Shape Defense

Ali Borji
Quintic AI, San Francisco, CA
aliborji@gmail.com

Abstract

Humans rely heavily on shape information to recognize objects. Conversely, convolutional neural networks (CNNs) are biased more towards texture. This fact is perhaps the main reason why CNNs are susceptible to adversarial examples. Here, we explore how shape bias can be incorporated into CNNs to improve their robustness. Two algorithms are proposed, based on the observation that edges are invariant to moderate imperceptible perturbations. In the first one, a classifier is adversarially trained on images with the edge map as an additional channel. At inference time, the edge map is recomputed and concatenated to the image. In the second algorithm, a conditional GAN is trained to translate the edge maps, from clean and/or perturbed images, into clean images. The inference is done over the generated image corresponding to the input's edge map. A large number of experiments with more than 10 data sets demonstrate the effectiveness of the proposed algorithms against FGSM, ℓ_∞ PGD, Carlini-Wagner, Boundary, and adaptive attacks. Further, we show that edge information can a) benefit other adversarial training methods, b) be even more effective in conjunction with background subtraction, c) be used to defend against poisoning attacks, and d) make CNNs more robust against natural image corruptions such as motion blur, impulse noise, and JPEG compression, than CNNs trained solely on RGB images. From a broader perspective, our study suggests that CNNs do not adequately account for image structures and operations that are crucial for robustness. The code is available at: `https://github.com/aliborji/ShapeDefense.git`

1 Introduction

Our primary goal here is to learn *robust models* for visual recognition inspired by the observation that object shape remains largely invariant to imperceptible adversarial perturbations (Fig. 1). Shape is the signature of an object and plays a vital role in recognition [1]. Humans rely heavily on edges and object boundaries, whereas CNNs emphasize more on texture [8, 11]. This may explain why adversarial examples are perplexing.

The convolution operation in CNNs is biased towards capturing texture since the number of pixels constituting texture far exceeds the number of pixels that fall on the object boundary. This in turn provides a big opportunity for adversarial image manipulation. Some attempts have been made to emphasize more on edges, for example by utilizing normalization layers (e.g., contrast and divisive normalization [14]). Such attempts, however, have not been fully investigated for adversarial defense. Overall, how shape and texture should be reconciled in CNNs continues to be an open question. Here we propose two solutions that can be easily implemented and integrated in existing defenses. We also investigate possible adaptive attacks against them. Extensive experiments across ten datasets, over which shape and texture have different relative importance, demonstrate the effectiveness of our solutions against strong attacks. Experiments on more than 10 data sets demonstrate the effectiveness of the proposed algorithms against FGSM, ℓ_∞ PGD, substitute, Carlini-Wagner, Boundary, and adaptive attacks (the latter are shown in appendices B, C, D, and E in order).

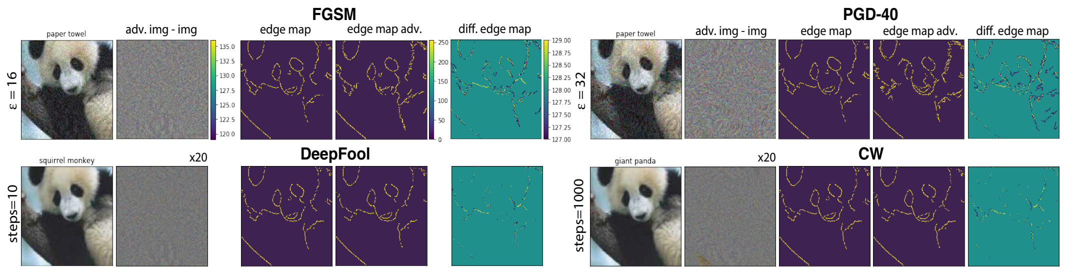


Figure 1: Adversarial attacks against ResNet152 over the giant panda image using FGSM [9], PGD-40 [16] (α=8/255), DeepFool [17] and Carlini-Wagner [5] attacks. The second columns in panels show the difference ($\mathcal{L}_2$) between the original image (not shown) and the adversarial one (values shifted by 128 and clamped). The edge map (using Canny edge detector) remains almost intact at small perturbations. Notice that edges are better preserved for the PGD-40.

Algorithm 1 Edge-guided adversarial training (EAT) for T epochs, perturbation budget ϵ, and loss balance ratio α, over a dataset of size M for a network f_θ (performed in minibatches in practice). $\beta \in \{edge, img, imgedge\}$ indicates network type and *redetect_train* means edge redetection during training.

```
for $t = 1 \dots T$ do
  for $i = 1 \dots M$ do
    // launch adversarial attack (here FGSM and PGD attacks)
    $\tilde{x}_i = \text{clip}(x_i + \epsilon\ sign(\nabla_x \ell(f_\theta(x_i), y_i)))$
    if $\beta$ == imgedge & redetect_train then
      $\tilde{x}_i = \text{detect\_edge}(\tilde{x}_i)$   // recompute and replace the edge map
    end if
    $\ell = \alpha\ \ell(f_\theta(x_i), y_i) + (1-\alpha)\ \ell(f_\theta(\tilde{x}_i), y_i)$   // here $\alpha = 0.5$
    $\theta = \theta - \nabla_\theta \ell$   // update model weights with some optimizer, e.g., Adam
  end for
end for
```

2 Proposed methods

Edge-guided Adversarial Training (EAT). In this approach, we perform adversarial training over the 2D (Gray+Edge) or 4D (RGB+Edge) input (i.e., number of channels; denoted as Img+Edge). Please see Appx A for illustration of this algorithm (Alg. 1). In another version of the algorithm, first, for each input (clean or adversarial), the old edge map is replaced with the newly extracted one. The edge map can be computed from the average of only image channels or all available channels (i.e., image plus edge). The latter can sometimes improve the results, since the old edge map, although perturbed, still contains unaltered shape structures. Then, adversarial training is performed over the new input. The reason behind adversarial training with redetected edges is to expose the network to possible image structure damage. The loss for training is a weighted combination of loss over clean images and loss over adversarial images. At inference time, first, the edge map is computed and then classification is done over the edge-augmented input. As a baseline model, we also consider first detecting the input's edge map and then feeding it to the model trained on the edges for classification. We refer to this model as Img2Edge.

GAN-based Shape Defense (GSD). Here, first, a conditional GAN is trained to map the edge image, from clean or adversarial images, to its corresponding clean image (Alg. 2). Any image translation method (here pix2pix by [12] using code at `https://github.com/mrzhu-cool/pix2pix-pytorch`) can be employed for this purpose. Next, a CNN is trained over the generated images. At inference time, first, the edge map is computed and then classification is done over the generated image for this edge image. The intuition is that the edge map remains nearly the same over small perturbation budgets (See Appx A). Notice that conditional GAN can also be trained on perturbed images (similar to [19] and [15] or edge-augmented perturbed images (similar to above).

Algorithm 2 GAN-based shape defense (GSD)

```
// Training
    1. Create a dataset of images $X = \{x_i, y_i\}^{i=1 \cdots N}$ including clean and/or perturbed images
    2. Extract edge maps ($e_i$) for all images in the dataset
    3. Train a conditional GAN $p_g(x|e)$ to map edge image $e$ to clean image $x$ // here pix2pix
    4. Train a classifier $p_c(y|x)$ to map generated image $x$ to class label $y$
// Inference
    1. For input image $x$, clean or perturbed, first compute the edge image $e$
    2. Then, compute $p_c(y|x')$ where $x'$ is the generated image corresponding to $e$
```

3 Experiments and results

3.1 Datasets and Models

Experiments are spread across 10 datasets covering a variety of stimulus types. Sample images from datasets are given in Fig. 2. Models are trained with cross-entropy loss and Adam optimizer [13] with a batch size of 100, for 20 epochs over MNIST and FashionMNIST, 30 over DogVsCat, and 10 over the remaining. Canny method [3] is used for edge detection over all datasets, except DogBreeds for which Sobel edge detection is used. Edge detection parameters are separately adjusted for each dataset. We did not carry out an exhaustive hyperparameter search, since we are interested in additional benefits edges may bring rather than training the best possible models. For attacks, we use `https://github.com/Harry24k/adversarial-attacks-pytorch`, except Boundary attack for which we use `https://github.com/bethgelab/foolbox`.

3.2 Results

3.2.1 Edge-guided Adversarial Training

Results over MNIST and CIFAR-10 are shown in Tables 2 and 3, respectively (please see Appx A). In these experiments, edge maps are computed only from the gray-level image (in turn computed from the image channels).

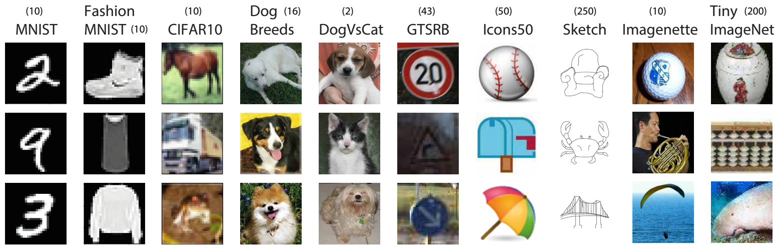


Figure 2: Sample images from the datasets. Numbers in parentheses denote the number of classes.

Over MNIST and FashionMNIST, robust models trained using edges outperform models trained on gray-level images (the last column). The naturally trained models, however, perform better using gray-level images than edge maps (Orig. model column). Adversarial training with augmented inputs improves the robustness significantly over both datasets, except the FGSM attack on FashionMNIST. Over CIFAR-10, incorporating the edges improves the robustness by a large margin against the PGD-40 attack. At $\epsilon = 32/255$, the performance of the robust model over clean and perturbed images is raised from (0.316, 0.056) to (0.776, 0.392). On average, the robust model shows 64% improvement over the RGB model (last column in Table 3). Over the TinyImageNet dataset, as in CIFAR-10, classification using edge maps is poor perhaps due to the background clutter. Nevertheless, incorporating edges improves the results. We expect even better results with more accurate edge detection algorithms (e.g., supervised deep edge detectors). Over these 4 datasets, the final model (i.e., adversarial training using image + redetected edge, and edge redetection at inference time) leads to the best accuracy. The improvement over the image is more pronounced at larger perturbations, in particular against the PGD-40 attack (as expected; please see Fig. 1).

