AID-PURIFIER: A LIGHT AUXILIARY NETWORK FOR BOOSTING ADVERSARIAL DEFENSE

Anonymous authors
Paper under double-blind review

ABSTRACT

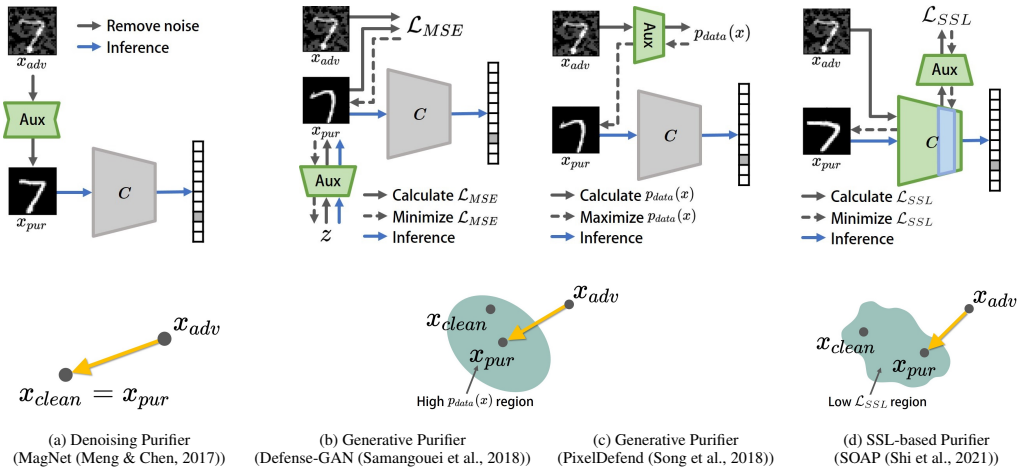
In this study, we propose *AID-purifier* that can boost the robustness of adversarially-trained networks by purifying their inputs. AID-purifier is an auxiliary network that works as an add-on to an already trained main classifier. To keep it computationally light, it is trained as a discriminator with a binary cross-entropy loss. To obtain additionally useful information from the adversarial examples, the architecture design is closely related to the information maximization principle where two layers of the main classification network are piped into the auxiliary network. To assist the iterative optimization procedure of purification, the auxiliary network is trained with AVmixup. AID-purifier can be also used together with other purifiers such as PixelDefend for an extra enhancement. Because input purification has been studied relative less when compared to adversarial training or gradient masking, we conduct extensive attack experiments to validate AID-purifier's robustness. The overall results indicate that the best performing adversarially-trained networks can be enhanced further with AID-purifier.

1 INTRODUCTION

Deep neural networks are vulnerable to adversarial examples generated by adding imperceptible adversarial perturbations to the original examples (Szegedy et al., 2013). To address this problem, various adversarial defense schemes have been proposed, where a vast majority of them can be grouped into three categories. The first category is *gradient masking* (Xiao et al., 2020; Athalye et al., 2018), where the gradient of the classifier is hidden or obfuscated to obstruct gradient-based adversarial attacks. The second category is *adversarial training* (Madry et al., 2017; Zhang et al., 2019; Lee et al., 2020), where the training dataset is augmented with adversarial examples. The third category is *adversarial purification* (Samangouei et al., 2018; Song et al., 2018; Meng & Chen, 2017; Shi et al., 2021), where a purification procedure is applied before the input example is passed to the main classifier.

Although gradient masking is an effective method against gradient-based adversarial attacks, a few critical limitations have been identified. Athalye et al. (2018) showed that defenses relying on obfuscated gradients can be circumvented. Papernot et al. (2017) demonstrated that black-box attacks often perform better than white-box attacks when the defense is a gradient-based method. Compared to gradient masking, adversarial training turned out to be much more robust against known adversarial attacks. Robustness is due to the inherent improvement in generalization over adversarial examples, and adversarial training is considered to be the most effective adversarial defense as of today. Madry et al. (2017) exploited a projected gradient descent (PGD) attack to augment the training dataset, Zhang et al. (2019) minimized the trade-off between the natural and robust errors when training, and Lee et al. (2020) suggested a soft-labeled data augmentation, called AVmixup, for improving adversarially robust generalization.

In this study, we focus on the third category, adversarial purification. Our objective is to develop a computationally light and easily attachable purifier such that it can be utilized as an add-on. Specifically, we show that we can boost the performance of Madry et al. (2017); Zhang et al. (2019), and Lee et al. (2020) with a light auxiliary network named *AID-Purifier*. AID-Purifier utilizes **A**Vmixup, **I**nformation maximization principles, and **D**iscriminative task as the underlying foundations. Before describing AID-Purifier, we first summarize previous works on adversarial purification methods.



(a) Denoising Purifier (MagNet (Meng & Chen, 2017))
(b) Generative Purifier (Defense-GAN (Samangouei et al., 2018))
(c) Generative Purifier (PixelDefend (Song et al., 2018))
(d) SSL-based Purifier (SOAP (Shi et al., 2021))

Figure 1: Summary of four existing purifiers. The upper diagrams show algorithm overviews. We denote main classification network as C, auxiliary network as Aux, network with frozen weights in gray, and network to be trained in green. The lower diagrams show the conceptual relationships between x_{clean}, x_{adv}, and x_{pur}.

Adversarial purification modifies input examples to increase adversarial robustness, and four well-known purification methods are shown in Figure 1. In (a), a *denoising purifier*, MagNet (Meng & Chen, 2017), is shown. It uses an auto-encoder, called a reformer, as an auxiliary network for denoising adversarial perturbations. It is a straightforward approach in that its objective is to return an adversarial example x_{adv} back to the original clean example x_{clean}. The resulting network as a whole, however, becomes just another feedforward network that is vulnerable to auxiliary-aware white-box attacks (Tramer et al., 2020). In (b), a *generative purifier*, Defense-GAN (Samangouei et al., 2018), is shown. It uses an independently trained GAN to project x_{adv} to a purified example x_{pur} that belongs to the clean distribution $p_{data}(x)$. Unlike denoising purifiers, there is no known auxiliary-aware attack that is effective. Defense-GAN, however, is not easy to train, and its performance is worse than that of another well-known generative purifier. In (c), another generative purifier, PixelDefend (Song et al., 2018), is shown. Its performance is much better than that of Defense-GAN, especially for more complex datasets. PixelDefend, however, is computationally heavy owing to its pixel-wise operation. In (d), a *self-supervised-learning-based purifier*, SOAP (Shi et al., 2021), is shown. It is similar to generative purifiers, but it projects x_{adv} to a purified example x_{pur} belonging to a low $\mathcal{L}_{SSL}$ region. SOAP yields competitive robust accuracy against state-of-the-art adversarial training and purification methods, but it needs to be jointly trained with the main classifier C. Of the four purification methods in Figure 1, SOAP is the only one that requires joint training and thus cannot be used as an add-on.

We herein propose a *discriminative purifier* named AID-Purifier, as shown in Figure 2. To the best of our knowledge, this is the first successful purification method based on a discrimination task. AID-Purifier uses an auxiliary discriminator network D to project x_{adv} to a purified example x_{pur} that belongs to a low $p_{adv}(x)$ region. Compared to the four methods in Figure 1, AID-Purifier is distinct because it has all the advantages of the four methods. Unlike denoising purifiers, it is robust against auxiliary-aware attacks. Unlike generative purifiers, it requires light computation and is easy to train. Unlike SOAP, it is an add-on that can be attached to any frozen state-of-the-art network. AID-Purifier is an effective stand-alone defense method; however, it can also create synergies with adversarially-trained networks or other purifier networks such as PixelDefend. For all the experiments we have performed under four strong white-box attacks, AID-Purifier was able to boost the robustness of the state-of-the-art adversarial training and purification methods. Furthermore, we

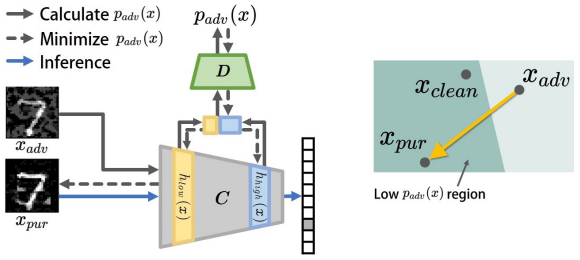


Figure 2: Algorithm overview and conceptual relationship of AID-Purifier. D is the discriminator network.

show that AID-Purifier is robust against a variety of adaptive attacks, including auxiliary-aware attack as a strong adaptive attack, and also against two representative black-box attacks.

