Appendix 1: Derivation

1.A Proof of Theorems

Proof of Theorem 1. Assume $p(\delta)$ is any distribution that only supports in Δ,

$$\mathscr{L}_{AR}^{P} = \mathop{\mathbb{E}}_{(X,Y)\sim D} \int_{\Delta} \mathbb{E}_{X'\sim\pi_\omega(X'|X+\delta)}\big[\mathscr{L}(X',Y;\theta)\big]p(\delta)\mathrm{d}\delta \tag{1a}$$

$$= \mathop{\mathbb{E}}_{(X,Y)\sim D} \int_{\Delta} p(\delta)\mathrm{d}\delta \int \mathrm{d}X'\pi_\omega(X' \mid X+\delta)\mathscr{L}(X',Y;\theta) \tag{1b}$$

$$= \mathop{\mathbb{E}}_{(X,Y)\sim D} \int_{\Delta} p(\delta)\mathrm{d}\delta \int \mathrm{d}X'\delta_{Dirac}(X'-X)\mathscr{L}(X',Y;\theta) \tag{1c}$$

$$= \mathop{\mathbb{E}}_{(X,Y)\sim D} \int_{\Delta} p(\delta)\mathrm{d}\delta\mathscr{L}(X,Y;\theta) \tag{1d}$$

$$= \mathop{\mathbb{E}}_{(X,Y)\sim D} \mathscr{L}(X,Y;\theta) \tag{1e}$$

□

Proof of Theorem 2. Since $\mathscr{L}(X+\beta,Y;\theta)$ is continuous and Δ is compact, $\beta_0 = \arg\min_{\beta\in\Delta}\mathscr{L}(X+\beta,Y;\theta)$ exists.

Assume $p(\delta)$ is any distribution that only supports in Δ and π_ω supports in $X+\Delta$,

$$\mathscr{L}_{AR}^{P} = \mathop{\mathbb{E}}_{(X,Y)\sim D} \int_{\Delta} \mathbb{E}_{X'\sim\pi_\omega(X'|X+\delta)}\big[\mathscr{L}(X',Y;\theta)\big]p(\delta)\mathrm{d}\delta \tag{2a}$$

$$= \mathop{\mathbb{E}}_{(X,Y)\sim D} \int_{\Delta} p(\delta)\mathrm{d}\delta \int_{X+\Delta} \mathrm{d}X'\pi_\omega(X' \mid X+\delta)\mathscr{L}(X',Y;\theta) \tag{2b}$$

$$\geq \mathop{\mathbb{E}}_{(X,Y)\sim D} \int_{\Delta} p(\delta)\mathrm{d}\delta \int_{X+\Delta} \mathrm{d}X'\pi_\omega(X' \mid X+\delta)\Big(\min_{X'\in\Delta}\mathscr{L}(X',Y;\theta)\Big) \tag{2c}$$

$$= \mathop{\mathbb{E}}_{(X,Y)\sim D} \int_{\Delta} p(\delta)\mathrm{d}\delta \int_{X+\Delta} \mathrm{d}X'\pi_\omega(X' \mid X+\delta)\mathscr{L}(X+\beta_0,Y;\theta) \tag{2d}$$

$$= \mathop{\mathbb{E}}_{(X,Y)\sim D} \int_{\Delta} p(\delta)\mathrm{d}\delta\mathscr{L}(X+\beta_0,Y;\theta) \tag{2e}$$

$$= \mathop{\mathbb{E}}_{(X,Y)\sim D} \mathscr{L}(X+\beta_0,Y;\theta) \tag{2f}$$

The equality is satisfied when $\pi_\omega(X'|X+\delta) = \delta_{Dirac}(X'-X-\beta_0)$. □