Over the DogVsCat dataset, as in FashionMNIST, the model trained on the edge map is much more robust than the image-only model (Table 7 in Appx. A). Over the DogBreeds dataset, utilizing edges does not improve the results significantly (compared to the image model). The reason could be that texture is more important than shape in this fine-grained recognition task (Table 8 Appx. A). Over GTSRB, Icons-50, and Sketch datasets, *image+edge* model results in higher robustness than the *image-only* model, but leads to relatively less improvement compared to the *edge-only* model. Please

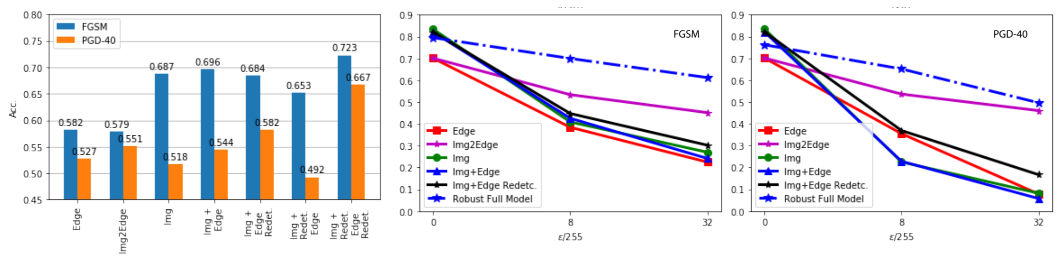


Figure 3: Left) Average results of the EAT defense on all datasets (last cols. in tables). Middle and Right) Comparison of natural (Orig. model column; solid lines) *vs.* adversarial training averaged over all datasets.

see Tables 9, 11, and 13. Over the Imagenette2-160 dataset (Table 15), classification using images does better than edges since the texture is very important on this dataset.

Average results over 10 datasets is presented in Fig. 3 (left panel). Combining shape and texture (full model) leads to a substantial improvement in robustness over the texture alone (5.24% improvement against FGSM and 28.76% imp. against PGD-40). Also, *image+edge* model is slightly more robust than the *image-only* model. Computing the edge map from all image channels improves the results on some datasets (e.g., GTSRB and Sketch) but hurts on some others (e.g., CIFAR-10) as shown in Appx. A. The right two panels in Fig. 3 show a comparison of natural (Orig. model column in tables; solid lines) *vs.* adversarial training. Natural training with *image+edge* and *redetection* at inference time leads to enhanced robustness with little to no harm to standard accuracy. Despite the Edge model only being trained on edges from clean images, the Img2Edge model does better than other naturally-trained models against attacks. The best performance, however, belongs to models trained adversarially. Notice that our results set a new record on adversarial robustness on some of these datasets even without exhaustive parameter search[1].

Robustness against Carlini-Wagner (CW) and Boundary attacks. Performance of our method against l_2 CW attack on MNIST dataset is shown in Appx. C. To make experiments tractable, we set the number of attack iterations to 10. With even 10 iterations, the original Edge and Img models are severely degraded. Img2Edge and Img+(Edge Redetect) models, however, remain robust. Adversarial training with CW attack results in robust models in all cases.

Results against the decision-based Boundary attack [2] are shown in Appx. D over MNIST and Fashion MNIST datasets. Edge, Img, and Img+Edge models perform close to zero over adversarial images. Img+(Edge Redetect) model remains robust since the Canny edge map does not change much after the attack, as is illustrated in Fig. 6.

Robustness against substitute model attacks. Following [18], we trained substitute models to mimic the robust models (with the same architecture but with RGB channels) using the cross-entropy loss over the logits of the two networks, for 5 epochs. The adversarial examples crafted for the substitute networks were then fed to the robust networks. Results are shown in *italics* in Tables 2, 3, 4 and 5 (performed only against the edge-redetect models). We find that this attack is not able to knock off the robust models. Surprisingly, it even improves the accuracy in some cases. Please refer to Appx. B for more details.

Robustness against adaptive attacks. So far we have been using the Canny edge detector which is non-differentiable. What if the adversary builds a differentiable edge detector to approximate the Canny edge detector and then utilizes it to craft adversarial examples? To study this, we run two experiments. In the first one, we build the following pipeline using the HED deep edge detector [24]: Img $\longrightarrow$ HED $\longrightarrow$ Classifier^{HED}. A CNN classifier (as above) is trained over the HED edges on the Imagenette2-160 dataset (See Appx. E). Attacking this classifier with FGSM and PGD-5 ($\epsilon = 8/255$) completely fools the network. The original classifier (Img2Edge here) trained on Canny edges, however, is still largely robust to the attacks (i.e., Img$^{adv-HED}$ $\longrightarrow$ Canny $\longrightarrow$ Classifier^{Canny}) as shown in Table 20. Notice that the HED edge maps are continuous in the range [0,1], whereas Canny edge maps are binary, which may explain why it is easy to fool the HED classifier (See Fig. 7).

Above, we used an off the shelf deep edge detector trained on natural scenes. As can be seen in Appx. E, its generated edge maps differ significantly from Canny edges. What if the adversary trains a model with the (*input, output*) pair as (*input image, Canny edge map*) to better approximate the Canny

[1]cf. [10]; the best robust accuracy on CIFAR-10 against PGD attack, ℓ_∞ of size 8/255, is about 67%.

edge detector? In experiment two, we investigate this possibility. We build a pipeline consisting of a convolutional autoencoder followed by a CNN on MNIST. Details regarding architecture and training procedure are given in Appx. E. As results in Fig. 10 reveal, FGSM and PGD-40 attacks against the pipeline are very effective. Passing the adversarial images through Canny and then a trained (naturally or adversarially) classifier on Canny edges (i.e., Img2Edge), still leads to high accuracy, which means that transfer was not successful. We attribute this feat to the binary output of Canny. Two important point deserve attention. First, here we used the Img2Edge model, which as shown above, is less robust compared to the full model (i.e., img+edge and redetection). Thus, adaptive attacks may be even less effective against the full model. Second, proposed methods perform better when edge map is less disturbed. For example, as shown in Fig. 10 (bottom), the PGD-40 adaptive attack is less effective against the shape defense since edges are preserved better.

Analysis of parameter α. By setting $\alpha = 0$, the network will be exposed only to adversarial examples (Alg. 1), which is computationally more efficient. However, it results in lower accuracy and robustness compared to when $\alpha = 0.5$, which means exposing the network to both clean and adversarial images is important (See Table 23; Appx. H). Nevertheless, here again incorporating edges improves the robustness significantly compared to the image-only case.

Speculation behind effectiveness of this method. The main reason is that the edge map acts as a checksum, and the network learns (through adversarial training) to rely more on the redetected edges when other channels are misleading. This aligns with prior observations such as shortcut learning in CNNs [7]. Also, our approach resembles adversarial patch or backdoor/trojan attacks where the goal is to fool a classifier by forcing it to rely on irrelevant cues. Conversely, here we use this trick to make a model more robust. Also, the Img2Edge model can purify the input before classifying it. Any adaptive attack against the EAT defense has to alter the edges which most likely will result in perceptible structural damages.

3.2.2 GAN-based Shape defense

We trained the pix2pix model for 10 epochs over MNIST and FashionMNIST datasets, and for 100 epochs over Icons-50 dataset. Sample generated images are shown in Fig. 11 (Appx. F). A CNN (same architecture as before) was trained for 10 epochs to classify the generated images. Results are shown in Fig. 4. The model trained over the images generated by pix2pix (solid lines in the figure) is compared to the model trained over the original clean training set (denoted by the dashed lines). Both models are tested over the clean and perturbed versions of the original test sets of the four datasets. Over MNIST and FashionMNIST datasets, GSD performs on par with the original model on clean test images. It is, however, much more robust than the original model against the attacks. When we trained the pix2pix over the edge maps from the perturbed images, the new CNN models became even more robust (stars in Fig. 4; top panels). We expect even better results with training over edge maps from both intact and perturbed images[2].

Over Icons-50 dataset, generated images are poor. Consequently, GSD underperforms the original model on clean images. Over the adversarial inputs, however, GSD wins, especially at high perturbation budgets and against the PGD-40 attack. With better edge detection and image generation methods (e.g., using perceptual loss), better results are expected.

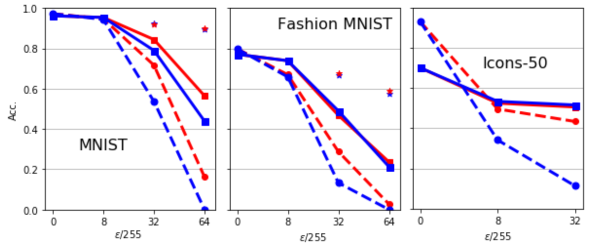


Figure 4: Results of GSD method.

Speculation behind effectiveness of this method. The main reason is that cGAN learns a function f that is invariant to adversarial perturbations. Since the edge map is not completely invariant to (especially large) perturbations, one has to train the cGAN on the augmented dataset composed of clean and perturbed images. One advantage of this approach is it computational efficiency since there is no need for adversarial training. Any adaptive attack against this defense has to fool the cGAN which is perhaps not feasible since it will be noticed from the generated images (i.e., cGAN will fail to generate decent images).

[2]Similarly, the edge map classifier used in the Img2Edge model in the previous section (EAT defense) can be trained on edge maps from both clean and adversarial examples to improve performance.

Table 1: Performance of edge-augmented *FastAT* and *FreeAT* adversarial defenses over clean and perturbed images (See Appx. G for extended algorithms). FastAT is trained with the FGSM adversary ($\epsilon = 0.1$ or $\epsilon = 0.3$) over MNIST and FashionMNIST datasets, and $\epsilon = 8/255$ over CIFAR-10). FreeAT is trained over CIFAR-10 with $\epsilon = 8/255$ and 8 minibatch replays. CIFAR-10 results are averaged over 3 runs (Appx. G). PGD attacks use 10 random restarts. The remaining settings and parameters are as in [22].