2 Related Works

2.1 Detecting adversarial examples with an auxiliary network

For humans, it is difficult to tell the difference between a clean example and its adversarial example. The difference, however, can be detected well by training a binary classification network (Gong et al., 2017; Metzen et al., 2017). A standard binary cross-entropy (BCE) loss can be used for training, where the loss is interpreted as the probability of an adversarial example. Gong et al. (2017) demonstrated that an independent and simple adversarial detector can be trained using a labeled dataset comprising clean and adversarial examples. Furthermore, they showed that the detector is robust to a second-round adversarial attack, similar to the auxiliary-aware attack in our work. Metzen et al. (2017) considered a small subnetwork and concluded that a reliable adversarial detector could be trained. The subnetwork is essentially an auxiliary network attached to the main classification network $C(x)$, and it utilizes one of the hidden layers' representation as the input. Even though such detectors clearly prove that the difference between $p_{clean}(x)$ and $p_{adv}(x)$ is large enough that a detector can be trained, they are limited in that they cannot improve the adversarial robustness. In this study, we extend the idea of adversarial detector and show that a light auxiliary network can improve the adversarial robustness. To the best of our knowledge, this is the first study to propose a purifier with a BCE loss over clean and adversarial examples.

2.2 Information maximization principles

Following the principle of maximum information preservation in (Linsker, 1988) and the information maximization approach in (Bell & Sejnowski, 1995), Hjelm et al. (2019) demonstrated that unsupervised learning of representations is possible by maximizing mutual information between an input and the output of a deep neural network. For the main classifier network in our study, we consider a lower layer's representation $h_{low}(x)$ and a higher layer's representation $h_{high}(x)$ for a given input image x. Subsequently, the information maximization principles state that there should be a strong information theoretic relationship between $h_{low}(x)$ and $h_{high}(x)$. The relationship, however, is likely to be weaker or different for adversarial examples because the network has not seen adversarial examples during training. With this motivation, we select two layers from the main classification network and pass their representations $h_{low}(x)$ and $h_{high}(x)$ to the auxiliary network. According to the information maximization principles, it would be natural to maximize the mutual information between $h_{low}(x)$ and $h_{high}(x)$. Unfortunately, the precise estimation of mutual information is known to be difficult (McAllester & Stratos, 2020; Song & Ermon, 2020). A known workaround for this problem is to evaluate BCE loss or Jensen-Shannon divergence between the positive example pairs of $(h_{low}(x_i), h_{high}(x_i))$ and the negative example pairs of $(h_{low}(x_i), h_{high}(x_j))$ (Hjelm et al., 2019; Brakel & Bengio, 2017; Veličković et al., 2019; Ravanelli & Bengio, 2018), known as contrastive learning (Hadsell et al., 2006). In our AID-Purifier, we also use a BCE loss but we discriminate between $(h_{low}(x_{adv}), h_{high}(x_{adv}))$ and $(h_{low}(x_{clean}), h_{high}(x_{clean}))$ instead. This can be a natural choice for adversarial defense, because the information theoretic relationship between $h_{low}(x)$ and $h_{high}(x)$ should be different for clean examples and adversarial examples. In particular, the perturbation of features induced by x_{adv} increases gradually as it passes through the network (Guo et al., 2017; Liao et al., 2018; Xie et al., 2019). The auxiliary network is denoted as $D(h_{low}(x), h_{high}(x))$.

2.3 AVmixup

Zhang et al. (2018) proposed mixup that is a data augmentation scheme with linearly interpolated training examples for regularizing deep networks. Mixup can be considered as a derivative of label smoothing (Szegedy et al., 2016). As a variant of the mixup, Lee et al. (2020) proposed AVmixup for performing data augmentation of adversarial examples. In AVmixup, a virtual example called an Adversarial Vertex (AV) is first defined. For a given pair of a clean example and the corresponding adversarial example, their AV lies in the same direction as the adversarial example, but γ times farther away from the clean example. By controlling γ, the effect of data augmentation can be controlled. Once an AV example is defined, AVmixup extends the training distribution via linear inter-

polations of the clean example and the corresponding AV example. While AVmixup was shown to be effective for the adversarial training of the main classification network $C(x)$, we apply AVmixup to train the auxiliary discriminator network $D(h_{low}(x), h_{high}(x))$. This data augmentation with linear interpolation plays a pivotal role in training AID-Purifier. As a basic iterative procedure is applied at the inference time for purification, the discriminator needs to learn how to purify not only a strong adversarial example but also a weak adversarial example. Ideally, we would like the discriminator to learn a continuous path for purifying an adversarial example with an iterative procedure.

3 AID-PURIFIER

3.1 DISCRIMINATOR: $D(h_{low}(x), h_{high}(x))$

A diagram of AID-Purifier is shown in Figure 3. The main classification network $C(x)$ can be any naturally or adversarially-trained network. The main network was frozen before attaching our auxiliary discriminator network $D(h_{low}(x), h_{high}(x))$. Following the information maximization principles, a lower layer representation $h_{low}(x)$ and a higher layer representation $h_{high}(x)$ are passed from classifier C to discriminator D. In contrast to the work in (Hjelm et al., 2019), we apply global average pooling as the first operation in the discriminator. Despite the loss of spatial resolution in each feature map, global average pooling is helpful for making D computationally light for two reasons. First, the size of the representation is significantly reduced by averaging the spatial dimensions where purification can still enforce spatial variations over the channels. Second, as the resulting representations are invariant to spatial translations (Lin et al., 2013), we can simply use a fully connected network of a small size as the discriminator. When $C(x)$ is a typical CNN network, the lower layer has fewer number of channels than the higher layer as illustrated in Figure 3. Fully connected layers follow the global average pooling. The discriminator is trained with a standard BCE loss over adversarial and clean examples, and the loss function is as follow:

$$\mathcal{L}_D = -t \log(D(h_{low}(x), h_{high}(x))) - (1-t)\log(1 - D(h_{low}(x), h_{high}(x))), \tag{1}$$

where x is the input example and t is the corresponding binary label (adversarial or clean). See Appendix A for the architecture details.

3.2 TRAINING: AVMIXUP

To train the discriminator, we apply AVmixup (Lee et al., 2020) such that the discriminator learns how to purify any strength of adversarial example. As explained in Section 2.3, this is an essential requirement for the iterative purification procedure to work well. The details of the AVmixup training are shown in Algorithm 1. We use only PGD to generate adversarial examples because PGD is the worst case attack for most scenarios. In Appendix G.2, we have investigated the sensitivity to the choice of attack type for training. The results indicate that PGD training outperforms others.

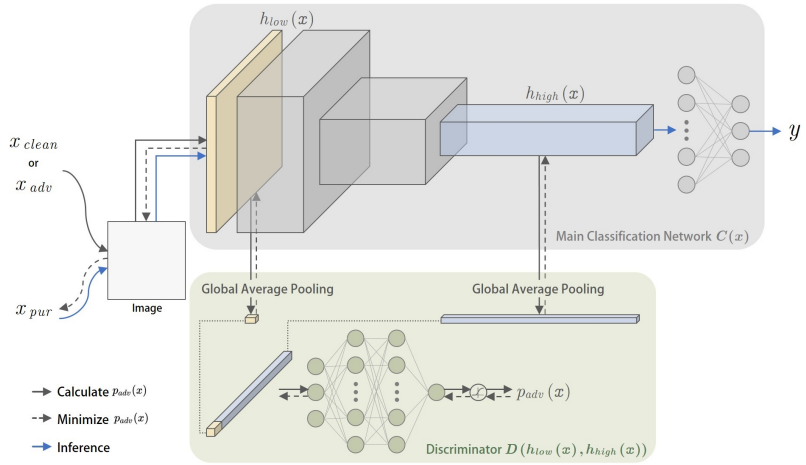


Figure 3: AID-Purifier. An auxiliary network D is attached to the main classification network C.

