A U-Net Based GAN for Adversarial Attacks

Emily Blixt Andreas Führ Erik Håkansson Lisa Lövgren Filip Olsson

Background

Despite recent advances in generative AI, many of the underlying architectures used, namely encoder-decoders and Generative Adversarial Networks (GANs [1]) remain fallible to the lacking invariance inherent in most neural network architectures. This poses security risks, as classifiers can be deceived by imperceptible alterations, termed ”adversarial attacks”. Research focuses mainly on non-generative methods, adding and adjusting noise to induce misclassifications. We propose a purely synthetic approach based on a U-Net [2] GAN were adversarial examples are synthetically created. We demonstrate the feasability of such an approach on the MNIST digits dataset, as well as contrast the performance with alternative methods. We also explore the possibility of relying on syntehtic generated adveserial examples as a data augmentation strategy to bolster classifier robustness.

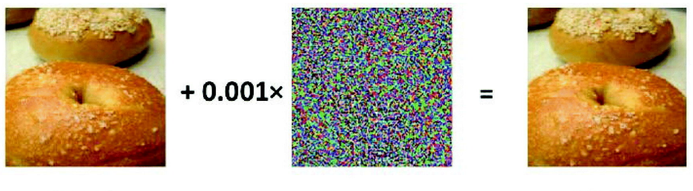


Figure 1. Illustration of an adversarial attack, where an image is subtly altered to fool a classifier [3]

Method

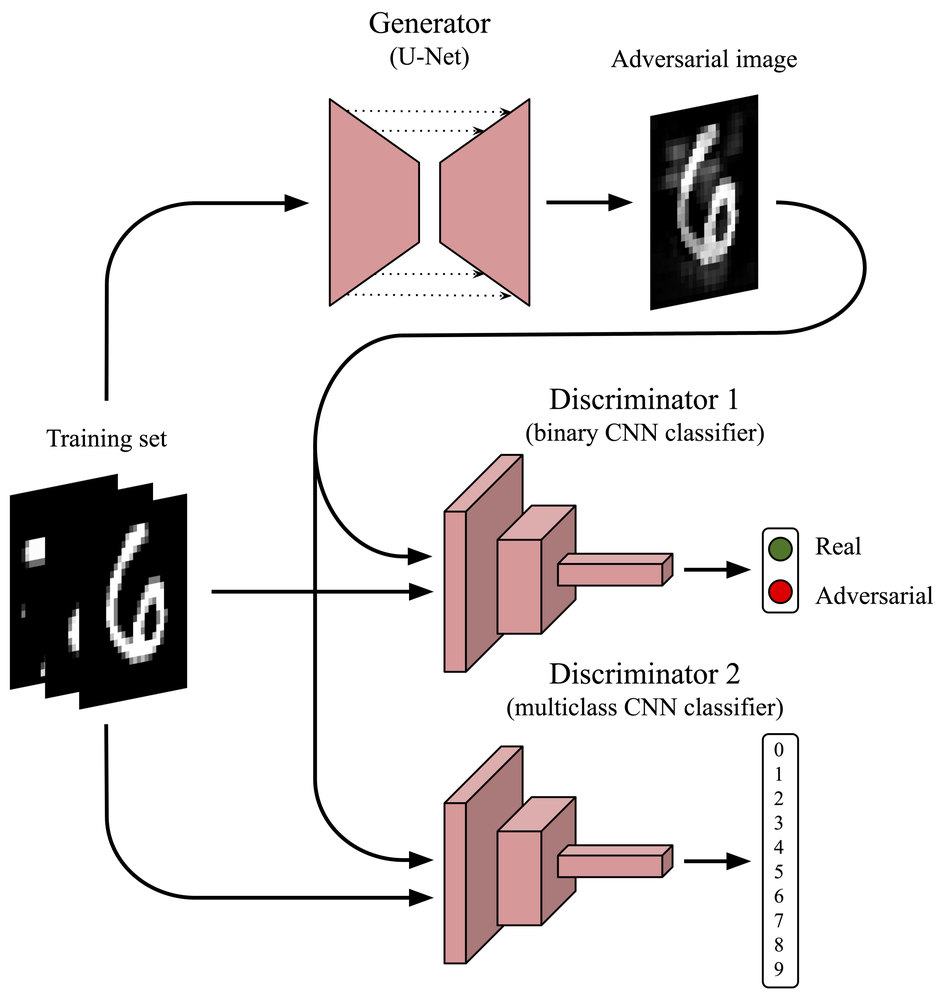


Figure 2. U-Net based GAN architecture with two discriminators. The discriminators are convolutional neural networks that make sure that (1) the images look like real MNIST digits, and (2) that they are misclassified.

