Dual Head Adversarial Training

Yujing Jiang[1], Xingjun Ma[2], Sarah Monazam Erfani[1], James Bailey[1]
[1]*School of Computing and Information Systems, The University of Melbourne*
{yujingj@student., sarah.erfani@, baileyj@}unimelb.edu.au
[2]*School of Information Technology, Deakin University*
daniel.ma@deakin.edu.au

Abstract—Deep neural networks (DNNs) are known to be vulnerable to adversarial examples/attacks, raising concerns about their reliability in safety-critical applications. A number of defense methods have been proposed to train robust DNNs resistant to adversarial attacks, among which adversarial training has so far demonstrated the most promising results. However, recent studies have shown that there exists an inherent tradeoff between accuracy and robustness in adversarially-trained DNNs. In this paper, we propose a novel technique Dual Head Adversarial Training (DH-AT) to further improve the robustness of existing adversarial training methods. Different from existing improved variants of adversarial training, DH-AT modifies both the architecture of the network and the training strategy to seek more robustness. Specifically, DH-AT first attaches a second network head (or branch) to one intermediate layer of the network, then uses a lightweight convolutional neural network (CNN) to aggregate the outputs of the two heads. The training strategy is also adapted to reflect the relative importance of the two heads. We empirically show, on multiple benchmark datasets, that DH-AT can bring notable robustness improvements to existing adversarial training methods. Compared with TRADES, one state-of-the-art adversarial training method, our DH-AT can improve the robustness by 3.4% against PGD^{40} and 2.3% against AutoAttack, and also improve the clean accuracy by 1.8%.

I. INTRODUCTION

Deep neural networks (DNNs) have been adopted to achieve state-of-the-art performance in a wide range of applications, such as computer vision [1], natural language processing [2] and speech recognition [3]. Despite the great success, DNNs have also been found to be extremely vulnerable to adversarial examples/attacks [4], [5]. With imperceptible but carefully-crafted perturbations, natural (clean) examples can be converted into adversarial examples to fool state-of-the-art DNNs [6], [7]. In recent research works, adversarial attacks have been demonstrated to be destructive to almost all kinds of DNNs including image models [7], [6], [8], video models [9], graph models [10] and even language models like BERT [11]. This has raised security concerns on the deployment of DNNs in safety-critical applications such as face recognition [12], autonomous driving [13], [14], [15], [16], [17], medical diagnosis [18], [19] and many others.

A number of methods have been proposed to defend DNNs against adversarial attacks adversarially robust DNNs [20], [21], [22], [23], [7], among which adversarial training (AT) has demonstrated the most promising results [24], [25], [26]. Adversarial Training (AT) [5] involves adversarial samples in each training step to enhance the model's robustness, which can be formulated as a min-max optimization problem

[7], [27]. Most existing adversarial training methods adopt WideResNets (WRNs) [28] to demonstrate the best robustness results. By upscaling ResNets (RNs) at the width dimension, WRNs introduce more capacity into RNs in an efficient manner, which has been found to be crucial for adversarial robustness [29], [30]. It has been observed that using WRNs can bring consistent robustness improvement over RNs [29], [30], [31]. So far, the two most commonly used WRNs are WRN-34-10 [7], [29], [32], [33] and WRN-34-20 [34], [35], [36], [37] (∼4× more parameters than WRN-34-10). In this paper, we aim to explore a more efficient and effective way to improve adversarial robustness with adversarial training that does not significantly increase the model's capacity.

Following the standard adversarial training (SAT) proposed by Madry et al. [7], many adversarial training techniques introduce new loss functions or training strategies with additional tunable hyperparameters to improve robustness under different settings [35]. For example, TRADES [29] adopts a hybrid of the cross-entropy (CE) and the Kullback–Leibler (KL) divergence loss with a balancing hyperparameter (λ) to explore different trade-offs between the clean accuracy and adversarial robustness. This has been found to be an important generalization of SAT with substantial ($> 5\%$) robustness improvement [29]. A closer look at TRADES under different λs, we find that the adversarial noise patterns generated from these models are quite different on test examples, and the transferability of these patterns to other models is very limited. Moreover, the attack will be significantly weakened if we generate adversarial examples using the averaged perturbation over two TRADES models trained with different λs. These results indicate that different λs produce models that are robust in distinctive ways. This motivates us to discover a novel technique that can exploit different training parameters in one single model via separate output heads, and effectively aggregate those heads to yield a more robust model.

In this paper, we propose *Dual Head Adversarial Training* (DH-AT), an improved variant of AT that attaches a second head to one intermediate layer of the network. In WRNs, the second head can be symmetrically attached to the end of the first residual block (illustrated in Fig. 2). When TRADES [29] training method is considered, the two heads can be trained either simultaneously or independently with different λs. The main (existing) head can also be directly loaded from a pretrained model without any modifications, in which case only one head requires training. After training the two heads, a

lightweight convolutional neural network (CNN) can then be adversarially trained to combine the two heads, which takes fewer than 20 epochs. In real-world scenarios, the second head and the lightweight CNN together form a strengthening mechanism to improve the adversarial robustness of any existing models. Note that the second head can also be switched off when robustness is no longer the primary concern.

In summary, our main contributions are:

- We propose a novel Dual Head Adversarial Training (DH-AT) method to improve the adversarial robustness of any existing models by attaching a second output head to the network. DH-AT can be easily incorporated into existing adversarial training methods with minimal modifications.
- Our DH-AT provides a novel defense strategy with one head is responsible for clean accuracy and the other head for adversarial robustness. With our DH-AT, achieving both clean accuracy and robustness at the same time is possible, as evidenced by our experimental results.
- Following DH-AT, we demonstrate a novel alternative to early-stopping by using the best epoch's weights as the main head and a "robust overfitted" [34] as the second head. On CIFAR-10, this leads to up to 2.01% robustness improvement against PGD^{40} and 1.19% against AutoAttack [25] on CIFAR-10, when compared with TRADES.

