## Appendices for *Towards Defending against Adversarial Examples via Attack-Invariant Features*

# A. Attack implementation

In this section, we present supplementary information on attack implementation. We use Advertorch Toolbox[1] to implement the projected gradient descent method (PGD), the decoupling direction and norm method (DDN), the Carlini and Wagner method (CW), the Jacobian-based saliency map attack method (JSMA) and the spatial transform attack method (STA). The autoattack method (AA)[2], the faster wasserstein attac method (FWA)[3] and the robust physical perturbation method ($RP_2$)[4] are implemented from their open source codes. On *MNIST*, the main parameters of attacks are as follows:

- **PGD**: We use the $L_\infty$ norm PGD method to craft adversarial examples. The default perturbation budget is set to 0.3. The default number of iterations is set to 40. The attack step size is set to 0.01.
- **DDN**: The number of iterations is set to 100. The factor to modify the norm at each iteration is set to 0.05. The number of quantization levels is set to 256.
- **CW**: We use the $L_2$ norm CW method to craft adversarial examples. The maximum number of iterations is set to 1000. The confidence of the adversarial examples is set to 1. The initial value of the constant is set to 1.
- **JSMA**: The highest percentage of pixels can be modified is set to 1.0. The perturb length is set to 1.0.
- **STA**: The maximum number of iterations is set to 500. The number of search times to find the optimum is set to 20.
- **AA**: The default perturbation budget is set to 0.3. The default number of iterations is set to 100.
- **FWA**: The wasserstein adversarial examples are crafted by exploiting PGD. The default perturbation budget is set to 0.3. The number of iterations is set to 40. The learning rate is set to 0.1.

On *CIFAR-10*, the main parameters of attacks are as follows:

[1] https://github.com/codeaudit/advertorch
[2] https://github.com/fra31/auto-attack
[3] https://github.com/watml/fast-wasserstein-adversarial
[4] https://github.com/tongwu2020/phattacks/tree/master/sign/experiment

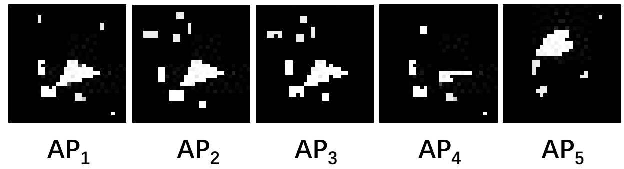


*Figure 1.* A visual illustration of five masks used to craft adversarial patches (AP).

- **PGD**: We use the $L_\infty$ norm PGD method to craft adversarial examples. The default perturbation budget is set to 8/255. The default number of iterations is set to 40. The attack step size is set to 0.01.
- **DDN**: The number of iterations is set to 100. The factor to modify the norm at each iteration is set to 0.05. The number of quantization levels is set to 256.
- **CW**: We use the $L_2$ norm CW method to craft adversarial examples. The maximum number of iterations is set to 500. The confidence of the adversarial examples is set to 1. The initial value of the constant is set to 1.
- **JSMA**: The highest percentage of pixels can be modified is set to 1.0. The perturb length is set to 1.0.
- **STA**: The maximum number of iterations is set to 200. The number of search times to find the optimum is set to 20.
- **AA**: The default perturbation budget is set to 8/255. The default number of iterations is set to 100.
- **FWA**: The wasserstein adversarial examples are crafted by exploiting PGD. The default perturbation budget is set to 8/255. The number of iterations is set to 40. The learning rate is set to 0.01.

On *LISA*, we use five different masks to implement $RP_2$ for crafting adversarial examples. The masks are shown in Figure 1. The number of iterations of $RP_2$ is set to 300 and the learning rate is set to 0.01.

# B. Defense results

In this section, we present supplementary information on defense results. We use two different combinations of seen types of attacks to train our ARN model: (i) the targeted

PGD and the non-targeted PGD ("ARN$_{PP}$"). (ii) the non-targeted DDN and the non-targeted PGD ("ARN$_{DP}$"). Figure 2, 3 shows examples which are restored by our *adversarial noise removing network* (ARN) against pixel-constrained attacks on *MNIST* and *CIFAR-10*. These attacks include non-targeted $L_\infty$ norm PGD (PGD$_N$), targeted $L_\infty$ norm PGD (PGD$_T$), non-targeted DDN (DDN$_N$), non-targeted $L_2$ norm CW (CW$_N$), targeted JSMA (JSMA$_T$) and non-targeted AA (AA$_N$). Figure 4 shows examples which are restored by our ARN against spatial-constrained attacks. These attacks include non-targeted STA (STA$_N$), targeted STA (STA$_T$), non-targeted FWA (FWA$_N$) and non-targeted RP$_2$ (RP$_N$). Figure 5 and 6 show adversarial examples and restored examples on *LISA*. Five types of adversarial patches (AP) are crafted by RP$_2$ and are added to natural examples to generate adversarial examples. We use adversarial examples with two types of adversarial patches (AP$_1$ and AP$_2$) as training data to train our ARN model. The categories corresponding to the class labels in *CIFAR-10* are as follows: 0) airplane, 1) car, 2) bird, 3) cat, 4) deer, 5) dog, 6) frog, 7) horse, 8) boat and 9) truck.

