Improving Robustness by Enhancing Weak Subnets

Yong Guo, David Stutz, and Bernt Schiele

Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken
{yongguo,david.stutz,schiele}@mpi-inf.mpg.de

Abstract. Despite their success, deep networks have been shown to be highly susceptible to perturbations, often causing significant drops in accuracy. In this paper, we investigate model robustness on perturbed inputs by studying the performance of internal sub-networks (subnets). Interestingly, we observe that most subnets show particularly poor robustness against perturbations. More importantly, these weak subnets are correlated with the overall lack of robustness. Tackling this phenomenon, we propose a new training procedure that **identifies and enhances weak subnets (EWS) to improve robustness**. Specifically, we develop a search algorithm to find particularly weak subnets and explicitly strengthen them via knowledge distillation from the full network. We show that EWS greatly improves both robustness against corrupted images as well as accuracy on clean data. Being complementary to popular data augmentation methods, EWS consistently improves robustness when combined with these approaches. To highlight the flexibility of our approach, we combine EWS also with popular adversarial training methods resulting in improved adversarial robustness.

Keywords: Model robustness, training method, sub-networks.

1 Introduction

Since 2012, when AlexNet won the first place in the ImageNet competition [46], deep (convolutional) networks [48] have been producing state-of-the-art results in many challenging tasks [35,49]. Recent work, however, highlights how brittle these models are when applied to images with simple corruptions, such as noise, blur, and pixelation [38]. In fact, these corruptions cannot fool the human vision system but often severely hamper the accuracy of deep networks [38,37]. Among an increasing body of work on developing robust models, data augmentation is particularly popular and effective. For example, AutoAugment [21], AugMix [40] or DeepAugment [37] improve the robustness against corrupted examples alongside the clean accuracy.

Complementary to this line of research, we study the robustness of nowadays overparameterized networks by analyzing their internal sub-networks (subnets). While it is well-known that few well-performing subnets, i.e., "winning tickets", exist within these large networks [27,73,50], the role of the remaining subnets in terms of robustness remains unexplored. In this paper, we find that most of these subnets perform rather poorly. Moreover, this is strongly correlated with the network's overall performance and becomes particularly apparent on corrupted or adversarial examples. As illustrated in Fig. 1, for a standard ResNet-50 [35] on ImageNet [67], the network's accuracy

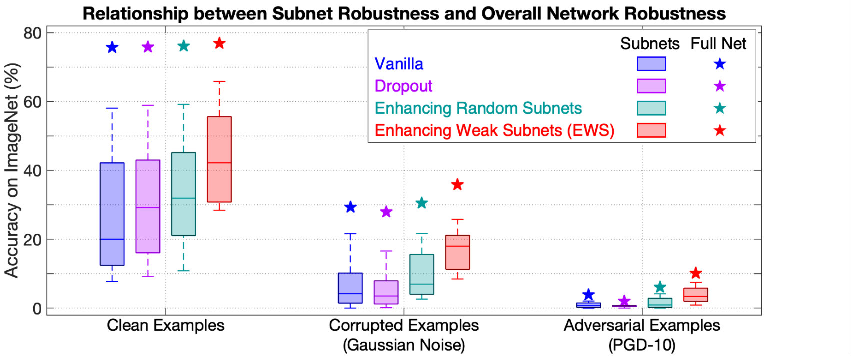

Fig. 1: On ImageNet, we plot accuracies on clean and perturbed examples for a standard ResNet-50 (blue stars) and 1K randomly sampled subnets (blue box plot) corresponding to 70% of the paths/channels each. On clean examples (left), subnets perform significantly worse than the overall network, with average accuracy reducing to roughly 20%. On examples perturbed with Gaussian (middle) or adversarial (right) noise, overall accuracy reduces severely alongside subnet accuracy, suggesting that *weak subnets* are responsible for this lack of robustness. By identifying and enhancing these weak subnets (EWS, red), we can improve robustness significantly. Importantly, we search for particularly weak subnets in order to enhance them through knowledge distillation, while encouraging distributed representations through dropout [76] (violet) or improving *random* subnets (green) does not improve accuracy or robustness as much.

(blur star) reduces significantly alongside the severely degraded subnets' accuracy (blue box) when facing perturbed examples, e.g., with Gaussian noise. Note that the blue box illustrates the performance range of 1K randomly sampled subnets, each corresponding to 70% of the overall network, and the mean performance (blue line in box) is at the lower end of this range. Overall, this leads us to the hypothesis that these *weak subnets* are, to some degree, responsible for the lack of robustness of the overall network.

In order to test this hypothesis and improve robustness, we intend to enhance subnet performance. Interestingly, dropout [76] can be regarded as a particular approach to construct and train subnets by randomly dropping the internal connections. Unfortunately, as shown in Fig. 1, it only slightly improves the subnets on clean data but yields much worse performance on perturbed data. It is worth noting that, the correlation between subnet and full network still holds, i.e., better subnets (blue boxes > violet boxes) always come along with better full network (blue stars > violet stars) on perturbed data (middle and right of Fig. 1). Thus, regarding this correlation, how to effectively improve subnets becomes an important problem to boost the overall robustness against perturbations.

Contributions: To address this, we propose to directly and very explicitly identify and improve weak subnets instead of the random ones, which is proved to be a more effective way (see Fig. 1 and Table 5). Specifically, we make three key contributions: *(1)* We propose a novel robust training method which identifies and **enhances weak subnets (EWS)** to improve the overall robustness of the full network. *(2)* To this end, we develop a search algorithm that obtains weak subnets by identifying particularly weak paths/channels inside the full network. Given a weak subnet, its performance is further enhanced by distilling knowledge from the full network. This approach is not only very scalable, it also adds negligible computational overhead (see results in Section 5.1). Note that identifying particularly weak subnets is crucial as shown in Fig. 1 where enhancing *random* subnets has little effect (red stars and boxes > green stars and boxes). *(3)* In experiments, we apply EWS on top of state-of-the-art data augmentation schemes to improve accuracy and corruption robustness on CIFAR-10/100-C and ImageNet-C [37]. Moreover, we also demonstrate the generality of our approach for improving adversarial robustness on top of recent adversarial training methods. Importantly, our approach is complementary to all these methods and improves consistently across a wide range of approaches. Our code is available at https://github.com/guoyongcs/EWS.

2 Related Work

Despite their outstanding performance, deep networks are not robust to image corruptions [37]. To address this, recent work explores re-calibrating batch normalization statistics [71,6,59], utilizing the frequency domain [68], or using vision transformers [57] to improve corruption robustness. However, data augmentation methods such as [66,28,21,40,37,11,69] represent the most prominent and successful line of work, ranging from simple Gaussian noise augmentation [66], over well-known schemes such as AutoAugment [21] to strategies specifically targeted towards corruption robustness such as AugMix [40] or DeepAugment [37].

Besides random corruptions, deep networks are susceptible to adversarial examples [79,29]. While plenty of approaches for defending against adversarial examples have been proposed [74,5,86,1,7,84,17,64,60,31], adversarial training (AT) has become the de facto standard [56]. This is also because other methods have repeatedly been "broken" using stronger or adaptive attacks, e.g., see [14,13,15,54,26,58,3,12,20]. Thus, recent work concentrates on various variants of AT: [90] adds an additional Kullback-Leibler term, [39] incorporates a complementary self-supervised loss, [2,16] use additional unlabeled examples. Many more variants exploring, e.g., instance-adaptive threat models [4,25], additional regularizers [88,63,61,61,75,80,8,51], curriculum training [10,85], weight perturbations [82,77], among many others [47,81,19,91,92,43,93], have been proposed. Moreover, AT has also been applied to corruption robustness [44,55].