Algorithm 1 AVmixup training of discriminator

Input: Dataset S, input example x, label y, main classifier C with weights θ_C, scaling factor γ, learning rate lr, $t_{clean} \triangleq 0$, $t_{adv} \triangleq 1$
Output: Discriminator D with weights θ_D

Freeze θ_C, initialize θ_D
for *epoch*=1, . . . , E **do**
  for mini-batch $\{x, y\}_{i=1}^{B} \sim S$ **do**
    $\delta \leftarrow PGD(x, y; \theta_C)$
    AVmixup:
    $x_{AV} \leftarrow x + \gamma \cdot \delta$
    $u \sim \textit{Uniform}(0, 1)$
    $\hat{x} \leftarrow u \cdot x + (1 - u) \cdot x_{AV}$
    $\hat{t} \leftarrow u \cdot t_{clean} + (1 - u) \cdot t_{adv}$
    Model update:
    $l \leftarrow \mathcal{L}_D(D(h_{low}(\hat{x}), h_{high}(\hat{x})), \hat{t})$
    $\theta_D \leftarrow \theta_D - lr \cdot \frac{1}{B} \Sigma_{i=1}^{B} \nabla_{\theta_D} l$
  end for
end for

Algorithm 2 Purification at inference time

Input: Main classifier C, discriminator D, input example x, number of iterations N, step size α, epsilon ε
Output: Purified example x_{pur}

$x_{pur} \leftarrow x$
for n=1, . . . , N **do**
  $p_{adv} \leftarrow D(h_{low}(x_{pur}), h_{high}(x_{pur}))$
  $x_{pur} \leftarrow x_{pur} - \alpha \cdot sign(\nabla_{x_{pur}} p_{adv})$
  $x_{pur} \leftarrow clip(x_{pur}, x - \varepsilon, x + \varepsilon)$
  $x_{pur} \leftarrow clip(x_{pur}, 0, 1)$
end for

3.3 Inference: iterative purification

For inference, the auxiliary discriminator network D is used to purify x into x_{pur}. The purification is applied to any x including both clean and adversarial examples. As in the PGD attack, a basic iterative procedure is applied, and the purification is summarized in Algorithm 2. Specifically, an iterative gradient sign method is applied with the goal of reducing the probability of an adversarial attack, $p_{adv}(x)$. We constrain the algorithm to keep the purified image x_{pur} within the ε-ball of x, because an x_{pur} far from x might alter the class output of the main classification network $C(x)$.

4 Experiments

It is certainly possible to use a purifier as a stand-alone defense, but some of the purifiers can be also used as an add-on defense for boosting the performance of another adversarial defense. In this section, we investigate the performance of AID-Purifier as a stand-alone and as an add-on.

We perform the experiments over four datasets - SVHN (Netzer et al., 2011), CIFAR-10 (Krizhevsky & Hinton, 2009), CIFAR-100 (Krizhevsky & Hinton, 2009), and TinyImageNet (Le & Yang, 2015). As in other studies (Madry et al., 2017; Zhang et al., 2019), we use a 10-widen Wide-ResNet-34 (Zagoruyko & Komodakis, 2016) as the main classification network $C(x)$. For the white-box adversarial attack, we consider PGD (Madry et al., 2017), C&W (Carlini & Wagner, 2017), DeepFool (DF) (Moosavi-Dezfooli et al., 2016), and MIM (Dong et al., 2018). Additional details can be found in Appendix A.

4.1 Comparison with other adversarial purifiers

Among the four adversarial purifiers in Figure 1, we exclude MagNet (Meng & Chen, 2017) (a denoising purifier) and SOAP (Shi et al., 2021) (a SSL-based purifier) from our experiments. As explained in Section 1, MagNet is vulnerable to auxiliary-aware attack and SOAP cannot be used as an add-on. Thus, we only focus on Defense-GAN (Samangouei et al., 2018) and PixelDefend (Song et al., 2018). As with AID-Purifier, both are robust against auxiliary-aware attack and can be used as an add-on. For the baseline adversarial training models of add-on experiments, we use Madry (Madry et al., 2017), Zhang (Zhang et al., 2019), and Lee (Lee et al., 2020) as the most representative set of defense models.

Robust accuracy: The robust accuracy results for SVHN are shown in Table 1. Defense-GAN performs well as a stand-alone, but its performance as an add-on is inferior to the other purifiers. In fact, as an add-on, it usually undermines the baseline performance for complex datasets as shown in Appendix B. For this reason, Defense-GAN is not investigated any further in Subsection 4.2.

Table 1: Robust accuracy: stand-alone and add-on performances of adversarial purifiers are shown for the *worst* white-box attack. SVHN dataset is evaluated below, and the results for other datasets can be found in Appendix B.

	Stand-alone	Add-on		
	Natural training	Madry et al. (2017)	Zhang et al. (2019)	Lee et al. (2020)
No purification	0.01	22.63	36.72	46.17
Defense-GAN	41.89	28.42	38.60	43.42
PixelDefend	23.34	52.83	55.42	64.14
AID-Purifier (Ours)	29.10	49.85	44.76	62.70
PixelDefend + **AID-Purifier (Ours)**	**42.67**	**64.35**	**56.68**	**65.61**

Table 2: Computational load: Purification time (i.e., inference time) and training time of the adversarial purifiers. Purification time was measured with batch size one. The reported values were measured with a single RTX2080ti, except for the training time of PixelDefend's TinyImageNet that was measured with four RTX2080ti's due to the memory requirement.

	SVHN		CIFAR-10		CIFAR-100		TinyImageNet	
	Purification time (sec/img)	Training time (min)	Purification time (sec/img)	Training time (min)	Purification time (sec/img)	Training time (min)	Purification time (sec/img)	Training time (min)
Defense-GAN	0.14	205	0.13	197	0.14	198	0.31	1385
PixelDefend	41.97	1185	40.54	1056	40.96	1056	166.31	*5131*
AID-Purifier (Ours)	0.29	23	0.29	15	0.29	16	0.30	147

Both of PixelDefend and AID-Purifier show positive improvements as a stand-along defense, but the performance is far from being impressive. When both are utilized together, however, they achieve 42.67% of robust accuracy that is better than Madry or Zhang as a stand-alone. As an add-on, each of PixelDefend and AID-Purifier creates large improvements over all three adversarial training models. In fact, the best performance is achieved when both are utilized together. This synergy is due to the diversity between PixelDefend and AID-Purifier, and the diversity will be discussed further in Section 6. In Table 1, PixelDefend tends to perform better than AID-Purifier as an individual model. An exhaustive comparison will be provided in Subsection 4.2.

Finally, we note that the stand-alone performances for complex datasets tend to be marginal for all of PixelDefend, AID-Purifier, and PixelDefend+AID-Purifier. Therefore, the purifiers will be more valuable as an add-on as long as they can consistently improve the baseline defense models of Madry, Zhang, and Lee. This is exactly what we will show in Subsection 4.2.

Computational load: For a high-throughput application such as an API service or a real-time object detection by an autonomous vehicle, it can be critical for an adversarial defense to have a minimum impact on the inference time. Training time is also important as explained in (Shafahi et al., 2019; Wong et al., 2020). For a defense based on an adversarial training, the increase in training time is generally known to be large but the increase in inference time is negligible or zero. For a defense based on an adversarial purification, however, the increase in inference time (i.e., purification time) can be significant.

We have measured the purification time and training time of the purifiers, and the results are shown in Table 2. For the purification time, both defense-GAN and AID-Purifier perform well but PixelDefend is up to 554 times slower than AID-Purifier. This is due to the pixel-wise operation of PixelDefend, and the slow speed can be a critical limitation for certain applications. For the training time, AID-Purifier is definitely faster than the other two purifiers. Training of AID-Purifier is fast because a binary cross-entropy loss is used and because the training is typically completed within one epoch. The result includes the time for generating adversarial examples. The other two purifiers are trained as generative models and the training is slow. In fact, training of a generative model can be tricky in general.

4.2 Boosting performance as an add-on

As a deep dive, we provide exhaustive add-on experiment results for PixelDefend and AID-Purifier in Table 3. The most important finding is that each of the two adversarial purifiers provides a pos-

Table 3: Robust accuracy: Exhaustive experiment results for PixelDefend and AID-Purifier are shown for SVHN, CIFAR-10, CIFAR-100, and TinyImageNet datasets. As an add-on, each of PixelDefend and AID-Purifier provides a positive improvement for almost any individual combination of dataset and attack method. *Worst* in the last column denotes the worst robustness among Clean, PGD, C&W, DF, and MIM. By inspecting the worst performance column of each dataset, it can be observed that PixelDefend+AID-Purifier achieves the best performance for three datasets and AID-Purifier achieves the best performance for TinyImageNet, which is the most complex dataset in our experiments.

