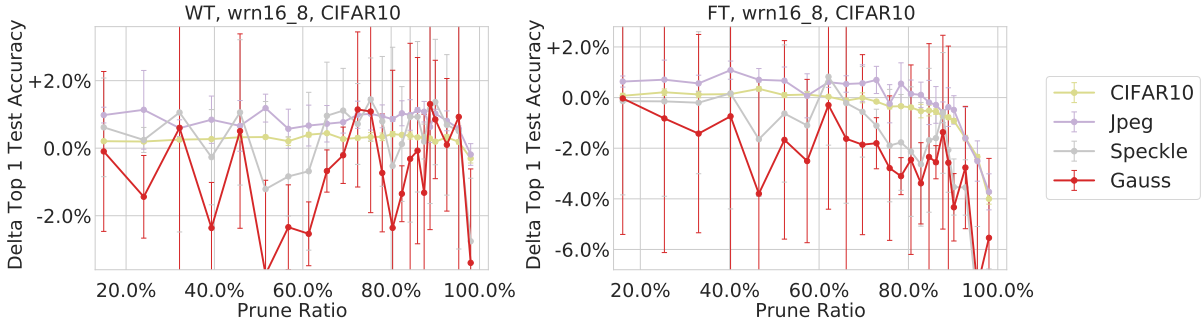
(d) FT, prune-test curve

Figure 33: The prune potential of a **DenseNet22** achievable for **CIFAR10** out-of-distribution data sets.



(a) WT, SiPP

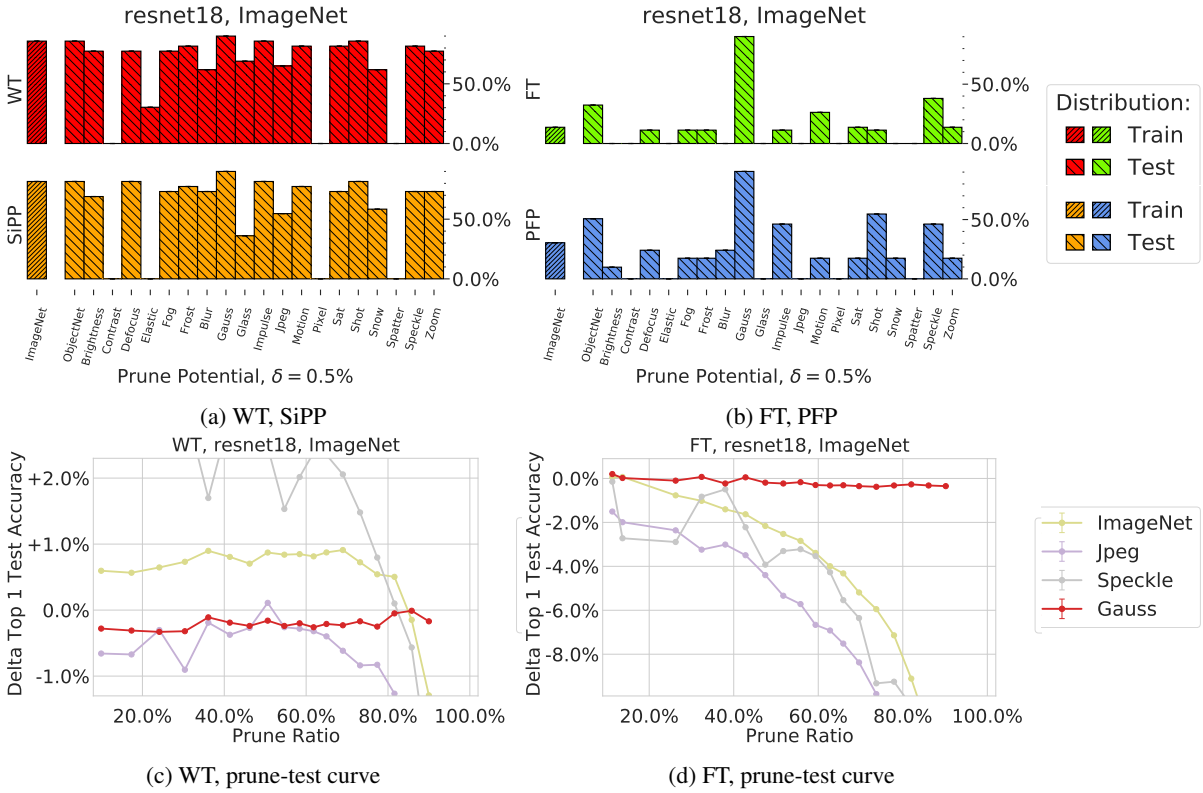
(b) FT, PFP



(c) WT, prune-test curve

(d) FT, prune-test curve

Figure 34: The prune potential of a **WRN16-8** achievable for **CIFAR10** out-of-distribution data sets.



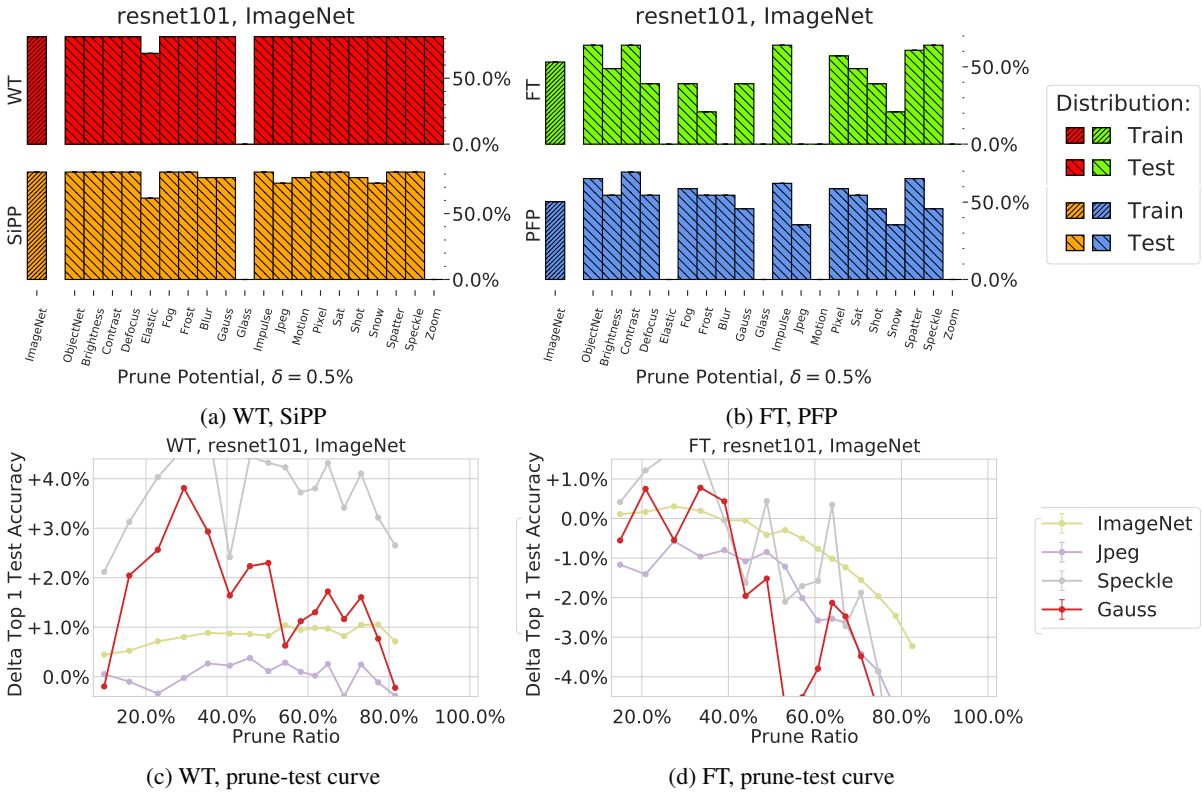
(a) WT, SiPP

(b) FT, PFP

(c) WT, prune-test curve

(d) FT, prune-test curve

Figure 35: The prune potential of a **ResNet18** achievable for **ImageNet** out-of-distribution data sets.



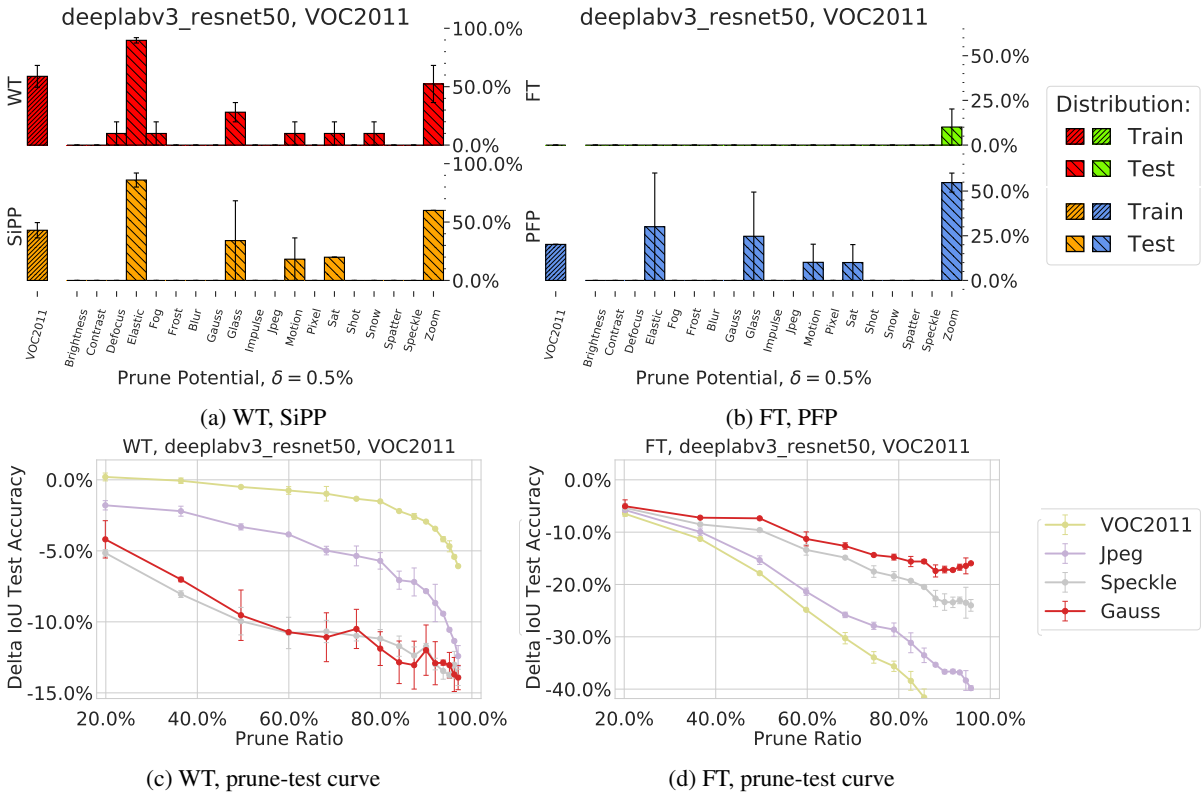
(a) WT, SiPP

(b) FT, PFP

(c) WT, prune-test curve

(d) FT, prune-test curve

Figure 36: The prune potential of a **ResNet101** achievable for **ImageNet** out-of-distribution data sets.



