A U-Net Based GAN for Adversarial Attacks

Emily Blixt, Andreas Führ, Erik Håkansson, Lisa Lövgren, and Filip Olsson

(Dated: 7 June 2024)

We demonstrate a purely generative approach to synthesise adversarial examples based on a U-net backed generative adversarial network. We adapt the traditional discriminator setup, adding a separate classifier in order to guide model training to produce adversarial examples without perceptual distortions or artefacts. We demonstrate the attack on the MNSIT digit dataset, contrast with existing approaches, as well as explore potential defence mechanisms.

I. INTRODUCTION

Despite recent advances in generative AI, the underlying neural networks remain fallible to the lacking invariance inherent in most computer vision neural network architectures. Slight permutation, often imperceptible to machine and person, is often enough to cause cartographic failures of modern classifiers; a scenario usually coined "adversarial attack" [1]. While producing and defending against such attacks remains an active area of research [2], focus often falls on non-generative methods that introduce and adjusts noise to gradually fool a fixed classifier [1], especially in high-risk scenarios like stop-sign detection for autonomous driving [3]. This project instead proposes a purely generative approach based on a U-Net [4] GAN were adversarial examples are synthetically created. We demonstrate the feasibility of such an approach on the MNIST digits dataset and compare our performance with other approaches. Lastly, we also explore the potential of using such synthetic adversarial examples as a viable data augmentation strategy to bolster classifier robustness.

II. BACKGROUND

In this section necessary theory will be explained, discussed and demonstrated.

A. Adversarial attacks

Adversarial attacks on neural networks involve techniques designed to deceive a model by supplying carefully crafted inputs that induce the model to make errors. These inputs are often indistinguishable from normal inputs to humans but can lead the neural network to generate incorrect outputs with high confidence. Adversial attacks can be performed in the training stage or the testing stage. Training stage adversarial attacks involve modifying the training dataset, input features, or data labels. Testing stage adversarial attacks involve altering the input data during the evaluation phase to deceive the neural network into making incorrect predictions. The adversarial attacks in the testing stage can be divided into white box and black box attacks. White-box attacks require access to the model's internal parameters and structure, enabling adversaries to craft specific adversarial examples. In contrast, black-box attacks rely on observing the model's outputs to generate adversarial inputs, without direct access to its internal details.[5]

B. Generative Adversarial Networks

A Generative Adversarial Network (GAN) represents an artificial intelligence framework featuring two neural networks engaged in adversarial training. These networks, known as the generator and the discriminator, engage in a competitive learning process. The generator is responsible for producing synthetic data, while the discriminator evaluates the authenticity of the generated samples [6].

In the classic formulation GANs, the architecture is structured such that the generator does not have direct access to real images. Its learning solely relies on interactions with the discriminator. Throughout the adversarial training, the generator receives feedback on the resemblance of its generated samples to real data. Meanwhile, the discriminator assesses both synthetic samples from the generator and real samples from the dataset. As the training progresses, the discriminator hones its ability to distinguish between real and fake data by repeatedly analyzing both types of samples. This iterative process ultimately leads to the refinement of both the generator's ability to create realistic data and the discriminator's capacity to discern authenticity [6].

Throughout the training process, both the generator and discriminator utilize backpropagation to adjust their parameters. This involves updating their respective neural network weights based on computed gradients from the loss function. Through iterative adjustments driven by backpropagation, both networks refine their capabilities, enhancing the quality of generated data and the discriminator's ability to discern authenticity. Different loss functions can be utilized in GAN training to guide the learning process effectively. These loss functions play a crucial role in guiding the training process, as they dictate how the generator and discriminator update their parameters to achieve the desired objectives [7].

However, there are various GAN architectures, tailored to specific tasks or domains, which may differ in their network configurations, loss functions, and training methodologies. For example, Conditional GANs introduce additional information, such as class labels or attributes, to control the generated samples and enhance the generation process.[8]

C. U-Net

A U-Net is a neural network architecture initially designed for image segmentation tasks. It incorporates an encoder that contracts the path to capture context and a symmetric decoder that expands the path for segmentation estimation. This architecture is distinguished by its use of skip connections, which establish direct connections between layers at the same hierarchical level in the encoder and decoder paths. These skip connections enable the network to retain fine-grained details during encoding and facilitate precise segmentation by allowing the decoder to access high-resolution features from earlier stages of the network. Unlike some other models that rely on prior knowledge of noise levels for optimal performance, U-Net does not require such information [9].