Appendix 2: End-to-End Joint Adversarial Attack

So far we have shown that our approach is able to defend a black-box CNN against adversarial attacks by transforming the perturbed images from its adversarial distribution space back to an adversarial-free zone around the original training/testing data distribution. A counter argument to this is whether an end-to-end attack jointly against the agent and the black-box CNN is possible. The answer to this is, yes, it is theoretically possible to perform an end-to-end joint attack but, empirically it is very expensive and time consuming. The reason for this is that, with the inclusion of the probabilistic generative network (PixelCNN) along with the black-box CNN, the system changes from being a deterministic system to a probabilistic system. In order to attack a probabilistic system, one needs to solve the stochastic gradient descent. The convergence rate for solving this is in the order of $O(1/\lambda)$, where λ is the convergence error. This is exponentially slower than the deterministic case and we verify this by crafting adversarial perturbations using the FGSM attack with $\epsilon_{\mathbf{attack}}$ = 16 against the joint system (ShieldNet + VGG) on the CIFAR10 dataset. We chose the FGSM attack because it is very fast in creating adversarial perturbations. It took approximately 27 hours to create 50 adversarial images using the NVIDIA Digits DevBox which consists of 4 TITAN X GPUs. Comparing this to the deterministic case, it takes just approximately 1 minute to generate 50 FGSM adversarial examples against the CNN (VGG). Moreover, the black-box CNN and the agent were trained separately and have independent parameters, hence a perturbation in a certain direction leading to a mis-classification in the black-box CNN does not necessarily lead to a similar perturbation in the same direction for the agent.

Appendix 3: Architecture of the Black-box CNN

3.A Architecture of ResNet for the CIFAR10 and Fashion MNIST datasets

Name	Configuration of the layer	
Initial Layer	Conv (filter size: 3 x 3, feature maps: 16(4), stride size: 1 x 1)	
Residual Block 1	Batch normalization & leaky relu Conv (filter size: 3 x 3, feature maps: 16(4), stride size: 1 x 1) Batch normalization & leaky relu Conv (filter size: 3 x 3, feature maps: 16(4), stride size: 1 x 1) Residual addition	**x10 times**
Residual block 2	Batch normalization & leaky relu Conv (filter size: 3 x 3, feature maps: 32(8), stride size: 2 x 2) Batch normalization & leaky relu Conv (filter size: 3 x 3, feature maps: 32(8), stride size: 1 x 1) Average pooling & padding & residual addition	
	Batch normalization & leaky relu Conv (filter size: 3 x 3, feature maps: 32(8), stride size: 1 x 1) Batch normalization & leaky relu Conv (filter size: 3 x 3, feature maps: 32(8), stride size: 1 x 1) Residual addition	**x9 times**
Residual block 3	Batch normalization & leaky relu Conv (filter size: 3 x 3, feature maps: 64(16), stride size: 2 x 2) Batch normalization & leaky relu Conv (filter size: 3 x 3, feature maps: 64(16), stride size: 1 x 1) Average pooling & padding & residual addition Batch normalization & leaky relu	
	Conv (filter size: 3 x 3, feature maps: 64(16), stride size: 1 x 1) Batch normalization & leaky relu Conv (filter size: 3 x 3, feature maps: 64(16), stride size: 1 x 1) Residual addition	**x9 times**
Pooling layer	Batch normalization & leaky relu & average pooling	
Output layer	fc_10 & Softmax	

3.B Architecture of VGG for the CIFAR10 and Fashion MNIST datasets

Name	Configuration of the layer	
Feature block 1	Conv (filter size: 3 x 3, feature maps: 16 (4), stride size: 1 x 1) Batch normalization & relu	**x2 times**
	Max pooling (stride size: 2 x 2)	
Feature block 2	Conv (filter size: 3 x 3, feature maps: 128 (32), stride size: 1 x 1) Batch normalization & relu	**x2 times**
	Max pooling (stride size: 2 x 2)	
Feature block 3	Conv (filter size: 3 x 3, feature maps: 512 (128), stride size: 1 x 1) Batch normalization & relu	**x3 times**
	Max pooling (stride size: 2 x 2)	
Feature block 4	Conv (filter size: 3 x 3, feature maps: 512 (128), stride size: 1 x 1) Batch normalization & relu	**x3 times**
	Max pooling (stride size: 2 x 2) & flatten	
Classifier block	Dropout & fc_512 (128) & relu Dropout & fc_10 & softmax	

* The same architecture of the CNN is used for both CIFAR10 and Fashion MNIST datasets, but different number of feature maps are used. The number of feature maps in parenthesis is for Fashion MNIST.

