

Appendix

A Implementation Details

In this section, we delve into the specifics of this study’s implementation. The related source code has been provided within the supplementary materials. In case of any remaining uncertainties, please refer to the included source code for further clarity.

A.1 Victim Classifiers

We utilize two distinct classifiers as the victim classifiers, each with a specific dataset: MadryNet [12] with MNIST, and ResNet18 [8] with the SVHN dataset.

A.1.1 MadryNet

We utilize MadryNet [12], a convolutional neural network (CNN), as the victim classifier for the MNIST dataset. The MadryNet model is trained using the Adam optimizer, with a learning rate set at 10^{-4} . Our training regimen comprises 14 epochs, with a batch size of 64. Any additional hyperparameters are retained at their default settings as prescribed by PyTorch.

During adversarial training on MadryNet, we implement a Projected Gradient Descent (PGD) untargeted attack on the training data, using the parameters $\epsilon = 0.3$, $\alpha = 0.036$, and 10 steps.

A.1.2 ResNet18

In the case of the SVHN dataset, we employ ResNet18 [8], another well-known architecture. The ResNet18 model is trained using the Adam optimizer, with a learning rate set at 10^{-4} . Our training regimen comprises 14 epochs, with a batch size of 64. Any additional hyperparameters are retained at their default settings as prescribed by PyTorch. Similarly to the MadryNet, adversarial training on ResNet18 also involves a PGD untargeted attack on the training data, but with different parameters: $\epsilon = 0.03$, $\alpha = 0.01$, and 10 steps.

A.2 Energy-based models

A.2.1 Neural network structure

For the MNIST dataset, we utilize a specialized convolutional neural network with the undermentioned architecture:

- The model commences with a 2D convolutional layer employing 64 filters of 5x5 kernel size, a stride of 2, and a larger padding of 4, effectively augmenting the input image size to 32x32. A ‘Swish’ activation function is then invoked to incorporate non-linearity.
- The second layer consists of another convolutional layer using 128 filters of 3x3 size, with a stride of 2 and padding of 1, followed by the ‘Swish’ activation function.
- The third layer is a replica of the previous one but escalates the filter count to 256 while preserving the filter size, stride, and padding, followed by a ‘Swish’ activation.
- The fourth convolutional layer utilizes 256 filters, similar to the third layer, with a 3x3 kernel size, stride of 2, and padding of 1. This is succeeded by a ‘Swish’ activation function.
- Post convolution, the output undergoes flattening to eliminate spatial dimensions.
- The flattened output is then passed through a fully connected layer with 256 units, followed by the ‘Swish’ activation function.
- The architecture culminates with a second fully connected layer mapping the 256 units to a determined output size. This size usually correlates to the number of classes in a classification task or the desired output size in regression tasks.

The neural network training employs the Adam optimizer with a learning rate of 10^{-4} , batch size of 128, and 200 epochs.

365 For the SVHN dataset, we utilize the WideResNet structure, specifically, the WRN-28-10 variant.
 366 Training the WRN-28-10 also involves the Adam optimizer with a learning rate of 10^{-4} , batch size
 367 of 128, and spans across 50 epochs.

368 A.3 Hardware Specifications

369 All of our experiments were conducted on a machine equipped with an Intel E5-2680 v3 CPU and an
 370 NVIDIA RTX 3090 GPU.