II. RELATED WORK

In this section, we briefly review existing works in both adversarial attack and defense. We focus on those methods that were developed in the white-box setting where model parameters are known to either the attacker or defender. And we will focus on those developed with image classification models, the major field of adversarial research.

A. Adversarial Attack

The vulnerability of DNNs to small adversarial perturbations was initially discovered by Szegedy et al. [4], where adversarial examples were crafted to fool state-of-the-art classification DNNs. By far, a significant amount of research has been conducted to either design more powerful adversarial attacking methods to examine the robustness of different DNNs. The most classic and efficient attacking method is the Fast Gradient Sign Method (FGSM) [5], which only takes one single step of gradient ascent to maximize the model's classification error. An iterative version of FGSM was then proposed to enhance the attack strength in physical-world scenarios. This attack is known as Basic Iterative Method (BIM) [38]. Proposed by Madry et al. [7], the Projected Gradient Descent (PGD) has been recognized as one of the strongest first-order adversarial attacks. PGD iteratively perturbs the input sample x with a smaller step size and clips the perturbation back to an ϵ-ball around x if it goes beyond. The ϵ perturbation constraint is defined by the l_∞ norm. Other well-known and effective adversarial attacks include DeepFool [39], Carlini and Wagner (CW) attacks [6], Jacobian-based Saliency Map Approach (JSMA) [40], Momentum Iterative Attack [41], Distributionally Adversarial Attack [42] and Margin Decomposition (MD)

attacks [26]. Recently, Croce et al. [25] proposed the AutoAttack (AA) which is a parameter-free ensemble of four different adversarial attack methods. AA has been shown to be the most reliable attack for robustness evaluation.

B. Adversarial Defense

A wide range of defense methods have been proposed to improve DNN adversarial robustness, such as defensive distillation [43], adversarial detection [44], [21], input denoising [45], [46], gradient regularization [47], [48], [49], model compression [50], [51] and adversarial training (AT) [5], [7]. While some these defense methods are still vulnerable to adaptive attacks [24], [26], AT methods have been found to be the most reliable defense.

AT gains robustness by training the model on adversarial (instead of clean) examples at each training iteration [7], [27]. Training with the PGD adversarial examples is known as the Standard Adversarial Training (SAT) [7]. SAT is the defense method that for the first time can bring considerable robustness into DNNs. A number of variants of SAT have recently been proposed to further improved the robustness of SAT. Ensemble adversarial training [52] loosens the model's decision boundary and augments clean training examples with perturbations transferred from diverse models. TRADES [29] exploits the KL distance between the model's outputs on clean versus adversarial examples to generate stronger (than PGD) attacks, thus can train more robust models. The objective of TRADES has two loss terms with one is the commonly used CE loss defined on clean examples and the other one is the KL divergence between the model's output on clean versus adversarial examples. It uses a regularization hyperparameter λ to exert different trade-offs between the clean accuracy and adversarial robustness. According to recent evaluations, TRADES improves the robustness of SAT by $\sim 5\%$ on CIFAR-10, which is a very significant improvement. Recently, Wang et al. [30] proposed the Misclassification Aware adveRsarial Training (MART), which is a variant of TRADES that improves robustness by differentiating misclassified from correctly-classified examples.

The recent trend of adversarial training also involves exploiting deeper or wider network architectures for better adversarial robustness, such as WRN-34-10 [7], [29], [32], [33], [53], WRN-34-20 [34], [35], [36], [37] or even WRN-106-8 [54]. Compared to WRN-34-10, WRN-34-20 and WRN-106-8 consist of $\sim 4\times$ and $\sim 2.3\times$ parameters, respectively. This tends to incur a significant amount of more training time. In this work, we explore smarter ways to improve robustness by attacking separate heads/branches to existing networks rather than simply scaling up the entire architecture like WRN-34-20 or WRN-106-8. Based on our analysis, the proposed DH-AT only introduces $\sim 0.95\times$ more parameters to WRN-34-10 at the cost of a linearly increased training time by $\sim 1.2\times$. This has led to much more robustness improvement than scaling WRN-34-10 up to WRN-34-20 or WRN-106-8. Moreover, it has been shown that clean accuracy may be inherently at odds with adversarial robustness [55].

In this work, we propose the use of separate heads in DNNs to achieve two purposes: 1) combining different levels of robustness into one single model via different heads; and 2) better trade-off between clean accuracy and adversarial robustness with each head is responsible for only one property (*e.g.*, accuracy or robustness). This admits more flexible defense strategies in real-world scenarios, for example, the robustness head can be switched off when robustness is no longer the primary concern, or secure a higher level of robustness using the more robust head or both. The most relevant work to ours is the recent use of adversarial examples along with a separate auxiliary batch norm to improve image recognition [56]. However, we focus more on the output of the network and its adversarial robustness, and more importantly, in the adversarial not the clean training setting. From the ensemble perspective, WRN-34-20 is a costly ensemble of two WRN-34-10 models, however, our dual head strategy provides a smarter way for ensembling. In other words, improved robustness can be achieved by simply ensembling multiple heads (rather than the entire network) into one single network.

III. Dual Head DNNs and Dual Head Adversarial Training

In this section, we first explain our intuition for designing dual head models, then describe how to design a dual head model for RNs or WRNs. Finally, we introduce the proposed dual head adversarial training strategy.