Competition through loss functions

- **Generator:** The loss function for the generator is a weighted sum of the losses from the discriminators, plus a penalty term for the norm.

$$\mathcal{L}_{\text{Gen}} = \alpha\mathcal{L}_{\text{Disc. 1}} + \beta\mathcal{L}_{\text{Disc. 2}} + \gamma\mathcal{L}_{\text{Norm}} \tag{1}$$

  The weighting of the three losses is determined by the coefficients α, β and γ, which were set to 2, 100 and 0,5 respectively, through trial and error. It was observed that the GAN is highly sensitive to the weighting of these loss functions. Discriminator 2 loss holds the most weight, since the generator’s main objective is to generate images that are misclassified.
- **Discriminator 1:** Ensures that the generated image looks like a ”valid MNIST image”. The loss function averages the loss of unaltered and adversarial images.
- **Discriminator 2:** Performs digit classification to push the generator to generate images that fool the classifier. The loss used for training is independent of the generator.
- **Norm metric:** Norm of difference between input image and generated image pixel values is used for the generator loss function. This is in order to visually imitate the input image.
- **Gaussian Noise:** Used as comparison to the adversarial noise to prove that our generated images can fool Discriminator 2 better, while modifying the images less.

References

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, *Generative adversarial networks*, 2014. arXiv: 1406.2661 [stat.ML].

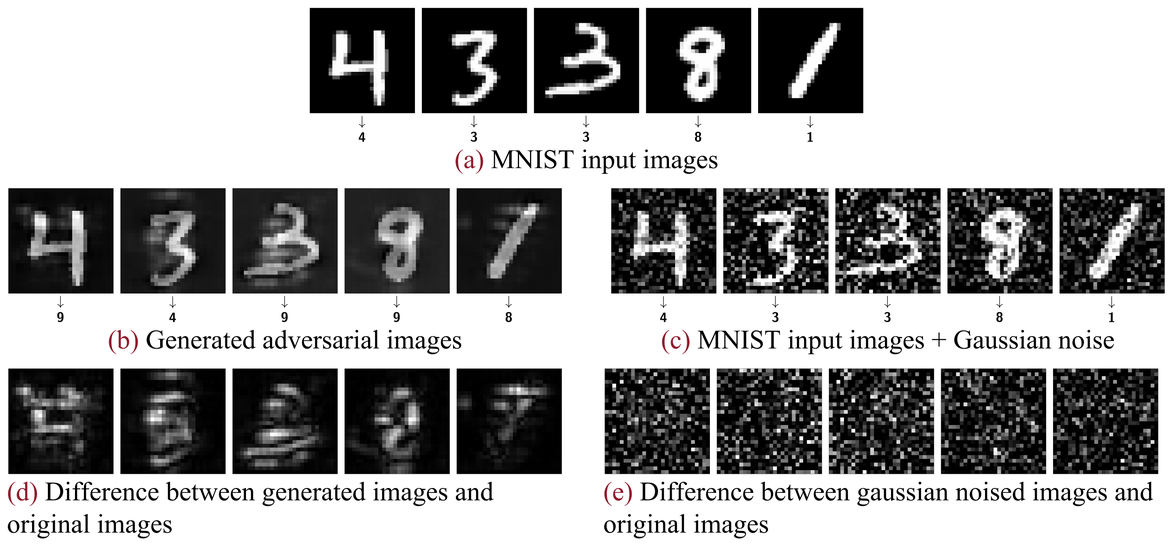
[2] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” *arXiv e-prints*, arXiv:1505.04597, arXiv:1505.04597, May 2015. DOI: 10.48550/arXiv.1505.04597. arXiv: 1505.04597 [cs.CV].

[3] C.-J. H. Yao Li Minhao Cheng and T. C. M. Lee, “A review of adversarial attack and defense for classification methods,” *The American Statistician*, vol. 76, no. 4, pp. 329–345, 2022. DOI: 10.1080/00031305.2021.2006781.

[4] W. Falcon and The PyTorch Lightning team, *PyTorch Lightning*, version 1.4, Mar. 2019. DOI: 10.5281/zenodo.3828935. [Online]. Available: https://github.com/Lightning-AI/lightning.