	SVHN						CIFAR-10					
Method	Clean	PGD	C&W	DF	MIM	Worst	Clean	PGD	C&W	DF	MIM	Worst
Natural training	96.19	0.01	44.98	0.57	0.04	0.01	95.44	0.00	2.98	0.01	0.00	0.00
Madry et al. (2017)	67.39	38.17	59.08	28.34	22.63	22.63	88.72	51.64	84.75	54.81	52.45	51.64
Zhang et al. (2019)	94.98	36.72	93.61	62.15	40.46	36.72	84.49	55.32	80.73	57.68	56.14	55.32
Lee et al. (2020)	97.29	55.64	**94.00**	52.45	46.17	46.17	90.46	46.44	86.41	54.14	49.32	46.44
Natural training + PixelDefend	88.46	37.70	83.68	82.89	23.34	23.34	85.45	40.41	82.13	**81.97**	29.41	29.41
Madry et al. (2017) + PixelDefend	74.56	52.83	72.66	74.29	55.77	52.83	87.31	54.75	85.63	72.71	54.88	54.75
Zhang et al. (2019) + PixelDefend	93.03	55.42	91.73	**90.30**	58.14	55.42	83.41	56.68	81.61	68.39	56.89	56.68
Lee et al. (2020) + PixelDefend	94.03	64.14	92.71	89.66	73.92	64.14	89.01	51.83	87.29	69.26	53.29	51.83
Natural training + **AID-Purifier (Ours)**	78.33	37.25	67.62	67.83	29.10	29.10	87.84	2.15	78.36	79.55	1.35	1.35
Madry et al. (2017) + **AID-Purifier (Ours)**	89.20	49.85	88.98	87.63	52.98	49.85	88.28	52.65	86.87	72.00	53.08	52.65
Zhang et al. (2019) + **AID-Purifier (Ours)**	93.04	45.73	91.78	82.77	44.76	44.76	84.59	56.05	83.07	69.19	56.57	56.05
Lee et al. (2020) + **AID-Purifier (Ours)**	95.28	62.70	**94.00**	88.91	70.41	62.70	89.59	49.13	**88.04**	67.29	51.24	49.13
Natural training + PixelDefend + **AID-Purifier (Ours)**	71.32	49.76	67.44	67.75	42.67	42.67	76.27	41.81	73.25	72.97	35.82	35.82
Madry et al. (2017) + PixelDefend + **AID-Purifier (Ours)**	88.71	64.35	88.39	87.34	72.27	64.35	86.66	55.07	85.28	72.07	55.10	55.07
Zhang et al. (2019) + PixelDefend + **AID-Purifier (Ours)**	90.28	56.68	87.51	84.35	58.80	56.68	83.50	**57.22**	82.06	69.75	**57.67**	**57.22**
Lee et al. (2020) + PixelDefend + **AID-Purifier (Ours)**	93.04	**65.61**	91.44	89.07	**74.23**	**65.61**	87.85	53.33	86.34	69.67	54.32	53.33

	CIFAR-100						TinyImageNet					
Method	Clean	PGD	C&W	DF	MIM	Worst	Clean	PGD	C&W	DF	MIM	Worst
Natural training	78.17	0.02	3.23	0.04	0.05	0.02	64.98	0.02	20.91	0.00	0.31	0.00
Madry et al. (2017)	64.69	25.09	58.31	43.08	25.91	25.09	58.46	20.74	52.71	38.27	21.41	20.74
Zhang et al. (2019)	56.90	29.59	51.68	39.89	29.98	29.59	50.28	23.81	45.46	33.48	24.08	23.81
Lee et al. (2020)	74.56	27.07	65.40	54.80	29.91	27.07	65.09	26.75	57.91	46.90	27.72	26.75
Natural training + PixelDefend	61.19	29.95	58.23	58.73	20.78	20.78	56.23	0.68	45.35	51.33	0.66	0.66
Madry et al. (2017) + PixelDefend	62.90	27.34	59.64	46.80	27.69	27.34	57.65	21.81	54.93	41.00	22.29	21.81
Zhang et al. (2019) + PixelDefend	55.43	30.61	52.53	42.36	30.85	30.61	49.53	24.21	47.51	35.61	24.34	24.21
Lee et al. (2020) + PixelDefend	69.87	30.84	**67.75**	57.32	32.13	30.84	58.17	29.82	57.11	49.46	30.15	29.82
Natural training + **AID-Purifier (Ours)**	67.15	0.52	60.27	63.27	0.35	0.35	58.00	0.26	45.41	52.73	0.40	0.26
Madry et al. (2017) + **AID-Purifier (Ours)**	63.94	26.34	61.11	46.77	26.92	26.34	58.69	21.15	55.11	41.36	21.73	21.15
Zhang et al. (2019) + **AID-Purifier (Ours)**	56.40	30.37	54.12	42.87	30.59	30.37	50.26	24.24	47.41	36.40	24.37	24.24
Lee et al. (2020) + **AID-Purifier (Ours)**	69.32	30.36	67.38	**64.07**	31.31	30.36	60.97	**31.05**	**59.28**	**56.67**	**30.54**	**30.54**
Natural training + PixelDefend + **AID-Purifier (Ours)**	54.56	27.36	52.47	52.23	23.06	23.06	50.22	3.26	43.47	46.64	2.02	2.02
Madry et al. (2017) + PixelDefend + **AID-Purifier (Ours)**	61.92	28.05	59.46	46.52	28.33	28.05	58.02	21.84	55.22	41.76	22.23	21.84
Zhang et al. (2019) + PixelDefend + **AID-Purifier (Ours)**	54.79	30.74	52.59	42.56	30.94	30.74	49.79	24.42	47.71	36.94	24.57	24.42
Lee et al. (2020) + PixelDefend + **AID-Purifier (Ours)**	66.55	**33.10**	64.78	57.76	**33.32**	**33.10**	59.13	30.52	58.02	50.88	30.48	30.48

itive enhancement for almost any individual combination of dataset and attack method. For the worst performance column, which can be considered as the overall conclusion for each dataset, PixelDefend+AID-Purifier achieves 42.11% (46.17 to 65.61), 3.43% (55.32 to 57.22), 11.86% (29.59 to 33.10), and 13.94% (26.75 to 30.48) of performance boosting for SVHN, CIFAR-10, CIFAR-100, and TinyImageNet, respectively. AID-Purifier alone, which requires only a light increase in the computational load, also achieves 35.80% (46.17 to 62.70), 1.32% (55.32 to 56.05), 2.64% (29.59 to 30.37), and 14.17% (26.75 to 30.54) of performance boosting.

5 Performance against adaptive attacks and black-box attacks

Because adversarial purification is a relatively less explored approach when compared to gradient masking or adversarial training approaches, it is essential to investigate as many attack types as possible. We first address the possibility of designing a direct adaptive attack for purifiers. Among the purifiers shown in Figure 2, denoising purifier can be easily attacked because the auxiliary network and the main network are directly cascaded and the overall gradient can be calculated in a straightforward way. The gradient calculation for other purifiers in Figure 2, however, is practically impossible because of the iterative nature of purification procedure (as in the PGD attack procedure) and because of the use of local gradients (Hessian is needed for an attack). The same is true for AID-purifier. A direct attack of AID-purifier needs to be based on the equation in line 4 of Algorithm 2, where gradient is already involved in the equation. Solving the equation as an adaptive attack requires a differentiation with respect to x_{pur}, and therefore Hessian needs to be calculated. Also, an iterative attack procedure needs to be designed to cope with the iterative nature of purification. Additionally, the $sign()$ and $clip()$ functions in Algorithm 2 need to be addressed. Overall, we believe a direct adaptive attack that fully utilizes the details of Algorithm 2 is an intractable problem even with iterative approximations. Instead, we perform an extensive investigation over the applicable attack methods.

A strong adaptive attack: SSL-based purifier was proposed by Shi et al. (2021). In their work, a new attack method called *auxiliary-aware attack* was provided where it is broadly applicable to any purifier that utilizes an auxiliary network. The basic idea of the attack is to find an adversarial example that is unlikely to be affected by the auxiliary network's purification while simultaneously being an effective attack to the main network. The auxiliary-aware attack can be considered as a strong adaptive attack because it is a complete white-box attack where both the main classification network and the auxiliary network are known to and utilized by the attacker. For AID-purifier, an auxiliary-aware attack is generated as

$$x_{adv} \leftarrow \underset{x_{adv} \in \mathcal{N}(x)}{argmax} \; (1-\lambda) \cdot \mathcal{L}_C - \lambda \cdot \mathcal{L}_D, \quad (2)$$

where λ is a trade-off parameter between the main cross-entropy loss $\mathcal{L}_C$ and the auxiliary discriminator loss $\mathcal{L}_D$. The results are shown in Figure 4. When λ is large, the attack successfully generates examples that cannot be purified, but instead the strength of the main network attack becomes weak. When λ is small, the main network attack is strong but instead the purification remains effective. Overall, $\lambda = 0$ turns out to be the optimal attack for minimizing robust accuracy, and the AID-purification is effective for improving robust accuracy. Additional results for natural training, Zhang et al. (2019), and Lee et al. (2020) are shown in Appendix D.1.