Complementary to the above lines of research, existing work [50,72,24] have shown the existence of particularly strong/robust subnets inside a large model. However, the impact of the remaining weak subnets on the overall robustness has not been investigated. Interestingly, as shown by Fig. 1, we observe that the majority of subnets are particularly weak and there is a clear correlation between the performance of subnets and overall robustness. This motivates us to explicitly enhance these weak subnets. To achieve this goal, our method is also inspired by recent work on neural architecture search (NAS) [95,52,9,18,34] and knowledge distillation (KD) [41,94]. Unlike existing NAS methods, we focus on improving the robustness of deep models and exploit NAS techniques to find weak subnets which may hamper the overall performance. Once we find these weak subnets, we seek to further enhance them using a KD loss, similar to learning lightweight student models in model compression [78,53,30].

3 EWS: Training by Enhancing Weak Subnets

This paper studies the robustness of deep, over-parameterized networks to image perturbations through the lens of subnets defined by a subselection of internal blocks/channels. After arguing that the robustness of subnets has a large impact on overall performance in Section 3.1, we seek to improve robustness through a novel training procedure: identifying and enhancing weak subnets through knowledge distillation, as summarized in Fig. 3. Specifically, in Section 3.2, we first discuss the construction and search of particularly weak subnets along with the motivation for strengthening them. Here, we present an effective and scalable search algorithm to find weak subnets. In Section 3.3, we then integrate the found weak subnets into our training procedure to explicitly enhance them. Our procedure is summarized in Algorithm 1. In this paper, we mainly concentrate on improving robustness to corrupted images, e.g., with noise or blur, on standard benchmarks such as CIFAR-10-C and ImageNet-C [37]. Besides, we also highlight the flexibility of our method by applying it on top of adversarial training [56].

3.1 Subnet Construction and Impact on Overall Performance

We intend to investigate the robustness of deep networks from the perspective of subnets. To this end, we define subnets as follows: Given a full (convolutional feedforward) network M, we construct a set of subnets $\alpha \in \Omega$ where Ω denotes the space of all the possible subnets of M. In this paper, we particularly consider deep models that consist of a stack of basic blocks, e.g., ResNets [35]. Actually, our approach can easily be adapted to any other architectures since most popular designs (e.g., MobileNet [42,70] and ResNeXt [83]) can be also decomposed into the basic units defined in Fig. 2. Without loss of generality, we allow each basic block i to have a specific number of paths n_i and each layer j in a block to have a specific number of channels c_j. For example, a basic residual block [35] contains $n_i = 2$ paths (i.e., the residual path and the skip connection). As shown in Fig. 2, we construct subnets by selecting a subset of paths and channels in each block and layer, respectively. Given a subnet width $\rho \in (0, 1)$, we select roughly $\rho \cdot n_i$ paths in each block i and $\rho \cdot c_j$ channels for each layer j. For example, $\rho = 0.7$ denotes selecting 70% of paths and channels.

We further investigate the impact of subnets on the overall performance. In Fig. 1, we quantify the robustness of such subnets (ρ=0.7) on corrupted and adversarial examples: Even on *clean* examples (left) where the full network tends to make correct predictions with relatively high accuracy (blue star), more than 50% of the subnets yield an accuracy below 20% (blue box). On corrupted or adversarial examples (middle and right), this phenomenon is further emphasized. More critically, as the performance of subnets deteriorates on such perturbed examples, the network's overall performance also reduces significantly. It is worth noting that, each subnet can be regarded as an indicator to show whether the corresponding parameters inside the full model are well learned or not. In this sense, one may easily improve the overall performance if we put more focus on these "weak" parameters during training. This motivates us to explicitly find and enhance these weak subnets. However, we emphasize that merely training random subnets, e.g., dropout [76] (violet in Fig. 1), does *not* contribute to the improved robustness.

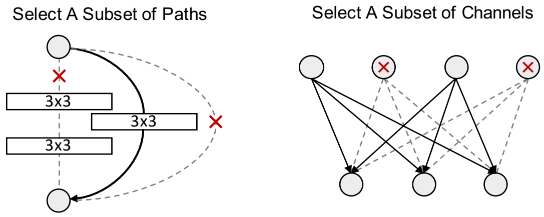

Fig. 2: We construct subnets by selecting a subset of paths (left) and channels (right) in each block of the network. Left: An example for a basic block with multiple paths where only one path is kept (in **bold**). Right: In the convolutional (fully connected) layers of each block, we select a subset of channels (in **bold**). This approach is applied block-by-block, layer-by-layer using a fixed fraction of paths/channels to be selected.

3.2 Finding Particularly Weak Subnets

In order to improve subnet performance and following Fig. 3, our proposed EWS has two core components: finding particular weak subnets and strengthening their performance. Regarding the first component (Fig. 3, left), given a search space Ω, the easiest way to identify/enhance such subnets is by random sampling, i.e., randomly selecting blocks and channels for a given ρ. However, Fig. 1 (green) shows that improving the performance of random subnets during training (as described in detail in Section 3.3) does not result in significantly more robust subnets (box plot) or improved overall performance (star). Instead, we propose a novel search algorithm that finds particularly poor subnets α, as evaluated based on their classification accuracy $R(\alpha)$. Following [62,33], the accuracy can be approximated based on a batch of sampled data in each iteration. To be specific, we learn a policy π_θ, parameterized by θ, and use it to generate candidates of weak subnets, i.e., $\alpha \sim \pi_\theta$, as identified based on particularly low accuracy. To learn the policy, we build a controller model to produce candidate subnets by minimizing the accuracy $R(\alpha)$ in expectation. Formally, we solve the following optimization problem

$$\min_{\theta} \mathbb{E}_{\alpha \sim \pi_\theta} \left[R(\alpha) \right]. \tag{1}$$

The controller's parameters can be updated using policy gradient based on a mini-batch of sampled subnets (see the supplementary material for details). This is made explicit in the first part (Lines 3-7) of Algorithm 1.

Architecture of the Controller Model. Since a network like ResNet can be represented by a series of tokens [62], the subnet generation task can be viewed as a sequential decision making problem. Following [62,33], we build an LSTM-based [32] controller model to learn the policy π_θ. Specifically, the controller takes an initial hidden state (kept constant during training) as input and sequentially predicts the selected paths/channels for each block/layer, as illustrated exemplarily in Fig. 3 (left). Specifically, for each block, we first select a subset of paths and subsequently select a subset of channels for each layer in the selected paths. We emphasize that the controller model scales linearly with the overall network and is easily applicable to a wide range of architectures. Please refer to our supplementary for further details.