A. Intuition for Using Two Heads

A recent study [34] shows that some AT methods heavily rely on early-stopping to achieve top-ranked robustness. Rice et al. [34] demonstrated that a similar performance gain can be achieved by a piece-wise learning rate scheduler along with smart early-stopping, which both can prevent "robust overfitting". In our investigation, TRADES [29] could achieve 55.88% robustness against PGD^{40} ($\epsilon = 8/255$) on CIFAR-10 dataset [34]. However, with only 15 more epochs of training after the best checkpoint, the robustness under the same attack drops by 2%+. This phenomenon leads us to inspect the differences in adversarial noises generated by PGD on models obtained at the best versus the last checkpoints. As shown in Fig. 1, the adversarial noises (right two columns) generated for the same clean image (left column) are notably different for models obtained at the 76th versus the 91st training epochs. This phenomenon is consistent across different CIFAR-10 test images. We also find that the adversarial noise generated on one model does not necessarily cause the same error on the other (see the prediction table in Fig. 1). This indicates that the two models have different levels of robustness and they are robust in distinctive ways. This phenomenon can also be expected to exist between models trained using different parameters. This motivates us to explore ways to combine different levels of robustness in one single model. Since robustness is more reliable at a later training stage when the shallow layers of the network are less likely to be significantly updated, we propose to use additional output branches (*i.e.*,
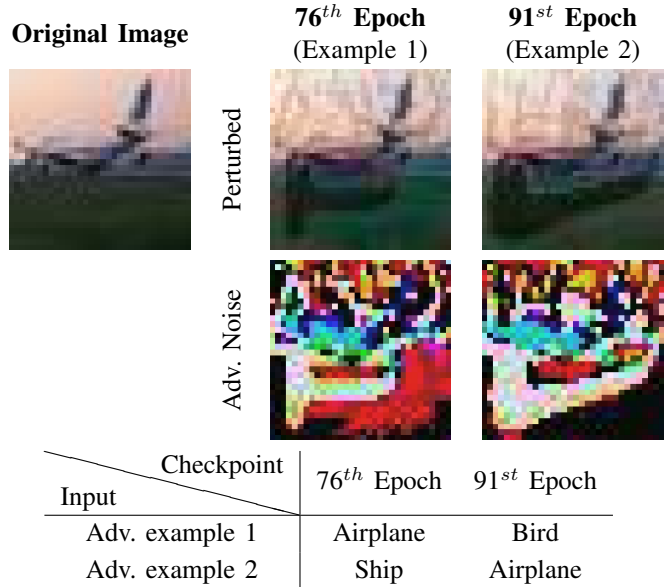

Original Image	**76th Epoch** (Example 1)	**91st Epoch** (Example 2)

Checkpoint / Input	76th Epoch	91st Epoch
Adv. example 1	Airplane	Bird
Adv. example 2	Ship	Airplane

Fig. 1: Adversarial examples (top row, right two columns) and noises (middle row, right two columns) generated by PGD^{100} ($\epsilon = 8/255$) for the same clean image (top row, left column) on different epochs of checkpoint. The model adopts WRN-34-10 trained with TRADES ($1/\lambda = 6$) on CIFAR-10 dataset (with image size 32×32), and the checkpoints are captured from the 76th and 91st epochs. The table reports the model's predictions on the two adversarial examples (true label is 'airplane'). The first adversarial example is generated from the 76th epoch, while the second example is generated from the same model at the 91st epoch. For better illustration, the adversarial noises are enhanced by $20\times$.

heads) to include more levels of robustness into a single network, *i.e.*, a dual head architecture.

B. Dual Head DNNs

We propose two types of duel head architectures: symmetric and asymmetric. The symmetric dual head architecture for WRN-34-10 is illustrated in Fig. 2. In this architecture, each head traverses the complete structure of a WRN-34-10 network, while sharing the first convolutional layer (Conv 1) and the first residual block (Block 1). In other words, the original model branches into two heads after the first residual block (Block 1), where we denote Block 1 as the "attaching point" of the model. To maximally employ convolutional layers that recognize higher-level features, the attaching point is preferably selected after the first group of repeated convolutional layers with identical filter arrangements. Particularly, for WRNs, we choose the output of Block 1 as the attaching point, regardless of the network's depth and width.

Asymmetric dual head architectures employ different depths of ResNets for the two heads. As illustrated in Fig. 3, the original ResNet-34 network serves as the main head and a revised ResNet-18 network is attached as the second head. Here, the attaching point is selected to be the output of the
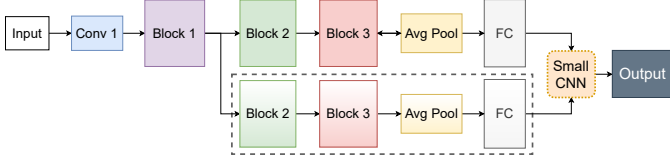
Fig. 2: The symmetric dual head architecture for WRN-34-10

first residual group (*i.e.*, the 4th convolutional block of the entire network). Note that, this asymmetric dual head version of WRN-34-10 employs all 3 residual blocks in the first residual group for both heads, while the standard ResNet-18 only employs 2 of them. After the attaching point, the second head (enclosed in the dashed box) has approximately 50% of the parameter size of the main head.
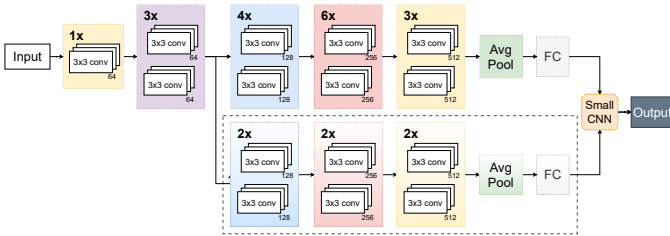