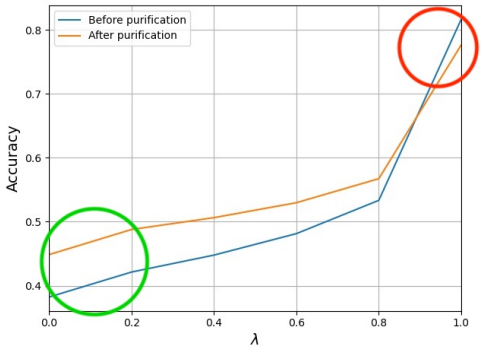


Figure 4: Robust accuracy against auxiliary-aware PGD attacks. When λ is large (red circle), the attack successfully generates examples that cannot be purified, but the attack to main network becomes weak. When λ is small (green circle), the main network attack is strong but the purification remains effective. The evaluation is over Madry for SVHN.

Other adaptive attacks: Several well-known adaptive attacks have been developed for general validation of defense methods. Tramer et al. (2020) and Dong et al. (2020) showed that evaluating defense methods across different attack configurations (attack iterations or epsilon) is important. Croce & Hein (2020) proposed AutoAttack that can achieve state-of-the-art attack performance and argued that it should be tested for any new defense. Hendrycks & Dietterich (2019) proposed new corruption datasets for testing robustness of classifiers. As shown in Appendix D, we have evaluated AID-Purifier over all of the aforementioned adaptive attacks. We have confirmed that AID-purifier can improve robustness even under attack configuration adjustments and AutoAttack. Additionally, AID-Purifier didn't show any significant performance drop for the corruption datasets.

Black-box attacks: We evaluated robustness against black-box attacks. In general, black-box attacks are much less effective than white-box attacks because of the lack of target classifier information. Despite this disadvantage, Athalye et al. (2018) showed that the defense methods that utilize target classifier's gradient can be vulnerable to black-box attacks.

For AID-Purifier, we consider two representative black-box attacks; Square attack as a score-based attack (Andriushchenko et al., 2020) and a transfer-based attack (Papernot et al., 2017). For transfer-based attacks, we test the robustness using two source networks. One of them is a 10-Wideresnet-34, the same as the target network, and the other is a VGG-19 (Simonyan & Zisserman, 2014). As shown in Table 4, the black-box attacks are not effective to the AID-Purifier.

6 Discussions

Ablation and sensitivity studies: To verify that the key features of AID-Purifier are effective, we have performed ablation studies and the results are shown in Appendix E. In Table 14 of Appendix E.1, all of the three key features are found to be helpful, where the choice of mutual information related architecture design, the choice of binary cross-entropy loss, and the choice of AVmixup augmentation are crucial for AID-Purifier's performance. In Table 15 of Appendix E.2, the number and choice of the layers to be connected to the discriminator D are investigated. We have limited the choice of candidate layers to avoid tuning effects, where only four candidates are considered. The best performance was observed for {10th block, 15th block}, but we have chosen {1st Conv, 15th block} because its performance is on par with the best and because it is more in line with our design principle. Competitiveness of connecting to two layers was confirmed as well.

Table 4: Robust accuracy under a score-based attack (Square attack) and transfer-based black-box attacks. (S=T) denotes the source model (Wideresnet-34) is the same as the target model and (S≠T) denotes the source model (VGG-19) is different from the target model. PGD, C&W, DF, and MIM attacks are used to evaluate the transfer-based attacks, and the results for the worst attacks are shown in the table. The results are shown for SVHN.

	Score-based	Tranfer-based (S=T)	Transfer-based (S≠T)
Natural training	0.213	11.955	38.55
Natural training+AID-Purifier	51.625	43.178	43.32
Madry et al. (2017)	16.745	40.731	87.31
Madry et al. (2017)+AID-Purifier	84.668	55.025	86.42

In addition to the ablation studies, we have investigated the sensitivity to the defense epsilon and the number of iterations N at the time of purification defense. The results are shown in Appendix G.1 where AID-purifier's robustness is confirmed.

Mutual information: A fundamental design principle of AID-purifier is the assumption that a smaller mutual information value between $h_{low}(x)$ and $h_{high}(x)$ should be observed for adversarial examples when compared to the clean examples. To confirm that the assumption is indeed true, we have measured the mutual information of clean examples and PGD examples (results in Appendix H). For both natural training and adversarial training, we have confirmed that mutual information is significantly larger for clean examples than for adversarial examples (0.622 vs 0.114 for natural training, 0.868 vs 0.478 for adversarial training).

Adversarial purifiers as performance boosters: In Section 4.2, it was shown that AID-Purifier and PixelDefend can be highly useful as add-on defenders that can be used in addition to a high-performance defender such as Madry, Zhang, and Lee. Not only that, AID-Purifier and PixelDefend can be used together for an extra improvement in robust accuracy. The extra improvement is due to the diversity between the two purifiers. PixelDefend's purification is based on the distribution $p_{data}(x)$ that is learned by PixelCNN. AID-Purifier's purification is based on the distributions p_{clean} and p_{adv} that are learned by the discriminator.

To investigate if AID-Purifier can create a positive synergy with purifiers other than PixelDefend, we have carried out an extra experiment. In Appendix C, NRP (Naseer et al., 2020) instead of PixelDefend is used to create a table in the same form as Table 3. NRP is a type of adversarial purifier, that trains a conditional GAN to learn an optimal input processing function that enhances model robustness. For NRP, the results in Table 10 of Appendix C are similar to PixelDefend's results - AID-Purifier and NRP can create positive synergies. For the case of Defense-GAN, we have shown in Appendix B that it can have negative effects for CIFAR-10, CIFAR-100, and TinyImageNet. In any case, we have tried using AID-Purifier together with Defense-GAN, and we have found that the loss by Defense-GAN can be mitigated by AID-Purifier as shown in Appendix F.

Even though we have considered only adversarially-trained models as the base models to improve, the base model does not need to be limited in such a way. As long as a purifier is light in computation and attachable to frozen networks, it can be used as an add-on to boost *any* adversarial defense network. In general, it remains as a future work to answer if adversarial purifiers should be always included as the last step add-on.

7 Conclusion

In this study, we have proposed AID-Purifier, a light auxiliary network for purifying adversarial examples. To the best of our knowledge, AID-Purifier is the first successful purification method that is based on a simple discriminator. It has a quite different characteristics from the previously known purifiers in terms of the purification objective, where a purified image x_{pur} is allowed to lie in an out-of-distribution region. It can consistently boost the performance of adversarially-trained networks, and it can create synergies with other adversarial purifiers such as PixelDefend and NRP. Whether adversarially-trained networks should be always used with one or more adversarial purifiers remains as an open question.

Ethics statement: Our study may increase privacy and security concerns by improving attackers' understanding on adversarial defense methods. However, we believe our study will contribute to the knowledge of the research community and be used in positive directions. Besides the privacy and security concerns, we believe our study does not contribute toward any other potential concerns.

Reproducibility statement: Our code is available as a supplementary item, and it will be made available on GitHub. Furthermore, we have included pseudo-codes in Section 3 and implementation details in Appendix A.

REFERENCES

Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pp. 484–501. Springer, 2020.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *International Conference on Machine Learning*, pp. 531–540. PMLR, 2018.

Anthony J Bell and Terrence J Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6):1129–1159, 1995.

Philemon Brakel and Yoshua Bengio. Learning independent features with adversarial nets for non-linear ica. *arXiv preprint arXiv:1710.05050*, 2017.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE symposium on security and privacy (sp)*, pp. 39–57. IEEE, 2017.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pp. 2206–2216. PMLR, 2020.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9185–9193, 2018.

Yinpeng Dong, Qi-An Fu, Xiao Yang, Tianyu Pang, Hang Su, Zihao Xiao, and Jun Zhu. Benchmarking adversarial robustness on image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 321–331, 2020.

Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. Adversarial and clean data are not twins. *arXiv preprint arXiv:1704.04960*, 2017.

Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pp. 1735–1742. IEEE, 2006.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HJz6tiCqYm`.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Bklr3j0cKX`.

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7:7, 2015.

Saehyung Lee, Hyungyu Lee, and Sungroh Yoon. Adversarial vertex mixup: Toward better adversarially robust generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 272–281, 2020.

Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1778–1787, 2018.

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

Ralph Linsker. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

David McAllester and Karl Stratos. Formal limitations on the measurement of mutual information. In *International Conference on Artificial Intelligence and Statistics*, pp. 875–884. PMLR, 2020.

Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pp. 135–147, 2017.

Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. In *Proceedings of 5th International Conference on Learning Representations (ICLR)*, 2017. URL https://arxiv.org/abs/1702.04267.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582, 2016.

Muzammal Naseer, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Fatih Porikli. A self-supervised approach for adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 262–271, 2020.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pp. 506–519, 2017.

Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017.

Mirco Ravanelli and Yoshua Bengio. Learning speaker representations with mutual information. *arXiv preprint arXiv:1812.00271*, 2018.

Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BkJ3ibb0-.

Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/7503cfacd12053d309b6bed5c89de212-Paper.pdf.

Changhao Shi, Chester Holtz, and Gal Mishne. Online adversarial purification based on self-supervised learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=_i3ASPp12WS.

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

Jiaming Song and Stefano Ermon. Understanding the limitations of variational mutual information estimators. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1x62TNtDS.

Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJUYGxbCW.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *arXiv preprint arXiv:2002.08347*, 2020.

Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rklz9iAcKQ.

Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJx040EFvH.

Chang Xiao, Peilin Zhong, and Changxi Zheng. Enhancing adversarial defense by k-winners-take-all. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Skgvy64tvr.

Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 501–509, 2019.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7472–7482. PMLR, 09–15 Jun 2019. URL http://proceedings.mlr.press/v97/zhang19p.html.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1Ddp1-Rb.

Supplementary materials for the paper "AID-Purifier: A Light Auxiliary Network for Boosting Adversarial Defense"

A EXPERIMENTAL DETAILS

In this appendix, we provide the details of the experiments conducted in our study. We described benchmark adversarial purifiers and their code sources in Appendix A.1, the detail of our discriminator architecture in Appendix A.2, hyperparameters of our discriminator in Appendix A.3, and attack hyperparameters in Appendix A.4.

A.1 BENCHMARK ADVERSARIAL PURIFIERS

The purifiers used in this paper are:

- Defense-GAN (Samangouei et al. (2018), Apache License) : `https://github.com/kabkabm/defensegan`.
- PixelDefend (Song et al. (2018), MIT License) : `https://github.com/microsoft/PixelDefend`.

A.2 AID-PURIFIER : DISCRIMINATOR ARCHITECTURE

For training discriminator, we first apply global average pooling to $h_{low}(x)$ and $h_{high}(x)$. Then, we pass the results to a fully-connected network described in Table 5. The discriminator architecture is for SVHN. We use one more linear layer both before and after concatenating for CIFAR-10, CIFAR-100, and TinyImageNet.

Table 5: Discriminator architecture

Operation	**Size**	**Activation**	**Output**
$h_{low} \rightarrow$ Linear	1024	ReLU	
Linear	1024	ReLU	Output 1
$h_{high} \rightarrow$ Linear	1024	ReLU	
Linear	1024	ReLU	Output 2
Concat (Output 1, Output 2)	2048		
Linear	1024	ReLU	
Linear	512	ReLU	
Linear	1		
Sigmoid	1		

A.3 AID-PURIFIER : DISCRIMINATOR HYPERPARAMETERS

Training hyperparameters: We train the network using SGD with learning rate 0.01, weight decay $2e{-}4$, and momentum 0.9 for 1 epoch. We use $\gamma = 2$ for SVHN and $\gamma = 1.5$ for CIFAR-10, CIFAR-100, and TinyImageNet.

Purification hyperparameters: Hyperparameters used for purification are described in Table 6.

Table 6: Hyperparameters in Algorithm 2

Dataset	N	ε	α
SVHN	10	12/255	3/255
CIFAR-10	10	8/255	2/255
CIFAR-100	10	16/255	2/255
TinyImageNet	10	8/255	2/255

A.4 ATTACK HYPERPARAMETERS

All attacks are evaluated under the l_2 metric for C&W and the l_∞ metric for the others. For SVHN, we use the perturbation size 12/255 and the step size 2/255. For CIFAR-10, CIFAR-100, and Tiny-ImageNet, we use the perturbation size 8/255 and the step size 1/255. We use Foolbox (Rauber et al., 2017), a third-party toolbox for evaluating adversarial robustness. All other parameters are set by Foolbox to be its default values.

B COMPARISON WITH OTHER PURIFIERS ON VARIOUS DATASETS

We repeated the same experiment of Table 1 for CIFAR-10 (Table 7), CIFAR-100 (Table 8), and TinyImageNet (Table 9).

For CIFAR-10, Defense-GAN performs better than AID-Purifier as a stand-alone, but its performance as an add-on is inferior to the other purifiers. PixelDefend performs better than AID-Purifier as a stand-alone and an add-on. PixelDefend+AID-Purifier performs best as a stand-alone and an add-on. For CIFAR-100, Defense-GAN performs worst and PixelDefend+AID-Purifier performs best again. For TinyImageNet, Defense-GAN performs best as a stand-alone, but PixelDefend+AID-Purifier outperforms other purifiers as an add-on.

Table 7: Robust accuracy: stand-alone and add-on performances of adversarial purifiers are shown for the *worst* white-box attack. CIFAR-10 dataset is evaluated below.

	Stand-alone	Add-on		
	Natural training	Madry et al. (2017)	Zhang et al. (2019)	Lee et al. (2020)
No purification	0.00	51.64	55.32	46.44
Defense-GAN	11.68	18.29	17.99	17.63
PixelDefend	29.41	54.75	56.68	51.83
AID-Purifier	1.35	52.65	56.05	49.13
PixelDefend+**AID-Purifier(Ours)**	**35.82**	**55.07**	**57.22**	**53.33**

Table 8: Robust accuracy: stand-alone and add-on performances of adversarial purifiers are shown for the *worst* white-box attack. CIFAR-100 dataset is evaluated below.

	Stand-alone	Add-on		
	Natural training	Madry et al. (2017)	Zhang et al. (2019)	Lee et al. (2020)
No purification	0.02	25.09	29.59	27.07
Defense-GAN	1.16	3.36	3.78	3.67
PixelDefend	20.78	27.34	30.61	30.84
AID-Purifier	0.35	26.34	30.37	30.36
PixelDefend+**AID-Purifier(Ours)**	**23.06**	**28.05**	**30.74**	**33.10**

Table 9: Robust accuracy: stand-alone and add-on performances of adversarial purifiers are shown for the *worst* white-box attack. TinyImageNet dataset is evaluated below.

	Stand-alone	Add-on		
	Natual training	Madry et al. (2017)	Zhang et al. (2019)	Lee et al. (2020)
No purification	0.00	20.74	23.81	26.75
Defense-GAN	**2.76**	4.95	4.79	4.74
PixelDefend	0.66	21.81	24.21	29.82
AID-Purifier	0.26	21.15	24.24	**30.54**
PixelDefend+**AID-Purifier(Ours)**	2.02	**21.84**	**24.42**	30.48

C BOOSTING PERFORMANCE AS AN ADD-ON TO NRP

We provide exhaustive add-on experiment results for NRP (Naseer et al., 2020) and AID-Purifier in Table 10. For the worst performance column, which can be interpreted as the overall conclusion for each dataset, AID-Purifier boosts the robustness in most scenarios. NRP+AID-Purifier

achieves 34.22% (46.17 to 61.97), 2.37% (55.32 to 56.63), 11.63% (29.59 to 33.03), and 14.43% (26.75 to 30.61) of performance boosting for SVHN, CIFAR-10, CIFAR-100, and TinyImageNet, respectively.

Table 10: Robust accuracy: exhaustive experiment results for NRP and AID-Purifier are shown for SVHN, CIFAR-10, CIFAR-100, and TinyImageNet datasets. As an add-on, each of NRP and AID-Purifier provides a positive improvement for almost any individual combination of dataset and attack method. By inspecting the worst performance column of each dataset, it can be observed that NRP+AID-Purifier achieves the best performance for two datasets, CIFAR-10 and CIFAR-100, and AID-Purifier achieves the best performance for two datasets, SVHN and TinyImageNet. Results of stand-alone and add-on using AID-Purifier only are duplicated from Table 3.