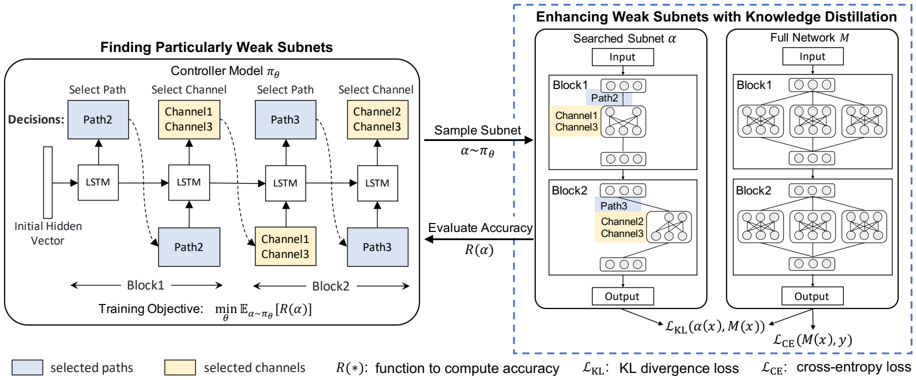

Fig. 3: Overview of our proposed **enhancing weak subnets (EWS)** training procedure. During training, we alternatingly perform subnet search based on a controller model (left) and train the full network with an additional distillation loss (right). As illustrated, the controller is trained using policy gradient every K iterations, while the full network can be any state-of-the-art network such as ResNets [35]. The distillation loss enforces predictions of the full network and the found weak subnet to be similar to improve subnet performance. We refer to Algorithm 1 for a detailed description.

3.3 EWS: Enhancing Weak Subnets with Knowledge Distillation

In the second step of EWS, corresponding to Fig. 3 (right) and based on the found weak subnets $\alpha \sim \pi_\theta$, we use a distillation loss to enhance subnet performance and thereby aim to improve the overall robustness of the full network. Let x be a training example with its label y. Besides the standard cross-entropy (CE) loss, we introduce an additional Kullback-Leibler (KL) divergence loss between the full network's predictions $M(x)$ and the subnet's predictions $\alpha(x)$. This can be thought of as distilling knowledge from the full network into the subnet and is meant to enhance the selected particularly weak subnets. Overall, the loss function to be minimized becomes

$$\mathcal{L}(x,y) = \mathcal{L}_{\text{CE}}\big(M(x),y\big) + \underbrace{\lambda\mathcal{L}_{\text{KL}}\big(\alpha(x),M(x)\big)}_{\text{enhance weak subnet}}, \tag{2}$$

where λ is a trade-off parameter determining the importance of enhancing weak subnets (see impact of λ in Section 5.1). While other losses for improving weak subnets are possible, e.g., a CE loss with the true labels $\mathcal{L}_{\text{CE}}\big(\alpha(x),y\big)$, we found that distillation with the KL divergence works best in practice (see supplementary for details). We emphasize that our method is very flexible in that it can easily be combined with different losses and is entirely complementary to data augmentation approaches.

As indicated in Algorithm 1, it is worth noting that, once we update the model parameters, the previously learned controller π_θ is outdated. This is problematic as the sampled subnets $\alpha \sim \pi_\theta$ might not be particularly weak anymore. Therefore, the model and controller are updated in an alternating fashion. However, we found that it is not necessary to update the controller model in each iteration. Instead, we only update the

Algorithm 1 Training by **enhancing weak subnets (EWS)**: We alternate between updating the controller model π_θ (every K iterations) and updating the model M. When training M, we sample a subnet $\alpha\sim\pi_\theta$ and exploit a distillation loss to enhance it.

```
Require: Training data $\mathcal{D}$, batch size for training model $N$, batch size for training controller $C$,
    hyper-parameter $\lambda$, training interval $K$, number of iterations $T$
for $t = 1, \cdots, T$ do
    Sample a batch of examples $\mathbb{B} = \{(x_i, y_i)\}_{i=1}^{N}$ from $\mathcal{D}$
    // Update the controller every $K$ iterations
    if $t \bmod K = 0$ then
        Sample a set of subnets $\{\alpha_i\}_{i=1}^{C}$ from $\pi_\theta$
        Compute subnet accuracy $R(\alpha)$ on $\mathbb{B}$
        Update $\theta$ using policy gradient:
            $\theta \leftarrow \theta + \eta \frac{1}{C} \sum_{i=1}^{C} \nabla_\theta \log \pi_\theta(\alpha_i) R(\alpha_i)$
    end if
    // Train the full model while enhancing weak subnets
    Sample a weak subnet $\alpha \sim \pi_\theta$
    Update $w$ using gradient descent:
        $w \leftarrow w - \eta \frac{1}{N} \sum_{i=1}^{N} \big(\nabla_w \mathcal{L}_{\text{CE}}(M(x_i), y_i) + \lambda \nabla_w \mathcal{L}_{\text{KL}}(\alpha(x_i), M(x_i))\big)$
end for
```

controller every K iterations where $K = 1$ represents fully alternating training, while $K \gg 1$ updates the controller more rarely. This also reduces the computational overhead of our methods significantly. As discussed in detail in Section 5.1, we find that $K = 10$ works very well in practice and leads to a negligible computational overhead.

3.4 Combining EWS with Adversarial Training

We further demonstrate the flexibility of EWS by combining it with adversarial training to improve adversarial robustness. Instead of identifying weak subnets by accuracy, we now consider *robust* accuracy on adversarial examples. Similarly, we enhance these subnets using a distillation loss on adversarial examples instead of the clean ones.

Searching for *Adversarially* Weak Subnets. Here, we seek to find subnets that are vulnerable to adversarial examples. We follow [56] and compute adversarial L_∞ perturbations by maximizing cross-entropy loss. Specifically, we construct the perturbed data as $x' = \arg\max_{\|x-x'\|_p \leq \epsilon} \mathcal{L}_{\text{CE}}(M(x'), y)$. While other objectives are possible, this is commonly achieved using projected gradient ascent (PGD), where the projection reduces to a simple clamping operation for ensuring $\|x - x'\| \leq \epsilon$. Finally, to find adversarially vulnerable subnets, we replace the (clean) accuracy in Equation (1) with the adversarial accuracy. We also note that the adversarial examples are computed with respect to the full model M and re-computed every iteration. Again, we follow Algorithm 1 and train the controller using policy gradient.

Adversarially Enhancing Weak Subnets. With the found weak subnets, we again enhance them to improve the overall adversarial robustness. Following the vanilla adversarial training [56], we optimize the cross-entropy loss on adversarial examples:

$$\min_{w} \; \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\mathcal{L}(x', y)\right] \quad \text{where } x' = \operatorname*{arg\,max}_{\|x-x'\|_p \leq \epsilon} \; \mathcal{L}_{\text{CE}}(M(x'), y). \tag{3}$$

Again, we additionally introduce a distillation-based KL divergence loss to enhance the subnets, following Section 3.3 above. Formally, the loss in Equation (3) becomes

$$\mathcal{L}(x', y) = \mathcal{L}_{\text{CE}}(M(x'), y) + \underbrace{\lambda \mathcal{L}_{\text{KL}}\big(\alpha(x'), M(x')\big)}_{\text{enhance subnet on x}'} \tag{4}$$

Note that the KL divergence is computed on the predictions $M(x')$ and $\alpha(x')$ based on the adversarial examples x'. As before, λ determines the importance of weak subnet performance. Even though this formulation follows vanilla adversarial training, our approach generalizes easily to other variants, including TRADES [89] where an additional loss on clean examples is minimized. In such cases, the KL divergence used for distillation is also computed on clean examples (see supplementary for more details). Additionally applying weight perturbations [82] or using additional unlabeled examples [16] is possible as well (see Table 4). While Fig. 1 only considers a model trained without adversarial examples, i.e., following Section 3.3, enhancing subnets during training clearly has a positive impact on robustness against adversarial examples (e.g., PGD ones as computed in [56]). Thus, we expect a similarly positive impact when explicitly finding and enhancing adversarially vulnerable subnets during adversarial training.