	MNIST (FastAT)					**Fashion MNIST (FastAT)**					**CIFAR-10 (FastAT)**			**CIFAR-10 (FreeAT)**		
ϵ	**0.1**		**0.3**		Avg.	**0.1**		**0.3**		Avg.	**8/255**		Avg.	**8/255**		Avg.
	0	**PGD-50**	**0**	**PGD-50**	Acc.	**0**	**PGD-50**	**0**	**PGD-50**	Acc.	**0**	**PGD-10**	Acc.	**0**	**PGD-10**	Acc.
Edge	0.986	0.940	0.113	0.113	0.538	0.844	0.753	0.786	0.110	0.623	0.582	0.386	0.484	0.679	**0.678**	**0.678**
Img	0.991	0.955	0.985	0.877	0.952	0.835	0.696	0.641	0.000	0.543	0.767	0.381	0.574	0.774	0.449	0.612
Img + Edge	**0.988**	0.968	0.980	0.922	0.965	0.851	0.780	**0.834**	0.769	0.809	**0.874**	0.386	0.630	**0.782**	0.442	0.612
Redetect	„	0.977	„	0.966	0.978	„	0.823	„	0.778	0.822	„	0.393	0.634	„	0.448	0.615
Img + Red. Edge	0.986	0.087	**0.986**	0.000	0.515	**0.857**	0.262	0.817	0.000	0.484	0.866	0.074	0.470	0.777	0.451	0.614
Redetect	„	**0.984**	„	**0.986**	**0.986**	„	**0.855**	„	**0.823**	**0.838**	„	**0.416**	**0.641**	„	0.452	0.615

4 Fast & free adversarial training with shape defense

Here, we examine whether incorporating shape bias can empower other defenses, in particular, a) fast adversarial training by [22], dubbed *FastAT*, and free adversarial training by [20], dubbed *FreeAT*. Wong et al. trained robust models using a much weaker and cheaper adversary to lower the cost of adversarial training. They showed that adversarial training with the FGSM adversary is as effective as PGD-based training. The key idea in Shafahi et al. 's work is to simultaneously update both the model parameters and image perturbations in one backward pass, rather than using separate gradient computations at each update step. Please see also Appx. G.

The same CNN architectures as in Wong et al. are employed here. For FastAT, we trained three models over MNIST (for 10 epochs), FashionMNIST (for 3 epochs), and CIFAR-10 (for 10 epochs & *early-stopping*) datasets. For FreeAT, we trained models only over CIFAR-10 for 10 epochs.

Results are shown in Table 1. Using shape-based FastAT and over MNIST, robust accuracy against PGD-50 grows from 95.5% (image-only model) to 98.4% (our full model) at $\epsilon = 0.1$ and from 87.7% to 98.6% at $\epsilon = 0.3$, which are even higher than what is reported by Wong et al. (97.5% at $\epsilon = 0.1$ and 88.8% at $\epsilon = 0.3$). Over FashionMNIST, the improvement is even more pronounced (from 69.6% to 85.5% at $\epsilon = 0.1$ and from 0% to 82.3% at $\epsilon = 0.3$). Over clean images, our full model outperforms other models in most of the cases. Over the CIFAR-10 dataset, the shape-based extension of the defenses results in high accuracy over both clean and perturbed images (using PGD-10 attack), compared to the image-only model. We expect similar improvements with the classic PGD adversarial training. Overall, our analyses in this section suggest that exploiting edges is not specific to the particular way we perform adversarial training (Algorithms 1&2), and be extended to other defense methods (e.g., TRADES algorithm by [25]).

5 Limitations and future work

Two algorithms are proposed to use shape bias and background subtraction to strengthen CNNs and defend against adversarial attacks and backdoor attacks. To fool these defenses, one has to perturb the image such that the new edge map is significantly different from the old one while preserving image shape and geometry, which does not seem to be trivial at low perturbation budgets. Our results without exhaustive parameter search (model architecture, epochs, edge detection, cGAN training, etc.) are very promising. However, comparison with other more complicated defenses, which often use a lot of tricks, bells and whistles, needs to be investigated. Our additional investigations, not shown here, reveal that combination of shape defense with background subtraction can help defend against backdoor attacks. Also, our results show that shape defense strengthens other adversarial defenses such as fast adversarial training [22] and free adversarial training [20].

Future work should test the proposed ideas against adversarial attacks such as gradient-free attacks, decision-based attacks, sparse attacks (e.g., the one pixel attack [21]), attacks that perturb only the edge pixels, attacks that manipulate the image structure [23], ad-hoc adaptive attacks, and backdoor [6]), other ℓ_p norms, and datasets. There might be also other ways to incorporate shape-bias in CNNs, such as 1) augmenting a dataset with edge maps or negative images, 2) overlaying texture from some objects onto some others as in [8], and 3) designing normalization layers [4].

References

[1] I. Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987.

[2] W. Brendel, J. Rauber, and M. Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *CoRR*, abs/1712.04248, 2017.

[3] J. Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[4] M. Carandini and D. J. Heeger. Normalization as a canonical neural computation. *Nature Reviews Neuroscience*, 13(1):51–62, 2012.

[5] N. Carlini and D. A. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, pages 39–57, 2017.

[6] X. Chen, C. Liu, B. Li, K. Lu, and D. Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[7] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann. Shortcut learning in deep neural networks. *arXiv preprint arXiv:2004.07780*, 2020.

[8] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

[9] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *Proc. ICLR*, 2015.

[10] S. Gowal, C. Qin, J. Uesato, T. Mann, and P. Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593*, 2020.

[11] M. A. Islam, M. Kowal, P. Esser, S. Jia, B. Ommer, K. G. Derpanis, and N. Bruce. Shape or texture: Understanding discriminative features in cnns. *arXiv preprint arXiv:2101.11604*, 2021.

[12] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[13] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[15] X. Li and S. Ji. Defense-vae: A fast and accurate defense against adversarial attacks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 191–207. Springer, 2019.

[16] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *CoRR*, abs/1706.06083, 2017.

[17] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *Proc. CVPR*, pages 2574–2582, 2016.

[18] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. Berkay Celik, and A. Swami. Practical black-box attacks against deep learning systems using adversarial examples. *CoRR*, abs/1602.02697, 2016.

[19] P. Samangouei, M. Kabkab, and R. Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018.

[20] A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein. Adversarial training for free! In *Advances in Neural Information Processing Systems*, pages 3358–3369, 2019.

[21] J. Su, D. V. Vargas, and K. Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[22] E. Wong, L. Rice, and J. Z. Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

[23] C. Xiao, J.-Y. Zhu, B. Li, W. He, M. Liu, and D. Song. Spatially transformed adversarial examples. *arXiv preprint arXiv:1801.02612*, 2018.

[24] S. Xie and Z. Tu. Holistically-nested edge detection. In *IEEE International Conference on Computer Vision*, 2015.

[25] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan. Theoretically principled trade-off between robustness and accuracy. *arXiv preprint arXiv:1901.08573*, 2019.

A Supplementary results

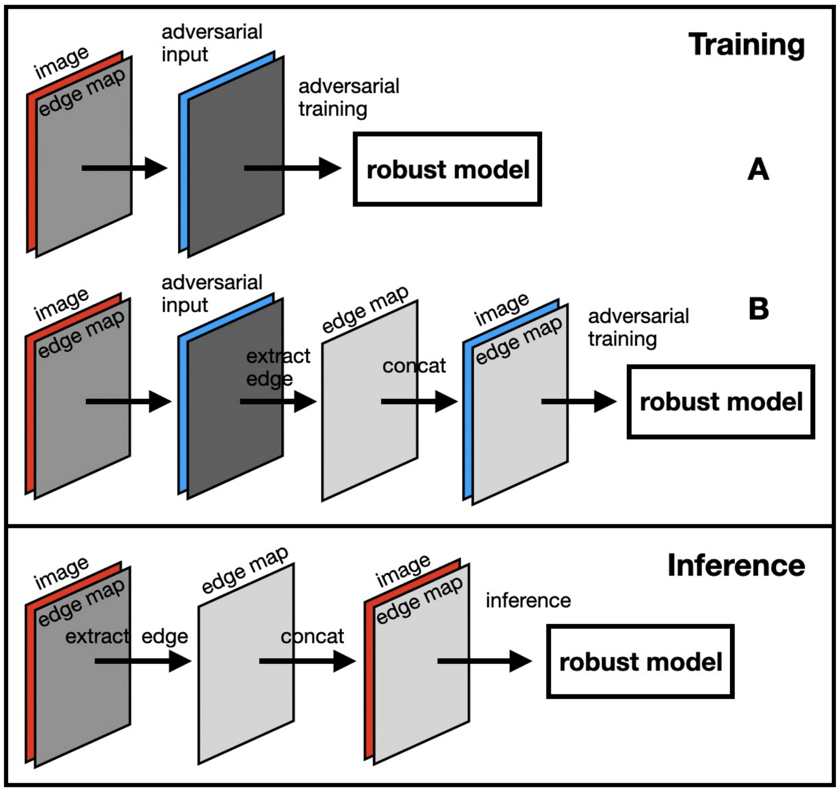


Figure 5: Edge-guided adversarial training (EAT). In its simplest form, adversarial training is performed over the 2D (Gray+Edge) or 4D (RGB+Edge) input (i.e., number of channels; denoted as Img+Edge). In a slightly more complicated form (B), first for each input (clean or adversarial), the old edge map is replaced with the newly extracted one. The edge map can be computed from the average of only image channels or all available channels (i.e., image plus edge).

Table 2: Results (Top-1 acc) over MNIST. The best accuracy in each column is highlighted in **bold**. In *italics* are the results of the substitute attack. Epsilon values are over 255. We used the ℓ_∞ variants of FGSM and PGD. Img2Edge means applying the Edge model (first row) to the edge map of the image.

		Orig. model				**Rob. model (8)**		**Rob. model (32)**		**Rob. model (64)**		**Average**
	ϵ	0/clean	8	32	64	0/clean	8	0/clean	32	0/clean	64	**Rob. models**
FGSM	**Edge**	0.964	0.925	0.586	0.059	0.973	0.954	0.970	0.892	0.964	0.776	0.921
	Img2Edge	„	**0.960**	**0.951**	**0.918**	„	**0.971**	„	**0.957**	„	0.910	**0.957**
	Img	**0.973**	0.947	0.717	0.162	**0.976**	0.955	**0.977**	0.892	0.970	0.745	0.919
	Img+Edge	0.972	0.941	0.664	0.089	**0.976**	0.958	**0.977**	0.902	**0.972**	0.782	0.928
	Redetect	”	0.950	0.803	0.356	”	0.962 *(0.968)*	”	0.919 *(0.947)*	”	0.843 *(0.881)*	0.941
	Img + Redetected Edge					0.974	0.950	0.970	0.771	0.968	0.228	0.810
	Redetect					”	0.958 *(0.966)*	”	0.929 *(0.947)*	”	**0.922** *(0.925)*	0.953
PGD-40	**Edge**	0.964	0.923	0.345	0.000	0.971	0.949	0.973	0.887	0.955	0.739	0.912
	Img2Edge	„	**0.961**	**0.955**	**0.934**	„	**0.970**	„	**0.958**	„	0.927	**0.960**
	Img	**0.973**	0.944	0.537	0.008	0.977	0.957	**0.978**	0.873	0.963	0.658	0.901
	Img+Edge	0.972	0.938	0.446	0.001	**0.978**	0.953	0.975	0.879	0.965	0.743	0.915
	Redetect	”	0.950	0.741	0.116	”	0.960 *(0.967)*	”	0.913 *(0.948)*	”	0.804 *(0.908)*	0.932
	Img + Redetected Edge					0.975	0.949	0.973	0.649	**0.968**	0.000	0.752
	Redetect					”	0.958 *(0.967)*	”	0.945 *(0.958)*	”	**0.939** *(0.942)*	**0.960**

Table 3: Results over the CIFAR-10 dataset.