371 B Annotator Interface

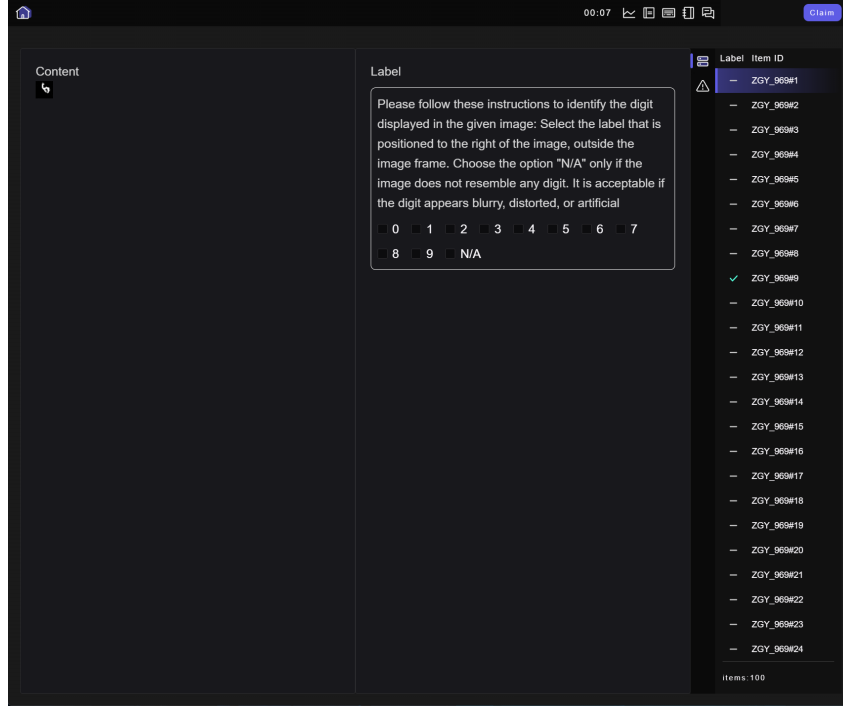


Figure 7: Annotator Interface

372 C Choosing a proper f

373 In this section, we propose two additional f functions, where the first is based on predictive entropy
 374 and the second is rooted in joint-energy.

375 The Predictive Entropy based f , denoted as f_{PE} , is formulated as follows:

$$f_{PE}(\mathbf{x}, y_{tar}) := -c_{PE} \sum_y \sigma(g_\phi(\mathbf{x}))[y] \log \sigma(g_\phi(\mathbf{x}))[y] + f_{CE}(\mathbf{x}, y_{tar})$$

376 Here, c_{PE} is a constant that determines the weight of the predictive entropy.

377 On the other hand, the Joint-Energy based f , denoted as f_{JE} , is given by:

$$f_{JE}(\mathbf{x}, y_{tar}) := -g_\phi(\mathbf{x})[y_{tar}] + c_{JE} \log \sum_y \exp(g_\phi(\mathbf{x})[y])$$

378 In this case, c_{JE} is a constant controlling the weight of the logsumexp term. It is worth noting that
 379 when $c_{JE} = 1$, f_{JE} simplifies to f_{CE} .

380 As shown in Figure 8, when p_{dis} is fixed, choosing f_{CW} results in better generation compared to f_{CE} ,
 381 f_{PE} , and f_{JE} .

382 An interesting visual interpretation of this phenomenon can be found in Figure 9. Here, we draw
383 samples from $p_{\text{vic}}(\cdot; y_{\text{tar}})$, observing that the samples drawn from the p_{vic} induced by f_{CW} contain the
384 least semantic information.