4 Experiments

In the following, we demonstrate that EWS allows to train more accurate and robust models, as tested mainly against corrupted examples in Section 4.1. Besides the standard training settings, we emphasize that EWS can be successfully used in a complementary fashion on top of popular data augmentation schemes, including AutoAugment [21], AugMix [40] or DeepAugment [37]. Moreover, we also show that EWS easily generalizes to adversarial training without significant modifications and greatly improves the adversarial robustness in Section 4.2. We provide additional ablations in Section 5.

4.1 Improving Corruption Robustness

We start by training our EWS on standard benchmark datasets, i.e., CIFAR-10 [45], CIFAR-100 [45], and ImageNet [23], and testing on both clean test examples and the corrupted ones, namely CIFAR-10-C, CIFAR-100-C, and ImageNet-C. On CIFAR-10-C and CIFAR-100-C, we report the test error on clean or corrupted examples. On ImageNet-C, in contrast, we report the mean corruption error (mCE) [38]. We also consider ImageNet-P which evaluates the prediction stability on videos using mean flip rate (mFR). For all the metrics, *lower is better*. In all the experiments, by default, we set $K = 10$, $\lambda = 1$, and $\rho = 0.7$ for our EWS.

Results on CIFAR. We compare different training methods based on a ResNet-50 model with 400 training epochs. Besides the standard training method, we also compare our method with Dropout [76]. By default, we use random cropping and horizontal flipping as the standard data augmentation. Moreover, we consider state-of-the-art augmentation methods that are commonly used to improve robustness, such as AutoAugment [21] and AugMix [40]. Note that our EWS is complementary to these methods and does not require any modifications.

Method		CIFAR-10		CIFAR-100	
		Clean Error (%) ↓	Corruption Error (%) ↓	Clean Error (%) ↓	Corruption Error (%) ↓
Standard	Vanilla	5.32 (-0.00)	26.46 (-0.00)	23.45 (-0.00)	50.76 (-0.00)
	Dropout	5.16 (-0.16)	26.17 (-0.29)	23.19 (-0.26)	50.43 (-0.33)
	EWS	**4.44 (-0.88)**	**24.94 (-1.52)**	**22.41 (-1.04)**	**40.08 (-1.68)**
AutoAugment	Vanilla	4.05 (-0.00)	16.19 (-0.00)	23.02 (-0.00)	44.37 (-0.00)
	Dropout	3.91 (-0.14)	16.04 (-0.15)	22.84 (-0.18)	44.09 (-0.28)
	EWS	**3.23 (-0.82)**	**14.31 (-1.88)**	**22.16 (-0.86)**	**42.40 (-1.97)**
AugMix	Vanilla	4.35 (-0.00)	13.57 (-0.00)	22.45 (-0.00)	38.28 (-0.00)
	Dropout	4.19 (-0.16)	13.44 (-0.13)	22.11 (-0.34)	37.97 (-0.31)
	EWS	**3.76 (-0.59)**	**10.80 (-2.77)**	**21.81 (-0.64)**	**35.24 (-3.04)**

Table 1: Clean and corrupted test error on CIFAR-10(-C) and CIFAR-100(-C). Our proposed EWS approach not only improves clean test error, but also consistently reduces corrupted test error as highlighted in **bold** on top of different data augmentation schemes.

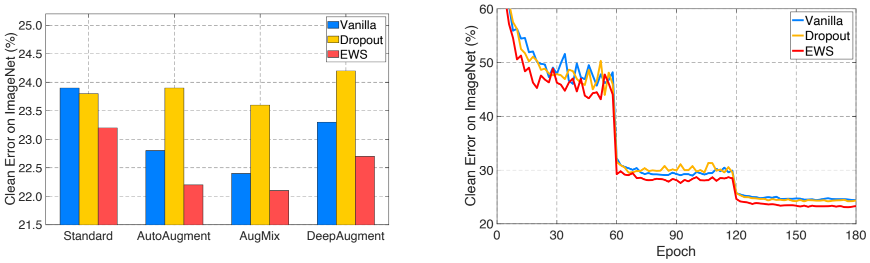

Fig. 4: *Left:* Comparisons of clean top-1 test error on ImageNet. EWS consistently reduces error across diverse augmentation schemes. *Right:* Training curves in terms of top-1 test error on ImageNet using the standard data augmentation scheme. Clearly, the improvement of EWS can be observed throughout training.

Table 1 shows that EWS is able to improve both clean and corrupted test errors over all three data augmentation schemes, i.e., “standard”, AutoAugment and AugMix. For example, on CIFAR-10, we are able to improve the clean accuracy over AugMix by 0.59% and significantly reduce the corruption error by 2.77%. Moreover, we also observe a similar performance improvement on CIFAR-100. To be specific, our EWS reduces the corruption error by 3.04% for AugMix while improving the clean accuracy by 0.64% on CIFAR-100. Note that the improvement is also significant for AutoAugment and the standard data augmentation, i.e., with the improvement of 1.88% and 1.52% on CIFAR-10, and the improvement of 1.97% and 1.68% on CIFAR-100. We emphasize that the improvement gets more significant with more complex data augmentation. This highlights that EWS is entirely complementary to state-of-the-art data augmentation. Furthermore, our method outperforms Dropout in all settings. As Dropout can be interpreted as randomly selecting channels in the preceding layers, it is a natural baseline to compare our EWS against. In addition, we also compare with a pruning based method CARDs [24] that yields state-of-the-art results on CIFAR-10-C. We highlight that, with AugMix, our EWS yields a larger relative improvement than CARDs [24] (2.77% vs 1.50%, see more details in our supplementary).

Method		mCE ↓	Gauss.	Shot	Imp.	Defoc.	Glass	Mot.	Zoom	Snow	Frost	Fog	Bright	Contra.	Elas.	Pixel	JPEG
Standard	Vanilla	76.5 (-0.0)	80	82	83	75	89	78	80	78	75	66	**57**	71	85	77	77
	Dropout	76.5 (-0.0)	77	79	80	78	90	79	87	**77**	77	67	58	**70**	84	75	76
	EWS	**75.1 (-1.4)**	**75**	**76**	**77**	**73**	**87**	**77**	**79**	80	**73**	**65**	58	73	**83**	**74**	**75**
AutoAugment	Vanilla	72.7 (-0.0)	69	68	72	**77**	83	80	81	79	75	64	56	70	88	57	**71**
	EWS	**71.7 (-1.0)**	**67**	**68**	**71**	78	**82**	**78**	**79**	**78**	**73**	**64**	**55**	**69**	**86**	**56**	72
AugMix	Vanilla	68.4 (-0.0)	65	66	67	70	**80**	66	66	75	72	67	58	**58**	79	69	**69**
	EWS	**67.5 (-0.9)**	**64**	**63**	**63**	**70**	81	**65**	**66**	**72**	**70**	**64**	**57**	63	**79**	**64**	70
DeepAugment	Vanilla	60.4 (-0.0)	49	50	47	59	73	65	76	64	**60**	58	51	61	76	48	67
	EWS	**58.7 (-1.7)**	**48**	**48**	**47**	**58**	**72**	**58**	**62**	**63**	62	**58**	**50**	**56**	**74**	**47**	**62**

Table 2: Corruption error on ImageNet-C. We consider the mean corruption error (mCE) as well as the individual ones. In all data augmentation settings, EWS reduces mCE significantly. We just report Dropout for the standard setting as it usually worsens mCE on top of complex augmentation and put the full comparisons in our supplementary.