		Orig. model			**Rob. model (8)**		**Rob. model (32)**		**Average**
	ϵ	0/clean	8	32	0/clean	8	0/clean	32	**Rob. models**
FGSM	**Edge**	0.490	0.060	0.015	0.535	0.323	0.382	0.199	0.360
	Img2Edge	„	0.258	0.258	„	0.270	„	0.217	0.351
	Img	**0.887**	0.359	0.246	**0.869**	0.668	**0.855**	0.553	0.736
	Img + Edge	0.860	0.366	0.169	0.846	0.611	0.815	0.442	0.679
	Redetect	„	**0.399**	**0.281**	„	0.569 *(0.631)*	„	0.417 *(0.546)*	0.662
	Img + Redetected Edge				0.846	0.530	0.832	0.337	0.636
	Redetect				„	**0.702** *(0.753)*	„	**0.569** *(0.678)*	**0.737**
PGD-40	**Edge**	0.490	0.071	0.000	0.537	0.315	0.142	0.119	0.278
	Img2Edge	„	0.259	**0.253**	„	0.274	„	0.253	0.301
	Img	**0.887**	0.018	0.000	0.807	0.450	0.316	0.056	0.407
	Img + Edge	0.860	0.019	0.000	0.788	0.429	0.176	0.119	0.378
	Redetect	„	**0.306**	0.093	„	0.504 *(0.646)*	„	0.150 *(0.170)*	0.404
	Img + Redetected Edge				**0.834**	0.155	**0.776**	0.006	0.443
	Redetect				„	**0.661** *(0.767)*	„	**0.392** *(0.700)*	**0.666**

Table 4: Results over the Fashion MNIST dataset (*)

	Orig. model				**Rob. model (8)**		**Rob. model (32)**		**Rob. model (64)**		**Average**
ϵ	0/clean	8	32	64	0/clean	8	0/clean	32	0/clean	64	**Rob. models**
FGSM											
Edge	0.775	0.714	0.497	0.089	0.776	0.740	0.766	**0.664**	0.748	**0.750**	**0.741**
Img2Edge	„	**0.755**	**0.679**	**0.452**	„	**0.762**	„	**0.664**	„	0.420	0.690
Img	0.798	0.670	0.288	0.027	**0.798**	0.722	0.764	0.584	**0.768**	0.505	0.690
Img+Edge	**0.809**	0.662	0.229	0.010	0.794	0.732	0.769	0.623	0.750	0.537	0.701
Redetect	”	0.691	0.326	0.053	”	0.739 *(0.761)*	”	0.616 *(0.660)*	„	0.491 *(0.496)*	0.693
Img + Redetected Edge					0.789	0.719	**0.775**	0.539	0.762	0.045	0.605
Redetect					”	0.739 *(0.753)*	”	**0.664** *(0.678)*	”	0.611 *(0.532)*	0.721
PGD-40											
Edge	0.775	0.711	0.370	0.002	0.783	0.744	0.769	0.661	0.743	0.574	0.712
Img2Edge	„	**0.757**	**0.683**	**0.380**	„	**0.762**	„	0.658	„	0.374	0.681
Img	0.798	0.659	0.133	0.000	0.792	0.713	0.760	0.515	0.734	0.324	0.640
Img+Edge	**0.809**	0.647	0.100	0.000	0.794	0.726	0.765	0.608	0.744	0.568	0.701
Redetect	”	0.682	0.235	0.014	”	0.734 *(0.760)*	”	0.629 *(0.666)*	-	0.607 *(0.426)*	0.712
Img + Redetected Edge					**0.800**	0.717	**0.779**	0.393	**0.771**	0.002	0.577
Redetect					”	0.743 *(0.766)*	”	**0.694** *(0.681)*	”	**0.690** *(0.504)*	**0.746**

Table 5: Results over the TinyImageNet dataset (*)

	Orig. model			**Rob. model (8)**		**Rob. model (32)**		**Average**
ϵ	0/clean	8	32	0/clean	8	0/clean	32	**Rob. models**
FGSM								
Edge	0.136	0.010	0.001	0.150	0.078	0.098	0.021	0.087
Img2Edge	„	0.097	**0.096**	„	0.094	„	0.077	0.105
Img	**0.531**	0.166	0.074	**0.512**	0.297	**0.488**	0.168	**0.366**
Img + Edge	0.522	0.152	0.050	0.508	0.273	0.471	0.148	0.350
Redetect	„	**0.171**	0.081	„	0.287 *(0.356)*	„	0.162 *(0.266)*	0.357
Img + Redetected Edge				0.505	0.264	0.482	0.111	0.340
Redetect				”	**0.305** *(0.371)*	„	**0.171** *(0.296)*	**0.366**
PGD-40								
Edge	0.136	0.007	0.000	0.148	0.077	0.039	0.014	0.069
Img2Edge	„	**0.094**	**0.092**	„	0.095	„	0.033	0.079
Img	**0.531**	0.019	0.000	0.392	0.150	0.191	0.019	0.188
Img + Edge	0.522	0.008	0.000	0.402	0.131	0.157	0.003	0.173
Redetect	„	0.074	0.009	„	0.198 *(0.353)*	„	0.019 *(0.103)*	0.194
Img + Redetected Edge				**0.425**	0.072	**0.328**	0.005	0.208
Redetect				”	**0.206** *(0.380)*	„	**0.073** *(0.279)*	**0.258**

Table 6: Results on CIFAR-10 dataset [edge map computed from 4 channels]

	Orig. model			**Rob. model (8)**		**Rob. model (32)**		**Average**
ϵ	0/clean	8	32	0/clean	8	0/clean	32	**Rob. models**
				FGSM				
Img+Edge	0.860	0.366	0.169	0.846	0.611	0.815	0.442	0.679
Redetect	”	0.415	0.280	”	0.574	„	0.416	0.663
Img + Redetected Edge				0.848	0.547	0.835	0.351	0.645
Redetect				”	0.696	”	0.553	0.733
				PGD-40				
Img+Edge	0.860		0.000	0.789	0.431	0.179	0.135	0.384
Redetect	”		0.087	„	0.501	„	0.152	0.405
Img + Redetected Edge				0.837	0.164	0.767	0.010	0.444
Redetect				”	0.648	„	0.352	0.651

Table 7: Results on DogVsCat dataset [edge map computed from 4 channels] (*)

	Orig. model			**Rob. model (8)**		**Rob. model (32)**		**Average**
ϵ	0/clean	8	32	0/clean	8	0/clean	32	**Rob. models**
				FGSM				
Edge	0.814	0.633	0.119	0.812	0.757	0.806	**0.999**	0.843
Img2Edge	„	**0.755**	**0.584**	„	**0.767**	„	0.576	0.740
Img	**0.863**	0.007	0.051	0.777	0.430	**0.819**	0.985	0.753
Img+Edge	0.823	0.007	0.000	0.782	0.641	0.808	0.992	0.806
Redetect	”	0.043	0.002	”	0.666	”	0.986	0.810
Img + Redetected Edge				**0.829**	0.615	0.812	0.853	0.778
Redetect				”	0.763	”	0.998	**0.850**
				PGD-40				
Edge	0.814	0.624	0.018	**0.820**	0.770	0.763	0.681	0.758
Img2Edge	„	**0.760**	**0.568**	„	**0.778**	„	0.656	0.754
Img	**0.863**	0.000	0.000	0.769	0.384	0.500	0.500	0.538
Img+Edge	0.823	0.000	0.000	0.785	0.689	0.816	0.496	0.696
Redetect	”	0.006	0.000	„	0.744	„	0.500	0.711
Img + Redetected Edge				0.819	0.600	**0.817**	0.009	0.561
Redetect				”	0.760	„	**0.972**	**0.842**

Table 8: Results on DogBreeds dataset using Sobel edge detection [edge map computed from 4 channels] (*)

	Orig. model			**Rob. model (8)**		**Rob. model (32)**		**Average**
ϵ	0/clean	8	32	0/clean	8	0/clean	32	**Rob. models**
				FGSM				
Edge	0.750	0.006	0.031	0.506	0.101	0.413	0.073	0.273
Img2Edge	„	0.236	**0.194**	„	0.362	„	0.241	0.380
Img	**0.899**	0.256	0.140	0.823	0.595	0.829	**0.449**	0.674
Img + Edge	0.896	0.225	0.098	**0.862**	0.534	0.820	0.385	0.650
Redetect	„	**0.244**	0.171	„	0.455	„	0.292	0.607
Img + Redetected Edge				0.843	0.506	**0.874**	0.298	0.630
Redetect				”	**0.618**	„	0.419	**0.689**
				PGD-40				
Edge	0.750	0.000	0.000	0.514	0.065	0.036	0.000	0.154
Img2Edge	„	**0.250**	**0.207**	„	0.301	„	0.037	0.222
Img	**0.899**	0.000	0.000	0.795	0.286	0.596	0.025	0.425
Img + Edge	0.896	0.000	0.000	0.789	0.225	0.567	0.042	0.406
Redetect	„	0.008	0.000	”	0.396	„	0.065	0.454
Img + Redetected Edge				**0.772**	0.028	**0.677**	0.000	0.369
Redetect				”	**0.393**	”	**0.149**	**0.498**

Table 9: Results on GTSRB dataset [edge map computed from 4 channels] (*)

	Orig. model			**Rob. model (8)**		**Rob. model (32)**		**Average**
ϵ	0/clean	8	32	0/clean	8	0/clean	32	**Rob. models**
				FGSM				
Edge	0.938	**0.683**	0.315	**0.947**	**0.863**	**0.946**	0.701	0.864
Img2Edge	„	0.501	0.451	„	0.516	„	0.469	0.719
Img	**0.955**	0.464	0.322	0.902	0.607	0.896	0.562	0.742
Img + Edge	0.951	0.624	0.382	0.940	0.842	0.943	0.686	0.853
Redetect	”	0.592	**0.471**	”	0.743	”	0.626	0.813
Img + Redetected Edge				0.925	0.801	0.939	0.616	0.820
Redetect				”	0.844	”	**0.766**	**0.869**
				PGD-40				
Edge	0.938	**0.618**	0.054	**0.950**	**0.861**	**0.937**	**0.598**	**0.836**
Img2Edge	„	0.501	**0.459**	„	0.506	„	0.462	0.714
Img	**0.955**	0.189	0.033	0.855	0.495	0.736	0.246	0.583
Img + Edge	0.951	0.271	0.021	0.943	0.750	0.839	0.342	0.718
Redetect	„	0.526	0.251	„	0.774	„	0.514	0.767
Img + Redetected Edge				0.929	0.505	0.893	0.134	0.615
Redetect				”	0.818	„	0.557	0.799