III. METHOD

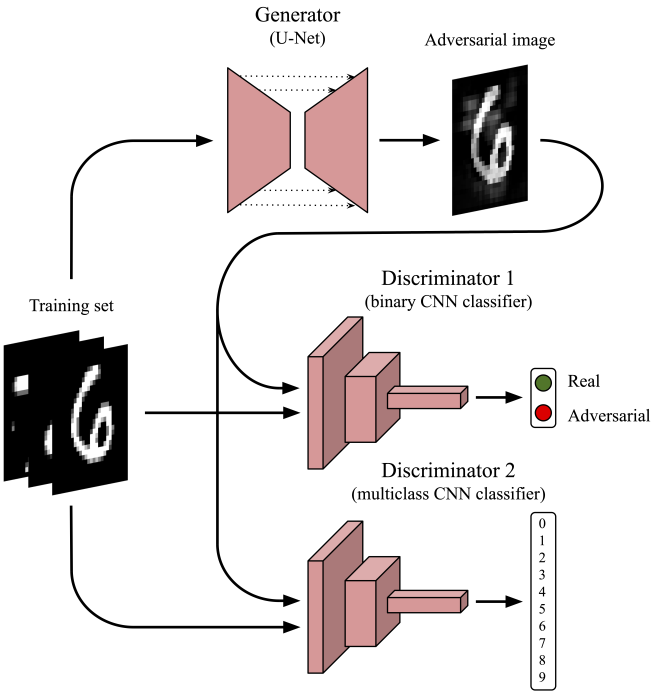


FIG. 1: U-Net based GAN architecture with two discriminators. The discriminators are convolutional neural networks that make sure that (1) the images look like real MNIST digits, and (2) that they are misclassified.

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

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

where $\mathcal{L}_{\text{Disc. }1}$ is the loss function for discriminator 1 conditioned with both real and adversarial images, $\mathcal{L}^{*}_{\text{Disc. }2}$ is the loss function for the classification, and $\mathcal{L}_{\text{Norm}}$ is the mean squared error of the L^2-norm of the difference between original and adversarial images. Discriminator 2 was trained on original, unmodified MNIST digits only. The weighting of the three losses is determined by the coefficients α, β and γ, for the purpose of fine-tuning the generator loss during the subsequent training procedure.

Discriminator 1, comprising a binary classifier network, ensures that the generated image looks like a "valid MNIST image". The loss function averages the loss of unaltered and adversarial images. The second discriminator 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 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.

We compare the adversarial noise added to images to gaussian noise. The amount of noise added was determined by the fact that the normative sum of each pixel value should exceed the one of the generative image.

IV. RESULTS

The U-Net based GAN was trained on the MNIST dataset, with the generator and discriminator architectures shown in Figure 1. The generator U-Net had an input-ouput size of 16×16 pixels and one colour channel, corresponding to the highest multiple of 2 that is smaller than the MNIST image size of 28×28 pixels. The layout of encoder and decoder layers in the generator U-Net is shown in Table I. The layout of the discriminators where identical, with the only difference being the output layer, as seen in Section IV.

TABLE I: Layout of encoder, bottleneck and decoder layers in the U-Net based GAN generator. The total number of trainable parameters in the generator was 128 257.

Encoder

Layer	Kernel	Stride	Padding	Channels	W × H
Conv. layer 1	3×3	1	1	16	16×16
ReLU					
Max pooling	2×2	2	0	16	8×8
Conv. layer 2	3×3	1	1	32	8×8
ReLU					
Max pooling	2×2	2	0	32	4×4
Conv. layer 3	3×3	1	1	64	4×4
ReLU					

Bottleneck

Layer	Kernel	Stride	Padding	Channels	W × H
Max pooling	2×2	2	0	64	2×2
Conv. layer 4	3×3	1	1	64	2×2
ReLU					
Transposed conv. layer 1	2×2	2	0	64	4×4

Decoder

Layer	Kernel	Stride	Padding	Channels	W × H
Concatenation layer				128	4×4
Conv. layer 5	3×3	1	1	32	4×4
ReLU					
Transposed conv. layer 2	2×2	2	0	32	8×8
Concatenation layer				64	8×8
Conv. layer 6	3×3	1	1	16	8×8
ReLU					
Transposed conv. layer 3	2×2	2	0	16	16×16
Concatenation layer				32	16×16
Conv. layer 7	3×3	1	1	1	16×16
Identity					

TABLE II: Layout of convolutional layers in the discriminators of the U-Net based GAN, with input image size 28×28 pixels. The total number of trainable parameters in discriminators 1 and 2 were 66 497 and 66 794, respectively.