Method		mFR ↓	Gaussian	Shot	Motion	Zoom	Snow	Bright	Translate	Rotate	Tilt	Scale
Standard	Vanilla	58.0 (-0.0)	**59**	58	64	72	63	62	**44**	52	57	**48**
	Dropout	57.8 (-0.2)	62	59	65	52	48	58	63	57	44	72
	EWS	**56.1 (-1.9)**	62	**55**	**62**	**49**	**45**	**52**	64	**52**	**42**	71
AutoAugment	Vanilla	51.7 (-0.0)	50	45	57	**68**	63	53	40	**44**	50	46
	EWS	**50.4 (-1.3)**	**48**	**44**	**53**	70	**62**	**52**	**36**	45	**49**	**45**
AugMix	Vanilla	37.4 (-0.0)	46	41	**30**	47	38	46	**25**	**32**	35	33
	EWS	**36.6 (-0.8)**	**45**	**39**	31	**42**	**33**	**43**	39	35	**27**	**32**
DeepAugment	Vanilla	32.1 (-0.0)	29	28	25	41	31	43	27	31	33	33
	EWS	**30.9 (-1.2)**	**28**	**26**	**25**	**40**	**28**	**41**	**26**	**30**	**32**	**33**

Table 3: Mean flip rate (mFR) on ImageNet-P, testing stability of predictions on (corrupted) videos. In line with Table 2, EWS improves consistently over all considered data augmentation schemes and nearly all corruption types. Similar to Table 2, Dropout did not improve over AutoAugment, AugMix or DeepAugment.

Results on ImageNet. Again, we adopt a ResNet-50 as the baseline model. Following [40], we use the learning rate warm-up for the first 5 epochs and train the model for 180 epochs in total. In addition to AutoAugment and Augmix, we consider a stronger augmentation scheme DeepAugment [37] designed for ImageNet.

In Fig. 4 (left), we first show that EWS consistently improves clean error on top of all considered data augmentation schemes. When equipped with AugMix, our EWS yields the best error of 22.1% across all the considered settings. The improvement of EWS can be also observed throughout the whole training process, as illustrated for the standard data augmentation scheme in Fig. 4 (right). More critically, in Table 2 and 3 we focus on the significant robustness improvements on ImageNet-C and ImageNet-P through EWS. Specifically, we consistently reduce mCE by $> 0.9\%$ across different augmentation schemes on ImageNet-C. We also highlight that EWS improves results across most included corruption types. Regarding ImageNet-P, these observations are further confirmed. For example, EWS reduces the mean flip rate (mFR) from 32.1% to 30.9% (1.2% improvement) on top of DeepAugment. Overall, these results indicate that improving the performance of subnets through EWS boosts clean and corrupted error across a range of state-of-the-art data augmentation schemes.

Method		PreAct ResNet-18			WRN-28-10			WRN-34-10		
		Clean ↓	PGD-20 ↓	AA ↓	Clean ↓	PGD-20 ↓	AA ↓	Clean ↓	PGD-20 ↓	AA ↓
AT	Vanilla [56]	17.54	49.18	52.96 (-0.00)	14.89	45.17	47.81 (-0.00)	14.74	45.39	47.47 (-0.00)
	EWS	**16.85**	**47.99**	**51.84 (-1.12)**	**14.57**	**44.24**	**47.17 (-0.64)**	**14.33**	**44.04**	**46.58 (-0.89)**
	AWP [82]	19.59	46.01	51.43 (-0.00)	15.89	42.93	46.41 (-0.00)	**14.17**	41.89	45.96 (-0.00)
	AWP-EWS	**19.25**	**44.98**	**50.48 (-0.95)**	**15.81**	**41.72**	**45.58 (-0.83)**	14.21	**41.07**	**45.29 (-0.67)**
TRADES	Vanilla [89]	17.42	46.88	50.84 (-0.00)	15.50	44.11	47.40 (-0.00)	15.32	43.84	46.89 (-0.00)
	EWS	**17.10**	**45.73**	**49.67 (-1.17)**	**15.09**	**43.45**	**46.72 (-0.68)**	**14.56**	**43.13**	**46.06 (-0.83)**
	AWP [82]	18.27	45.36	49.62 (-0.00)	14.84	41.25	44.86 (-0.00)	15.55	40.85	43.90 (-0.00)
	AWP-EWS	**17.67**	**44.20**	**48.58 (-1.04)**	**14.30**	**40.40**	**44.22 (-0.64)**	**14.13**	**40.05**	**43.17 (-0.73)**
TRADES-AWP*		17.13	43.68	48.37 (-0.00)	13.37	38.51	41.97 (-0.00)	12.73	35.97	40.74 (-0.00)
TRADES-AWP-EWS*		**16.62**	**42.33**	**47.23 (-1.14)**	**12.59**	**37.60**	**41.23 (-0.74)**	**11.90**	**35.19**	**40.05 (-0.69)**

Table 4: Clean and robust test error, i.e., on adversarial examples generated using PGD-20 and AutoAttack, for three different architectures: PreAct ResNet-18, WRN-28-10 and WRN-34-10. We consider 'vanilla" AT and TRADES as well as their AWP variants as the baselines. We also highlight the improvement against AutoAttack in parentheses. * denotes the models trained with additional 500K unlabeled data [16]. Across all the settings, EWS reduces both the robust test error and the clean error.

4.2 Improving Adversarial Robustness

Besides corruption robustness, on CIFAR-10, we also apply EWS on top of several popular adversarial training approaches, including vanilla adversarial training (AT) [56], TRADES [89], AT with adversarial weight perturbations (AT-AWP) [82], and using additional unlabeled examples [16]. Our setup follows the settings of [65]. For AWP, we follow the hyper-parameters of the original paper [82]. Note that combinations with these approaches are also possible, e.g., TRADES with AWP *and* EWS, highlighting the flexibility of EWS. We consider PreAct ResNet-18 [36], WRN-28-10 and WRN-34-10 [87] and employ early stopping [65]. We train and evaluate using an ϵ of $^8/_{255}$. During training we use 10 iterations of PGD. At test time, we evaluate models under AutoAttack (AA) [20] and PGD with 20 iterations.

In Table 4, our EWS consistently yield significant improvement in term of adversarial robustness across both architectures and adversarial training variants. For example, considering vanilla AT, EWS reduces clean error of PreAct ResNet-18 by 1.19% and robust error against AA by 1.12%. While the improvement gets slightly smaller for larger models, i.e., WRN-28-10 and WRN-34-10, EWS improves both clean and robust error consistently. On a WRN-34-10, the improvement in robust error is still 0.89%. We highlight that we can yield a similar improvement of 0.77% against AA when we consider stronger training tricks specified by [60], i.e., additionaly using label smoothing and Softplus (see more details in the supplementary). This improvement also generalizes to TRADES and AWP both of which generally improve adversarial robustness. Moreover, EWS is able to improve over TRADES and AWP when using additional pseudo-labeled training examples, which performs best in our experiments. Here, EWS reduces the robust test error from 40.74% to 40.05% (0.69% improvement). At the same time, EWS also reduces the clean error by 0.83%. EWS improving over AWP also indicates that utilizing adversarial weight perturbations [82] *on* weak subnets instead of the overall network is more beneficial. We also compare our method with a very recent adversarial

Method	Clean Error (%) ↓	Corruption Error (%) ↓
Baseline	5.32	26.46
Random Search	4.49	25.81
L_1-norm Selection	4.37	25.45
Ours	**4.12**	**24.94**

Table 5: Clean and corrupted test error on CIFAR-10 and CIFAR-10-C. We compare our search method with random search and an L_1-norm based heuristic method. While randomly selecting subnets to improve already reduces both clean and corrupted error, finding particularly weak subnets obtains the lowest errors.

training method LBGAT [22]. We show that, using the same settings of LBGAT [22], EWS yields better adversarial robustness and clean accuracy than LBGAT on both CIFAR-10 and CIFAR-100. We put the detailed comparisons in supplementary due to page limit. Overall, these experiments highlight the generality of our method, improving not only corruption robustness but also adversarial robustness.