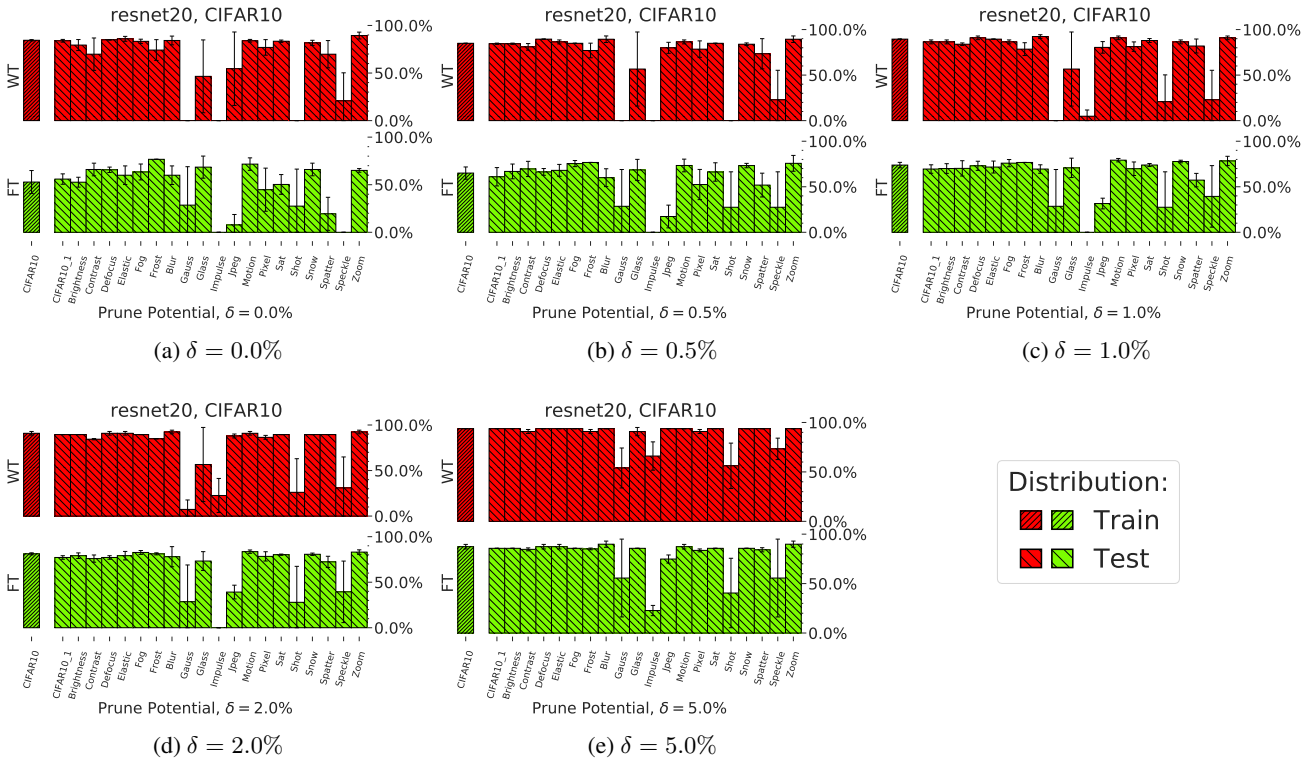
(a) WT, SiPP

(b) FT, PFP

(c) WT, prune-test curve

(d) FT, prune-test curve

Figure 37: The prune potential of a **DeeplabV3** achievable for **Pascal VOC** out-of-distribution data sets.



(a) $\delta = 0.0\%$

(b) $\delta = 0.5\%$

(c) $\delta = 1.0\%$

(d) $\delta = 2.0\%$

(e) $\delta = 5.0\%$

Figure 38: The prune potential of a ResNet20 trained on CIFAR10 for WT and FT, respectively. In each figure the same experiment is repeated with different values of δ ranging from 0% to 5%.

D.4 Choice of δ

We evaluated the prune potential for a ResNet20 trained on CIFAR10 for a range of possible values for δ to ensure that our observations hold independent of the specific choice of δ. Recall that δ denotes the amount of "slack" when evaluating the prune potential, i.e., the difference in test accuracy between the pruned and unpruned network for which the pruned network's performance is considered commensurate. Specifically, as shown in Figure 38, we consider a range of δ between 0% and 5%. Naturally, the prune potential is higher overall for larger values of δ. However, we can see that our main observations essentially remain unchanged, that is the prune potential still significantly varies across different tasks and distributions. Overall, these results confirm our previous findings.

D.5 Additional Results for Excess Error Based on Corruptions

Recall that the excess error is defined as the additional error incurred on the test distribution compared to the error on the train distribution. The difference in excess error between pruned and unpruned networks thus quantifies the *additional error* incurred by the pruned network on the test distribution *on top of* the error increase incurred by the pruned network compared to the unpruned network according to the prune-accuracy curve on the train distribution.

We used ordinary least squares (linear regression) to compute the prediction of the relationship between prune ratio and difference in excess error. The y-intercept is set to 0 since by the definition the difference in excess error is 0% for a prune ratio of 0%. The shaded regions describe the 95% confidence intervals, which were computed based on bootstrapping.

The results for the CIFAR10 network architectures ResNet20, ResNet56, ResNet110, VGG16, DenseNet22, and WRN16-8 are shown in Figures 39, 40, 41, 42, 43, and 44, respectively. The results for ImageNet network architectures ResNet18 and ResNet101 are shown in Figures 45 and 46, respectively. The results for the VOC network architecture DeeplabV3 is shown in Figure 47. Note that ideally the slope would be zero indicating that the prune-accuracy curve on nominal data is predictive of o.o.d. data. However, as shown most pruned networks exhibit a significant increase in excess error that increases with higher prune ratios. These results further corroborate our understanding that networks cannot be pruned to full extent when faced with o.o.d. data.

Notable exceptions include WRN16-8 (CIFAR10) and ResNet101 (ImageNet) with little correlation between the prune ratio and the difference in excess error indicating that those networks may be genuinely overparameterized in a robust sense confirming our findings from previous sections.

The negative delta in excess error for FT on DeeplabV3 (Figure 47), on the other hand, is a spurious consequence of the prune potential of FT for nominal data already being zero rather than a consequence of the amount of overparameterization in the network.

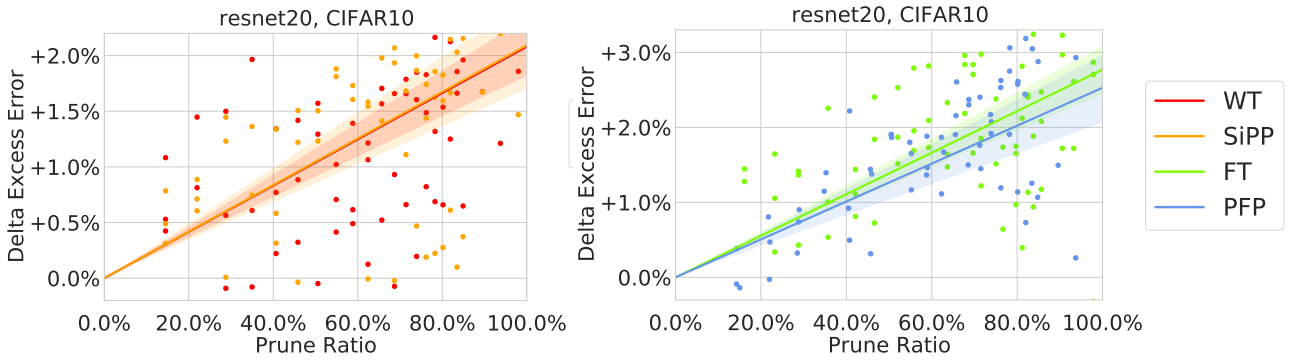


Figure 39: The difference in excess error for a **ResNet20** trained on **CIFAR10**.

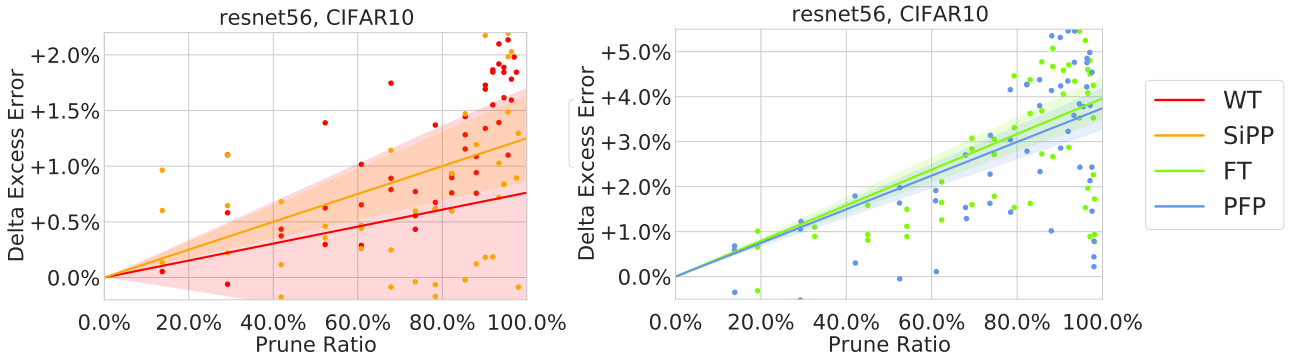


Figure 40: The difference in excess error for a **ResNet56** trained on **CIFAR10**.