Fig. 3: The asymmetric dual head architecture for WRN-34-10

Different from conventional multi-head networks like the Siamese network [57], here we further combine the two heads into one single specifically designed output subnetwork. We design a lightweight CNN to aggregate the output from each head. The CNN subnetwork comprises two types of conventional filters: head-wise logits convolution (shown in Fig. 5) and class-wise logits convolution (shown in Fig. 6), where the logits refer to the output of the fully connected (FC) layers (no softmax). The logits output of each head (originally being two 1×10 arrays) will be merged into a $1\times10\times2$ tensor before feeding into the lightweight CNN (illustrated in the left part of Fig. 5). For both types of convolution, they use the same 1×2 convolutional kernels, except traversing through different dimensions.
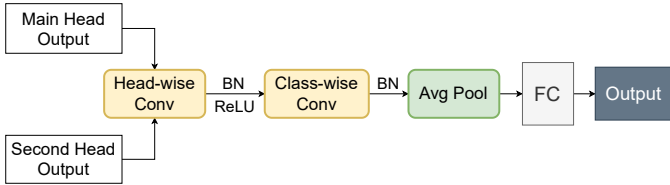

Fig. 4: The architecture of the lightweight CNN

The head-wise logits convolution applies 8 convolutional kernels with the size of $1\times1\times2$ to traverse across different classes. For each convolutional operation, it involves logits from each head regarding the same class (examples can be found in Fig. 5). Accordingly, a feature map of size $8\times10\times1$ will be generated for each input. This head-wise logits convolution is intuitively designed to associate the logits of each

class from the two heads. Then, batch normalization and ReLU activation are performed on the feature maps.
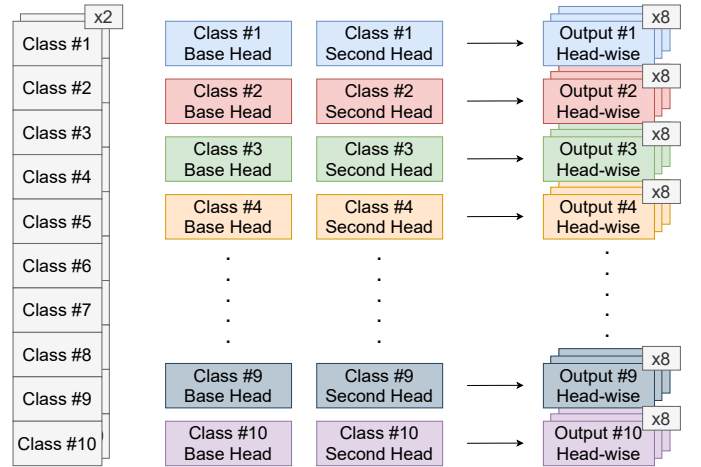

Fig. 5: Head-wise logits convolution

For the class-wise feature-map convolution, it utilizes 16 convolutional kernels with size $1\times2\times1$ to traverse the feature maps previously generated by the head-wise logits convolution. The class-wise feature-map convolution is different from ordinary convolutional kernels. Fig. 6 shows an example with 10 classes. It will be applied to logits in all 45 combinations of classes in the first dimension of the input feature map (*i.e.*, the dimension of the kernels in the previous convolution, namely the one with the size of 8). Accordingly, a new feature map of size $16\times8\times45\times1$ will be generated in this layer. The class-wise convolution enables the network to learn the class-wise correlations between clean and adversarial examples. For instance, a clean example with ground truth Class #3 may be predicted to be Class #7 after adversarial perturbation. Such class cross information is important for adversarial robustness, yet has been overlooked in existing adversarial training methods.
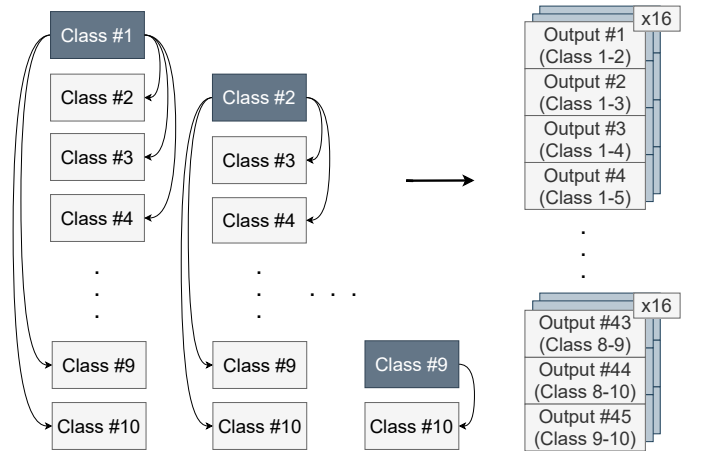

Fig. 6: Class-wise feature-map convolution

Then, we use an average pooling layer with stride 2 to reduce the feature maps' dimensionality along the second

dimension, the same dimension where class-wise feature-map convolution is applied (the one with size 16). After this pooling layer, the dimension of the feature maps will be reduced to $8 \times 8 \times 45$. Finally, the feature maps will be flattened to size 2880×1 before passing into a fully connected (FC) layer (with softmax activation) to output the final predictions.