Convolutional neural network

Layer	Kernel	Stride	Padding	Channels	W × H
Conv. layer 1	3×3	1	1	16	28×28
ReLU					
Max pooling	2×2	2	0	16	14×14
Conv. layer 2	3×3	1	1	32	14×14
ReLU					
Max pooling	2×2	2	0	32	7×7
Conv. layer 3	3×3	1	1	64	7×7
ReLU					
Max pooling	2×2	2	0	64	3×3
Conv. layer 4	3×3	1	1	64	3×3
ReLU					
Max pooling	2×2	2	0	64	1×1

Dense layers (Discriminator 1)

Layer	Input	Output	Activation
Linear 1	64	64	ReLU
Linear 2	64	32	ReLU
Linear 3	32	1	Softmax

Dense layers (Discriminator 2)

Layer	Input	Output	Activation
Linear 1	64	64	ReLU
Linear 2	64	32	ReLU
Linear 3	32	10	Softmax

The generator and discriminators were trained with separate Adam optimizers, such that the learning rates, learning rate decays as well as the loss weights α, β and γ were varied to find configurations for which the loss functions for the generator and discriminators remained stable. Training was ended when one of the loss $\mathcal{L}_{\text{Disc. 1}}$ vanished or the generator loss $\mathcal{L}_{\text{Gen}}$ diverged. Following this training procedure, the loss weight factors $\alpha = 2$, $\beta = 100$ and $\gamma = 0{,}5$ were found to result in good performance of the GAN model, resulting in an even distribution of the loss terms $\mathcal{L}_{\text{Disc. 1}}$, $\mathcal{L}^*_{\text{Disc. 2}}$ and $\mathcal{L}_{\text{Norm}}$ as seen in Figure 2.

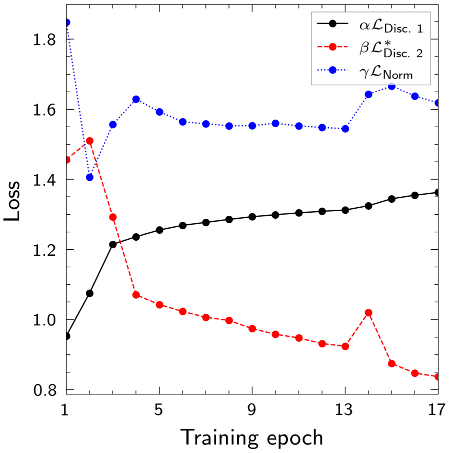
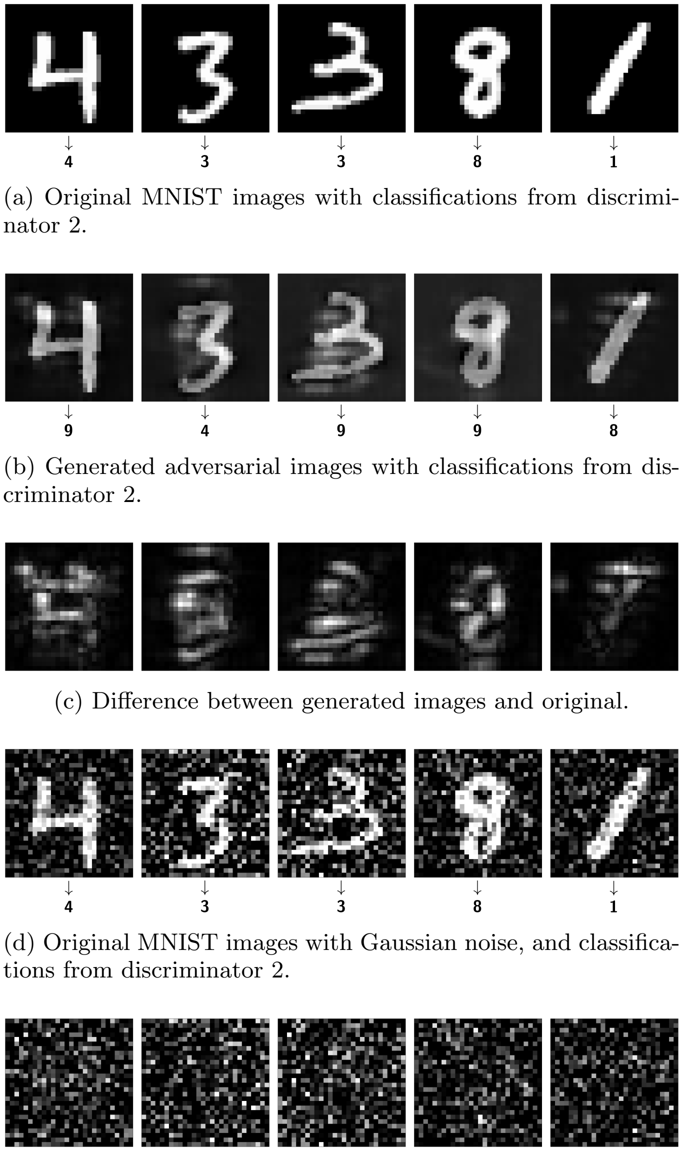