Table 10: Results on GTSRB dataset [edge map computed from 3 channels]

	Orig. model			**Rob. model (8)**		**Rob. model (32)**		**Average**
ϵ	0/clean	8	32	0/clean	8	0/clean	32	**Rob. models**
				FGSM				
Img + Edge	0.951	0.624	0.382	0.940	0.842	0.943	0.686	0.853
Redetect	„	0.500	0.395	„	0.558	„	0.492	0.733
Img + Redetected Edge				0.889	0.699	0.891	0.549	0.757
Redetect				„	0.610	„	0.577	0.742

Table 11: Results on Icons-50 dataset [edge map computed from 4 channels] (*)

	Orig. model			**Rob. model (8)**		**Rob. model (32)**		**Average**
ϵ	0/clean	8	32	0/clean	8	0/clean	32	**Rob. models**
				FGSM				
Edge	0.883	0.545	0.210	**0.904**	0.771	**0.889**	0.594	0.789
Img2Edge	„	**0.713**	**0.690**	„	0.746	„	0.730	**0.817**
Img	**0.930**	0.495	0.433	0.772	0.789	0.836	0.720	0.779
Img + Edge	0.929	0.569	0.433	0.829	0.818	0.844	**0.745**	0.809
Redetect	„	0.470	0.414	„	0.730	„	0.732	0.784
Img + Redetected Edge				0.841	**0.837**	0.849	0.688	0.804
Redetect				„	0.817	„	0.710	0.804
				PGD-40				
Edge	0.883	0.423	0.000	**0.902**	0.769	**0.846**	0.404	0.730
Img2Edge	„	**0.706**	**0.683**	„	0.753	„	**0.695**	**0.799**
Img	**0.930**	0.341	0.113	0.765	0.663	0.736	0.453	0.654
Img + Edge	0.929	0.320	0.011	0.800	0.678	0.785	0.366	0.657
Redetect	„	0.416	0.248	„	0.738	„	0.660	0.746
Img + Redetected Edge				0.838	0.644	0.824	0.097	0.601
Redetect				”	**0.792**	„	0.539	0.748

Table 12: Results on Icons-50 dataset [edge map computed from 3 channels]

	Orig. model			**Rob. model (8)**		**Rob. model (32)**		**Average**
ϵ	0/clean	8	32	0/clean	8	0/clean	32	**Rob. models**
				FGSM				
Img+Edge	0.929	0.569	0.433	0.829	0.818	0.844	0.745	0.809
Redetect	„	0.520	0.460	„	0.737	„	0.731	0.785
Img + Redetected Edge				0.831	0.788	0.870	0.725	0.804
Redetect				”	0.783	„	0.765	0.812

Table 13: Results on Sketch dataset [edge map computed from 2 channels] (*)

	Orig. model			**Rob. model (8)**		**Rob. model (32)**		**Average**
ϵ	0/clean	8	32	0/clean	8	0/clean	32	**Rob. models**
				FGSM				
Edge	0.479	0.167	**0.041**	0.502	0.343	**0.483**	**0.216**	**0.386**
Img2Edge	„	**0.464**	0.014	„	**0.494**	„	0.022	0.375
Img	**0.532**	0.109	0.021	**0.530**	0.278	0.474	0.144	0.356
Gray + Edge	0.486	0.097	0.019	0.513	0.286	0.440	0.167	0.352
Redetect	„	0.263	0.004	„	0.355	„	0.013	0.330
Img + Redetected Edge				0.497	0.180	0.420	0.071	0.292
Redetect				”	0.416	„	0.162	0.374
				PGD-40				
Edge	0.480	0.106	0.000	0.508	0.341	0.401	0.068	0.330
Img2Edge	„	**0.471**	**0.127**	„	**0.499**	„	**0.214**	**0.405**
Img	**0.532**	0.028	0.000	**0.538**	0.260	0.018	0.000	0.204
Gray + Edge	0.486	0.034	0.000	0.500	0.279	0.026	0.000	0.201
Redetect	„	0.277	0.024	„	0.360	„	0.004	0.223
Img + Redetected Edge				0.502	0.121	**0.448**	0.000	0.268
Redetect				”	0.423	„	0.212	0.396

Table 14: Results on Sketch dataset [edge map computed from 1 channel]

	Orig. model			**Rob. model (8)**		**Rob. model (32)**		**Average**
ϵ	0/clean	8	32	0/clean	8	0/clean	32	**Rob. models**
				FGSM				
Gray + Edge	0.486	0.097	0.019	0.513	0.286	0.440	0.167	0.352
Redetect	„	0.213	0.005	„	0.388	„	0.022	0.341
Img + Redetected Edge				0.519	0.296	0.445	0.191	0.363
Redetect				„	0.397	„	0.020	0.345

Table 15: Results on Imagenette2-160 dataset [edge map computed from 4 channels] (*)

	Orig. model			**Rob. model (8)**		**Rob. model (32)**		**Average**
ϵ	0/clean	8	32	0/clean	8	0/clean	32	**Rob. models**
				FGSM				
Edge	0.780	0.101	0.436	0.781	0.520	0.664	0.245	0.553
Img2Edge	„	0.599	**0.598**	„	0.603	„	0.578	0.656
Img	**0.969**	0.617	0.409	**0.959**	0.827	0.946	0.710	0.860
Img + Edge	0.959	0.613	0.373	0.951	0.801	0.935	0.643	0.832
Redetect	„	**0.652**	0.471	„	0.812	„	0.687	0.846
Img + Redetected Edge				0.950	0.747	**0.949**	0.592	0.810
Redetect				„	**0.834**	„	**0.732**	**0.866**
				PGD-40				
Edge	0.780	0.064	0.000	0.794	0.526	0.577	0.071	0.492
Img2Edge	„	**0.601**	**0.577**	„	0.610	„	0.381	0.591
Img	**0.969**	0.052	0.005	0.918	0.599	0.808	0.221	0.636
Img + Edge	0.959	0.045	0.000	0.909	0.558	0.762	0.151	0.595
Redetect	”	0.445	0.069	”	0.743	”	0.305	0.680
Img + Redetected Edge				**0.944**	0.246	**0.883**	0.046	0.530
Redetect				”	**0.757**	”	**0.432**	**0.754**

B Robustness against substitute model attacks

Following [18], we trained substitute models to mimic the robust models (with the same architecture but with RGB channels) using the cross-entropy loss over the logits of the two networks, for 5 epochs. The adversarial examples crafted for the substitute networks were then fed to the robust networks. Results are shown in *italics* in Tables 2, 3, 4 and 5 (performed only against the edge-redetect models). We find that this attack is not able to knock off the robust models. Surprisingly, it even improves the accuracy in some cases.

Table 16: Results of the substitute attack against the robust Img + Edge models (redetect and full model).

	MNIST			**Fashion MNIST**			**CIFAR**		**TinyImgNet**	
ϵ	8	32	64	8	32	64	8	32	8	32
FGSM										
Img + edge model (redetect inference)										
Substitute model on clean images	0.94	0.9365	0.9314	0.7515	0.7393	0.7311	0.8079	0.7766	0.008	0.008
Substitute model on adversarial images	0.8941	0.5858	0.0992	0.6484	0.3701	0.0967	0.2716	0.2049	0.004	0.003
Robust model on clean images	0.9761	0.9766	0.9722	0.7939	0.7692	0.75	0.8463	0.8463	0.508	0.471
Robust model on adversarial images	0.9623	0.9189	0.842	0.7391	0.6156	0.4908	0.5695	0.4186	0.287	0.161
Robust model on substitute adv. images	0.9678	0.9472	0.8813	0.7609	0.6604	0.4955	0.6307	0.5463	0.356	0.266
Img + redetected edge model (redetect inference)										
Substitute model on clean images	0.9381	0.9335	0.9326	0.7513	0.7431	0.7388	0.8104	0.7966	0.008	0.008
Substitute model on adversarial images	0.89	0.5696	0.0989	0.6538	0.3663	0.08	0.2879	0.1988	0.004	0.002
Robust model on clean images	0.9742	0.9699	0.9681	0.7891	0.7746	0.7617	0.8456	0.8328	0.495	0.482
Robust model on adversarial images	0.9583	0.9283	0.9216	0.7392	0.664	0.6115	0.7032	0.5684	0.380	0.170
Robust model on substitute adv. images	0.9657	0.9469	0.9249	0.7529	0.6776	0.5318	0.7528	0.7528	0.371	0.296
PGD-40										
Img + edge model (redetect inference)										
Substitute model on clean images	0.9391	0.9344	0.9257	0.7531	0.7408	0.7303	0.756	0.194	0.008	0.006
Substitute model on adv. images	0.8906	0.4455	0.0196	0.6473	0.2745	0.0096	0.020	0.003	0.000	0.000
Robust model on clean images	0.9782	0.9751	0.9654	0.7938	0.7652	0.7442	0.788	0.179	0.395	0.157
Robust model on adv. images	0.9599	0.9132	0.8039	0.7336	0.6289	0.6068	0.504	0.152	0.242	0.018
Robust model on substitute adv. images	0.9667	0.9477	0.9079	0.7603	0.6656	0.4263	0.646	0.170	0.352	0.103
Img + redetected edge model (redetect inference)										
Substitute model on clean images	0.9385	0.9363	0.9329	0.7503	0.7471	0.7415	0.804	0.730	0.008	0.008
Substitute model on adv. images	0.8888	0.4617	0.0211	0.6458	0.2687	0.01	0.016	0.000	0.000	0.000
Robust model on clean images	0.975	0.9732	0.9682	0.7998	0.7793	0.7715	0.834	0.766	0.425	0.328
Robust model on adv. images	0.9581	0.9449	0.9386	0.7435	0.6943	0.6902	0.662	0.375	0.206	0.074
Robust model on substitute adv. images	0.9665	0.9575	0.9417	0.7661	0.681	0.5037	0.767	0.700	0.380	0.279

C Robustness against Carlini-Wagner (CW) and Boundary attacks

Performance of our method against l_2 CW attack [5] on MNIST dataset is shown in Table 17. To make experiments tractable, we set the number of attack iterations to 10. With even 10 iterations, the original Edge and Img models are severely degraded. Img2Edge and Img+(Edge Redetect) models, however, remain robust. Adversarial training with CW attack results in robust models in all cases.