C. Dual Head Adversarial Training

The training procedure of DH-AT associates the three components in a particular order: the main head, then the second head, and finally the lightweight CNN. Given a specific adversarial training method and a dual head DNN, DH-AT first adversarially trained the main head from scratch. Note that, any pre-trained model can be used as the main head. Next, we attach the second head to the main head at the specified attaching point and freeze the parameters of every component of the main head before the attaching point. We then train the second head using the same adversarial training method but with different hyperparameter settings, for example, by altering the attack intensity ϵ or changing loss hyperparameters (*e.g.*, in TRADES or MART). The last step of training is to train the lightweight CNN as follows:

$$\min_{f} \mathbb{E}\{CE(f(\boldsymbol{x}), \boldsymbol{y}) + \max_{\boldsymbol{x}' \in \mathbb{E}(\boldsymbol{x}, \epsilon)} KL(f(\boldsymbol{x}), f(\boldsymbol{x}'))/\lambda\}, \quad (1)$$

where $\boldsymbol{x}'$ is the adversarial example generated using PGD for clean example $\boldsymbol{x}$, and λ with $1/\lambda = 2$ is balancing parameter on the adversarial output of the lightweight CNN. Note that the above loss is also the loss function used in TRADES [29]. In this training stage, each training clean sample is paired with its PGD adversarial example which together is passed into the network in batches for model training. On CIFAR-10, when WRN-34-10 is used as the main head, the lightweight CNN can converge within 20 epochs of adversarial training.

Our dual head design and DH-AT training strategy provide a flexible way to keep two different levels of robustness in one single model. For example, keeping the best and the last checkpoints of the same model, or keeping models trained using different methods or with different hyper-parameters. Taking the best-last checkpoint case, for example, one can first train the main head using an existing adversarial training method, and at the same time, monitoring the validation robustness on a small validation set. When the validation robustness starts to drop, stop training the network and leave it as the main head. We can then copy all the weights to the second head, freeze the parameters of the main head before the attaching point and train the second head for a certain number of epochs until the training loss converges. We can then free both heads to train the lightweight CNN, again using the same adversarial training method. Note that, due to the special design of the lightweight merging CNN, the second head can be easily switched on and off to meet different application scenarios, which is also the case for the main head.

IV. EXPERIMENTS: SYMMETRIC DH-AT WITH WRN-34-10

We first test our symmetric DH-AT in which the two heads of the network are identical. All baseline models (SAT, TRADES, and MART) are trained following their settings specified in the original papers, except for utilizing 2 GPUs in parallel. During training, we apply PGD^{10} with $\epsilon = 8/255$ and step size $2/255$ to generate training adversarial examples. We evaluate all baseline and our models using 1) untargeted PGD attack, 2) targeted Auto-PGD (APGD) attack [25], and AutoAttack [25] which is an ensemble of 4 different attacks including the APGD. We follow existing works to use the same $\epsilon = 8/255$ for testing for all attacks. The step sizes for PGD attacks are set to $2.5 \cdot \epsilon/num\_steps$ when different perturbation steps are considered. The complete robustness results are summarized in Table I. Next, we will detail these results according to the adversarial training method used.

A. Detailed Experimental Settings

1) DH-AT with TRADES: To combine our DH-AT with TRADES, we investigate two different settings: 1) training the two heads with different λ hyperparameters, and 2) training the two heads by different numbers of epochs. Same as the original TRADES, each head in our DH-AT model is trained for 100 epochs using Stochastic Gradient Descent (SGD) with initial learning rate 0.1, momentum 0.9, and weight decay $2e^{-4}$. The learning rate is decayed by a factor of $1/10$ at the 75th, 90th, and 100th epochs.

2) Heads trained by TRADES with different λs: In this setting, the two heads are trained using TRADES with different λ hyperparameters. We denote the λ used by the main and second head as λ_1 and λ_2, and test two sets of combinations: 1) $1/\lambda_1 = 6$ and $1/\lambda_2 = 3$, and 2) $1/\lambda_1 = 3$ and $1/\lambda_2 = 2$. The training of the two heads follows the procedure described in Section III-C. With a small validation set (1000 CIFAR-10 test images), we find that the best checkpoint of TRADES is the 76th epoch. We then freeze the main head to the 76th epoch, and continue to train the second head using $1/\lambda_2 = 3$ for another 76 epochs. After this, we freeze both heads, attach the lightweight CNN and train the CNN for 15 epochs using TRADES loss with $1/\lambda = 2$ and a learning rate of 0.02. Note that, for the second experiment with $1/\lambda_1 = 3$ and $1/\lambda_2 = 2$, the best checkpoint is found to be the 78th epoch. Apart from these changes, the remaining procedures are identical to the first experiment. We take the best checkpoints of the standalone TRADES as our baselines. The results are reported in the top 2 - 5 rows in Table I (the first row is the standard TRADES result).

3) Heads trained by TRADES for different epochs: In this setting, we use the best checkpoint of TRADES as the main head, and a "robust overfitted" [34] subnetwork as the second head. The two heads are trained for different numbers of epochs, but using the same TRADES training technique with the same λ. Here, we test two different λs: $1/\lambda = 6$ and $1/\lambda = 3$. The best checkpoints under the two λs are the 76th

TABLE I: Robustness results of different defense methods under various parameter settings, on CIFAR-10 data. For all DH-AT models (WRN-34-10s), the left cell shows the results for the individual head (upper: main head; lower: second head), while the right cell shows the result for the final model. Robustness is evaluated on CIFAR-10 test set against $\epsilon = 8/255$ attacks.