	SVHN						CIFAR-10					
Method	Clean	PGD	C&W	DF	MIM	Worst	Clean	PGD	C&W	DF	MIM	Worst
Natural training	96.19	0.01	44.98	0.57	0.04	0.01	95.44	0.00	2.98	0.01	0.00	0.00
Madry et al. (2017)	67.39	38.17	59.08	28.34	22.63	22.63	88.72	51.64	84.75	54.81	52.45	51.64
Zhang et al. (2019)	94.98	36.72	93.61	62.15	40.46	36.72	84.49	55.32	80.73	57.68	56.14	55.32
Lee et al. (2020)	97.29	55.64	**94.00**	52.45	46.17	46.17	90.46	46.44	86.41	54.14	49.32	46.44
Natural training+NRP	88.42	33.52	80.44	77.81	19.70	19.70	70.95	41.02	62.82	62.79	6.60	6.60
Madry et al. (2017)+NRP	82.29	49.54	81.05	81.44	51.83	49.54	86.08	44.02	84.06	**70.69**	54.96	44.02
Zhang et al. (2019)+NRP	93.00	52.71	91.68	84.94	50.63	50.63	81.90	55.83	79.69	66.04	56.15	55.83
Lee et al. (2020)+NRP	94.23	62.68	92.71	83.51	65.48	62.68	87.26	52.98	85.60	69.02	54.46	52.98
Natural training+**AID-Purifier (Ours)**	78.33	37.25	67.62	67.83	29.10	29.10	87.84	2.15	78.36	79.55	1.35	1.35
Madry et al. (2017)+**AID-Purifier (Ours)**	89.20	49.85	88.98	87.63	52.98	49.85	88.28	52.65	86.87	72.00	53.08	52.65
Zhang et al. (2019)+**AID-Purifier (Ours)**	93.04	45.73	91.78	82.77	44.76	44.76	84.59	56.05	83.07	69.19	56.57	56.05
Lee et al. (2020)+**AID-Purifier (Ours)**	95.28	**62.70**	**94.00**	**88.91**	**70.41**	**62.70**	89.59	49.13	**88.04**	67.29	51.24	49.13
Natural training+NRP+**AID-Purifier (Ours)**	66.89	42.09	62.72	62.59	36.02	36.02	66.38	40.58	61.04	60.86	20.46	20.46
Madry et al. (2017)+NRP+**AID-Purifier (Ours)**	85.33	53.59	84.62	83.16	56.49	53.59	86.14	46.27	84.53	**70.69**	55.31	46.27
Zhang et al. (2019)+NRP+**AID-Purifier (Ours)**	87.02	53.39	83.65	73.59	50.27	50.27	82.22	**56.63**	81.03	67.26	**56.71**	**56.63**
Lee et al. (2020)+NRP+**AID-Purifier (Ours)**	92.05	61.97	90.35	81.34	65.22	61.97	86.25	53.92	84.40	68.57	55.06	53.92

	CIFAR-100						TinyImageNet					
Method	Clean	PGD	C&W	DF	MIM	Worst	Clean	PGD	C&W	DF	MIM	Worst
Natural training	78.17	0.02	3.23	0.04	0.05	0.02	64.98	0.02	20.91	0.00	0.31	0.00
Madry et al. (2017)	64.69	25.09	58.31	43.08	25.91	25.09	58.46	20.74	52.71	38.27	21.41	20.74
Zhang et al. (2019)	56.90	29.59	51.68	39.89	29.98	29.59	50.28	23.81	45.46	33.48	24.08	23.81
Lee et al. (2020)	74.56	27.07	65.40	54.80	29.91	27.07	65.09	26.75	57.91	46.90	27.72	26.75
Natural training+NRP	40.25	6.23	35.92	36.44	5.85	5.85	49.28	9.13	43.41	45.71	7.22	7.22
Madry et al. (2017)+NRP	61.50	27.43	58.77	45.69	27.74	27.43	55.70	22.78	53.79	40.21	23.24	22.78
Zhang et al. (2019)+NRP	54.74	30.25	52.58	42.43	30.55	30.25	47.70	23.98	45.80	33.32	24.00	23.98
Lee et al. (2020)+NRP	65.15	32.77	63.43	54.27	33.68	32.77	55.38	29.76	54.52	47.25	30.39	29.76
Natural training+**AID-Purifier (Ours)**	67.15	0.52	60.27	63.27	0.35	0.35	58.00	0.26	45.41	52.73	0.40	0.26
Madry et al. (2017)+**AID-Purifier (Ours)**	63.94	26.34	61.11	46.77	26.92	26.34	58.69	21.15	55.11	41.36	21.73	21.15
Zhang et al. (2019)+**AID-Purifier (Ours)**	56.40	30.37	54.12	42.87	30.59	30.37	50.26	24.24	47.41	36.40	24.37	24.24
Lee et al. (2020)+**AID-Purifier (Ours)**	69.32	30.36	**67.38**	**64.07**	31.31	30.36	60.97	**31.05**	**59.28**	**56.67**	30.54	30.54
Natural training+NRP+**AID-Purifier (Ours)**	42.73	11.67	39.18	39.48	10.11	10.11	43.77	15.46	39.98	41.75	12.40	12.40
Madry et al. (2017)+NRP+**AID-Purifier (Ours)**	61.14	27.84	59.41	46.67	27.95	27.84	55.70	22.66	54.02	40.48	23.17	22.66
Zhang et al. (2019)+NRP+**AID-Purifier (Ours)**	54.76	30.55	52.46	43.07	31.04	30.55	47.90	24.24	46.09	33.93	23.87	23.87
Lee et al. (2020)+NRP+**AID-Purifier (Ours)**	64.78	**33.03**	63.13	54.64	**33.77**	**33.03**	58.49	30.61	57.28	50.51	**30.93**	**30.61**

D ADAPTIVE ATTACK

In this appendix, we evaluate the performance of AID-Purifier under additional adaptive attacks. We described the robustness under a strong adaptive attack in Appendix D.1, the performance sensitiveness to the attack's hyper-parameters such as epsilon and iteration in Appendix D.2, to common corruptions in Appendix D.3, and to a recent strong white-box attack in Appendix D.4.

D.1 A STRONG ADAPTIVE ATTACK:

We evaluate AID-Purifier over the auxiliary-aware PGD attack over natural training, Madry, Zhang, and Lee for SVHN, and the results are shown in Figure 5. The results are consistent with Figure 4 in the manuscript.

D.2 ATTACK EPSILON AND ATTACK ITERATION

The robust accuracy results for the sensitiveness of attack hyper-parameters are shown in Table 11. In Table 11-(a), AID-Purifier improves the performance at least 28% until the attack epsilon is smaller than 32/255. the robust accuracy drops terribly when the attack epsilon is 32/255 because we fix the defense epsilon as 12/255. In Table 11-(b), the performances are not sensitive to the attack's iteration number.

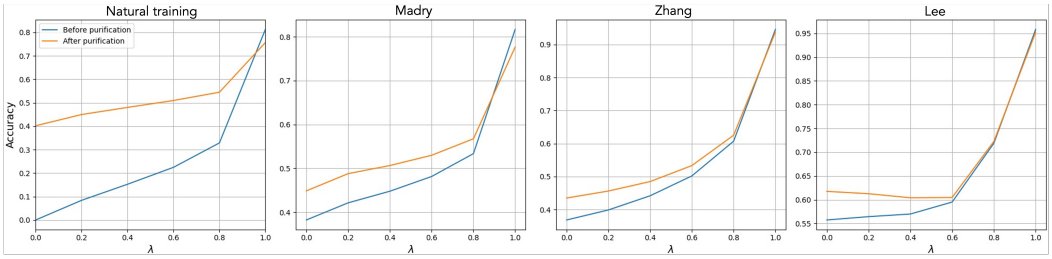


Figure 5: Robust accuracy before (blue) and after (orange) purification are evaluated against auxiliary-aware PGD attacks. The evaluation is over natural training, Madry, Zhang, and Lee for SVHN.

Table 11: Robust accuracy: (SVHN under PGD attack; Madry is used to train the main classification network) (a) Performance of Madry and Madry+AID-Purifier for varying the attack epsilon of PGD are shown for SVHN. (b) Performance of Madry and Madry+AID-Purifier for varying the attack iteration of PGD are shown for SVHN.