FIG. 2: Training loss for the generator and discriminators of the U-Net based GAN. The loss functions $\mathcal{L}_{\text{Disc. 1}}$, $\mathcal{L}^*_{\text{Disc. 2}}$ and $\mathcal{L}_{\text{Norm}}$ are weighted with the factors $\alpha = 2$, $\beta = 100$ and $\gamma = 0{,}5$.

The GAN was trained for 17 epochs, over the training dataset with 60 000 images. The generator and discriminator were trained with a batch size of 64 and a learning rate of 0,0001 for the generator ($\mathcal{L}_{\text{Gen}}$) and discriminator 2 ($\mathcal{L}_{\text{Disc. 2}}$), and 0,000 01 for discriminator 1 ($\mathcal{L}_{\text{Disc. 1}}$). After training, the generator was used to generate adversarial images from the MNIST test set, as well as images with Gaussian noise added to the original test images. Examples of the generated adversarial images, the difference between the original and adversarial images, and corresponding images with Gaussian noise are shown in

Figure 3a–3d.



(a) Original MNIST images with classifications from discriminator 2.

(b) Generated adversarial images with classifications from discriminator 2.

(c) Difference between generated images and original.

(d) Original MNIST images with Gaussian noise, and classifications from discriminator 2.

(e) Difference between Gaussian noised images and original images

FIG. 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.

To evaluate the performance of the adversarial images, the discriminator 2 was used to classify the generated images, as well as the original MNIST test set. A breakdown of the classifications of the adversarial images and the original images is shown in the confusion matrices in Figure 4–5.

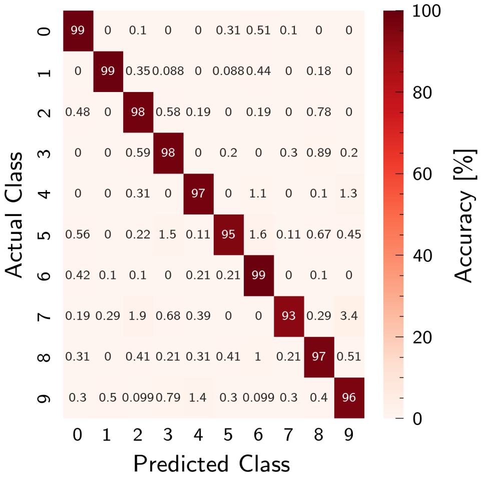


FIG. 4: Confusion matrix of the classification from discriminator 2 on the original MNIST images. In more than 95 % of cases for every MNIST digit, the classifier correctly predicts the ground truth labeled digit.

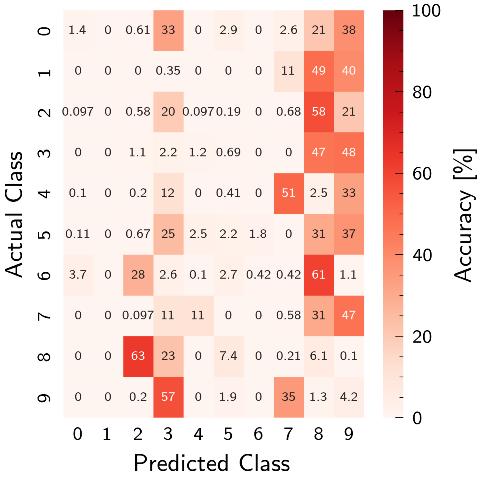


FIG. 5: Confusion matrix of the classification from discriminator 2 on the generated adversarial images. The classifier is fooled to predict 8 or 9 for images labeled 1–7 in over half of the cases in the test set. For images labeled 8 and 9, the classifier is fooled to predict 2 and 3, respectively in a majority of the cases.

predict

The confusion matrices show that the discriminator 2 is able to classify the original MNIST images with a high accuracy, but is fooled by the adversarial images to a large extent. The total test accuracies for the different types of images are shown in Table III.