Appendix 4: Original and Generated Images

4.A Examples of original and generated images of the CIFAR10 dataset

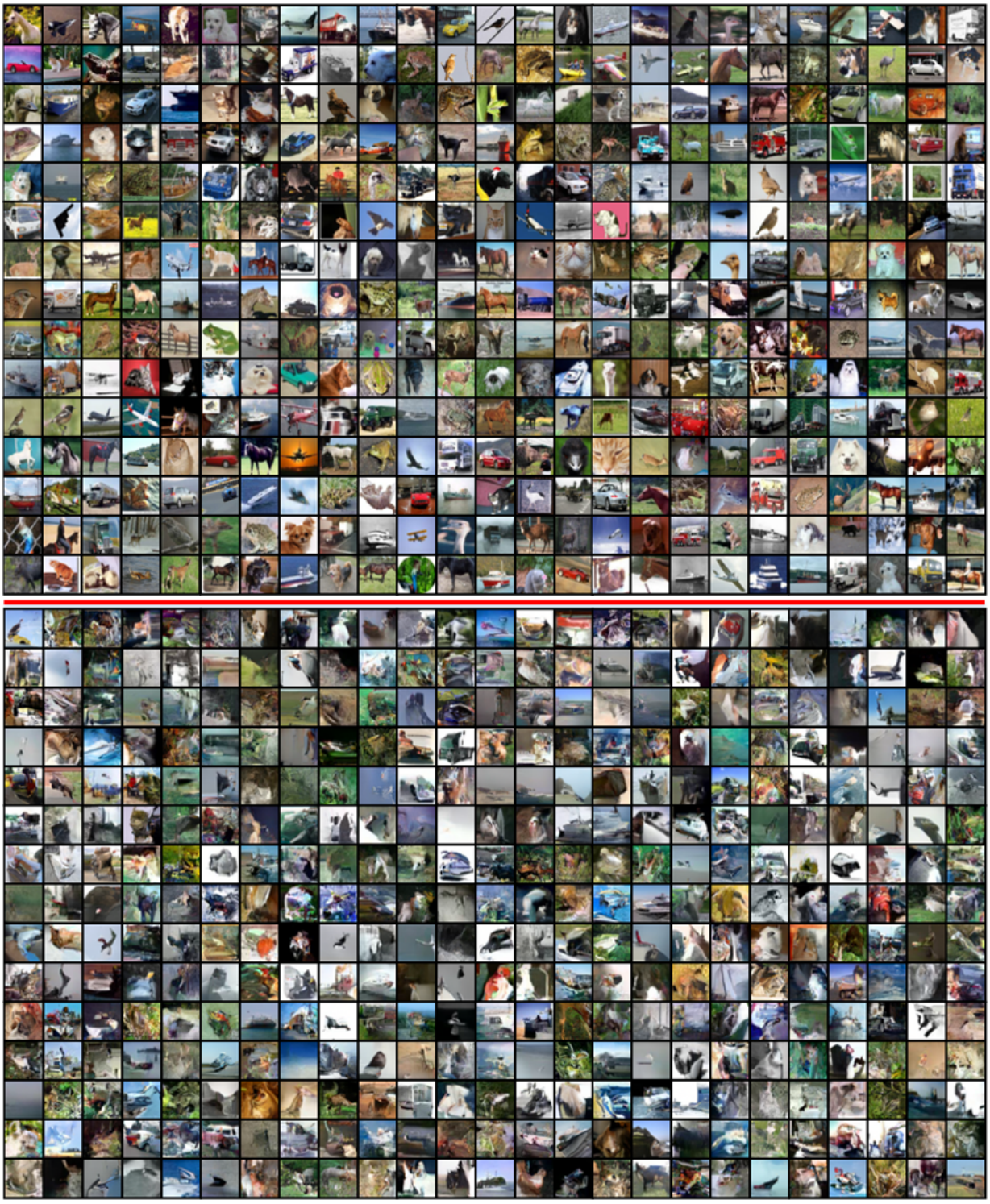

Figure 1. The images above the red line are original images from the CIFAR10 dataset. The images below the red line are images generated by the PixelCNN model.