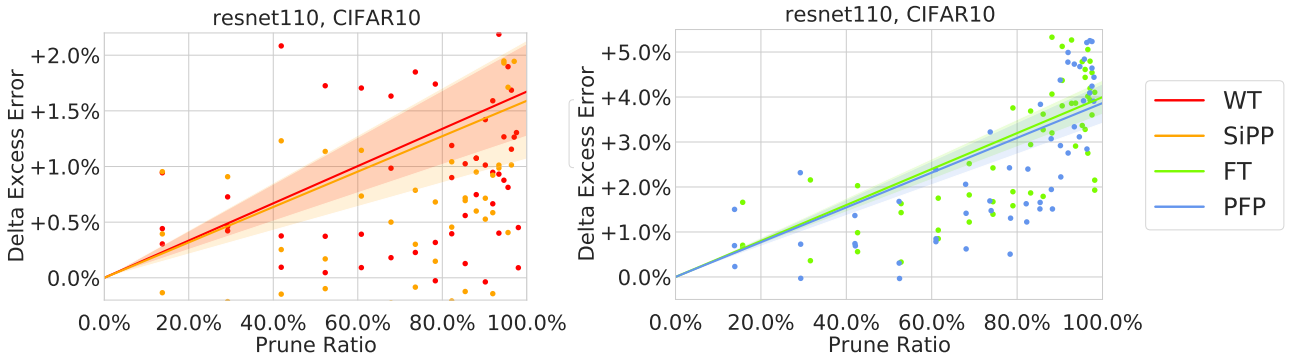


Figure 41: The difference in excess error for a **ResNet110** trained on **CIFAR10**.

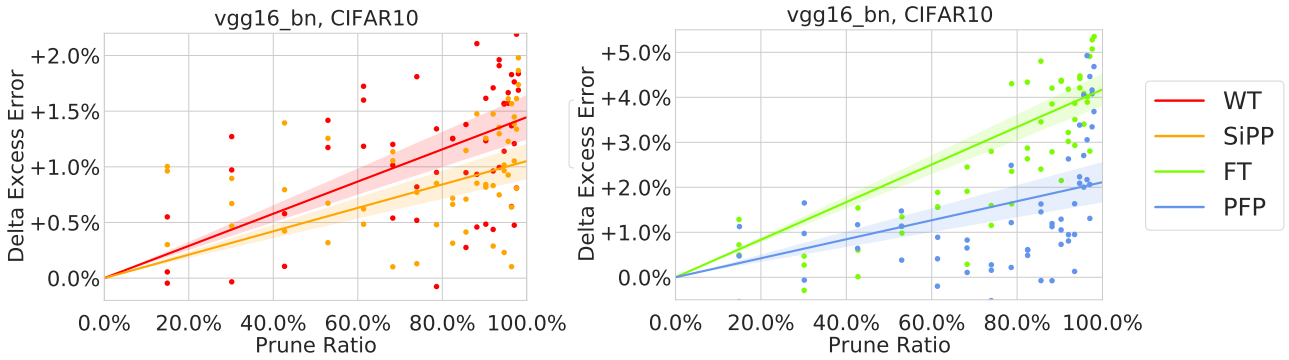


Figure 42: The difference in excess error for a **VGG16** trained on **CIFAR10**.

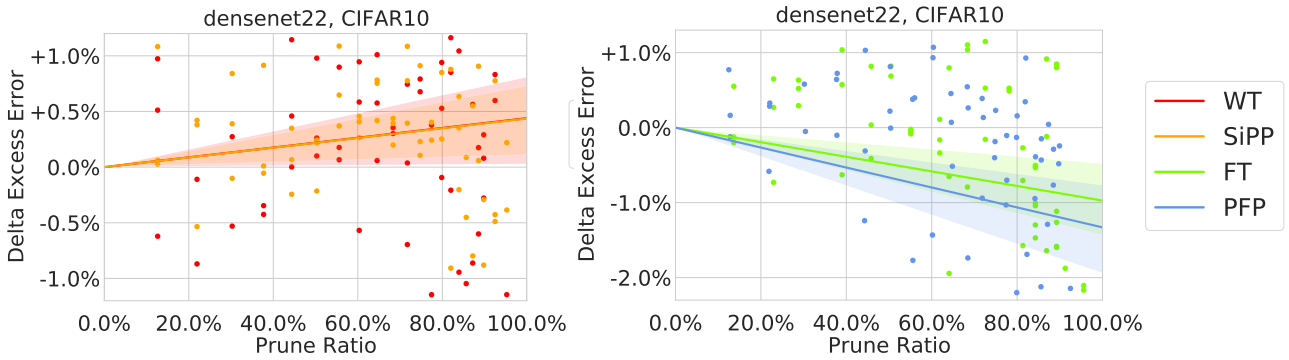


Figure 43: The difference in excess error for a **DenseNet22** trained on **CIFAR10**.

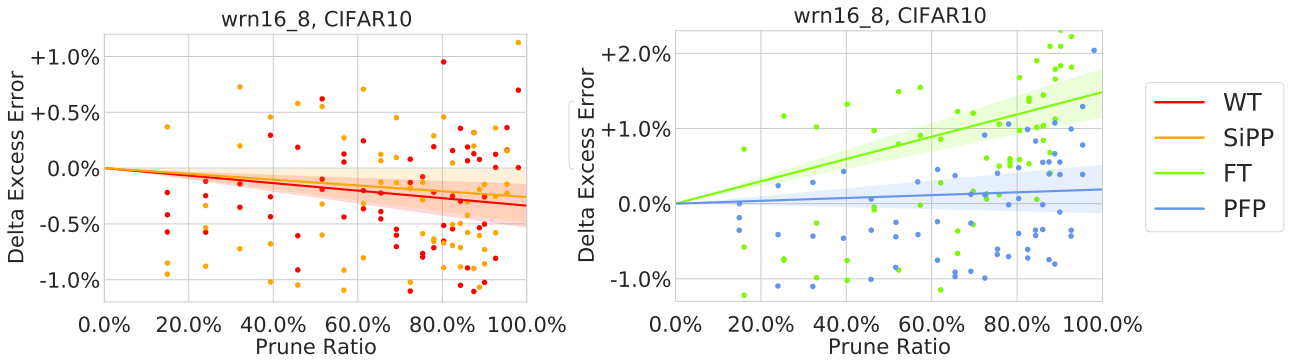


Figure 44: The difference in excess error for a **WRN16-8** trained on **CIFAR10**.

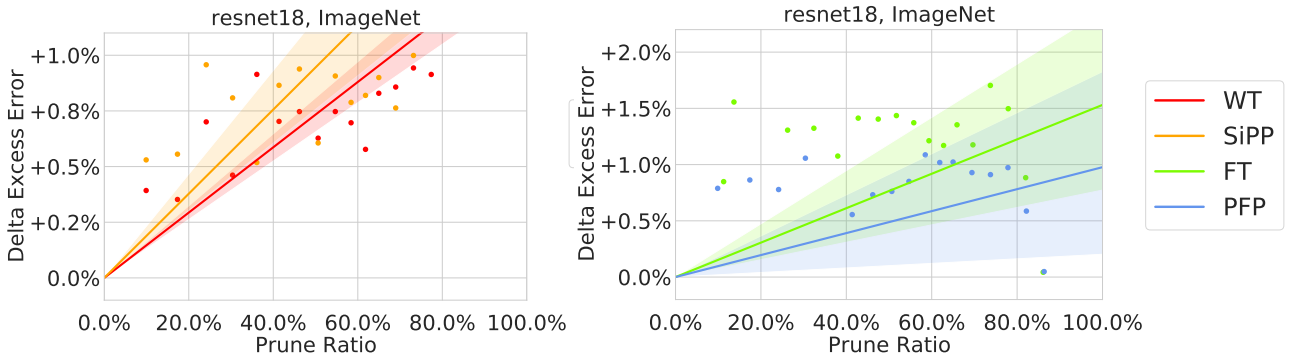


Figure 45: The difference in excess error for a **ResNet18** trained on **ImageNet**.

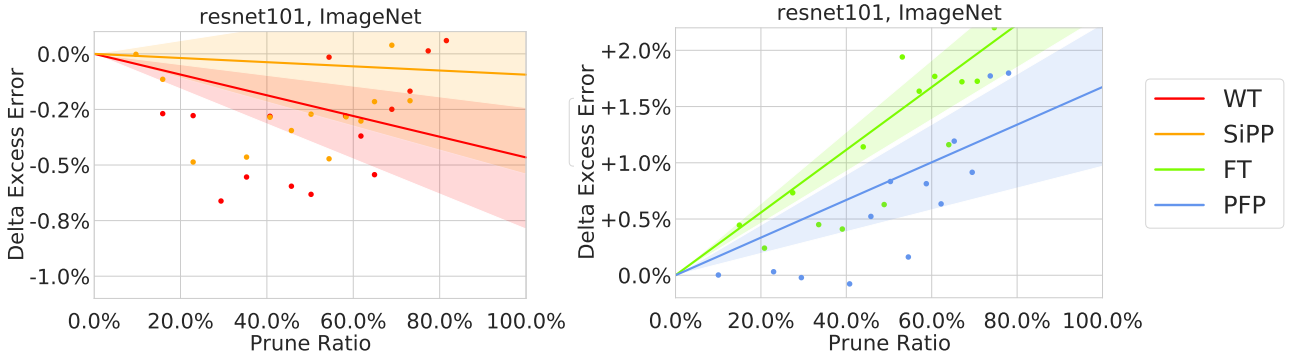


Figure 46: The difference in excess error for a **ResNet101** trained on **ImageNet**.