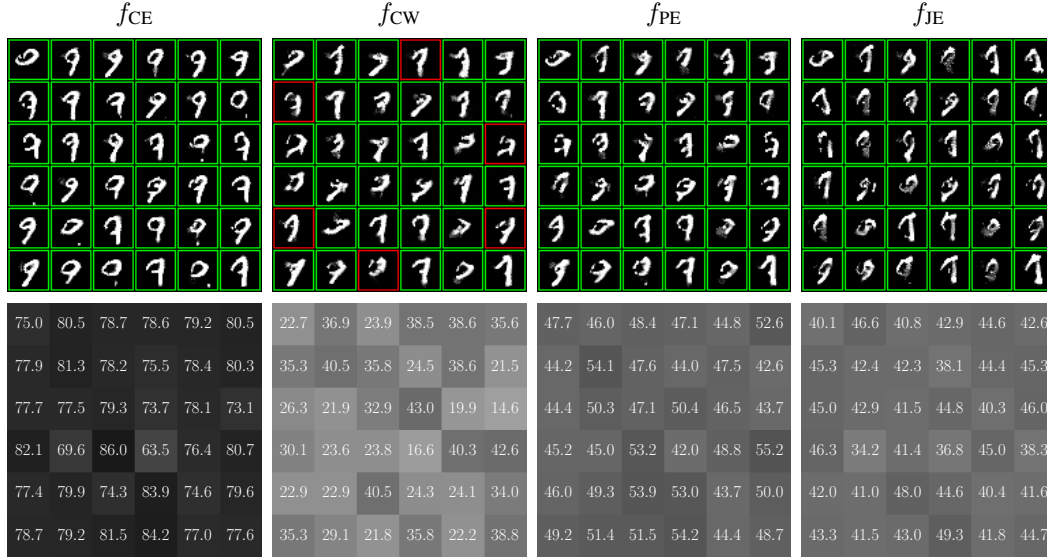


Figure 8: Targeted Attack on a Single Image: The source image belongs to class 7, and the target class is 9. The first row displays samples drawn from $p_{\text{adv}}(\cdot, \mathbf{x}_{\text{ori}}, y_{\text{tar}})$, where \mathbf{x}_{ori} is an image from class 7 and $y_{\text{tar}} = 9$. All cases share the same random seed and the same $p_{\text{dis}}(\cdot; \mathbf{x}_{\text{ori}})$ trained on augmentations of \mathbf{x}_{ori} . The key distinction among the plots is the function f used, in this order: f_{CE} , f_{CW} , f_{PE} , f_{JE} . The second row of plots showcases the predictive probability (softmax probability) of the target class, corresponding to each digit in the first row on a one-to-one basis. A green border signifies a successful deception of the victim classifier, while a red border indicates failure.

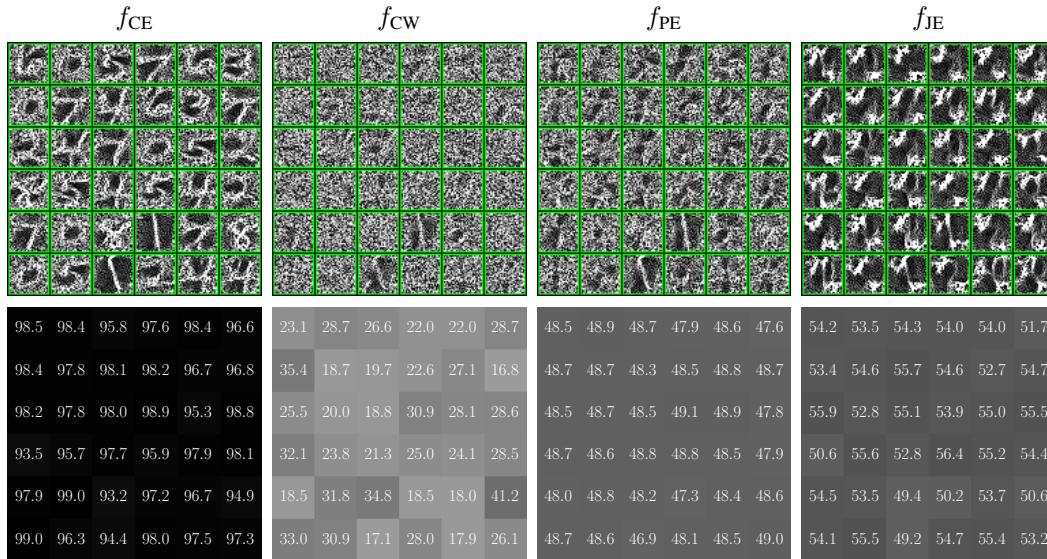


Figure 9: Samples drawn from the victim distribution $p_{\text{vic}}(\cdot; y_{\text{tar}})$ with randomly sampled y_{tar} . All four cases share the same random seed. The parameters are consistent with those in Figure 8.

D Low-visual quality samples with high likelihood

During our experiments, we observed that high likelihood samples do not invariably exhibit high visual quality. This phenomenon is showcased in Figure 10, where, despite being sorted by energy (a parameter proportional to likelihood), the earliest samples do not always deliver high visual quality. Further, we empirically identified a pattern: high likelihood samples that possess low visual quality often correspond to a low softmax probability for class y_{ori} (the label of the original image). Leveraging this observation, we decided to retain only the top few percentile of samples that have the highest softmax probability for class y_{ori} within an auxiliary classifier, and then sort the remaining samples by energy.