5 Ablation and Discussions

In the following, we present further ablation experiments and discussions. Specifically, in Section 5.1, we demonstrate that finding particularly weak subnets is important and discuss training cost as well as hyper-parameters. This also shows that computational overhead is minimal, even for adversarial training. Furthermore, we study the weight of the distillation loss λ and the width of the selected subnets ρ. Finally, in Section 5.2, we show that EWS allows to analyze the vulnerability of individual blocks and layers.

5.1 Search Strategies and Hyper-Parameters

We perform ablations on CIFAR-10(-C) regarding our search strategy and hyper-parameters. For all the experiments, we use a ResNet-50 and adopt the same settings as before.
Search strategies. In Table 5, we conduct an ablation study to investigate the effectiveness of our search method. We compare our method to two baselines: random search selects subnets entirely at random; and L_1-norm selection chooses the channels/paths with lowest L_1-norm of weights. For simplicity, we compare clean and corrupted test error on CIFAR-10(-C). Since the randomly sampled subnets may contain some weak components, we are able to reduce both the clean and corrupted error. Using L_1-norm is a simple heuristic to find weaker subnets than random search. This can be seen by a slightly reduced clean and corrupted error. Nevertheless, the L_1-norm may not be highly correlated with accuracy and the search results are often suboptimal. In contrast, our method yields the lowest clean and corrupted errors as the controller directly finds subnets with low accuracy. This also confirms our results in Fig. 1.
Training interval K and training cost. As detailed in Section 3.3 and Algorithm 1, we train the controller model every K iterations. Thus, we expect a trade-off between training cost and performance: a stronger controller model should improve performance, but needs to be updated more often (i.e., lower K) which, however, increases the training

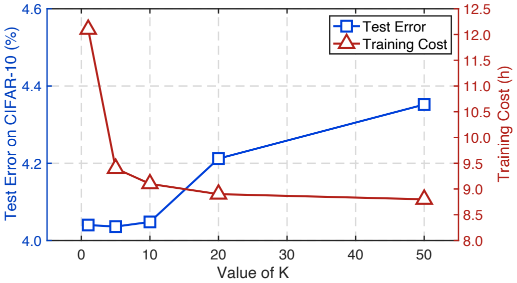

Fig. 5: We plot (clean) test error on CIFAR-10 (blue) and training cost in hours (red) against the controller training interval K, see Algorithm 1. Clearly, increasing K reduces training cost significantly. At the same time, clean error reduces significantly for $K \leq 10$. In practice, $K = 10$ yields a good trade-off and only introduces little computational overhead (roughly 3%) for both corruption and adversarial experiments.

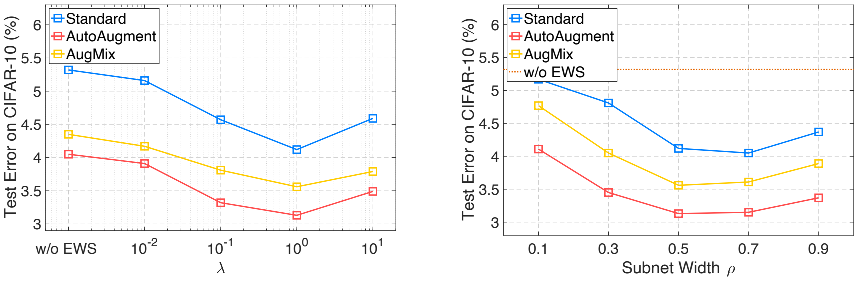

Fig. 6: Clean test error on CIFAR-10 plotted against the weight of the distillation loss (left, see Algorithm 1) and the subnet width ρ (right). *Left:* Across all tested data augmentation schemes, including AutoAugment and AugMix, $\lambda = 1$ performs best. *Right:* Too small or too large subnets during training reduce the benefit of EWS. We found that $\rho = 0.5$ and $\rho = 0.7$ perform best in most cases.

cost. As shown in Fig. 5, gradually increasing K greatly reduces the training cost. Note that we obtain promising results for $K{\leq}10$ but observe a significant increase of test error when $K{>}10$. We observe that $K{=}10$ also works well for ImageNet(-C) and the data dependent tuning is not necessary. This justifies our choice of $K{=}10$ which also results in small computational overhead of roughly 3% on the adversarial training settings.

Weight of distillation loss. In Fig. 6 (left), we change the value of λ in Eqn. (2). Note that $\lambda = 0$ corresponds to training *without* EWS. Larger λ, in contrast, increases the effect of EWS, i.e., trying to enhance weak subnets more aggressively. Given a set of values $\lambda \in \{0, 0.001, 0.01, 0.1, 1, 10\}$, we gradually reduce clean test error up until $\lambda = 1$. Larger values will result in a small performance degradation. However, even $\lambda = 10$ still outperforms training without EWS. This generalizes across all considered data augmentation schemes, including AutoAugment and AugMix.

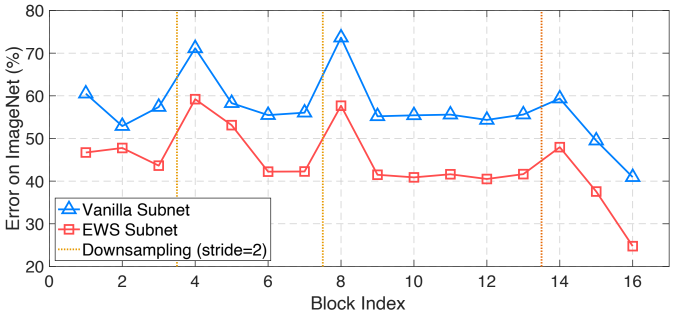

Fig. 7: Test error on ImageNet when constructing subnets by randomly selecting 50% of paths/channels in a specific block while keeping the rest unchanged. The results are averaged across 100 sampled subnets. The dotted line marks the position of downsampling operation. Clearly, blocks *after* downsampling blocks are most vulnerable and the last two blocks are least vulnerable. In all cases, EWS reduces error on subnets significantly.

Subnet width. In Fig. 6 (right), we investigate the impact of subnet width ρ on the performance improvement obtained using EWS. To this end, we consider a set of candidate widths $\rho \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. We suspect that very small subnets (ρ=0.1) cause the distillation loss $\mathcal{L}_{\text{KL}}\big(\alpha(x), M(x)\big)$ to be very large, hampering the training process. When we increase ρ to 0.5 or 0.7, EWS yields significantly better results. For a larger $\rho \geq 0.9$, the distillation loss can be very small, performance drops again.

5.2 Vulnerability of Blocks and Layers

As mentioned in Section 3.1, we construct subnets by selecting a subset of paths and channels. This also allows us to investigate the vulnerability of each block. To this end, we construct subnets by randomly selecting a subset of paths/channels in a specific block, while keeping all other blocks unchanged. In Fig. 7, we report test error on ImageNet when randomly sampling 100 subnets for each block. We observed that the blocks behind the downsampling operation (dotted lines) tend to be more vulnerable. More critically, EWS consistently reduces test error across all blocks compared to standard training.