4.B Examples of original and generated images of the Fashion MNIST dataset

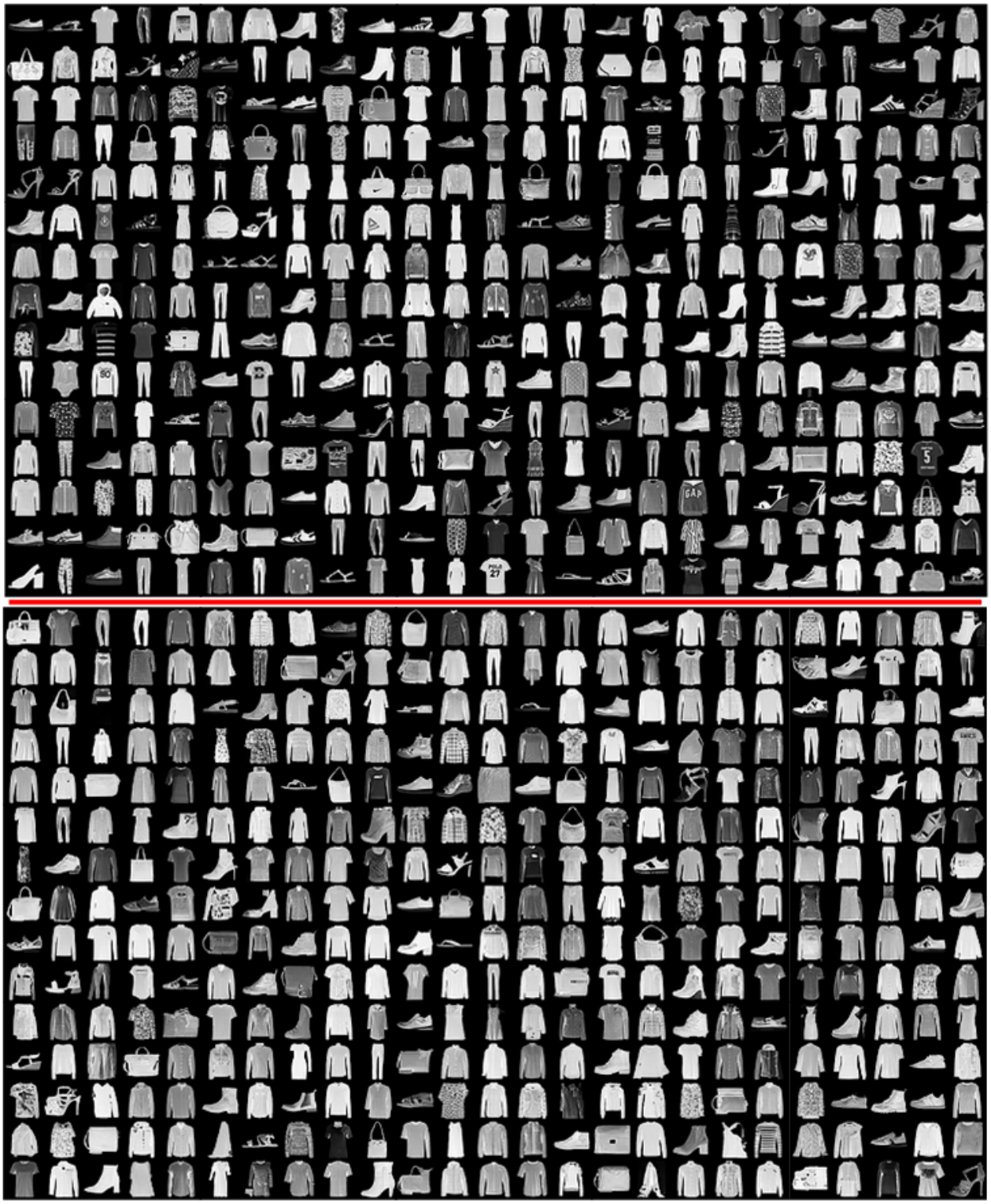

Figure 2. The images above the red line are original images from the Fashion MNIST dataset. The images below the red line are images generated by the PixelCNN model.