	Natural		PGD^{40}		PGD^{100}		APGD-DLR-T		AutoAttack	
TRADES [29]	84.97%		55.88%		55.64%		53.10%		53.08%	
TRADES $1/\lambda = 6.0$ (76th Epoch)	84.97%		55.88%		55.64%		53.10%		53.08%	
TRADES $1/\lambda = 3.0$ (76th Epoch)	86.45%	86.78%	53.17%	**59.31%**	52.65%	**58.92%**	50.36%	**56.84%**	50.31%	**55.38%**
TRADES $1/\lambda = 3.0$ (78th Epoch)	86.97%		53.30%		52.92%		50.93%		50.80%	
TRADES $1/\lambda = 2.0$ (78th Epoch)	87.73%	**87.97%**	51.73%	57.34%	51.37%	57.01%	49.78%	54.25%	49.64%	53.41%
TRADES $1/\lambda = 6.0$ (76th Epoch)	84.97%		55.88%		55.48%		53.10%		53.08%	
TRADES $1/\lambda = 6.0$ (91st Epoch)	85.87%	85.93%	53.52%	57.89%	53.06%	57.58%	52.75%	54.63%	52.53%	54.27%
TRADES $1/\lambda = 3.0$ (78th Epoch)	86.97%		53.30%		52.92%		50.93%		50.80%	
TRADES $1/\lambda = 3.0$ (93th Epoch)	87.50%	87.63%	51.55%	56.08%	50.91%	55.62%	50.19%	53.03%	50.07%	52.84%
SAT [7]	87.32%		46.75%		45.26%		44.78%		44.23%	
SAT (160th Epoch)	87.32%		46.75%		45.26%		44.78%		44.23%	
SAT (175th Epoch)	**87.98%**	87.84%	45.82%	**47.81%**	45.01%	**47.24%**	44.52%	**45.48%**	43.96%	**45.06%**
MART [30]	84.15%		58.25%		57.78%		55.24%		54.68%	
MART $\lambda = 6.0$ (84th Epoch)	84.15%		58.25%		57.78%		55.04%		54.68%	
MART $\lambda = 3.0$ (84th Epoch)	85.48%	**85.51%**	56.83%	**59.94%**	56.17%	**59.36%**	53.97%	**56.23%**	53.11%	**55.75%**
MART $\lambda = 6.0$ (84th Epoch)	84.15%		58.25%		57.78%		55.24%		54.68%	
MART $\lambda = 6.0$ (99th Epoch)	84.29%	84.52%	57.23%	59.06%	56.84%	58.57%	54.60%	55.36%	54.02%	55.23%

and 78th epoch respectively. The results are reported in the top 6 - 9 rows in Table I.

4) DH-AT with SAT and MART: We also experiment on DH-AT with SAT [7] and MART [30] to demonstrate the compatibility of DH-AT to different adversarial training methods. Since SAT [7] loss function does not have tunable parameters, we only apply DH-AT with different numbers of epochs for the two heads. For the original SAT, we train the networks for 200 epochs using SGD with an initial learning rate 0.1, momentum 0.9, and weight decay $2e^{-4}$. The learning rate is divided by 10 at the 100th and 150th epochs. The best checkpoint at the 160th epoch is selected as the main head. The second head is trained for another 15 epochs. The results are shown in rows 10 - 12 in Table I.

Improved from TRADES, MART [30] also has a tuneable hyperparameter λ in its loss function. Following the above DH-AT experiment with TRADES, here we train the main head using $1/\lambda_1 = 6$ for 85 epochs, then train the second head with $1/\lambda_2 = 3$ for the same number of epochs. We also evaluate DH-AT on MART ($1/\lambda = 6$) with different numbers of epochs for the two heads, where the second head is trained for another 15 epochs after copied over from the main head, the best checkpoint obtained at the 85th epoch). The results are reported in the bottom 5 rows in Table I.

B. Results Analysis

Compared with standard TRADES ($1/\lambda = 6$), using DH-AT with the two heads having different hyperparameters $1/\lambda_1 = 6$ and $1/\lambda_2 = 3$ demonstrates a considerable robustness improvement by 3.43% against PGD^{40} and 2.30%

against AutoAttack [25] (top 5 rows in Table I). Moreover, the clean accuracy is also improved by more than 2%, which indicates that clean accuracy and robustness can be improved simultaneously. Note that, the training time of this type of DH-AT is approximately $2.2\times$ compared to standard TRADES. According to the results in rows 6 - 9 in Table I, applying our DH-AT with TRADES can effectively exploit the best and the last checkpoints, leading to 2.01% robustness boost against PGD^{40} and 1.19% against AutoAttack. This verifies the effectiveness of our proposed DH-AT for overcoming the overfitting issue in adversarial training. Note that, in this setting, the training time of DH-AT is approximately $\sim 1.4\times$ compared to standard TRADES. These two sets of experiments of DH-AT with TRADES demonstrate the flexibility of our DH-AT in different settings. It can be applied to incorporate two different levels of robustness or checkpoints into one single model.

Results in the bottom 8 rows in Table I demonstrate the good compatibility of our DH-AT strategy with different adversarial training methods. When utilizing MART with either $\lambda = 6$ or $\lambda = 3$ for both heads, our DH-AT method is able to deliver the highest adversarial robustness of 59.94% (improved by 1.69% compared to standard MART $\lambda = 6$) against PGD^{40} and 55.75% (improved by 1.07% compared to standard MART) against AutoAttack. Meanwhile, the clean accuracy is also improved compared to standalone SAT/MART, though only slightly. Since SAT uses the cross-entropy (CE) loss and the learning rate is decayed to 0.001 before the best checkpoint, its performance is relatively stable during additional training. However, applying our proposed DH-AT strategy can still

improve its robustness by 1.06% against PGD^{40} and 0.83% against AutoAttack.

V. Experiments: Asymmetric DH-AT with ResNet

We further evaluate our DH-AT strategy on the CIFAR-100 dataset using its asymmetric variant, *i.e.*, the main head is a ResNet-34 and the second head is a revised Resnet-18 (since it has one more residual block than its standard version). We select SAT and TRADES as the baseline adversarial training methods and report the best checkpoint's performance. For robustness evaluation, we use the PGD^{20} (no random restarts), PGD^{100} (with 5 random restarts), and AutoAttack (AA). All attacks are bounded by $\epsilon = 8/255$ and the step size of PGD (including its variants in AA) is set to $1/255$.