6 Conclusion

In this paper, we described the phenomenon that most subnets of deep networks perform rather poorly, especially on perturbed examples. Interestingly, these weak subnets are highly correlated with the lack of robustness of the full model. To address this issue, we focused on improving model robustness by identifying and enhancing these particularly weak subnets. This leads to our proposed training method, EWS, which specifically employs a search algorithm to find weak subnets and then strengthens them through knowledge distillation. With minimal computational overhead, EWS can be applied on top of popular data augmentation schemes as well as adversarial training variants. Experiments show that EWS improves robustness consistently over these methods.

References

1. Akhtar, N., Mian, A.: Threat of adversarial attacks on deep learning in computer vision: A survey. arXiv.org **abs/1801.00553** (2018)
2. Alayrac, J., Uesato, J., Huang, P., Fawzi, A., Stanforth, R., Kohli, P.: Are labels required for improving adversarial robustness? In: NeurIPS (2019)
3. Athalye, A., Carlini, N.: On the robustness of the CVPR 2018 white-box adversarial example defenses. arXiv.org **abs/1804.03286** (2018)
4. Balaji, Y., Goldstein, T., Hoffman, J.: Instance adaptive adversarial training: Improved accuracy tradeoffs in neural nets. arXiv.org **abs/1910.08051** (2019)
5. Barreno, M., Nelson, B., Sears, R., Joseph, A.D., Tygar, J.D.: Can machine learning be secure? In: AsiaCCS (2006)
6. Benz, P., Zhang, C., Karjauv, A., Kweon, I.: Revisiting batch normalization for improving corruption robustness. arXiv.org **abs/2010.03630** (2020)
7. Biggio, B., Roli, F.: Wild patterns: Ten years after the rise of adversarial machine learning. arXiv.org **abs/1712.03141** (2018)
8. Bui, A., Le, T., Zhao, H., Montague, P., deVel, O., Abraham, T., Phung, D.Q.: Improving adversarial robustness by enforcing local and global compactness. In: ECCV (2020)
9. Cai, H., Zhu, L., Han, S.: ProxylessNAS: Direct neural architecture search on target task and hardware. In: ICLR (2019)
10. Cai, Q., Liu, C., Song, D.: Curriculum adversarial training. In: IJCAI. pp. 3740–3747 (2018)
11. Calian, D.A., Stimberg, F., Wiles, O., Rebuffi, S., György, A., Mann, T.A., Gowal, S.: Defending against image corruptions through adversarial augmentations. arXiv.org **abs/2104.01086** (2021)
12. Carlini, N.: Is ami (attacks meet interpretability) robust to adversarial examples? arXiv.org **abs/1902.02322** (2019)
13. Carlini, N., Wagner, D.: Adversarial examples are not easily detected: Bypassing ten detection methods. arXiv.org **abs/1705.07263** (2017)
14. Carlini, N., Wagner, D.A.: Defensive distillation is not robust to adversarial examples. arXiv.org **abs/1607.04311** (2016)
15. Carlini, N., Wagner, D.A.: Magnet and ”efficient defenses against adversarial attacks” are not robust to adversarial examples. arXiv.org **abs/1711.08478** (2017)
16. Carmon, Y., Raghunathan, A., Schmidt, L., Duchi, J.C., Liang, P.: Unlabeled data improves adversarial robustness. In: NeurIPS (2019)
17. Chaubey, A., Agrawal, N., Barnwal, K., Guliani, K.K., Mehta, P.: Universal adversarial perturbations: A survey. arXiv.org **abs/2005.08087** (2020)
18. Chen, Y., Guo, Y., Chen, Q., Li, M., Zeng, W., Wang, Y., Tan, M.: Contrastive neural architecture search with neural architecture comparators. In: CVPR. pp. 9502–9511 (2021)
19. Cheng, M., Lei, Q., Chen, P., Dhillon, I.S., Hsieh, C.: CAT: customized adversarial training for improved robustness. arXiv.org **abs/2002.06789** (2020)
20. Croce, F., Hein, M.: Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In: ICML (2020)
21. Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation policies from data. In: CVPR (2019)
22. Cui, J., Liu, S., Wang, L., Jia, J.: Learnable boundary guided adversarial training. In: ICCV. pp. 15721–15730 (2021)
23. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. pp. 248–255 (2009)
24. Diffenderfer, J., Bartoldson, B., Chaganti, S., Zhang, J., Kailkhura, B.: A winning hand: Compressing deep networks can improve out-of-distribution robustness. NeurIPS **34** (2021)

25. Ding, G.W., Sharma, Y., Lui, K.Y.C., Huang, R.: MMA training: Direct input space margin maximization through adversarial training. In: ICLR (2020)
26. Engstrom, L., Ilyas, A., Athalye, A.: Evaluating and understanding the robustness of adversarial logit pairing. arXiv.org **abs/1807.10272** (2018)
27. Frankle, J., Carbin, M.: The lottery ticket hypothesis: Training pruned neural networks. arXiv.org **abs/1803.03635** (2018)
28. Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F.A., Brendel, W.: Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. ICLR (2019)
29. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv.org **abs/1412.6572** (2014)
30. Gou, J., Yu, B., Maybank, S.J., Tao, D.: Knowledge distillation: A survey **129**(6), 1789–1819 (2021)
31. Gowal, S., Qin, C., Uesato, J., Mann, T.A., Kohli, P.: Uncovering the limits of adversarial training against norm-bounded adversarial examples. arXiv.org **abs/2010.03593** (2020)
32. Graves, A.: Long short-term memory. In: Supervised sequence labelling with recurrent neural networks, pp. 37–45. Springer (2012)
33. Guo, Y., Chen, Y., Zheng, Y., Zhao, P., Chen, J., Huang, J., Tan, M.: Breaking the curse of space explosion: Towards efficient nas with curriculum search. In: ICML. pp. 3822–3831. PMLR (2020)
34. Guo, Y., Zheng, Y., Tan, M., Chen, Q., Li, Z., Chen, J., Zhao, P., Huang, J.: Towards accurate and compact architectures via neural architecture transformer. PAMI (2021)
35. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
36. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: ECCV. pp. 630–645. Springer (2016)
37. Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., et al.: The many faces of robustness: A critical analysis of out-of-distribution generalization. In: ICCV. pp. 8340–8349 (2021)
38. Hendrycks, D., Dietterich, T.G.: Benchmarking neural network robustness to common corruptions and perturbations. In: ICLR (2019)
39. Hendrycks, D., Mazeika, M., Kadavath, S., Song, D.: Using self-supervised learning can improve model robustness and uncertainty. In: NeurIPS (2019)
40. Hendrycks, D., Mu, N., Cubuk, E.D., Zoph, B., Gilmer, J., Lakshminarayanan, B.: Augmix: A simple data processing method to improve robustness and uncertainty. In: ICLR (2020)
41. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: NeurIPS Deep Learning and Representation Learning Workshop (2014)
42. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv.org **abs/1704.04861** (2017)
43. Jeddi, A., Shafiee, M.J., Wong, A.: A simple fine-tuning is all you need: Towards robust deep learning via adversarial fine-tuning. arXiv.org **abs/2012.13628** (2020)
44. Kireev, K., Andriushchenko, M., Flammarion, N.: On the effectiveness of adversarial training against common corruptions. arXiv.org **abs/2103.02325** (2021)
45. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
46. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NeurIPS. pp. 1097–1105 (2012)
47. Lamb, A., Verma, V., Kannala, J., Bengio, Y.: Interpolated adversarial training: Achieving robust neural networks without sacrificing too much accuracy. In: AISec (2019)

48. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural Computation **1**(4), 541–551 (1989)
49. Lee, C.Y., Xie, S., Gallagher, P., Zhang, Z., Tu, Z.: Deeply-supervised nets. In: AISTATS (2015)
50. Li, B., Wang, S., Jia, Y., Lu, Y., Zhong, Z., Carin, L., Jana, S.: Towards practical lottery ticket hypothesis for adversarial training. arXiv.org **abs/2003.05733** (2020)
51. Li, Y., Min, M.R., Lee, T., Yu, W., Kruus, E., Wang, W., Hsieh, C.J.: Towards robustness of deep neural networks via regularization. In: ICCV (2021)
52. Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. In: ICLR (2019)
53. Liu, J., Zhuang, B., Zhuang, Z., Guo, Y., Huang, J., Zhu, J., Tan, M.: Discrimination-aware network pruning for deep model compression. arXiv preprint arXiv:2001.01050 (2020)
54. Liu, Y., Zhang, W., Li, S., Yu, N.: Enhanced attacks on defensively distilled deep neural networks. arXiv.org **abs/1711.05934** (2017)
55. Madaan, D., Shin, J., Hwang, S.J.: Learning to generate noise for robustness against multiple perturbations. arXiv.org **abs/2006.12135** (2020)
56. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: ICLR (2018)
57. Mao, X., Qi, G., Chen, Y., Li, X., Duan, R., Ye, S., He, Y., Xue, H.: Towards robust vision transformer. arXiv.org **abs/2105.07926** (2021)
58. Mosbach, M., Andriushchenko, M., Trost, T.A., Hein, M., Klakow, D.: Logit pairing methods can fool gradient-based attacks. arXiv.org **abs/1810.12042** (2018)
59. Nado, Z., Padhy, S., Sculley, D., D'Amour, A., Lakshminarayanan, B., Snoek, J.: Evaluating prediction-time batch normalization for robustness under covariate shift. arXiv.org **abs/2006.10963** (2020)
60. Pang, T., Yang, X., Dong, Y., Su, H., Zhu, J.: Bag of tricks for adversarial training. In: ICLR (2021)
61. Pang, T., Yang, X., Dong, Y., Xu, T., Zhu, J., Su, H.: Boosting adversarial training with hypersphere embedding. In: NeurIPS (2020)
62. Pham, H., Guan, M.Y., Zoph, B., Le, Q.V., Dean, J.: Efficient neural architecture search via parameter sharing. In: ICML. pp. 4092–4101 (2018)
63. Rakin, A.S., He, Z., Fan, D.: Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack. CVPR (2019)
64. Rebuffi, S.A., Gowal, S., Calian, D.A., Stimberg, F., Wiles, O., Mann, T.: Fixing data augmentation to improve adversarial robustness. arXiv preprint arXiv:2103.01946 (2021)
65. Rice, L., Wong, E., Kolter, J.Z.: Overfitting in adversarially robust deep learning. In: ICML (2020)
66. Rusak, E., Schott, L., Zimmermann, R.S., Bitterwolf, J., Bringmann, O., Bethge, M., Brendel, W.: A simple way to make neural networks robust against diverse image corruptions. In: ECCV (2020)
67. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. IJCV **115**(3), 211–252 (2015)
68. Saikia, T., Schmid, C., Brox, T.: Improving robustness against common corruptions with frequency biased models. arXiv.org **abs/2103.16241** (2021)
69. Salman, H., Ilyas, A., Engstrom, L., Kapoor, A., Madry, A.: Do adversarially robust imagenet models transfer better? arXiv.org **abs/2007.08489** (2020)
70. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: CVPR. pp. 4510–4520 (2018)

71. Schneider, S., Rusak, E., Eck, L., Bringmann, O., Brendel, W., Bethge, M.: Improving robustness against common corruptions by covariate shift adaptation. arXiv.org **abs/2006.16971** (2020)
72. Sehwag, V., Wang, S., Mittal, P., Jana, S.: HYDRA: pruning adversarially robust neural networks. In: NeurIPS (2020)
73. Sehwag, V., Wang, S., Mittal, P., Jana, S.: On pruning adversarially robust neural networks. arXiv.org **abs/2002.10509** (2020)
74. Silva, S.H., Najafirad, P.: Opportunities and challenges in deep learning adversarial robustness: A survey. arXiv.org **abs/2007.00753** (2020)
75. Singla, S., Feizi, S.: Second-order provable defenses against adversarial attacks. In: ICML (2020)
76. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. JMLR **15**(1), 1929–1958 (2014)
77. Stutz, D., Hein, M., Schiele, B.: Relating adversarially robust generalization to flat minima. In: ICCV (2021)
78. Sun, S., Cheng, Y., Gan, Z., Liu, J.: Patient knowledge distillation for bert model compression. EMNLP (2019)
79. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv.org **abs/1312.6199** (2013)
80. Wag, W., Chen, J., Yang, M.H.: Adversarial training with bi-directional likelihood regularization for visual classification. In: ECCV (2020)
81. Wang, H., Chen, T., Gui, S., Hu, T., Liu, J., Wang, Z.: Once-for-all adversarial training: In-situ tradeoff between robustness and accuracy for free. In: NeurIPS (2020)
82. Wu, D., Xia, S.T., Wang, Y.: Adversarial weight perturbation helps robust generalization. NeurIPS (2020)
83. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: CVPR. pp. 1492–1500 (2017)
84. Xu, H., Ma, Y., Liu, H., Deb, D., Liu, H., Tang, J., Jain, A.K.: Adversarial attacks and defenses in images, graphs and text: A review. arXiv.org **abs/1909.08072** (2019)
85. Yu, H., Liu, A., Liu, X., Yang, J., Zhang, C.: Towards noise-robust neural networks via progressive adversarial training. arXiv.org **abs/1909.04839** (2019)
86. Yuan, X., He, P., Zhu, Q., Bhat, R.R., Li, X.: Adversarial examples: Attacks and defenses for deep learning. arXiv.org **abs/1712.07107** (2017)
87. Zagoruyko, S., Komodakis, N.: Wide residual networks. In: BMVC (2016)
88. Zhang, H., Wang, J.: Defense against adversarial attacks using feature scattering-based adversarial training. In: NeurIPS (2019)
89. Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., Jordan, M.: Theoretically principled trade-off between robustness and accuracy. In: ICML. pp. 7472–7482. PMLR (2019)
90. Zhang, H., Yu, Y., Jiao, J., Xing, E.P., Ghaoui, L.E., Jordan, M.I.: Theoretically principled trade-off between robustness and accuracy. In: ICML (2019)
91. Zhang, H., Chen, H., Song, Z., Boning, D., inderjit dhillon, Hsieh, C.J.: The limitations of adversarial training and the blind-spot attack. In: ICLR (2019)
92. Zhang, J., Xu, X., Han, B., Niu, G., Cui, L., Sugiyama, M., Kankanhalli, M.S.: Attacks which do not kill training make adversarial learning stronger. In: ICML (2020)
93. Zi, B., Zhao, S., Ma, X., Jiang, Y.: Revisiting adversarial robustness distillation: Robust soft labels make student better. In: ICCV (2021)
94. Zi, B., Zhao, S., Ma, X., Jiang, Y.G.: Revisiting adversarial robustness distillation: Robust soft labels make student better. CVPR (2021)
95. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. In: ICLR (2017)