Performance of the the EAT defense against the l_2 Carlini-Wagner attack [5] with the following parameters:

```
attack = CW(net, targeted=False, c=1e-4, kappa=0, iters=10, lr=0.001)
```

Table 17: Robustness against Carlini-Wagner (CW) and Boundary attacks

	Orig. model		**Robust model**		**Average Rob. models**
	0/clean	adv.	0/clean	adv.	
Edge	0.964	0.106	0.948	0.798	0.873
Img2Edge	„	0.962	„	0.949	**0.949**
Img	0.973	0.103	0.949	0.856	0.903
Img+Edge	0.972	0.097	0.945	0.845	0.895
Redetect	„	0.971	„	0.942	0.944
Img + Redetected Edge			0.947	0.819	0.883
Redetect			„	0.946	0.946

D Robustness against Boundary attack

Results against the decision-based Boundary attack [2] over MNIST and Fashion MNIST datasets are shown below. Edge, Img, and Img+Edge models perform close to zero over adversarial images. Img+(Edge Redetect) model remains robust since the Canny edge map does not change much after the attack, as is illustrated in Fig. 6.

Performance of the the edge augmented model against the Boundary attack [2] with the following parameters:

```
BoundaryAttack(init_attack=None, steps=25000, spherical_step=0.01,
               source_step=0.01, source_step_convergance=1e-07,
               step_adaptation=1.5, tensorboard=False,
               update_stats_every_k=10)
```

Table 18: Results over 500 images from the MNIST dataset

	Orig. model	
	0/clean	adv. (boundary)
Edge	0.964	0.000
Img	0.973	0.003
Img+Edge	0.972	0.000
Redetect	,,	**0.945**
Img+Redetected Edge (adversarially trained using FGSM $\epsilon = 8/255$)	0.974	0.001
Redetect	,,	**0.965**

Table 19: Results over 500 images from the Fashion MNIST dataset

	Orig. model	
	0/clean	adv. (boundary)
Edge	0.776	0.005
Img	0.798	0.018
Img+Edge	0.809	0.003
Redetect	,,	**0.747**
Img+Redetected Edge (adversarially trained using FGSM $\epsilon = 8/255$)	0.789	0.003
Redetect	,,	**0.770**

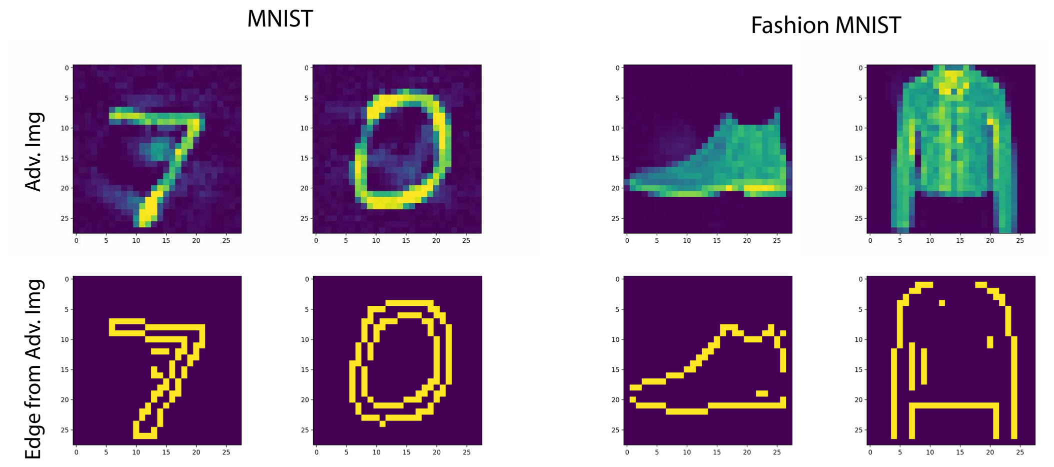


Figure 6: Sample images from the Boundary attack.

E Robustness against adaptive attacks

E.1 Robustness against adaptive attacks over Imagenette2-160 dataset

We use the PyTorch implementation[3] of the HED edge detector proposed by [24]. Here, a classifier is first trained on top of the edge maps from the HED. Then, the entire pipeline (Img $\longrightarrow$ HED $\longrightarrow$ Classifier^{HED}) is attacked to generate an adversarial image. The performance of this classifier is measured on both clean and adversarial images. The adversarial image is also fed to the classifier trained on Canny edge maps (i.e., Img$^{adv-HED}$ $\longrightarrow$ Canny $\longrightarrow$ Classifier^{Canny}). Results are shown in Table below. As it can be seen, adversarial examples crafted for HED fail to completely fool the model trained on Canny edges (i.e., they do not transfer).

Table 20: Results over 500 images from the Imagenette2-160 dataset against the FGSM and PGD-5 ($\epsilon = 8/255$) attacks.

	Orig. model		
	0/clean	adv. (FGSM)	adv. (PGD-5)
Img2Edge (Img $\longrightarrow$ HED $\longrightarrow$ Classifier^{HED})	0.793	0.052	0.003
Img2Edge (Img$^{adv-HED}$ $\longrightarrow$ Canny $\longrightarrow$ Classifier^{Canny})	0.767	0.542	0.548

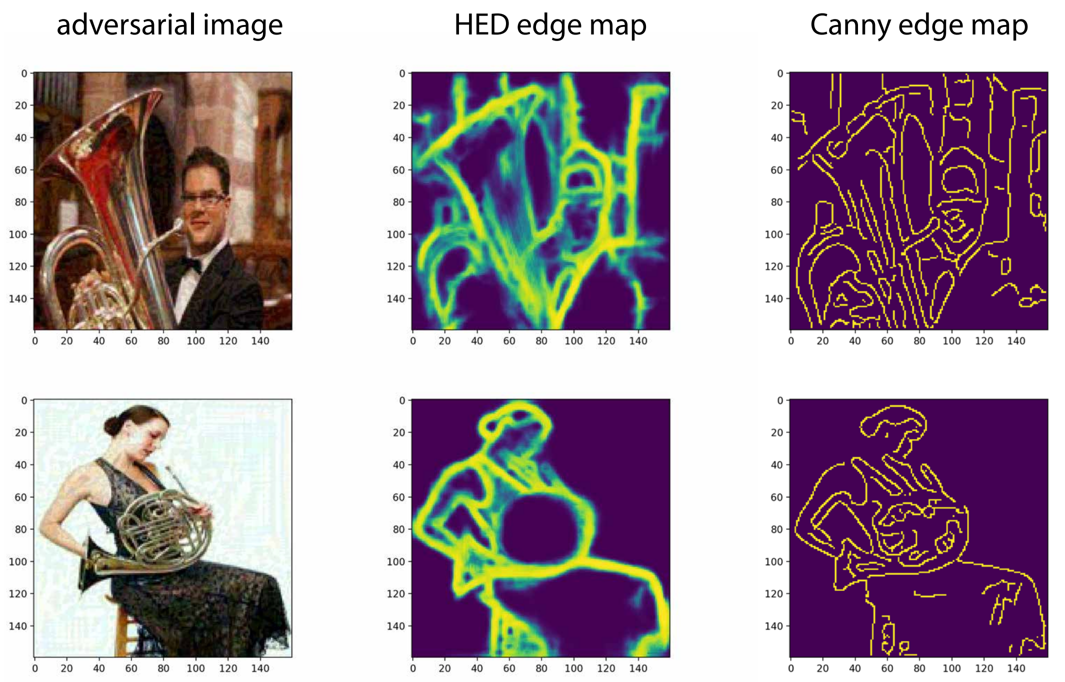


Figure 7: Two sample adversarial images (FGSM) along with their edge maps using HED and Canny edge detection methods.

E.2 Robustness against adaptive attacks over MNIST dataset

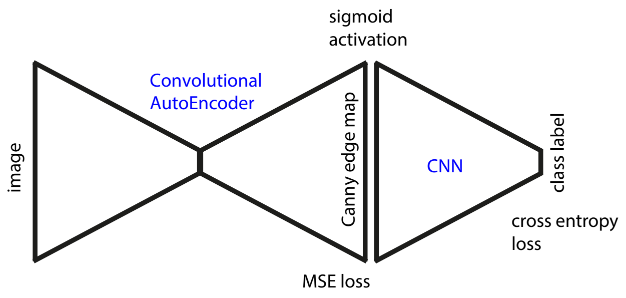
Here, we attempt to explicitly approximate the Canny edge detector using a differentiable convolutional autoencoder. In our pipeline, a classifier (CNN) is stacked after the convolutional autoencoder (with sigmoid output neurons). We first freeze the classifier and train the autoencoder using the MSE loss with (input, output) pair being (image, canny edge map). We then freeze the autoencoder and train the classifier using Cross Entropy loss. After training the network, we then craft adversarial examples for it and feed them to a classifier trained on

[3]`https://github.com/sniklaus/pytorch-hed`

Canny edges (original models or robust models as was mentioned in the main text). Fig. 8 shows the pipeline and some sample approximated edge maps. Fig. 9 shows the architecture details in PyTorch.

The top panel in Fig. 10 shows results using the FGSM and PGD-40 attacks against the pipeline itself, and also against the Img2Edge model (trained over clean edges or adversarial ones[4]). As can be seen, both attacks are very successful against the pipeline but they do not perform well against the Canny edge map classifier (i.e., crafted adversarial examples for the pipeline do not transfer well to the Imge2Edge trained over Canny Edge map; img$\longrightarrow$ Canny $\longrightarrow$ class label). Notice, that here we only used the model trained on edge maps. It is likely to gain even better robustness against the adaptive attacks in using the img+edge+redetect.

The bottom panel in Fig. 10 shows sample adversarial digits (constructed using the adaptive attack) and their edge maps under the FGSM and PGD-40 attacks. Notice how PGD-40 attack preserves the edges (compered to FGSM). This is because it needs less perturbation to fool the classifier. Also, notice that the perturbations shown are perceptible which results in edges maps having noise. If we limit ourselves to imperceptible perturbations, then edge maps will not change much compared to the original edge maps on clean images.