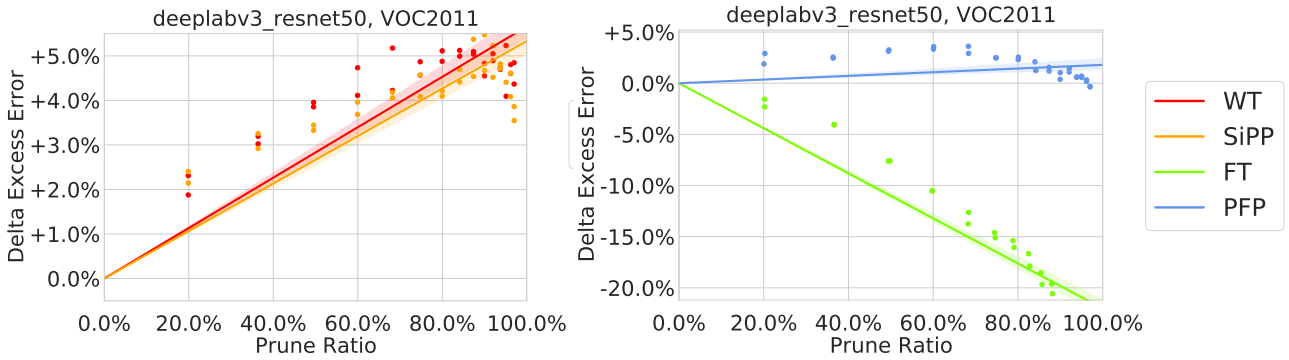


Figure 47: The difference in excess error for a **DeeplabV3** trained on **Pascal VOC**.

D.6 Results for Overparameterization

We summarize our results pertaining to using the prune potential as a way to gauge the amount of overparameterization. Specifically, for each network and prune method, we evaluate the *average* and *minimum* prune potential for both the train and test distribution. The average and minimum are hereby computed over the corruptions/variations that are included in each distribution. Note that for these experiments the train distribution only contains the nominal data; thus the average and minimum coincides. For the test distribution we take the average and minimum over all the respective corruptions. The mean and standard deviation reported are computed over three repetitions of the same experiment.

The resulting prune potentials are listed in Tables 9 and 10 for weight pruning (WT, SiPP) and filter pruning (FT, PFP), respectively. Note that for most networks we can observe around 20% drop in average prune potential between train and test distribution while most networks have 0% (!) minimum prune potential for data from the test distribution. As previously observed some networks may be considered *genuinely* overparameterized in the robust sense including WRN16-8, ResNet101, which manifests itself with a very stable prune potential across both train and test distribution.

Model	WT - Prune Potential (%)				SiPP - Prune Potential (%)			
	Average		Minimum		Average		Minimum	
	Train Dist.	Test Dist.	Train Dist.	Test Dist.	Train Dist.	Test Dist.	Train Dist.	Test Dist.
ResNet20	84.9 ± 0.0	$\mathbf{66.7 \pm 3.3}$	84.9 ± 0.0	$\mathbf{0.0 \pm 0.0}$	86.4 ± 2.2	$\mathbf{70.4 \pm 4.3}$	86.4 ± 2.2	$\mathbf{0.0 \pm 0.0}$
Resnet56	94.6 ± 0.0	$\mathbf{82.3 \pm 4.2}$	94.6 ± 0.0	$\mathbf{4.6 \pm 6.5}$	94.5 ± 0.9	$\mathbf{78.5 \pm 1.0}$	94.5 ± 0.9	$\mathbf{0.0 \pm 0.0}$
ResNet110	96.3 ± 0.6	$\mathbf{77.9 \pm 1.4}$	96.3 ± 0.6	$\mathbf{0.0 \pm 0.0}$	96.5 ± 0.7	$\mathbf{78.8 \pm 2.8}$	96.5 ± 0.7	$\mathbf{0.0 \pm 0.0}$
VGG16	98.0 ± 0.0	$\mathbf{80.9 \pm 2.2}$	98.0 ± 0.0	$\mathbf{0.0 \pm 0.0}$	98.0 ± 0.0	$\mathbf{80.7 \pm 1.9}$	98.0 ± 0.0	$\mathbf{0.0 \pm 0.0}$
DenseNet22	79.8 ± 1.9	$\mathbf{76.0 \pm 6.7}$	79.8 ± 1.9	$\mathbf{24.9 \pm 35.2}$	79.8 ± 1.9	$\mathbf{74.1 \pm 9.1}$	79.8 ± 1.9	$\mathbf{21.5 \pm 30.4}$
WRN16-8	98.0 ± 0.0	$\mathbf{95.7 \pm 0.7}$	98.0 ± 0.0	$\mathbf{90.0 \pm 2.1}$	95.3 ± 0.0	$\mathbf{94.1 \pm 0.8}$	95.3 ± 0.0	$\mathbf{78.1 \pm 15.1}$
ResNet18	85.8 ± 0.0	$\mathbf{63.6 \pm 0.0}$	85.8 ± 0.0	$\mathbf{0.0 \pm 0.0}$	81.6 ± 0.0	$\mathbf{57.8 \pm 0.0}$	81.6 ± 0.0	$\mathbf{0.0 \pm 0.0}$
ResNet101	81.6 ± 0.0	$\mathbf{76.8 \pm 0.0}$	81.6 ± 0.0	$\mathbf{0.0 \pm 0.0}$	81.6 ± 0.0	$\mathbf{70.7 \pm 0.0}$	81.6 ± 0.0	$\mathbf{0.0 \pm 0.0}$
DeeplabV3	58.9 ± 9.3	$\mathbf{11.6 \pm 2.7}$	58.9 ± 9.3	$\mathbf{0.0 \pm 0.0}$	43.0 ± 6.6	$\mathbf{11.5 \pm 3.1}$	43.0 ± 6.6	$\mathbf{0.0 \pm 0.0}$

Table 9: The average and minimum prune potential computed on the train and test distribution, respectively, for weight prune methods (WT, SiPP). The train distribution hereby consists of nomimal data, while the test distribution consists of the CIFAR10-C, ImageNet-C, VOC-C corruptions.

Model	FT - Prune Potential (%)				PFP - Prune Potential (%)			
	Average		Minimum		Average		Minimum	
	Train Dist.	Test Dist.	Train Dist.	Test Dist.	Train Dist.	Test Dist.	Train Dist.	Test Dist.
ResNet20	65.0 ± 6.7	$\mathbf{55.3 \pm 4.8}$	65.0 ± 6.7	$\mathbf{0.0 \pm 0.0}$	66.5 ± 3.0	$\mathbf{53.9 \pm 4.4}$	66.5 ± 3.0	$\mathbf{0.0 \pm 0.0}$
ResNet56	86.6 ± 1.2	$\mathbf{64.8 \pm 3.7}$	86.6 ± 1.2	$\mathbf{0.0 \pm 0.0}$	88.1 ± 0.0	$\mathbf{64.9 \pm 4.0}$	88.1 ± 0.0	$\mathbf{0.0 \pm 0.0}$
ResNet110	88.1 ± 3.5	$\mathbf{68.5 \pm 3.2}$	88.1 ± 3.5	$\mathbf{0.0 \pm 0.0}$	92.2 ± 1.8	$\mathbf{71.6 \pm 1.8}$	92.2 ± 1.8	$\mathbf{0.0 \pm 0.0}$
VGG16	85.4 ± 2.4	$\mathbf{66.3 \pm 0.5}$	85.4 ± 2.4	$\mathbf{0.0 \pm 0.0}$	95.0 ± 0.5	$\mathbf{77.9 \pm 2.1}$	95.0 ± 0.5	$\mathbf{0.0 \pm 0.0}$
DenseNet22	$\mathbf{47.4 \pm 2.2}$	58.2 ± 5.3	47.4 ± 2.2	$\mathbf{9.6 \pm 13.6}$	$\mathbf{51.8 \pm 5.3}$	59.6 ± 6.0	51.8 ± 5.3	$\mathbf{12.6 \pm 17.8}$
WRN16-8	86.2 ± 1.3	$\mathbf{75.7 \pm 4.1}$	86.2 ± 1.3	$\mathbf{36.6 \pm 28.6}$	86.9 ± 1.9	$\mathbf{86.6 \pm 1.0}$	86.9 ± 1.9	$\mathbf{66.7 \pm 1.7}$
ResNet18	13.7 ± 0.0	$\mathbf{13.5 \pm 0.0}$	13.7 ± 0.0	$\mathbf{0.0 \pm 0.0}$	30.4 ± 0.0	$\mathbf{22.5 \pm 0.0}$	30.4 ± 0.0	$\mathbf{0.0 \pm 0.0}$
ResNet101	53.1 ± 0.0	$\mathbf{33.5 \pm 0.0}$	53.1 ± 0.0	$\mathbf{0.0 \pm 0.0}$	50.3 ± 0.0	$\mathbf{43.0 \pm 0.0}$	50.3 ± 0.0	$\mathbf{0.0 \pm 0.0}$
DeeplabV3	$\mathbf{0.0 \pm 0.0}$	0.5 ± 0.5	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	20.2 ± 0.1	$\mathbf{6.8 \pm 2.6}$	20.2 ± 0.1	$\mathbf{0.0 \pm 0.0}$

Table 10: The average and minimum prune potential computed on the train and test distribution, respectively, for filter prune methods (FT, PFP). The train distribution hereby consists of nomimal data, while the test distribution consists of the CIFAR10-C, ImageNet-C, VOC-C corruptions.