## C. Leaked defenses

In this section, we present supplementary information on defending under difficult scenarios where defenses are leaked. We use ARN$_{PP}$, APE-G$_{PP}$ and HGD$_{PP}$ as the leaked defense models to craft adversarial examples by distinct attacks. Figure 7 shows defense results of our ARN against BPDA. The adversarial examples are crafted by jointly using PGD$_N$ and BPDA against our ARN. Figure 8, 9 and 10 show defense results of our ARN against white-box and gray-box adaptive attacks. To be specific, the leaked defense in Figure 8 is our ARN$_{PP}$ and the attack is PGD$_T$. The leaked defense in Figure 9 is APE$_{PP}$ and the attacks are PGD$_T$, CW$_N$ and CW$_T$. The leaked defense in Figure 10 is HGD$_{PP}$ and the attacks are PGD$_T$, DDN$_N$ and DDN$_T$.


*Figure 2.* A visual illustration of adversarial examples and their restored examples. These adversarial examples are crafted by pixel-constrained attacks.


*Figure 3.* A visual illustration of adversarial examples and their restored examples. These adversarial examples are crafted by pixel-constrained attacks.


*Figure 4.* A visual illustration of adversarial examples and their restored examples. These adversarial examples are crafted by spatially-constrained attacks.

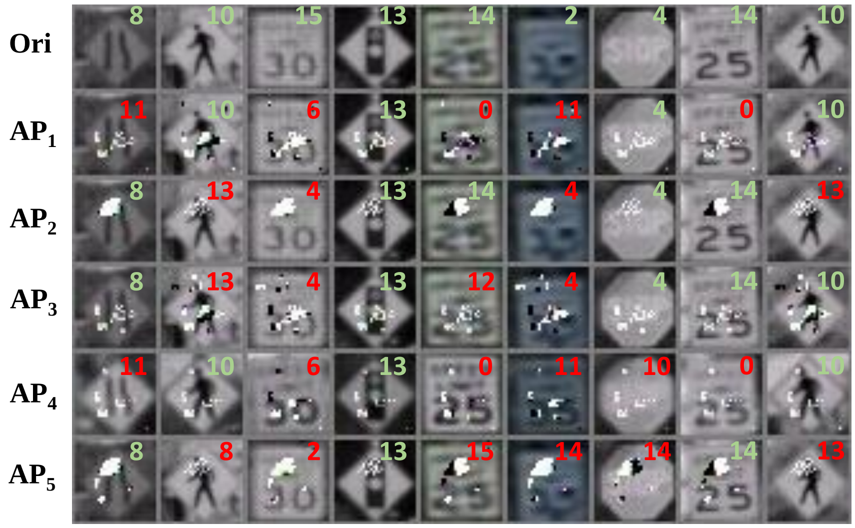


*Figure 5.* A visual illustration of adversarial examples on *LISA*. Five types of adversarial patches (AP) are crafted by $RP_2$ and are added to natural examples to generate adversarial examples.