1. Freeze the CNN (requires_grad = False) and train the AutoEncoder
2. Freeze the AutoEncoder (requires_grad = False) and train the CNN
3. Unfreeze all the network (requires_grad = True) and attack it
4. Feed the adeversarial image to a CNN trained with Canny edge maps

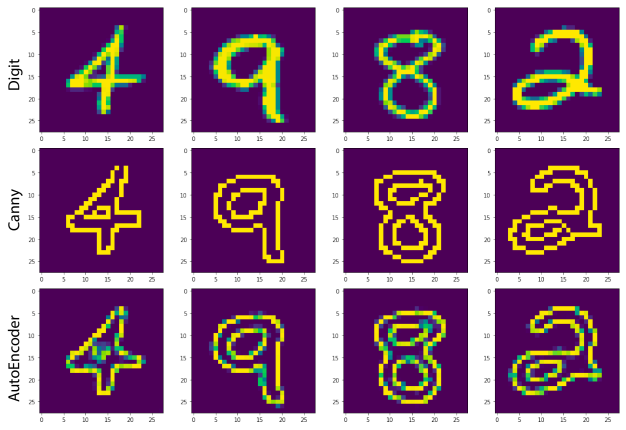
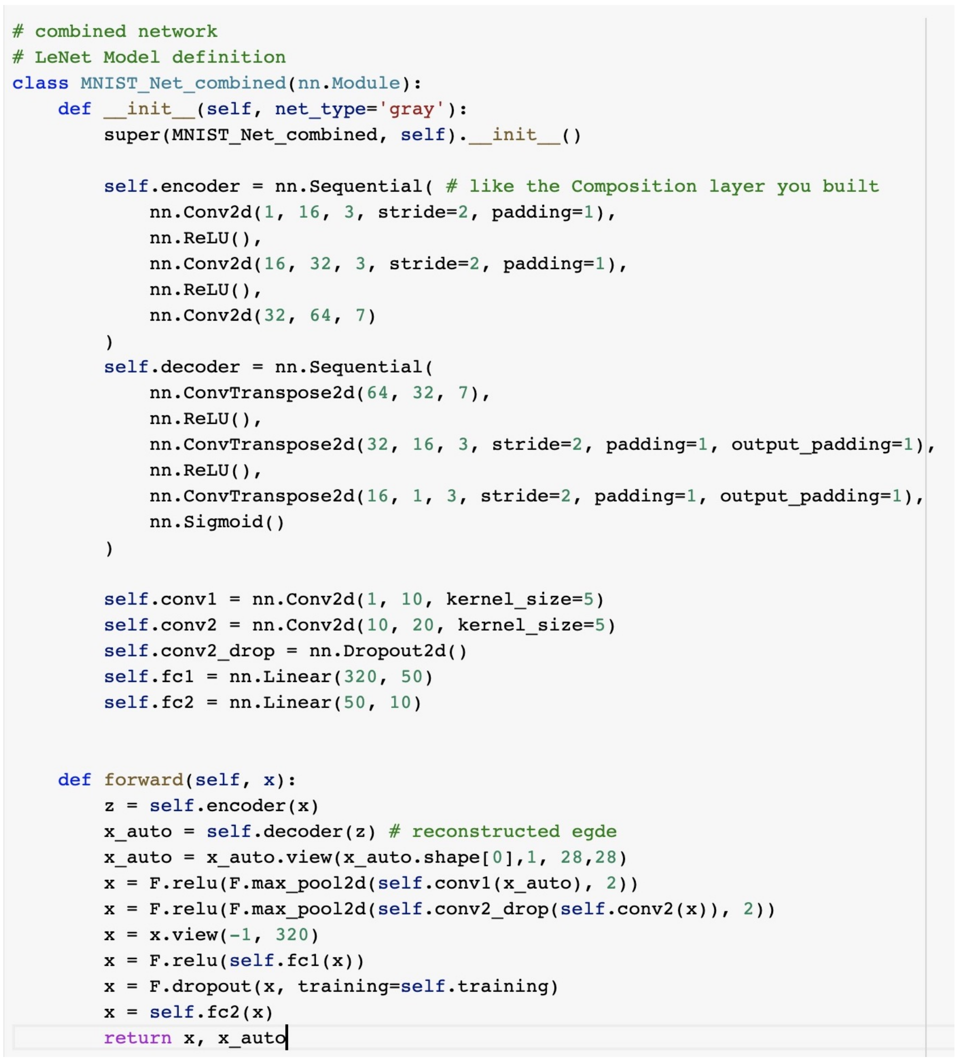


Figure 8: Top: our pipeline to approximate the Canny edge detector and our approach for crafting adversarial examples, Bottom: Sample digits and their generated edge maps.

[4]Here we used the model adversarially trained at eps=8/255 and test it against other perturbations; unlike the main text where we trained robust models separately for each epsilon.



```python
# combined network
# LeNet Model definition
class MNIST_Net_combined(nn.Module):
    def __init__(self, net_type='gray'):
        super(MNIST_Net_combined, self).__init__()

        self.encoder = nn.Sequential( # like the Composition layer you built
            nn.Conv2d(1, 16, 3, stride=2, padding=1),
            nn.ReLU(),
            nn.Conv2d(16, 32, 3, stride=2, padding=1),
            nn.ReLU(),
            nn.Conv2d(32, 64, 7)
        )
        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(64, 32, 7),
            nn.ReLU(),
            nn.ConvTranspose2d(32, 16, 3, stride=2, padding=1, output_padding=1),
            nn.ReLU(),
            nn.ConvTranspose2d(16, 1, 3, stride=2, padding=1, output_padding=1),
            nn.Sigmoid()
        )

        self.conv1 = nn.Conv2d(1, 10, kernel_size=5)
        self.conv2 = nn.Conv2d(10, 20, kernel_size=5)
        self.conv2_drop = nn.Dropout2d()
        self.fc1 = nn.Linear(320, 50)
        self.fc2 = nn.Linear(50, 10)


    def forward(self, x):
        z = self.encoder(x)
        x_auto = self.decoder(z) # reconstructed egde
        x_auto = x_auto.view(x_auto.shape[0],1, 28,28)
        x = F.relu(F.max_pool2d(self.conv1(x_auto), 2))
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)), 2))
        x = x.view(-1, 320)
        x = F.relu(self.fc1(x))
        x = F.dropout(x, training=self.training)
        x = self.fc2(x)
        return x, x_auto
```

Figure 9: PyTorch code of our pipeline shown in Fig 8.

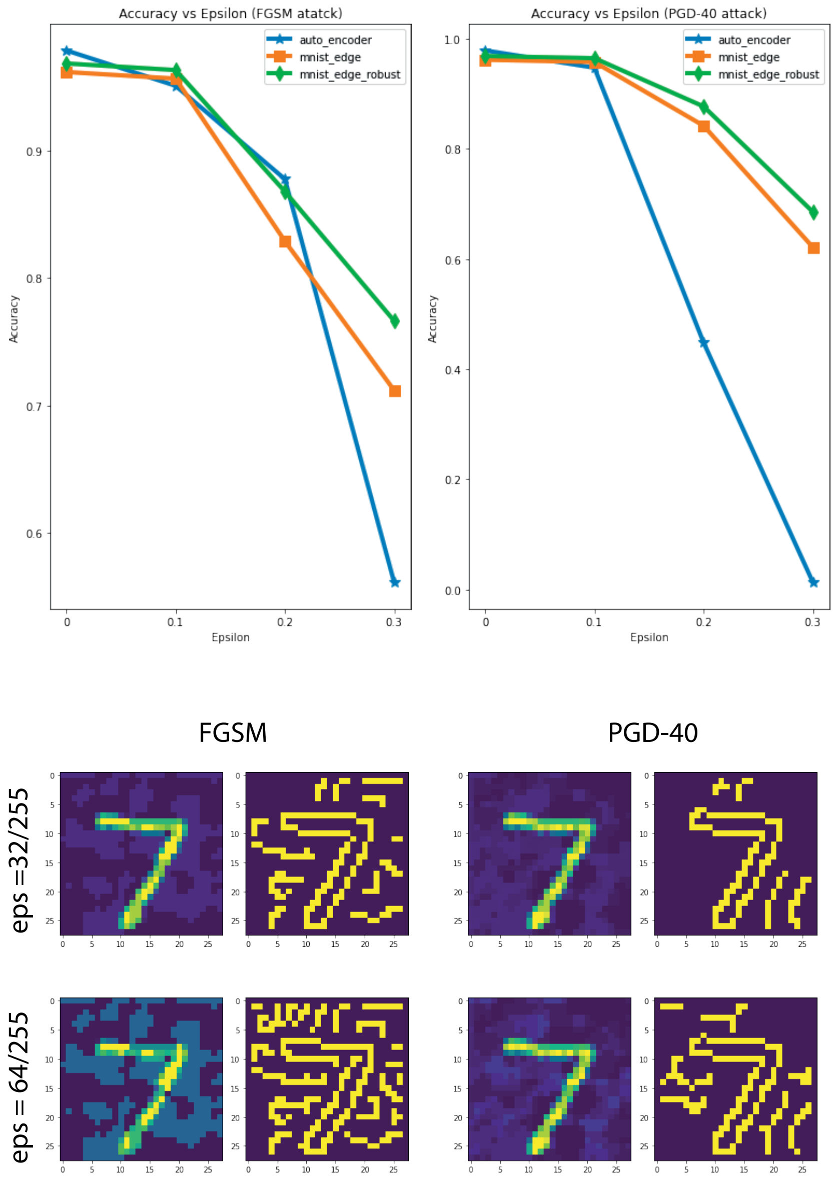


Figure 10: Top: Performance of the adaptive attack, Bottom: Samples adversarial images and their edge maps using the Canny edge detector.