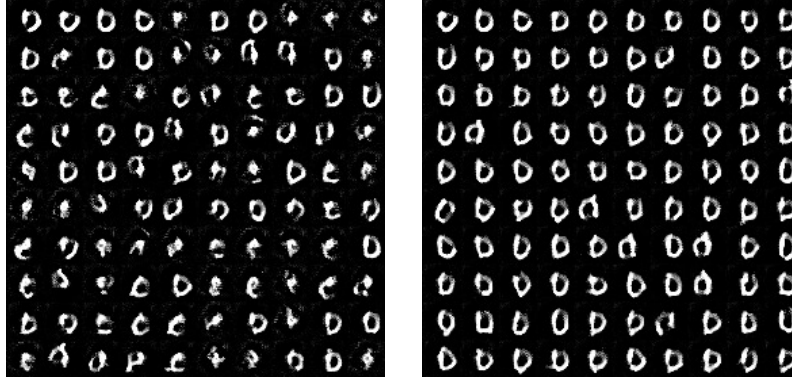


Figure 10: In this instance, the source image, denoted as \mathbf{x}_{ori} , represents the digit ‘0’ as displayed in Figure 1, while the target is class 1. We derived 4641 samples from $p_{\text{adv}}(\cdot; 0, 1)$ via rejection sampling. The **Left** portion of the figure shows the initial 100 samples, ordered by energy. The **Right** section, on the other hand, depicts the same initial 100 samples, also sorted by energy, but only after retaining the top 10 percent of samples with the highest softmax probability of class 0 in the auxiliary classifier.

E PGD attacks on adversarially trained victim classifiers

This section is dedicated to showcasing the application of Projected Gradient Descent (PGD) attacks on robustly trained classifiers, employing a variety of parameters.

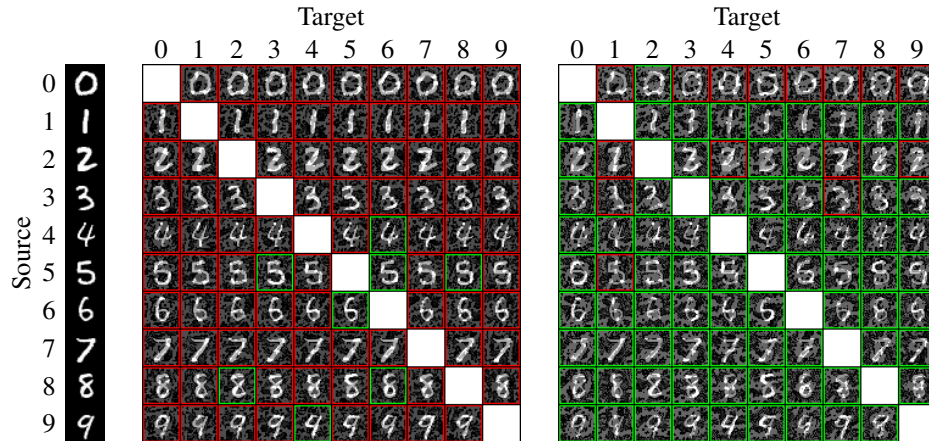


Figure 11: Targeted attacks on an adversarially trained MadryNet [12] using Projected Gradient Descent (PGD) with L_{∞} norm, $\alpha = 0.04$, and 100 steps. Left: $\epsilon = 0.3$. Right: $\epsilon = 0.4$.

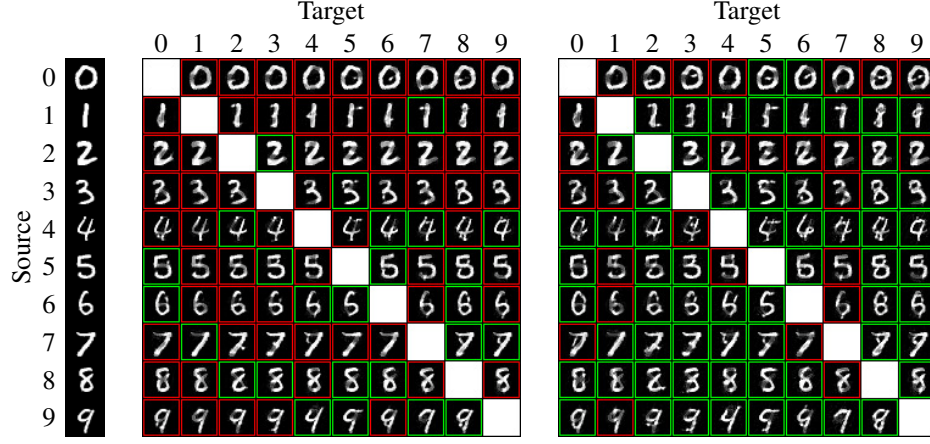


Figure 12: Targeted attacks on an adversarially trained MadryNet [12] using Projected Gradient Descent (PGD) with L_2 norm, $\alpha = 0.2$, and 100 steps. Left: $\epsilon = 3$. Right: $\epsilon = 4$.