Key Results

Image classification



(a) MNIST input images

(b) Generated adversarial images

(c) MNIST input images + Gaussian noise

(d) Difference between generated images and original images

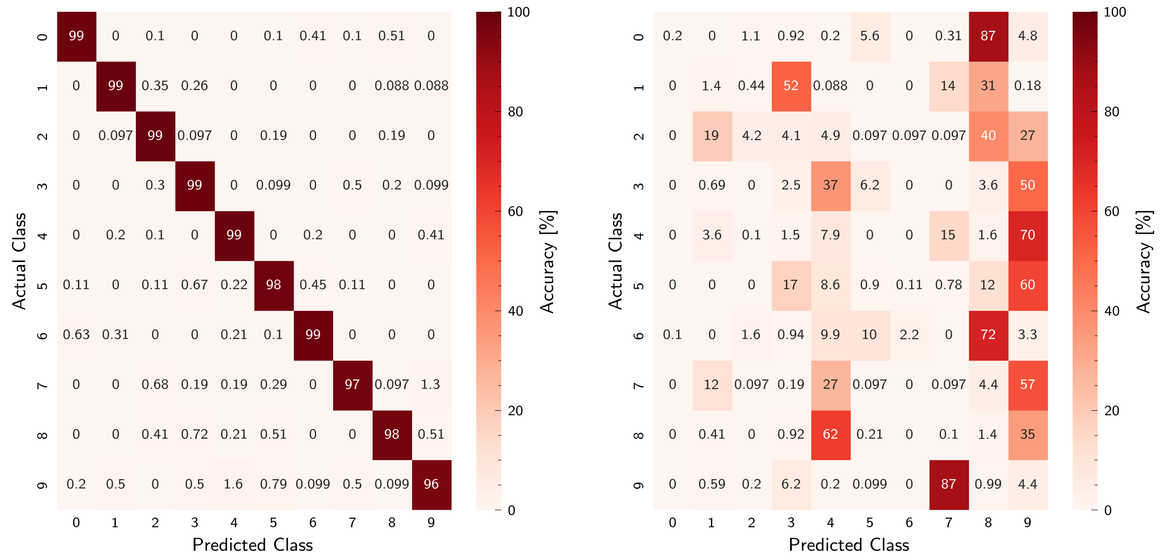
(e) Difference between gaussian noised images and original images

Figure 3. Comparison of input, generated and noisy images. The input (a) is fed into the generator, which outputs the generated adversarial images (b). Subfigures (d) and (e) show the difference between input images and the produced images to illustrate what was changed.

Classification accuracy

Table 1. Test accuracies for different types of images. The same test set of 10 000 images were noised and fed through the generator.

Image type	Accuracy
Ground truth	**98,3 %**
Added Gaussian noise	**90,3 %**
Adversarial	**2,5 %**



(a) **Unaltered images** from MNIST.

(b) **Adversarial images** generated from MNIST.

Figure 4. Confusion matrices of the classifications from discriminator 2 for unaltered images (a) and generated adversarial images (b) from the MNIST dataset. The confusion matrices were generated using the same test set of 10 000 images and discriminator 2 model with identical weights and biases.

Structural similarity of altered images

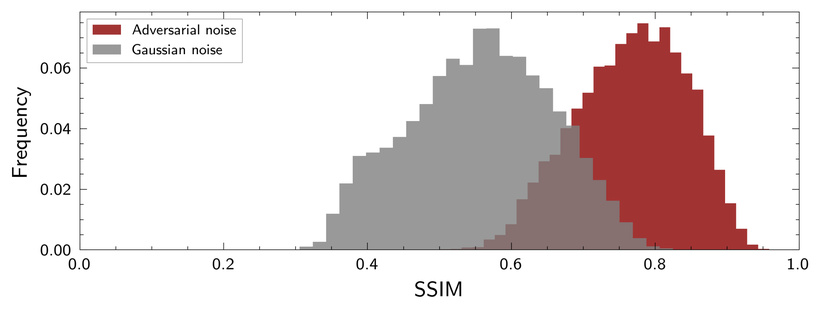


Figure 5. Histogram of the structural similarity index measure (SSIM) forgenerated adversarial images and images with Gaussian noise from the MNIST test set, as compared to the original images. The mean SSIM for the adversarial images was 0,77 and for the Gaussian noised images 0,56. The SSIM was calculated with default parameters [4].

Discussion & Conclusions

- Our generator U-Net have learned to alter images from the MNIST dataset *with high structural similarity* that fool our CNN classifier discriminator (trained with independent loss in parallel) in over 97 % of cases. The discriminator shows an average accuracy of 98,3 % across the same test set.
- A majority of the generated adversarial images are misclassified as *either the digit 8 or 9*, which are two of the *most difficult digits to classify correctly* in the original MNIST dataset in terms of accuracy.
- Gaussian noise added to images gives a *significantly higher accuracy* of 90,3 % compared to the generated adversarial images, while on averege having a *lower structural similarity index measure (SSIM)*.
- The generator (16×16 pixel input/output) has learned to add adversarial noise to images through *local, overlapping image patches* of MNIST digits (28×28 pixels). This is a strong indication that there exists *inherently local features* in the classifier discriminator.