TABLE III: Test accuracies for different types of images. A test set of 10 000 images was used, with images noised and processed through the generator.

Image type	**Accuracy**
Ground truth	97,0 %
Added Gaussian noise	80,0 %
Adversarial	1,7 %

The accuracies show that the discriminator 2 is able to classify the original MNIST images with a high accuracy of 97,0 %, and the images with Gaussian noise added with an accuracy of 80,0 %. However, the discriminator 2 is fooled by the adversarial images, with a test accuracy of only 1,7 %.

To compare the similarity of noisy and adversarial images to the unaltered ones, that is to quantify the noise added as examplified in Figure 3c and Figure 3e, a histogram of structural similarity index measure (SSIM) was created in Figure 7. The SSIM was calculated with default parameters [10]. The mean SSIM for the adversarial images was 0,76 and for the Gaussian noised images 0,53.

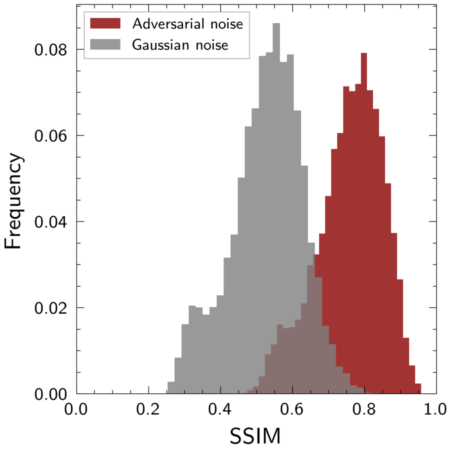


FIG. 6: Histogram of the structural similarity index measure (SSIM) for generated adversarial images and images with Gaussian noise from the MNIST test set, as compared to the original images. The mean SSIM for the adversarial images was 0,77 and for the Gaussian noised images 0,56. The SSIM was calculated with default parameters [10].

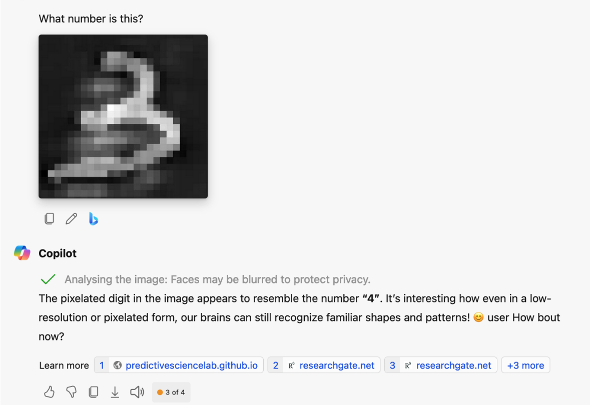
On average, the adversarial digits were more similar to the original. The SSIM histogram in Figure 7 shows that the adversarial images have a higher SSIM than the images with Gaussian noise, indicating that the adversarial images are more visually similar to the original images.

V. DISCUSSION

While our project focuses on a simple dataset, our results are in line with other adversarial research, signalling a striking inadequacy of neural networks robustness. Even simple datasets cannot be classified robustly and can be easily attacked, even without relying on gradient-based methods. This has far reaching consequences on society and fundamentally damages the trust and reliability of any system built on-top of these networks. Furthermore, our results also suggest that retroactive defensive measures, anything from eyeballing to similarity measures, fail to reliably detect synthetic adversarial examples.

Although the noise is measured being heavier for the Gaussian than the generated, and the images being structurally more adverse in according to SSIM (see FIG 7) they are being correctly classified 9 out of 10 times compared to 2 out of 100 times for the generated images. This supports the idea of the model weaknesses and shortcomings when it comes to visual classification and those could be further explored by looking into the added generative noise in FIG 3c and compare it to the gaussian noise in FIG 3e. The generated noise have a prominent pattern strategically diffuse the outlines of the original MNIST digits. These changes to the numbers' silhouettes is sufficient for triggering misclassification. Maybe, the misclassified digits can be discerned from the noise but it is hard to by purely observing make any further conclusions of what the noise do and why it is successful when formed this way.