E Additional Details for Prune Potential with robust training

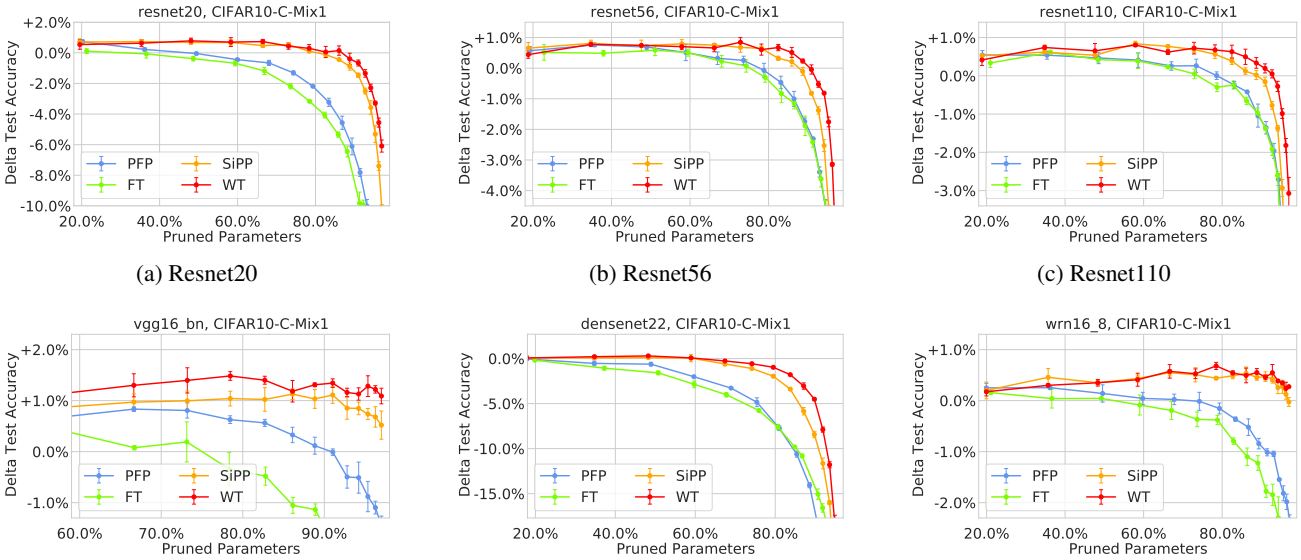
In this section, we consider whether including additional data augmentation techniques derived from the corruptions of CIFAR10-C can boost and/or stabilize the prune potential of a network. Specifically, we incorporate a subset of the corruptions into the training pipeline to train and retrain the pruned network in a robust manner. Below, we list details pertaining to the experimental setup as well as report the results on the conducted experiments.

	Train Distribution	Test Distribution
Nominal	CIFAR10 (no corruption)	CIFAR10.1
Noise	Impulse, Shot	Gauss
Blur	Motion, Zoom	Defocus, Glass
Weather	Snow	Brightness, Fog, Frost
Digital	Contrast, Elastic, Pixel	Jpeg

Table 11: The list of corruptions used for the train and test distribution, respectively, categorized according to type.

E.1 Experimental Setup and Prune Results

To train, prune, and retrain networks we consider the same prune pipeline and experimental setting as described in Section D.2. In addition, we incorporate a subset of the CIFAR10-C corruptions into the training and retraining pipeline by corrupting the training data with the respective corruption technique. That is, when sampling a batch of training data each training image is corrupted with a CIFAR10-C corruption (or no corruption) uniformly at random. The subset of corruptions used as part of the train and test distribution are listed in Table 11. Note that the train and test distribution are mutually exclusive, i.e., they do not share any of the corruptions. However, as shown in Table 11 each category of corruption is used in both the train and test distribution. For each corruption, we choose severity level 3 out of 5 just as before. The nomimal prune-accuracy curves (CIFAR10) for each of the trained and pruned networks are shown in Figure 48.



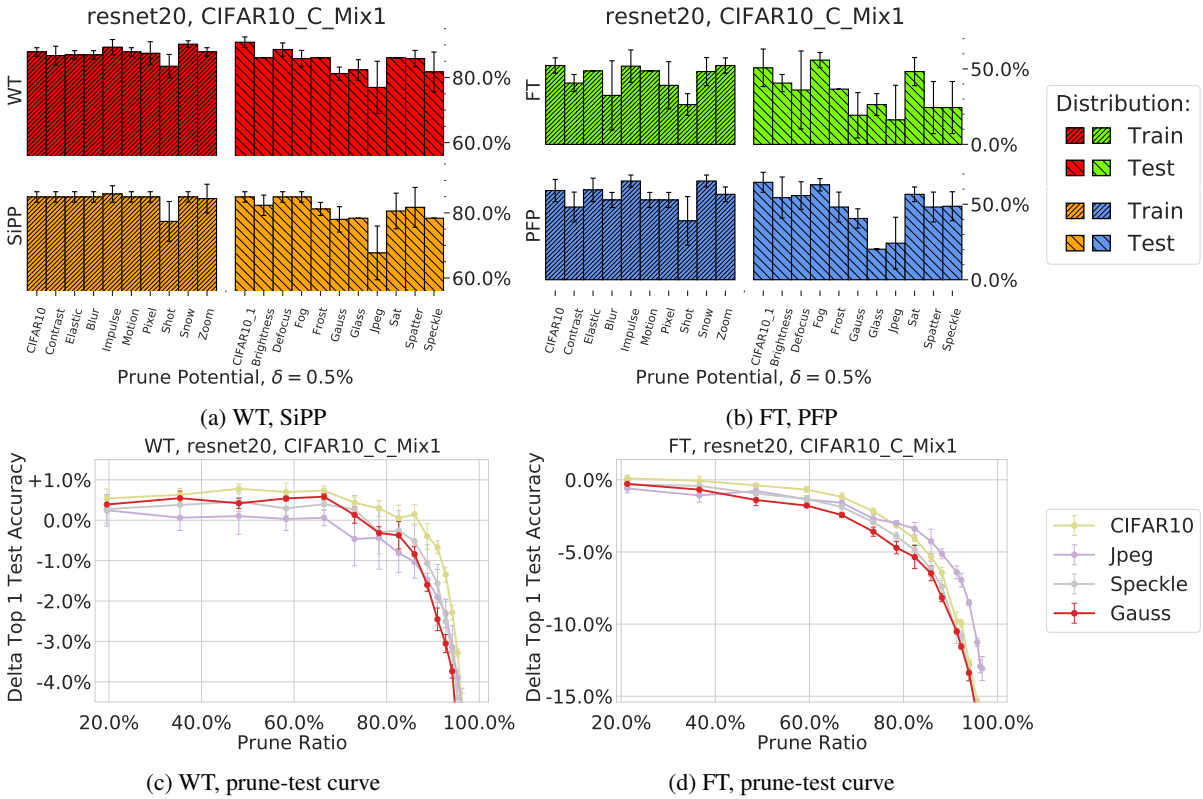
(a) Resnet20 (b) Resnet56 (c) Resnet110

(d) VGG16 (e) DenseNet22 (f) WRN16-8

Figure 48: The difference in test accuracy (nominal CIFAR10) to the uncompressed network.

E.2 Results for Prune Potential

Our results for the prune potential of the network architectures ResNet20, ResNet56, ResNet110, VGG16, DenseNet22, and WRN16-8 are shown in Figures 49, 50, 51, 52, 53, and 54, respectively. We note that overall the prune potential for corruptions from the train distribution can be well preserved since we already included the respective corruptions during training and we can predict the prune potential accurately. However, we can also observe that the prune potential improves for some of the corruptions that were not included during retraining. Despite training in a robust manner, however, the prune potential can still be significantly lower for corruptions from test distribution and/or exhibit high variance (low predictability).



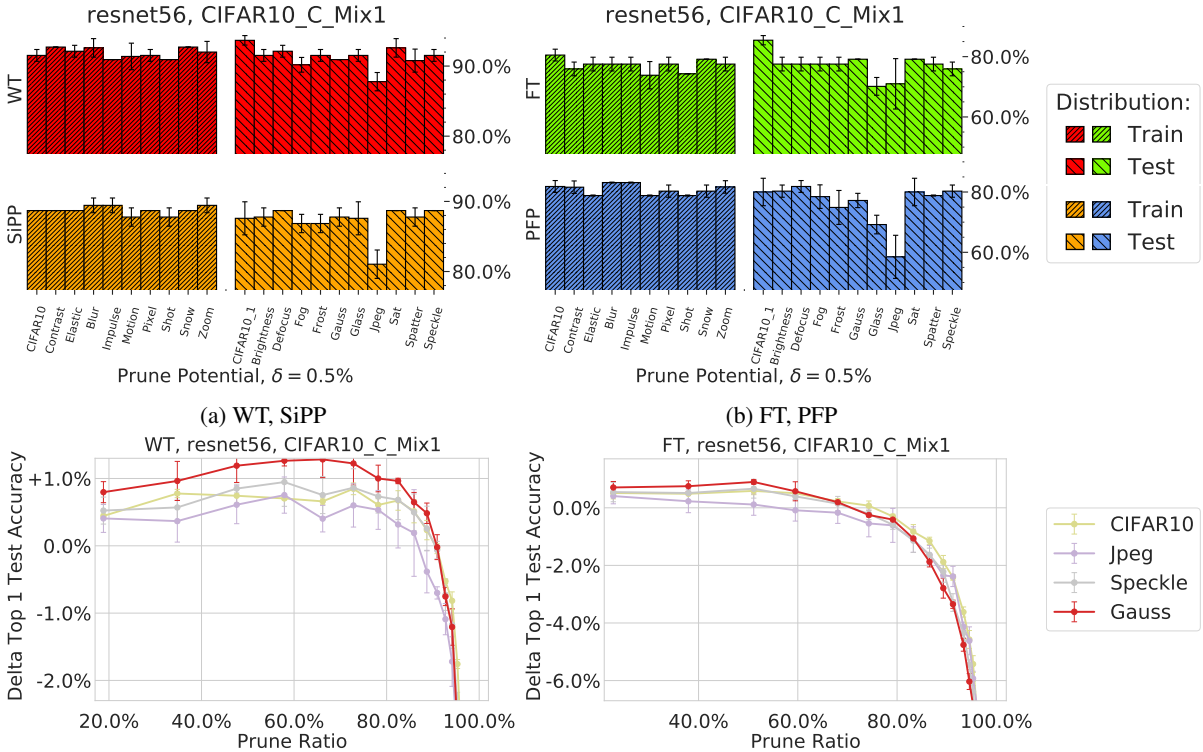
(a) WT, SiPP

(b) FT, PFP

(c) WT, prune-test curve

(d) FT, prune-test curve

Figure 49: The prune potential of a **robustly pruned ResNet20** achievable for **CIFAR10** out-of-distribution data sets.