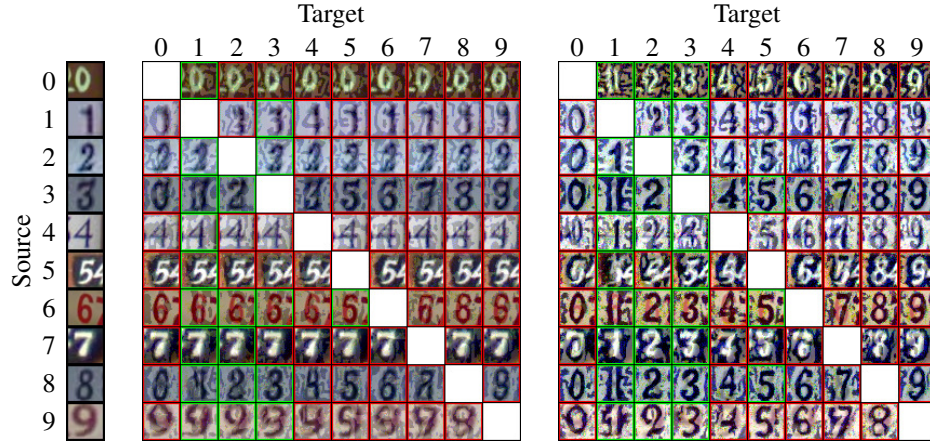


Figure 13: Targeted attacks on an adversarially trained ResNet18 [8] using Projected Gradient Descent (PGD) with L_∞ norm, $\alpha = 0.005$, and 100 steps. Left: $\epsilon = 0.1$. Right: $\epsilon = 0.3$.

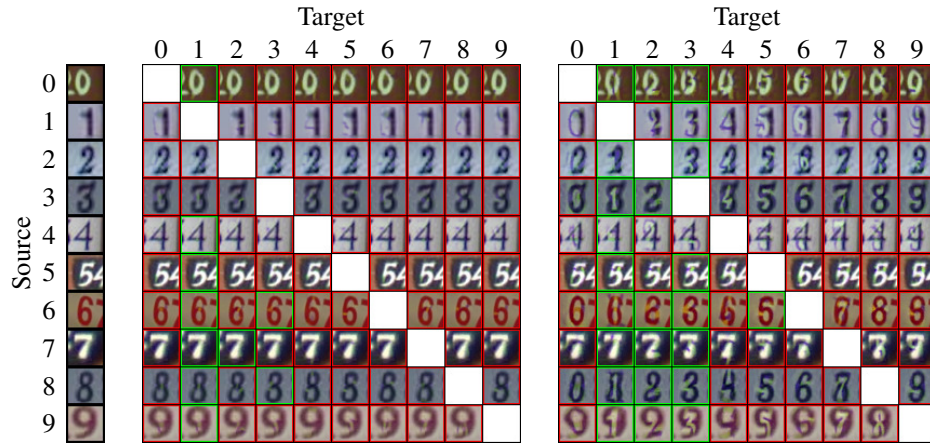


Figure 14: Targeted attacks on an adversarially trained MadryNet [12] using Projected Gradient Descent (PGD) with L_2 norm, $\alpha = 0.1$, and 100 steps. Left: $\epsilon = 1$. Right: $\epsilon = 3$.

397 **F Broader Impacts of this work**

398 The present study introduces a novel approach: the semantics-aware adversarial attack. This method
399 provides significant insights into the resilience and vulnerability of sophisticated classifiers.

400 From an advantageous perspective, it highlights the inherent risks associated with robust classifiers.
401 By exposing potential weak points in such systems, the study underscores the necessity for further
402 improvements in classifier security. This can pave the way for building more resilient artificial
403 intelligence systems in the future.

404 Conversely, the work also presents potential pitfalls. There is a risk that malicious entities might
405 exploit the concepts discussed here for nefarious purposes. It is crucial to take into account the
406 potential misuse of this semantics-aware adversarial attack and accordingly develop preventive
407 measures to deter its utilization for unethical ends.