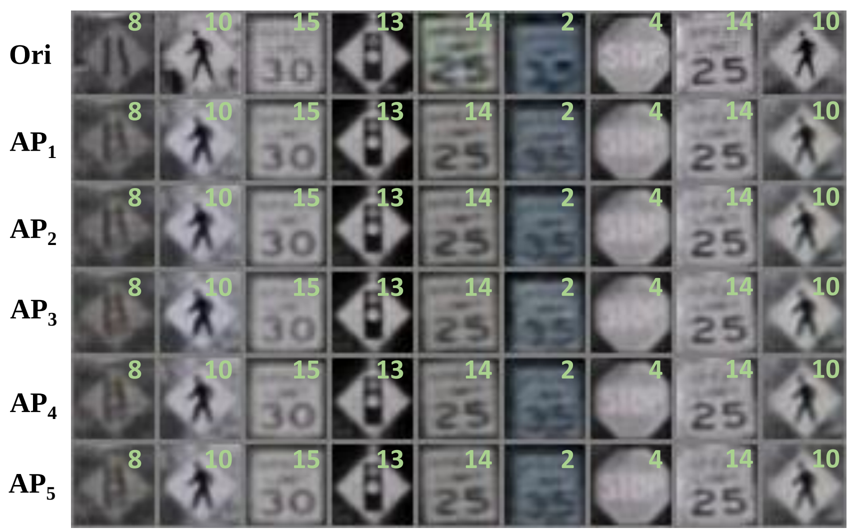


*Figure 6.* A visual illustration of restored examples on *LISA*. Five types of adversarial patches (AP) are crafted by $RP_2$ and are added to natural examples to generate adversarial examples.

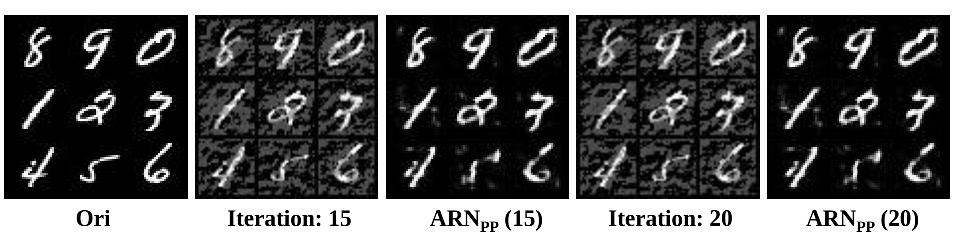


*Figure 7.* A visual illustration of adversarial examples and their restored examples against BPDA. The adversarial examples are crafted by jointly using BPDA and $PGD_N$ against our ARN. The number of iterations of $PGD_N$ is 15 and 20 respectively.

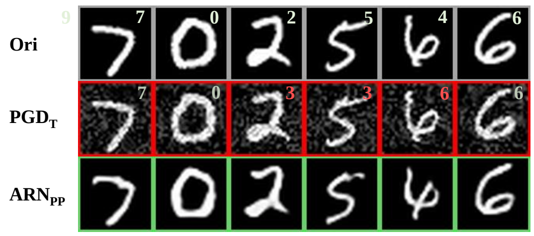


*Figure 8.* A visual illustration of defense results of our ARN against the white-box adaptive attack.

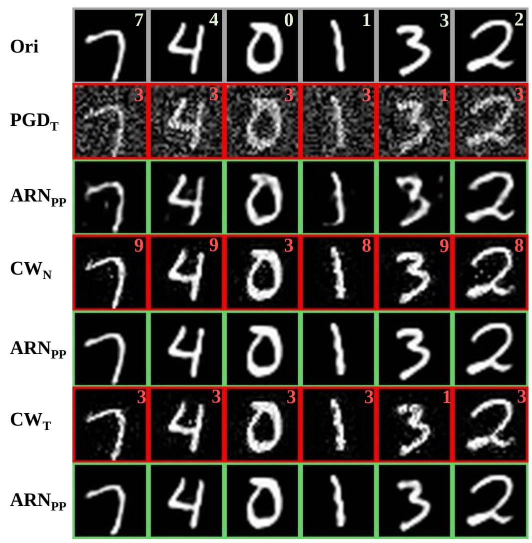


*Figure 9.* A visual illustration of defense results of our ARN against the gray-box adaptive attack. The local defense model is $\text{APE}_{PP}$.

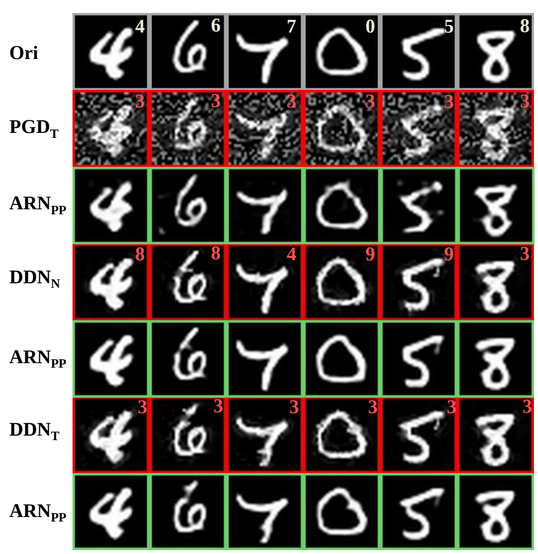


*Figure 10.* A visual illustration of defense results of our ARN against the gray-box adaptive attack. The local defense model is $\text{HGD}_{PP}$.