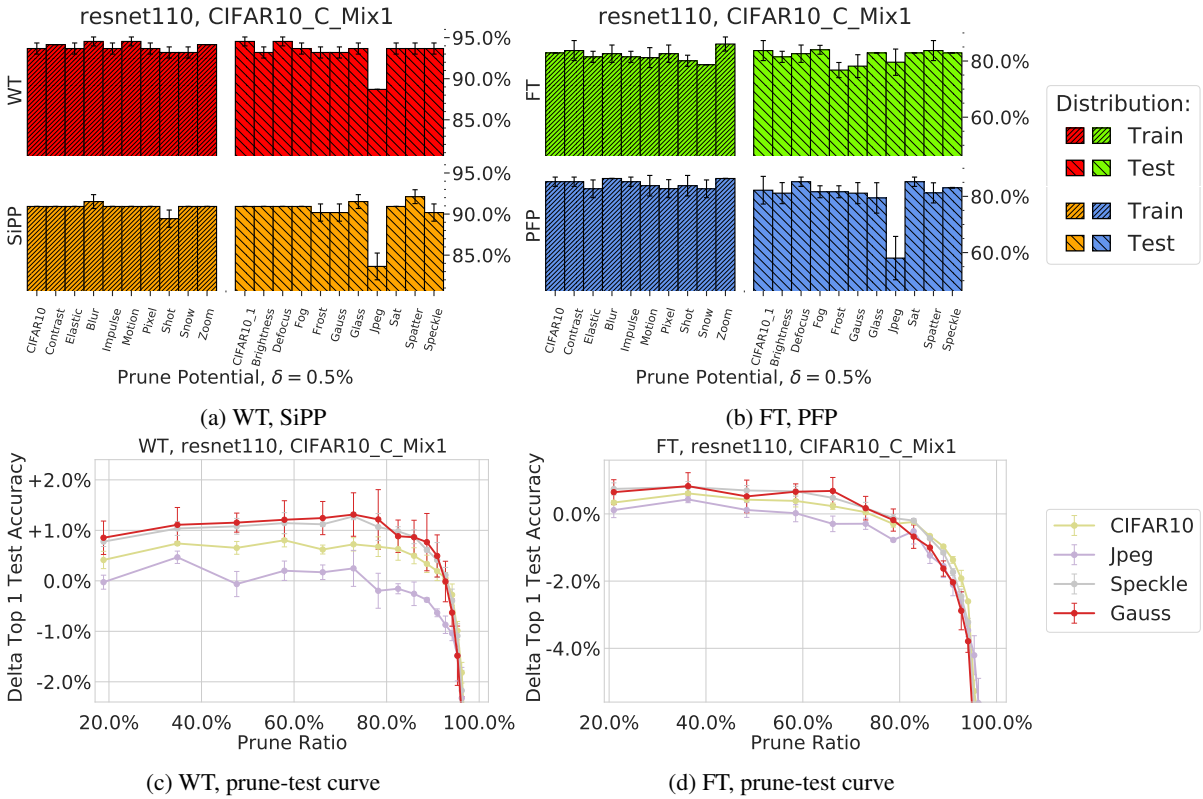
(a) WT, SiPP

(b) FT, PFP

(c) WT, prune-test curve

(d) FT, prune-test curve

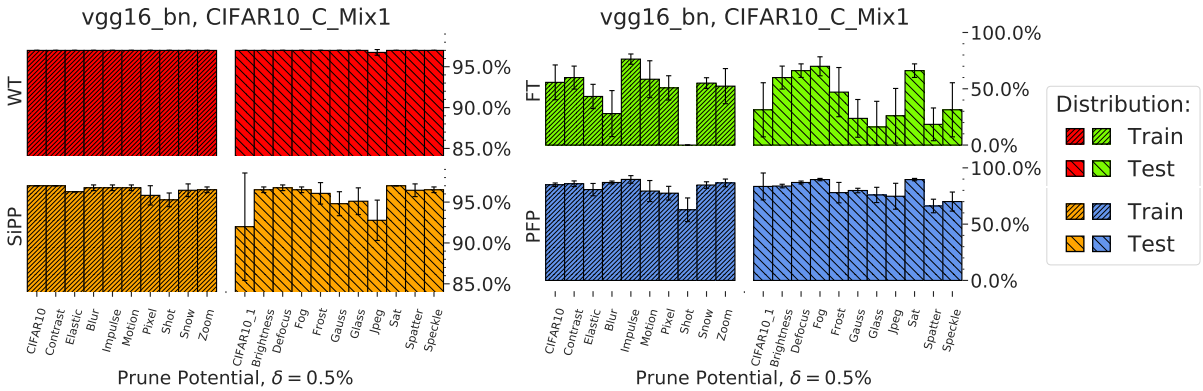
Figure 50: The prune potential of a **robustly pruned ResNet56** achievable for **CIFAR10** out-of-distribution data sets.



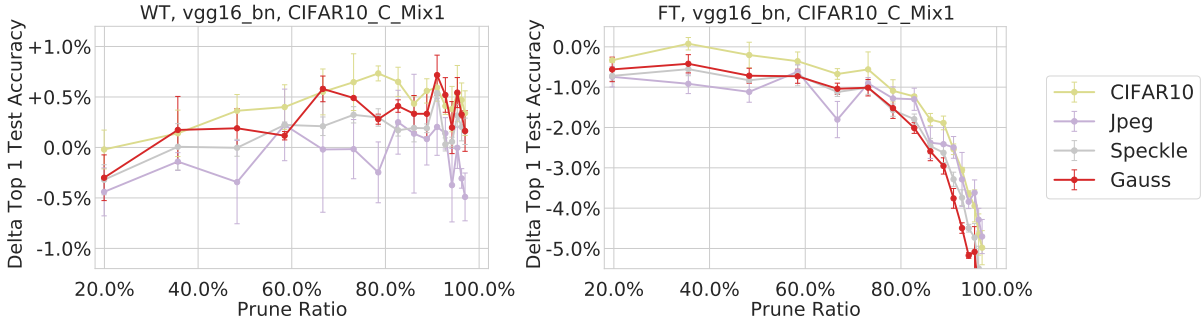
(a) WT, SiPP (b) FT, PFP

(c) WT, prune-test curve (d) FT, prune-test curve

Figure 51: The prune potential of a **robustly pruned ResNet110** achievable for **CIFAR10** out-of-distribution data sets.