TABLE II: Robustness results on CIFAR-100 for different defense methods under various hyperparameter settings. All baseline methods (Natural, SAT and TRADES) use the ResNet-34 model. All DH-AT models utilize ResNet-34 as the main head and a revised ResNet-18 as the second head.

	Natural	PGD^{20}	PGD^{100}	AA
Natural Training	**78.56%**	0.02%	0.00%	0.00%
Natural (DH w/ TRADES)	73.86%	**14.89%**	**13.63%**	**10.65%**
Natural (DH w/ SAT)	74.64%	13.27%	12.39%	9.71%
TRADES [29]	58.13%	27.59%	26.37%	26.18%
TRADES (DH)	59.57%	**28.91%**	**27.28%**	**26.36%**
TRADES (DH w/ PGD^2)	**68.72%**	22.23%	21.31%	20.30%
SAT [7]	60.48%	24.48%	23.45%	23.31%
SAT (DH)	61.39%	**25.67%**	**24.41%**	**23.32%**
SAT (DH w/ PGD^2)	**71.25%**	19.86%	18.57%	16.09%

We train our DH-AT models using adversarial training (SAT or TRADES) with three different strengths for the main head: 1) natural (clean) training, PGD^2 and PGD^{10}. For PGD^{10} adversarial training, we follow the same setting as SAT or TRADES, while for PGD^2 adversarial training, the step size is set to $2/255$ with $\epsilon = 4/255$. Similar to previous experiments, we select the best checkpoint for the main head. For natural training, we train the ResNet-34 model for 40 epochs with the CE loss. Then, we add a second head and a lightweight CNN to this naturally-trained network, and train both of them using PGD^{10} adversarial training. The use of naturally-trained models as the main head is to demonstrate the effectiveness and practicability of our DH-AT strategy in more complex real-world defense scenarios. For example, the naturally-trained main head can be easily extracted from our DH-AT model to achieve high clean accuracy, while the adversarially-trained second head can be switched on to obtain robustness.

The results are reported in Table II. Compared with the naturally-trained model, using DH-AT with TRADES for the second head can improve the robustness from 0.02% to

14.89% against PGD^{20} attack, while the clean accuracy drops by only 4.7% to 73.86%. Additionally, higher robustness can be achieved by applying DH-AT with a stronger adversary for the main head. When using DH-AT with SAT where the main head is trained with a PGD^2 adversary, the clean accuracy can be improved by 10.77%, while the robustness drops by 4.62%. Ultimately, utilizing DH-AT on the standard TRADES can improve the robustness by 1.32%, and at the same time, improve the clean accuracy by 1.44%. While there is still large room for improvement, these results demonstrate the effectiveness and flexibility of our DH-AT strategy to meet the diverse accuracy and robustness requirements of real-world applications.

VI. Conclusion

In this work, we proposed a Dual Head Adversarial Training (DH-AT) strategy to combine different levels of adversarial robustness into one single model. DH-AT introduces both architectural and training modifications to existing deep neural networks (DNNs) and adversarial training methods. The two heads in DH-AT models can be trained differently to improve the overall robustness, while maintaining or slightly improving clean accuracy. We showed that our DH-AT strategy can be readily implemented into different DNNs and adversarial training methods with minimal modifications. Our proposed DH-AT strategy can be used as a practical tool to obtain both clean accuracy and adversarial robustness, or different levels of accuracy-robustness trade-off in a single model.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, pp. 770–778, 2016.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[3] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *ICML*, pp. 173–182, PMLR, 2016.

[4] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *ICLR*, 2014.

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

[6] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*, pp. 39–57, IEEE, 2017.

[7] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *ICLR*, 2018.

[8] D. Wu, Y. Wang, S.-T. Xia, J. Bailey, and X. Ma, "Skip connections matter: On the transferability of adversarial examples generated with resnets," *ICLR*, 2020.

[9] L. Jiang, X. Ma, S. Chen, J. Bailey, and Y.-G. Jiang, "Black-box adversarial attacks on video recognition models," in *ACM MM*, pp. 864–872, 2019.

[10] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, "Adversarial attack on graph structured data," in *ICML*, pp. 1115–1124, PMLR, 2018.

[11] L. Sun, K. Hashimoto, W. Yin, A. Asai, J. Li, P. Yu, and C. Xiong, "Adv-bert: Bert is not robust on misspellings! generating nature adversarial samples on bert," *arXiv preprint arXiv:2003.04985*, 2020.

[12] G. Goswami, N. Ratha, A. Agarwal, R. Singh, and M. Vatsa, "Unravelling robustness of deep learning based face recognition against adversarial attacks," in *AAAI*, vol. 32, 2018.

[13] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *CVPR*, pp. 1625–1634, 2018.

[14] Z. Kong, J. Guo, A. Li, and C. Liu, "Physgan: Generating physical-world-resilient adversarial examples for autonomous driving," in *CVPR*, pp. 14254–14263, 2020.

[15] R. Duan, X. Ma, Y. Wang, J. Bailey, A. K. Qin, and Y. Yang, "Adversarial camouflage: Hiding physical-world attacks with natural styles," in *CVPR*, pp. 1000–1008, 2020.

[16] J. Tu, M. Ren, S. Manivasagam, M. Liang, B. Yang, R. Du, F. Cheng, and R. Urtasun, "Physically realizable adversarial examples for lidar object detection," in *CVPR*, pp. 13716–13725, 2020.

[17] Y. Cao, C. Xiao, D. Yang, J. Fang, R. Yang, M. Liu, and B. Li, "Adversarial objects against lidar-based autonomous driving systems," *arXiv preprint arXiv:1907.05418*, 2019.