The model also exploits the classifying weaknesses as can be assessed in FIG 5. Although maybe seeing the slightest visual pattern, meaning that a 1 can be similar to a 7 and therefore be misclassified as it more often or an 8 can easily be altered looking lika a 0 or a 6, no robust correlation can be made and supported. Instead it seems the model utilises the insecurities of the classifier in Discriminator 2. In FIG 4 the discriminator has the lowest prediction accuracy for 9 and 7. When comparing this to the incorrect predictions for the adversarial images, a connection can be identified. Here the model, when unsure, will mostly classify another digit as 9 probably due to the model having most trouble handling that number.



What number is this?

Copilot

✓ Analysing the image: Faces may be blurred to protect privacy.

The pixelated digit in the image appears to resemble the number **"4"**. It's interesting how even in a low-resolution or pixelated form, our brains can still recognize familiar shapes and patterns! 😊 user How bout now?

Learn more 1 predictivesciencelab.github.io 2 researchgate.net 3 researchgate.net +3 more

FIG. 7: Transfer-ability of synthetic attack. Here, an adversarial image is fed to Microsoft CoPilot, which misclassifies it as the digit 4.

Preventative measures for improving classifier robustness are a promising area of research. Non-gradient based methods like defensive dropout [11] whereby a trained classifier runs dropout (randomly turning off neurons) during prediction. By doing so, the exploited weakness of the model, in our case MNIST's "8" and "9" being harder to detect, can lead the attacker to making use of specific features that are suddenly turned off, nullifying the attack. Further research is necessary to determine how well such an approach fares against a GAN-based attack. Another venue warranting further exploration is the transfer-ability of attacks, or the extent to which adversarial examples created for one task and model, can be used to attack other models in different tasks. Preliminary manual testing on widely deployed systems such as Microsoft CoPilot hint in that direction, but further research is needed (on open-sourced model architectures) to measure and definitely estimate the extent.

VI. CONCLUSION

We have demonstrated a viable generative approach to constructing adversarial examples for the MNIST dataset in a sample efficient manner using a U-net based GAN. These examples lead to catastrophic failure in a pre-trained MNIST classifier indicating successful attacks. The adversarial examples generated by the U-Net based GAN is able to fool the classifier to a much larger extent than images with Gaussian noise added, while maintaining a higher structural similarity to the original images. The results show that the U-Net based GAN as a generative approach to constructing adversarial examples is a promising alternative to traditional non-generative methods, and that the adversarial examples can be used as a data augmentation strategy to bolster classifier robustness.

VII. ACKNOWLEDGEMENTS

The authors would like to thank Yu-Wei Chang for his guidance and support throughout the project.

VIII. CONTRIBUTIONS

Andreas, Emily, Filip and Lisa conducted literature review, theorised, developed and ran experiments on the U-Net based GAN, as well as produced relevant results from these experiments. Erik developed comparisons with gradient-based methods and conducted literature review. All authors participated fully in writing the poster and this report.

[1] J. C. Costa, T. Roxo, H. Proença, and P. R. M. Inácio, How deep learning sees the world: A survey on adversarial attacks amp; defenses, IEEE Access **12**, 61113–61136 (2024).

[2] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, Intriguing properties of neural networks (2014), arXiv:1312.6199 [cs.CV].

[3] T. Gu, B. Dolan-Gavitt, and S. Garg, Badnets: Identifying vulnerabilities in the machine learning model supply chain (2019), arXiv:1708.06733 [cs.CR].

[4] O. Ronneberger, P. Fischer, and T. Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, arXiv e-prints , arXiv:1505.04597 (2015), arXiv:1505.04597 [cs.CV].

[5] S. Qiu, Q. Liu, S. Zhou, and C. Wu, Review of artificial intelligence adversarial attack and defense technologies, Applied Sciences **9**, 10.3390/app9050909 (2019).

[6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial networks (2014), arXiv:1406.2661 [stat.ML].

[7] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, and F.-Y. Wang, Generative adversarial networks: introduction and outlook, IEEE/CAA Journal of Automatica Sinica **4**, 588 (2017).

[8] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, Generative adversarial networks: An overview, IEEE Signal Processing Magazine **35**, 53 (2018).

[9] H. Miao, Z. Zhao, C. Sun, B. Li, and R. Yan, A u-net-based approach for tool wear area detection and identification, IEEE Transactions on Instrumentation and Measurement **70**, 1 (2021).

[10] W. Falcon and The PyTorch Lightning team, PyTorch Lightning (2019).

[11] S. Wang, X. Wang, P. Zhao, W. Wen, D. Kaeli, P. Chin, and X. Lin, Defensive dropout for hardening deep neural networks under adversarial attacks, in *Proceedings of the International Conference on Computer-Aided Design*, ICCAD '18 (ACM, 2018).