F Sample generated images by the conditional GAN in GAN-based Shape Defense (GSD)

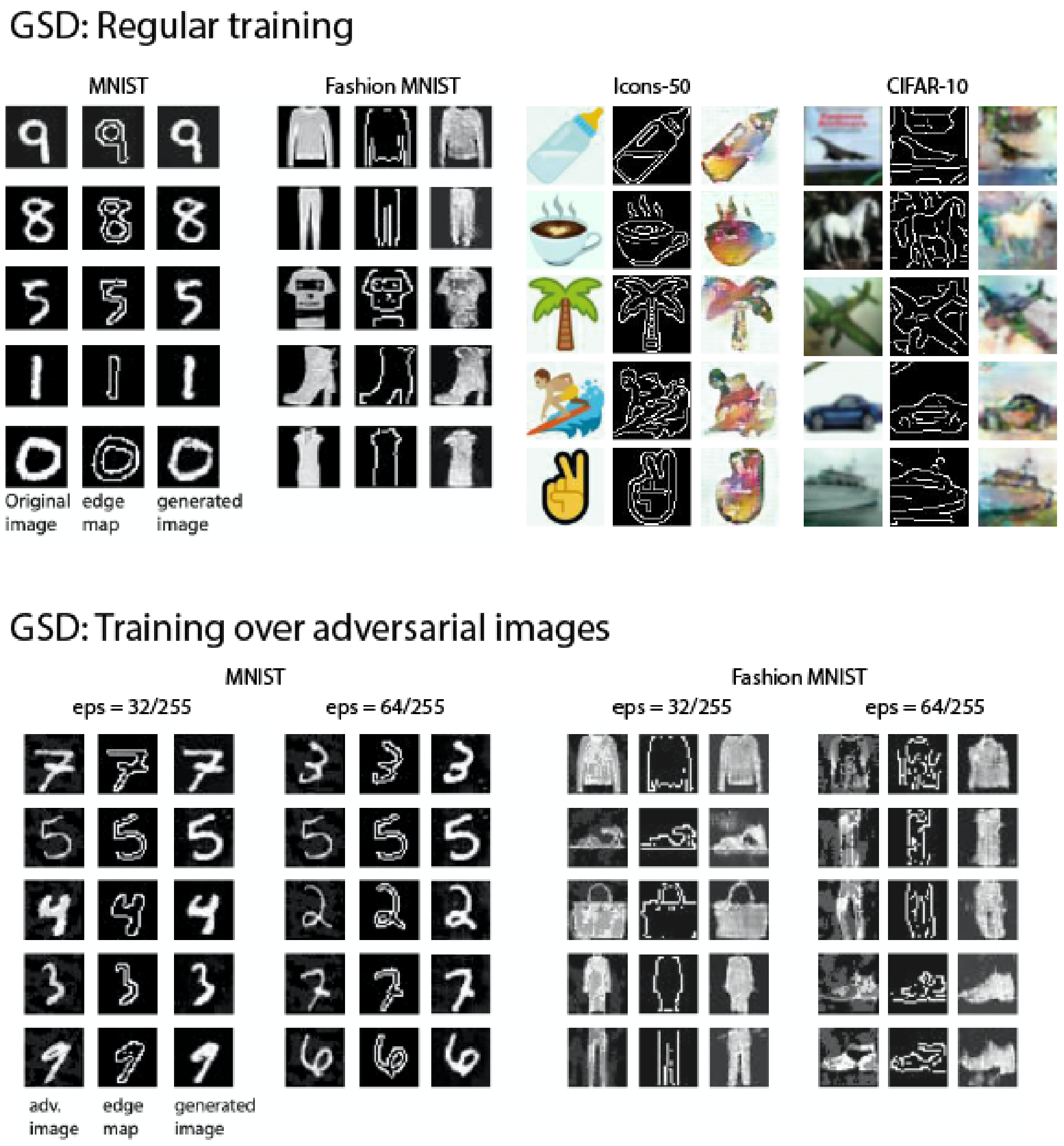


Figure 11: Top) GSD with a classifier trained on images generated (by pix2pix) only from the edge maps of the clean images, Bottom) GSD with edge maps derived from adversarial examples. Columns from left to right: adversarial images by the FGSM attack, their edge maps, and generated images by pix2pix.

G Shape-based extensions of vanilla PGD adversarial training, free adversarial training (FreeAT), and fast adversarial training (FastAT) algorithms

Algorithm 3 Shape-based PGD adversarial training for T epochs, given some radius ϵ, adversarial step size α and N PGD steps and a dataset of size M for a network f_θ. $\beta \in \{edge, img, imgedge\}$ indicates the net_type and *redetect_train* mean edge redetection during training.

```
for $t = 1 \dots T$ do
  for $i = 1 \dots M$ do
    // Perform PGD adversarial attack
    $\delta = 0$ // or randomly initialized
    for $j = 1 \dots N$ do
      $\delta = \delta + \alpha \cdot \mathrm{sign}(\nabla_\delta \ell(f_\theta(x_i + \delta), y_i))$
      $\delta = \max(\min(\delta, \epsilon), -\epsilon)$
    end for
    $\tilde{x}_i = x_i + \delta$
    if redetect_train & $\beta$ == imgedge then
      $\tilde{x}_i =$ detect_edge($\tilde{x}_i$)    // recompute and replace the edge map
    end if
    $\theta = \theta - \nabla_\theta \ell(f_\theta(\tilde{x}_i), y_i)$ // Update model weights with some optimizer, e.g. SGD
  end for
end for
```

Algorithm 4 Shape-based "Free" adversarial training for T epochs, given some radius ϵ, N minibatch replays, and a dataset of size M for a network f_θ. $\beta \in \{edge, img, imgedge\}$ indicates the net_type and *redetect_train* mean edge redetection during training.

```
$\delta = 0$
// Iterate T/N times to account for minibatch replays and run for T total epochs
for $t = 1 \dots T/N$ do
  for $i = 1 \dots M$ do
    // Perform simultaneous FGSM adversarial attack and model weight updates $T$ times
    for $j = 1 \dots N$ do
      $\tilde{x}_i = x_i + \delta$
      if redetect_train & $\beta$ == imgedge then
        $\tilde{x}_i =$ detect_edge($\tilde{x}_i$)    // recompute and replace the edge map
      end if
      // Compute gradients for perturbation and model weights simultaneously
      $\nabla_\delta, \nabla_\theta = \nabla \ell(f_\theta(\tilde{x}_i), y_i)$
      $\delta = \delta + \epsilon \cdot \mathrm{sign}(\nabla_\delta)$
      $\delta = \max(\min(\delta, \epsilon), -\epsilon)$
      $\theta = \theta - \nabla_\theta$ // Update model weights with some optimizer, e.g. SGD
    end for
  end for
end for
```

Algorithm 5 Shape-based FGSM adversarial training for T epochs, given some radius ϵ, N PGD steps, step size α, and a dataset of size M for a network f_θ. $\beta \in \{edge, img, imgedge\}$ indicates the net_type and *redetect_train* mean edge redetection during training.

```
for $t = 1 \ldots T$ do
  for $i = 1 \ldots M$ do
    // Perform FGSM adversarial attack
    $\delta = \text{Uniform}(-\epsilon, \epsilon)$
    $\delta = \delta + \alpha \cdot \text{sign}(\nabla_\delta \ell(f_\theta(x_i + \delta), y_i))$
    $\delta = \max(\min(\delta, \epsilon), -\epsilon)$
    $\tilde{x}_i = x_i + \delta$
    if redetect_train & $\beta$ == imgedge then
      $\tilde{x}_i = \text{detect\_edge}(\tilde{x}_i)$    // recompute and replace the edge map
    end if
    $\theta = \theta - \nabla_\theta \ell(f_\theta(\tilde{x}_i), y_i)$ // Update model weights with some optimizer, e.g. SGD
  end for
end for
```

Table 21: Performance of the Fast Adversarial Training (FastAT) method over three runs.

	Run 1		Run 1		Run 1		Average	
Model	Clean	PGD-10	Clean	PGD-10	Clean	PGD-10	Clean	PGD-10
Edge	0.559	0.384	0.581	0.187	0.608	0.586	0.582	0.386
RGB	0.813	0.368	0.598	0.205	0.889	0.569	0.767	0.381
Img + Edge	0.863	0.590	0.882	0.334	0.878	0.878	**0.874**	0.386
Redetect	„	0.593	„	0.341	„	0.245	„	0.393
RGB + Redet. Edge	0.892	0.001	0.817	0.115	0.889	0.105	0.866	0.074
Redetect	„	0.265	„	0.656	„	0.326	„	**0.416**

Table 22: Performance of the Free Adversarial Training (FreeAT) method over three runs.

	Run 1		Run 1		Run 1		Average	
Model	Clean	PGD-10	Clean	PGD-10	Clean	PGD-10	Clean	PGD-10
Edge	0.674	0.672	0.704	0.702	0.660	0.659	0.679	**0.678**
RGB	0.783	0.450	0.768	0.450	0.772	0.447	0.774	0.449
Img + Edge	0.784	0.432	0.779	0.447	0.782	0.448	**0.782**	0.442
Redetect	„	0.447	„	0.448	„	0.449	„	0.448
RGB + Redet. Edge	0.776	0.451	0.776	0.454	0.780	0.447	0.777	0.451
Redetect	„	0.452	„	0.456	„	0.448	„	0.452

H Analysis of parameter α in Alg. 1 (EAT defense)

Table 23: Results (Top-1 acc.) over MNIST corresponding to $\alpha = 0$ (i.e., adversarial training only on adversarial examples taking part in the loss function). See also Table 1 in the main text.

	Rob. model (8)		**Rob. model (32)**		**Rob. model (64)**		**Average**
ϵ	0/clean	8	0/clean	32	0/clean	64	**Rob. models**
			FGSM				
Img+Edge	**0.963**	0.938	**0.959**	0.869	0.931	0.684	0.891
Redetect	”	0.943	„	0.887	„	0.727	0.902
Img + Redetected Edge	**0.963**	0.936	0.944	0.588	**0.937**	0.030	0.733
Redetect	”	**0.948**	„	**0.911**	„	**0.916**	**0.937**
			PGD-40				
Img+Edge	**0.966**	0.940	**0.960**	0.859	0.928	0.607	0.877
Redetect	”	**0.946**	„	0.883	„	0.657	0.890
Img + Redetected Edge	0.963	0.933	0.947	0.469	**0.936**	0.000	0.708
Redetect	”	**0.946**	„	**0.913**	„	**0.915**	**0.937**

Table 24: Results (Top-1 acc.) over Fashion MNIST corresponding to $\alpha = 0$ (i.e., adversarial training only on adversarial examples taking part in the loss function). See also Table 4 in the main text.

	Rob. model (8)		**Rob. model (32)**		**Rob. model (64)**		**Average**
ϵ	0/clean	8	0/clean	32	0/clean	64	**Rob. models**
			FGSM				
Img+Edge	0.756	0.701	0.732	0.619	0.683	0.487	0.663
Redetect	”	0.707	„	0.635	„	0.481	0.666
Img + Redetected Edge	**0.768**	0.705	**0.739**	0.481	**0.693**	0.040	0.571
Redetect	”	**0.727**	„	**0.660**	„	**0.635**	**0.704**
			PGD-40				
Img+Edge	0.768	0.702	0.749	0.573	0.718	0.432	0.657
Redetect	”	0.714	„	0.593	„	0.510	0.675
Img + Redetected Edge	**0.778**	0.702	**0.762**	0.414	**0.750**	0.001	0.568
Redetect	”	**0.725**	„	**0.632**	„	**0.615**	**0.710**