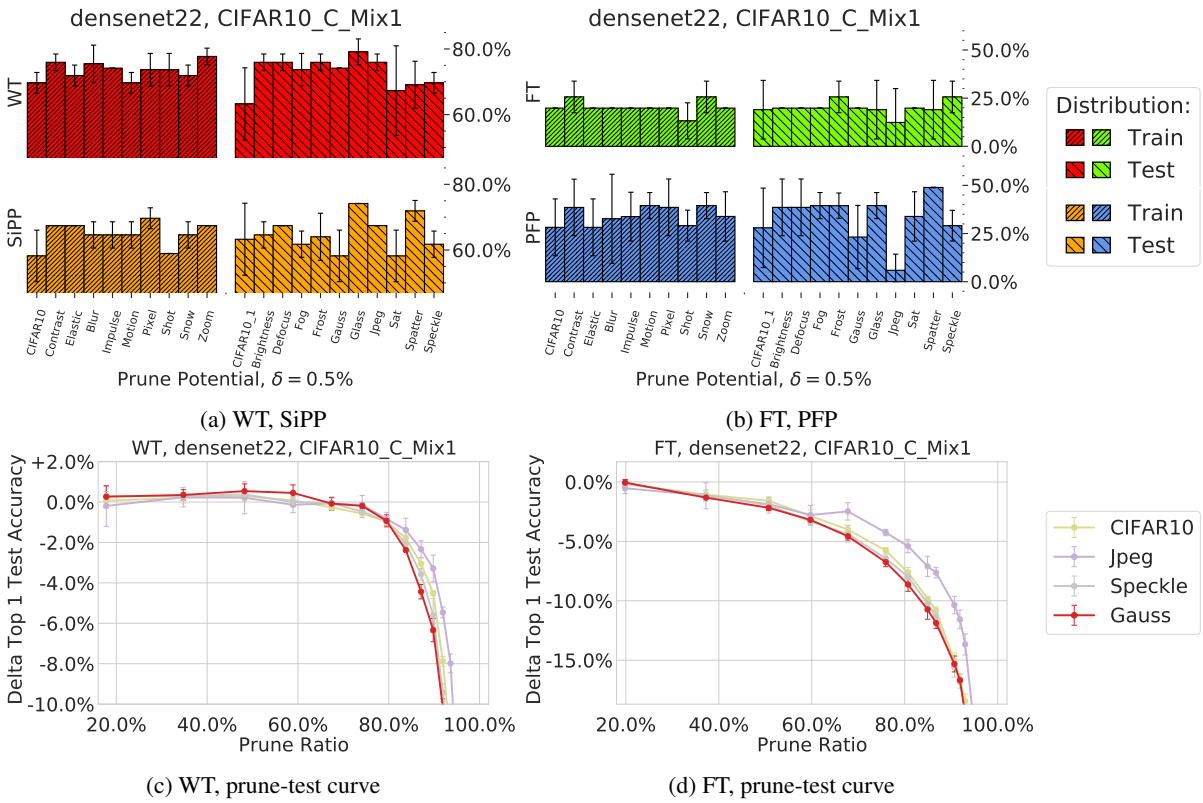


(a) WT, SiPP (b) FT, PFP



(c) WT, prune-test curve (d) FT, prune-test curve

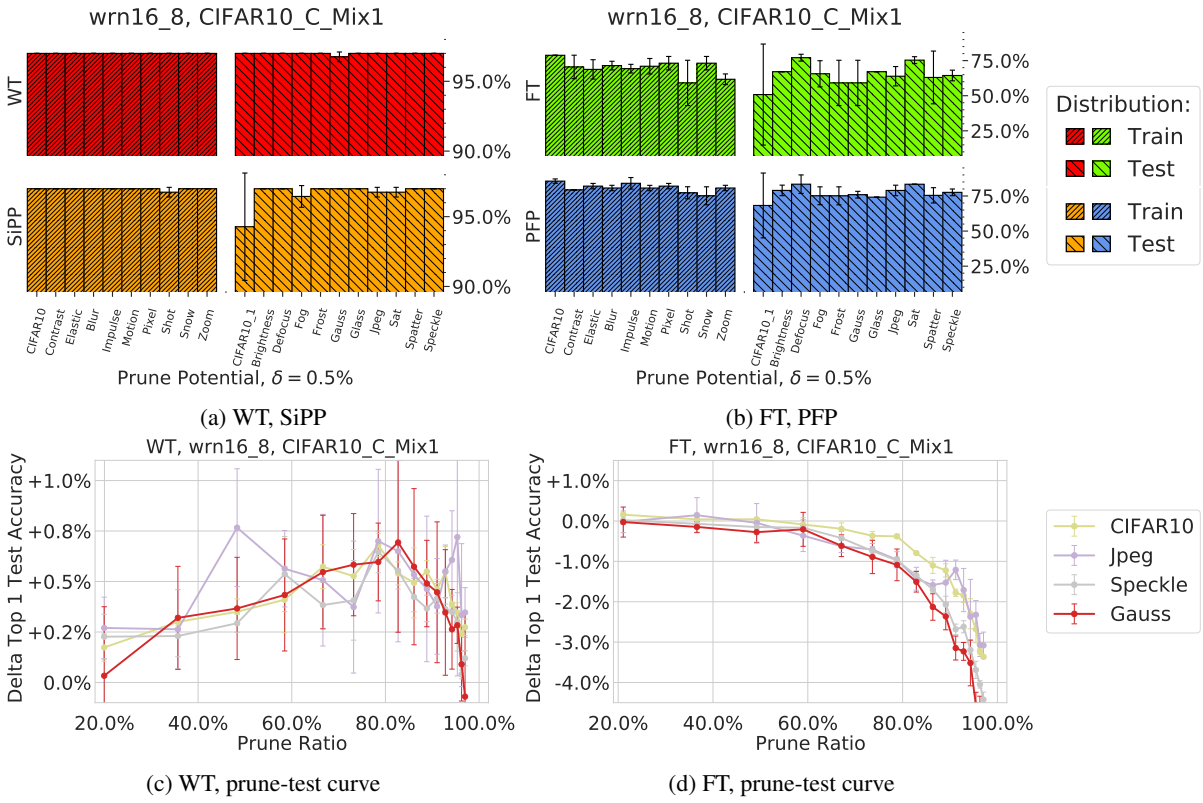
Figure 52: The prune potential of a **robustly pruned VGG16** achievable for **CIFAR10** out-of-distribution data sets.



(a) WT, SiPP (b) FT, PFP

(c) WT, prune-test curve (d) FT, prune-test curve

Figure 53: The prune potential of a **robustly pruned DenseNet22** achievable for **CIFAR10** out-of-distribution data sets.



(a) WT, SiPP (b) FT, PFP

(c) WT, prune-test curve (d) FT, prune-test curve

Figure 54: The prune potential of a **robustly pruned WRN16-8** achievable for **CIFAR10** out-of-distribution data sets.

E.3 Results for Excess Error

Following the approach described in Section D.5 we also evaluated the resulting difference in excess error between pruned and unpruned networks. Our results are shown in Figures 55, 56, 57, 58, 59, and 60 for ResNet20, ResNet56, ResNet110, VGG16, DenseNet22, and WRN16-8, respectively, all of which have been trained in a robust manner. We note that for most networks, except for smaller ones like ResNet20, the correlation between prune ratio and difference in excess error almost disappears. These results encourage the use of robust pruning techniques in order to ensure that pruned networks perform reliably. However, we note that the excess error is computed as an *average* over all corruptions included in the train and test distribution, respectively. Thus, it is not an appropriate measure to estimate whether particular corruptions could still impact the prune potential more significantly than others.

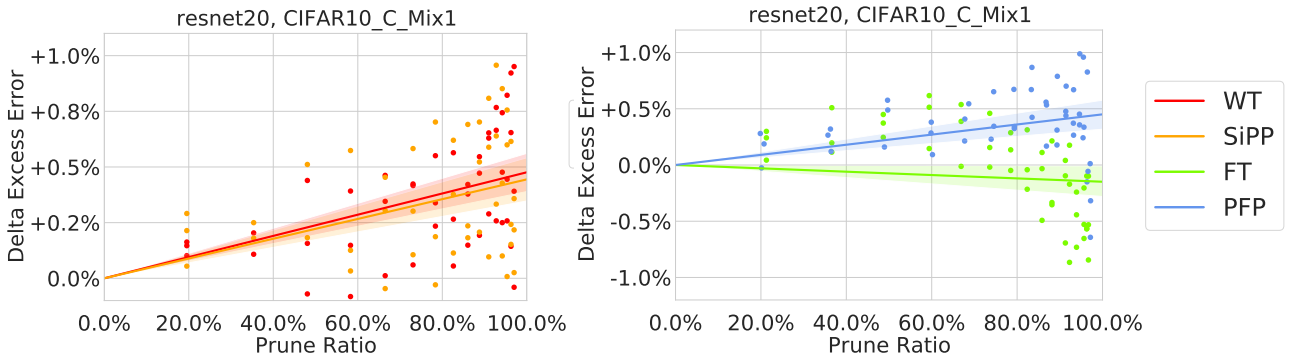


Figure 55: The difference in excess error for a **robustly pruned ResNet20** trained on **CIFAR10**.

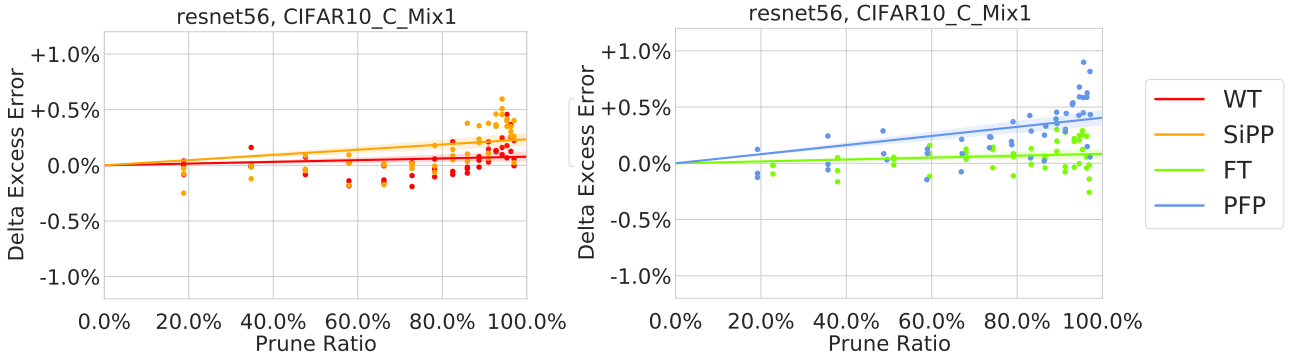


Figure 56: The difference in excess error for a **robustly pruned ResNet56** trained on **CIFAR10**.

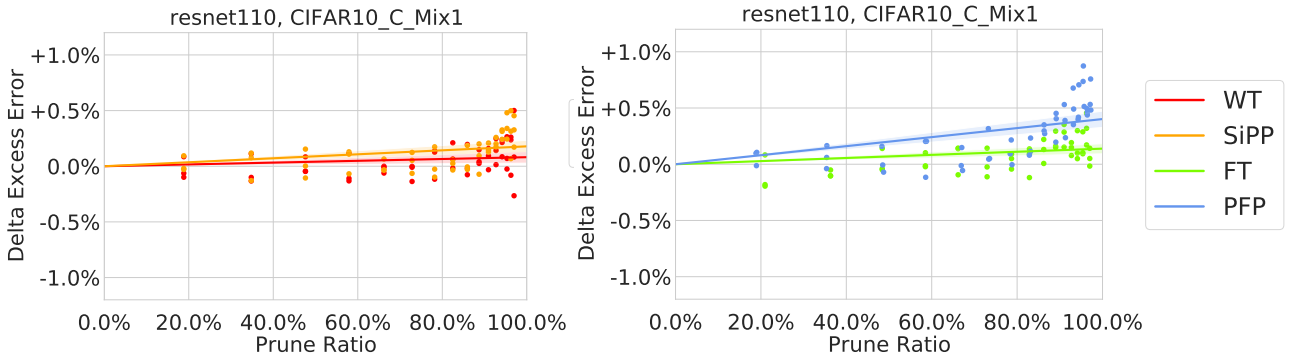


Figure 57: The difference in excess error for a **robustly pruned ResNet110** trained on **CIFAR10**.

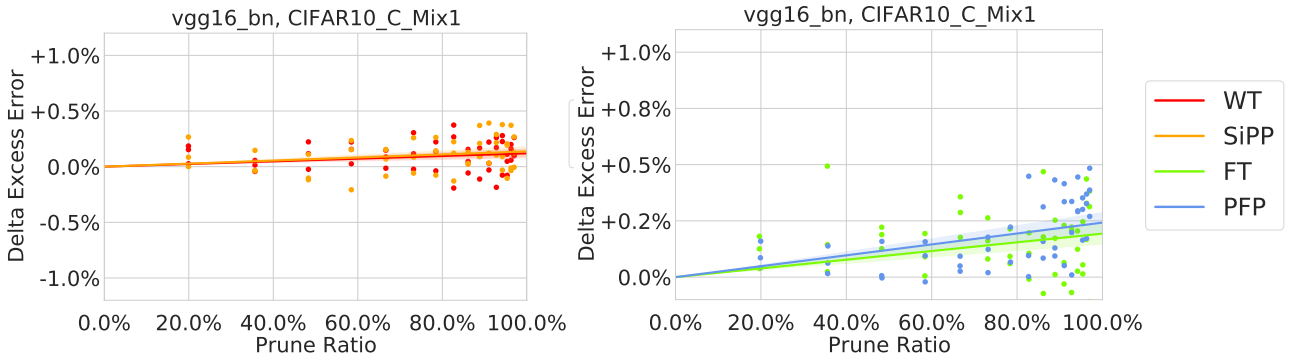


Figure 58: The difference in excess error for a **robustly pruned VGG16** trained on **CIFAR10**.