Appendix 5: Perturbed and Purified Images

5.A Examples of Perturbed and Purified images of the CIFAR10 dataset

Figure 3. The images above the red line are perturbed images from the CIFAR10 dataset. The images below the red line are the corresponding purified images. The perturbed images were created using the BIM attack with $\epsilon_{\mathbf{attack}} = 16$

5.B Examples of Perturbed and Purified images of the Fashion MNIST dataset

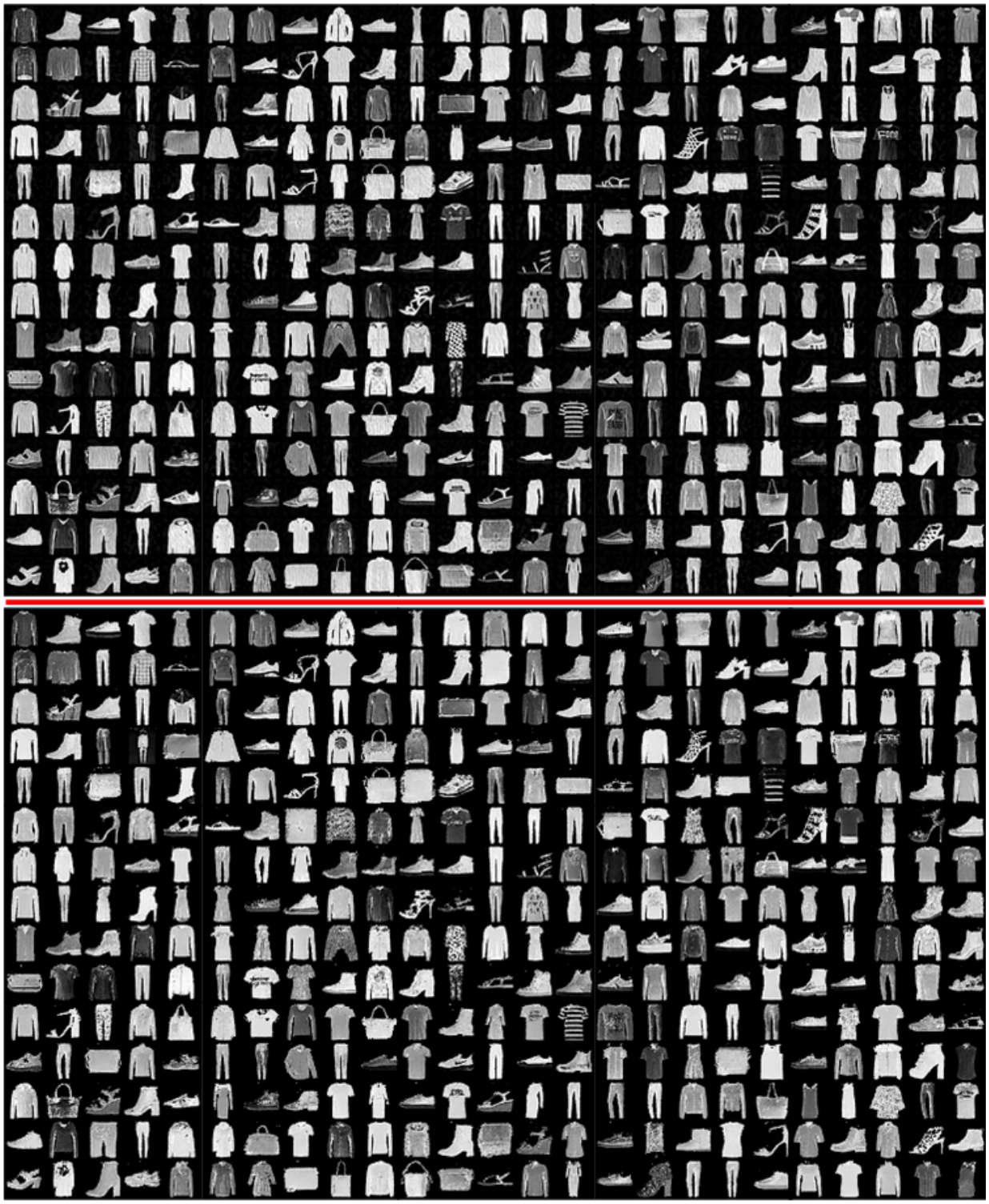

Figure 4. The images above the red line are perturbed images from the Fashion MNIST dataset. The images below the red line are the corresponding purified images. The perturbed images were created using the BIM attack with $\epsilon_{\mathtt{attack}} = 25$