[18] S. G. Finlayson, J. D. Bowers, J. Ito, J. L. Zittrain, A. L. Beam, and I. S. Kohane, "Adversarial attacks on medical machine learning," *Science*, vol. 363, no. 6433, pp. 1287–1289, 2019.

[19] X. Ma, Y. Niu, L. Gu, Y. Wang, Y. Zhao, J. Bailey, and F. Lu, "Understanding adversarial attacks on deep learning based medical image analysis systems," *Pattern Recognition*, vol. 110, p. 107332, 2021.

[20] C. Guo, M. Rana, M. Cissé, and L. van der Maaten, "Countering adversarial images using input transformations," in *ICLR*, 2018.

[21] X. Ma, B. Li, Y. Wang, S. M. Erfani, S. Wijewickrema, G. Schoenebeck, D. Song, M. E. Houle, and J. Bailey, "Characterizing adversarial subspaces using local intrinsic dimensionality," *ICLR*, 2018.

[22] G. S. Dhillon, K. Azizzadenesheli, Z. C. Lipton, J. Bernstein, J. Kossaifi, A. Khanna, and A. Anandkumar, "Stochastic activation pruning for robust adversarial defense," in *ICLR*, 2018.

[23] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. L. Yuille, "Mitigating adversarial effects through randomization," in *ICLR*, 2018.

[24] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *ICML*, pp. 274–283, 2018.

[25] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," *arXiv preprint arXiv:2003.01690*, 2020.

[26] L. Jiang, X. Ma, Z. Weng, J. Bailey, and Y.-G. Jiang, "Imbalanced gradients: A new cause of overestimated adversarial robustness," *arXiv preprint arXiv:2006.13726*, 2020.

[27] Y. Wang, X. Ma, J. Bailey, J. Yi, B. Zhou, and Q. Gu, "On the convergence and robustness of adversarial training," in *ICML*, pp. 6586–6595, 2019.

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

[29] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan, "Theoretically principled trade-off between robustness and accuracy," *ICML*, 2019.

[30] Y. Wang, D. Zou, J. Yi, J. Bailey, X. Ma, and Q. Gu, "Improving adversarial robustness requires revisiting misclassified examples," in *ICLR*, 2020.

[31] B. Wu, J. Chen, D. Cai, X. He, and Q. Gu, "Does network width really help adversarial robustness?," *arXiv preprint arXiv:2010.01279*, 2020.

[32] L. Huang, C. Zhang, and H. Zhang, "Self-adaptive training: beyond empirical risk minimization," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[33] D. Wu, S.-T. Xia, and Y. Wang, "Adversarial weight perturbation helps robust generalization," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[34] L. Rice, E. Wong, and Z. Kolter, "Overfitting in adversarially robust deep learning," in *ICML*, pp. 8093–8104, PMLR, 2020.

[35] T. Pang, X. Yang, Y. Dong, H. Su, and J. Zhu, "Bag of tricks for adversarial training," *arXiv preprint arXiv:2010.00467*, 2020.

[36] T. Pang, X. Yang, Y. Dong, K. Xu, H. Su, and J. Zhu, "Boosting adversarial training with hypersphere embedding," *arXiv preprint arXiv:2002.08619*, 2020.

[37] J. Cui, S. Liu, L. Wang, and J. Jia, "Learnable boundary guided adversarial training," *arXiv preprint arXiv:2011.11164*, 2020.

[38] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[39] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *CVPR*, pp. 2574–2582, 2016.

[40] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *EuroS&P*, pp. 372–387, IEEE, 2016.

[41] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *CVPR*, pp. 9185–9193, 2018.

[42] T. Zheng, C. Chen, and K. Ren, "Distributionally adversarial attack," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 2253–2260, 2019.

[43] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *SP*, pp. 582–597, 2016.

[44] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *NDSS*, 2017.

[45] C. Guo, M. Rana, M. Cissé, and L. van der Maaten, "Countering adversarial images using input transformations," in *ICLR*, 2018.

[46] Z. Niu, Z. Chen, L. Li, Y. Yang, B. Li, and J. Yi, "On the limitations of denoising strategies as adversarial defenses," *arXiv preprint arXiv:2012.09384*, 2020.

[47] A. Ross and F. Doshi-Velez, "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[48] D. Jakubovitz and R. Giryes, "Improving dnn robustness to adversarial attacks using jacobian regularization," in *ECCV*, pp. 514–529, 2018.

[49] C. Finlay and A. M. Oberman, "Scaleable input gradient regularization for adversarial robustness," *Machine Learning with Applications*, vol. 3, p. 100017, 2021.

[50] S. Ye, K. Xu, S. Liu, H. Cheng, J.-H. Lambrechts, H. Zhang, A. Zhou, K. Ma, Y. Wang, and X. Lin, "Adversarial robustness vs. model compression, or both?," in *ICCV*, pp. 111–120, 2019.

[51] Y. Guo, C. Zhang, C. Zhang, and Y. Chen, "Sparse dnns with improved adversarial robustness," in *NeurIPS*, 2018.

[52] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[53] Y. Bai, Y. Zeng, Y. Jiang, S.-T. Xia, X. Ma, and Y. Wang, "Improving adversarial robustness via channel-wise activation suppressing," in *ICLR*, 2021.

[54] J.-B. Alayrac, J. Uesato, P.-S. Huang, A. Fawzi, R. Stanforth, and P. Kohli, "Are labels required for improving adversarial robustness?," in *Advances in Neural Information Processing Systems*, pp. 12214–12223, 2019.

[55] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," *ICLR*, 2019.

[56] C. Xie, M. Tan, B. Gong, J. Wang, A. L. Yuille, and Q. V. Le, "Adversarial examples improve image recognition," in *CVPR*, pp. 819–828, 2020.

[57] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, no. 04, pp. 669–688, 1993.