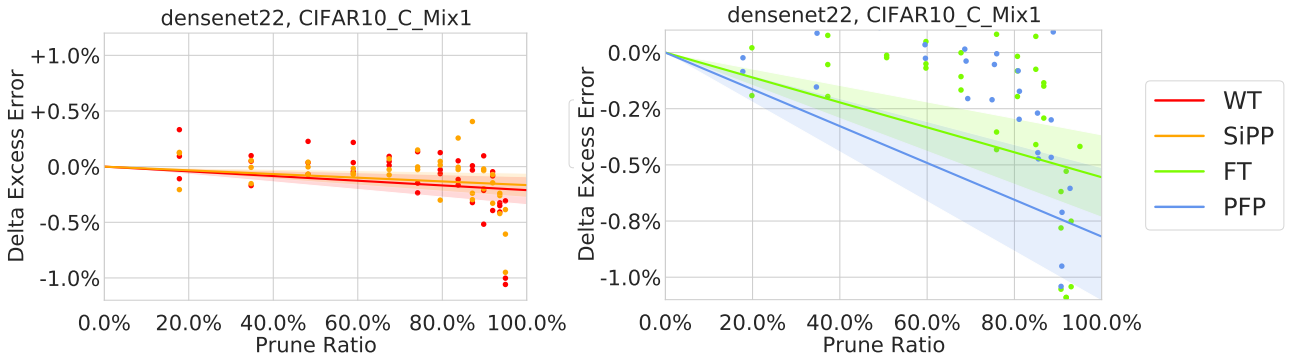


Figure 59: The difference in excess error for a **robustly pruned DenseNet22** trained on **CIFAR10**.

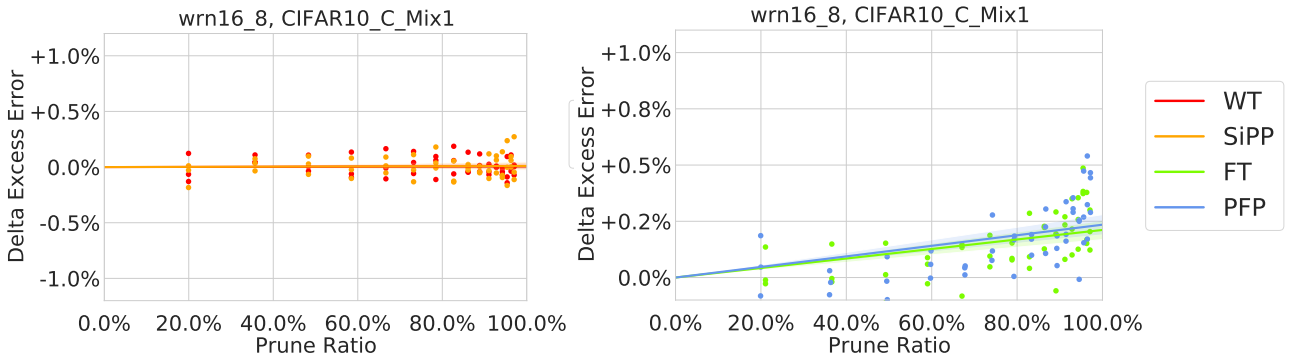


Figure 60: The difference in excess error for a **robustly pruned WRN16-8** trained on **CIFAR10**.

E.4 Results for Overparameterization

Finally, we report the average and minimum prune potential across all networks and prune methods for the train and test distribution, respectively. As highlighted in Section D.6, the prune potential is used to gauge the amount of overparameterization in the network.

The resulting prune potentials are listed in Tables 12 and 13 for weight prune methods (WT, SiPP) and filter prune methods (FT, PFP), respectively. In contrast to Section D.6 the minimum and average prune potential on the train distribution differ since here the train distribution contains multiple corruptions. We note that with robust training the average prune potential remains almost unaffected by changes in the distribution as also apparent from the results in Section E.3. In addition, for most networks even the minimum prune potential on the test distribution is nonzero. These results further encourage the use of robust training techniques when pruning neural networks.

As elaborated upon in Section 6, we observe that we can regain much of the prune potential by *explicitly regularizing* the pruned network during retraining. In other words, the amount of overparameterization is not only a function of the data set and network architecture, but of the training procedure as well.

However, we note that these observations hinge upon the particular choice of the train and test distribution, which share certain commonalities in this case. Potentially, it might be possible to construct test distributions that differ significantly from the train distribution, in which case pruned networks might suffer disproportionally more from the distribution change compared to unpruned networks. These results would then be analogous to the ones without robust training presented in Section D.

Model	WT - Prune Potential (%)				SiPP - Prune Potential (%)			
	Average		Minimum		Average		Minimum	
	Train Dist.	Test Dist.	Train Dist.	Test Dist.	Train Dist.	Test Dist.	Train Dist.	Test Dist.
ResNet20	87.5 ± 1.9	$\mathbf{84.7 \pm 0.9}$	83.5 ± 3.6	$\mathbf{72.6 \pm 4.9}$	84.2 ± 1.7	$\mathbf{80.3 \pm 1.6}$	77.4 ± 6.1	$\mathbf{65.9 \pm 6.1}$
ResNet56	91.8 ± 0.5	$\mathbf{91.3 \pm 0.4}$	90.2 ± 1.0	$\mathbf{87.8 \pm 1.3}$	88.7 ± 0.5	$\mathbf{87.2 \pm 0.1}$	87.8 ± 1.3	$\mathbf{81.0 \pm 2.0}$
ResNet110	93.8 ± 0.3	$\mathbf{93.2 \pm 0.4}$	92.7 ± 0.0	$\mathbf{88.7 \pm 0.0}$	90.8 ± 0.1	$\mathbf{90.2 \pm 0.1}$	89.4 ± 1.0	$\mathbf{83.6 \pm 1.6}$
VGG16	97.0 ± 0.0	$\mathbf{97.0 \pm 0.0}$	97.0 ± 0.0	$\mathbf{96.8 \pm 0.3}$	96.5 ± 0.1	$\mathbf{95.5 \pm 0.5}$	94.6 ± 0.5	$\mathbf{88.3 \pm 3.9}$
DenseNet22	73.4 ± 1.1	$\mathbf{72.7 \pm 1.9}$	67.4 ± 0.0	$\mathbf{51.8 \pm 5.0}$	$\mathbf{64.7 \pm 1.3}$	64.8 ± 0.6	55.4 ± 5.0	$\mathbf{51.8 \pm 5.0}$
WRN16-8	97.0 ± 0.0	$\mathbf{97.0 \pm 0.0}$	97.0 ± 0.0	$\mathbf{96.8 \pm 0.3}$	97.0 ± 0.0	$\mathbf{96.7 \pm 0.5}$	96.8 ± 0.3	$\mathbf{94.3 \pm 3.8}$

Table 12: The average and minimum prune potential computed on the train and test distribution, respectively, for weight prune methods (WT, SiPP). The train and test distribution hereby each consist of a mutually exclusive subset of corruptions as listed in Table 11.

Model	FT - Prune Potential (%)				PFP - Prune Potential (%)			
	Average		Minimum		Average		Minimum	
	Train Dist.	Test Dist.	Train Dist.	Test Dist.	Train Dist.	Test Dist.	Train Dist.	Test Dist.
ResNet20	44.0 ± 6.3	$\mathbf{34.4 \pm 6.1}$	19.3 ± 15.0	$\mathbf{7.1 \pm 10.0}$	55.2 ± 6.2	$\mathbf{47.7 \pm 6.4}$	39.0 ± 16.1	$\mathbf{13.3 \pm 9.4}$
ResNet56	77.1 ± 0.4	$\mathbf{77.1 \pm 0.9}$	72.2 ± 2.9	$\mathbf{67.3 \pm 6.1}$	80.8 ± 0.4	$\mathbf{76.3 \pm 0.5}$	78.8 ± 0.2	$\mathbf{58.5 \pm 7.1}$
ResNet110	82.1 ± 1.8	$\mathbf{81.7 \pm 0.8}$	78.7 ± 0.0	$\mathbf{74.9 \pm 2.7}$	84.4 ± 2.1	$\mathbf{80.0 \pm 2.2}$	81.6 ± 2.1	$\mathbf{58.0 \pm 7.7}$
VGG16	48.0 ± 3.3	$\mathbf{41.5 \pm 5.2}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	81.9 ± 1.8	$\mathbf{79.8 \pm 2.3}$	$\mathbf{62.7 \pm 10.5}$	63.9 ± 3.9
DenseNet22	20.3 ± 0.7	$\mathbf{20.0 \pm 5.7}$	13.2 ± 9.4	$\mathbf{6.6 \pm 9.4}$	34.2 ± 8.8	$\mathbf{33.1 \pm 6.5}$	17.5 ± 14.2	$\mathbf{0.0 \pm 0.0}$
WRN16-8	69.7 ± 4.1	$\mathbf{64.8 \pm 2.2}$	54.2 ± 12.9	$\mathbf{24.4 \pm 17.3}$	80.7 ± 0.7	$\mathbf{76.9 \pm 2.9}$	72.1 ± 3.1	$\mathbf{57.2 \pm 14.9}$

Table 13: The average and minimum prune potential computed on the train and test distribution, respectively, for filter prune methods (FT, PFP). The train and test distribution hereby each consist of a mutually exclusive subset of corruptions as listed in Table 11.