Attack eps	Madry	Madry+AID-Purifier
1/255	57.29	88.20
2/255	49.49	86.92
4/255	42.18	83.82
8/255	34.94	73.14
12/255 (our work)	38.17	49.85
16/255	22.28	28.62
32/255	1.27	1.59

(a)

Attack iteration	Madry	Madry+AID-Purifier
40 (our work)	38.17	49.85
100	35.64	48.60
200	34.86	48.34

(b)

D.3 COMMON CORRUPTION

The CIFAR10-C benchmark (Hendrycks & Dietterich, 2019) consists of 15 diverse corruption types applied to test images of CIFAR10. The corruptions are drawn from four main categories; noise, blur, weather, and digital. We investigate the robustness for CIFAR10-C, and the results are shown in Table 12. Compared to Madry, Madry+AID-Purifier showed a similar performance. The results show that AID-Purifier is not sensitive to common corruptions.

Table 12: Robust accuracy results for different corruptions. Results are shown for Madry and Madry+AID-Purifier on CIFAR10-C.

	Noise				**Blur**			
	Gaussian	**Shot**	**Speckle**	**Impulse**	**Defocus**	**Gaussian**	**Motion**	**Zoom**
Madry	83.11	84.45	84.14	76.77	82.93	80.48	78.82	81.79
Madry+AID-Purifier	83.38	84.56	84.30	76.44	82.52	80.22	78.47	81.16

	Weather				**Digital**					
	Snow	**Fog**	**Brightness**	**Frost**	**Contrast**	**Elastic**	**Pixelate**	**JPEG**	**Spatter**	**Saturate**
Madry	82.87	61.94	85.64	79.87	44.84	81.92	86.38	85.96	83.86	85.46
Madry+AID-Purifier	82.22	63.45	85.05	79.24	45.87	81.48	86.10	85.78	83.48	84.73

D.4 AUTOATTACK

We evaluated AID-Purifier under a recent strong ensemble white-box attack, AutoAttack (Croce & Hein, 2020), and AID-Purifier significantly improves robust accuracy as shown in Table 13.

Table 13: Robust accuracy under AutoAttack. The results are shown for SVHN.

Base model	Before purification	After purification
Natural training	0.004	35.679
Madry	10.134	64.363

E ABLATION STUDY

E.1 KEY FEATURES

AID-Purifier utilizes **A**Vmixup, **I**nformation maximization principles, and **D**iscriminative task as the underlying foundations. We perform ablation experiments over the three key features. In (a) of Table 14, we demonstrate the importance of the number of $h(x)$. The result shows that the MI-based architecture (exploiting two $h(x)$ as inputs) outperforms the simple architecture used in adversarial discriminators (Gong et al., 2017; Metzen et al., 2017) (exploiting one $h(x)$ as an input). In (b) of Table 14, we explore the effect of training targets. As we mentioned, using BCE loss between clean and adversarial examples has better performance than contrastive loss. In (c) of Table 14, the result shows that AVmixup helps the purification performance as we expected.

Table 14: Ablation test results over the key features of AID-Purifier: The evaluations are over Madry, SVHN, and PGD. The baseline performance of Madry without any add-on is 38.17%. (a) Number of intermediate layers connected from $C(x)$ to the discriminator. (b) Training targets. (c) Data augmentation method for training.

Number of $h(x)$	Accuracy (%)
1	48.93
2	**49.85**

(a) Number of $h(x)$

Targets	Accuracy (%)
Contrastive	44.87
Clean vs. adv.	**49.85**

(b) Training targets

Augmentation	Accuracy (%)
None	46.02
AV	47.73
Mixup	48.49
AVmixup	**49.85**

(c) Data augmentation

E.2 INTERMEDIATE LAYERS CONNECTED FROM $C(x)$ TO THE DISCRIMINATOR D

Table 15: Ablation test of the intermediate layers connected from $C(x)$ to the discriminator: The evaluations are over Madry, SVHN, and PGD. 1st Conv denotes the output of the first convolution layer and n-th Block denotes the output of the n-th residual block, where downsampling is performed. Check symbols indicate the connected layers. The best performing combination is {10th block, 15th block}, but we have used {1st Conv, 15th block} in our main experiments because the combination performs almost equally well and because it is more consistent with our design principle.

	Used intermediate representation				Worst white-box attack
Number of $h(x)$	1st Conv	5th Block	10th Block	15th Block	**AID-Purifier (Ours)**
1	✓				48.28
		✓			48.66
			✓		48.92
				✓	<u>48.93</u>
2	✓	✓			44.43
	✓		✓		48.97
	✓			✓	49.85 (manuscript)
		✓	✓		44.51
		✓		✓	45.77
			✓	✓	<u>**50.10**</u>

The ablation test results of the intermediate layers connected from $C(x)$ to the discriminator D are presented in Table 15. We have limited the choice of layers to avoid tuning effects (only four were considered as shown in Table 15), and studied the robust accuracy. The best performance was observed for {10th block, 15th block}, but we have chosen {1st Conv, 15th block} because its performance was on par with the best and because it is more in line with our design principle.

While the best performing options are based on two layers, one layer options also perform reasonably well. We have also investigated three layers and found that three can be also a reasonable option in terms of performance. But three-layer provides at most a marginal improvement while incurring an unnecessary complication. Therefore, we considered the marginal improvement as a tuning effect. Overall, it looks clear that two layers should be used because it is in line with our design principle and because it provides the best performance.

F BOOSTING PERFORMANCE OF AID-PURIFIER AS AN ADD-ON TO THE DEFENSE-GAN

We evaluate the robust accuracy of Defense-GAN+AID-Purifier as an add-on for SVHN in Table 16. Defense-GAN is actually harmful, but AID-Purifier can recover part of the performance loss created by Defense-GAN.

Table 16: Robust accuracy results over Madry, SVHN, and PGD.

	PGD
Madry	38.17
Madry+Defense-GAN	28.59
Madry+Defense-GAN+AID-Purifier	32.29

G SENSITIVITY STUDIES

G.1 DEFENSE EPSILON AND DEFENSE ITERATION

The results of sensitivity to the defense epsilon are shown in Figure 6, and the robust accuracy is insensitive to the choice of defense epsilon larger than 0.04.

The results of sensitivity to the number of iterations N at the time of purification defense are shown in Table 17. In general, the performance improves as N is increased. We have chosen $N = 10$ in our work such that the computational load can be kept small as shown in Table 2.

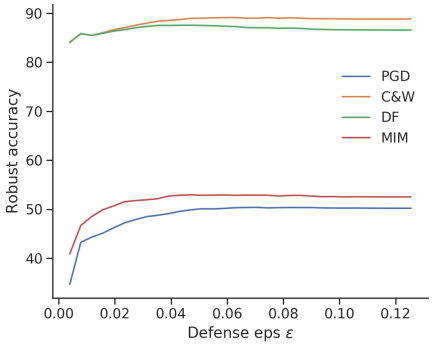


Figure 6: Robust accuracy for Madry+AID-Purifier is shown with respect to the variations in the defense epsilon (SVHN under PGD attack; Madry is used to train the main classification network).

Number of iterations	Madry+AID-Purifier
20	50.62
10 (our work)	49.85
5	49.26
4	48.95
2	46.68

Table 17: Robust accuracy for Madry+AID-Purifier with respect to the variations in the number of defense iterations (SVHN under PGD attack; Madry is used to train the main classification network).

G.2 ATTACK METHOD FOR TRAINING DISCRIMINATOR D

When training the discriminator, the attack method for generating adversarial examples need to be decided. The sensitivity study results are shown in Table 18 for SVHN dataset, and we have used PGD training in our work.

Table 18: Attack method used at the time of training and the resulting robust accuracy (SVHN under PGD attack; Madry is used to train the main classification network).

	PGD	C&W	DF	Worst
PGD training (our work)	**37.25**	67.62	67.83	**37.25**
C&W training	11.36	52.27	45.73	11.36
DF training	0.39	**74.12**	**74.65**	0.39
PGD + CW training	8.51	54.77	48.85	8.51
PGD + DF training	1.16	68.01	69.74	1.16
CW + DF training	19.14	68.73	64.16	19.14
PGD + CW + DF training	8.73	67.83	62.94	8.73

H MUTUAL INFORMATION ESTIMATION BETWEEN TWO LAYERS

For the estimation, we use MINE (Mutual Information Neural Estimation) (Belghazi et al., 2018). The results are shown for SVHN in Table 19. As expected, there is a large difference between clean examples and adversarial examples. For natural training, mutual information is significantly larger for clean examples (0.622) than for adversarial examples (0.114). For adversarial training, the same observation holds where the values are 0.868 for clean examples and 0.478 for adversarial examples.

Table 19: Estimation results of mutual information between the two layers ($h_{low}(x)$ and $h_{high}(x)$).

	Natural training	Madry et al. (2017)
Clean	0.622	0.868
PGD	0.114	0.478