Neeko: Model Hijacking Attacks Against Generative Adversarial Networks

1st Junjie Chu
CISPA Helmholtz Center for Information Security
junjie.chu@cispa.de

2nd Yugeng Liu
CISPA Helmholtz Center for Information Security
yugeng.liu@cispa.de

3rd Xinlei He
The Hong Kong University of Science and Technology (Guangzhou)
xinleihe@hkust-gz.edu.cn

4th Michael Backes
CISPA Helmholtz Center for Information Security
director@cispa.de

5th Yang Zhang
CISPA Helmholtz Center for Information Security
zhang@cispa.de

6th Ahmed Salem
Microsoft Security Response Center (MSRC)
ahmsalem@microsoft.com

***Abstract*—Generative models have garnered significant interest in the realm of machine learning but are costly to produce and face growing regulatory constraints, requiring resource-heavy training and collaboration with various stakeholders, especially data providers. Such collaborative environments have given rise to a new threat known as model hijacking attacks. Adversaries can tamper with the training process to embed a hidden task, so that train/hijack high-end models at minimal costs or even sidestep regulations. In this paper, we extend the scope of model hijacking from classifiers to generative models by introducing the first model hijacking attack tailored for Generative Adversarial Networks (GANs), namely *Neeko*. Neeko is based on a novel U-Net-based Disguiser and allows a compromised GAN to generate authentic-looking images from its original distribution, but when downscaled, these images are visually changed to be from the hijacking dataset distribution. Through experiments on different image benchmark datasets, we demonstrate the efficacy and stealthiness of Neeko. Neeko poses security and accountability risks associated with training public GANs on potentially malicious or illegal datasets and raises concerns about evading those regulations addressing deepfakes and synthetic images.**

***Index Terms*—GANs, model hijacking, data poisoning**

I. Introduction

While GANs hold immense promise, their training process presents inherent vulnerabilities. The acquisition of vast amounts of data and substantial computational power offers adversaries an attack surface to exploit. A novel attack, known as the model hijacking attack [1], [2], capitalizes on this vulnerability. By subtly poisoning the training dataset, adversaries can implement an additional, often malicious, hijacking task in the target model. However, the previous model hijacking attacks only work on classifiers instead of generative models. In this paper, we fill this gap by introducing the first model hijacking attack against GANs, namely Neeko.

Partially funded by the European Health and Digital Executive Agency (HADEA) within the project "Understanding the individual host response against Hepatitis D Virus to develop a personalized approach for the management of hepatitis D" (DSolve, grant agreement number 101057917) and the BMBF with the project "Repräsentative, synthetische Gesundheitsdaten mit starken Privatsphärengarantien" (PriSyn, 16KISAO29K).

Neeko is a training time attack and adopts the same threat model as data poisoning attacks and previous model hijacking attacks [1]–[4]. Concretely, the adversary conducts data poisoning to repurpose a target GAN designed for a hijackee image generation task (original task) to be able to complete a hijacking image generation task (the adversary's task). The hijacked GAN attacked by Neeko is capable of seamlessly generating synthetic images from both the original and hijacking datasets' distributions. To maintain the covert nature of data poisoning during the training phase, Neeko needs to preserve the utility of the target GAN in its original image generation task, while ensuring that camouflaged training samples (used to poison the training set) are visually highly similar to clean training samples.

Motivation: The adversary can hijack a target model to perform an unintended image generation task by using Neeko, without the model's owner noticing. This poses accountability risks for the model owner, as it could lead to allegations that their model is providing illegal or unethical services. For example, an adversary could hijack a benign GAN, initially designed for generating facial images, to produce synthetic pornography pictures. In short, an adversary can hijack a publicly available GAN (especially those high-quality and expensive GANs), leading to the GAN providing illegal or unethical services, unintentionally implicating the hijacked GAN's owner. Neeko also poses the risk of parasitic computing. The adversary could exploit a publicly accessible GAN for their own applications, bypassing the need to train and host their own GANs, thus saving on the associated costs.

Methodology: The objective of the Neeko is two-fold: Firstly, to allow a GAN to execute the adversary's hijacking agenda seamlessly and stealthily, and secondly, to ensure the GAN retains its initial functionality. We initiate Neeko using an image scaling method via quadratic programming (QP), embedding smaller "hijacking" images within larger "original" ones. The camouflaged images, visually similar to the original, reveal the hijacking images upon downscaling. This technique is used to create a camouflaged dataset for training the target

GAN, allowing it to learn the original task explicitly and learn hijacking tasks implicitly at the same time. The compromised GAN then could produce images from both the original and the camouflaged dataset's distributions. The QP-based Neeko is effective but very time-consuming, so we additionally introduce the Disguiser, a U-Net-based model inspired by diffusion model advancements. This model can implement functions similar to QP but is much more efficient.

Evaluation: We evaluate Neeko using various datasets: for original tasks, we use medical (`NIH Chest X-ray 14`), human facial (`CelebA`, `FFHQ`); and for hijacking tasks, we use cartoon avatars (`Konachan`, `Anime`) and numerical digits (`MNIST`). The evaluation focuses on the attack's stealthiness (similarity between the camouflaged and original images), efficiency (time consumption), and the hijacked GAN's utility on both tasks. The results of extensive experiments demonstrate the good performance of Neeko across these criteria. For instance, using `MNIST` to hijack a StyleGAN v3 trained on `CelebA`, both QP and Disguiser methods produce highly stealthy camouflaged training images, as reflected by high PSNR values (> 25 dB). The Disguiser significantly cut attack time by over 90% compared to QP. For the synthetic images, Neeko achieves the FID score of 4.99 which is only a minor increase of 1.36 compared to training a clean GAN, showing that the hijacked GAN maintains high utility. Defense testing reveals Neeko's covert nature, posing detection challenges for even knowledgeable defenders, highlighting its stealth and difficulty to detect.

In summary, we make the following contributions:

- We extend model hijacking to generative models by proposing the first model hijacking attack against GANs.
- We propose a new approach to implement model hijacking attacks, specifically through efficient Disguiser-based image scaling attacks.
- Our comprehensive empirical experiments reveal the efficacy of our proposed attack methods.

II. Preliminaries

A. Image Scaling Attack

Image scaling attacks [5], [6] are training time attacks that target the preprocessing phase of machine learning workflows. These attacks alter input images by quadratic programming (QP) so that when the images undergo the preprocessing steps, such as downscaling, their appearance is drastically changed. This enables adversaries to discreetly embed malicious images within seemingly benign target ones.

B. Model Hijacking Attack

Model hijacking attacks [1], [2] are another training time attack wherein adversaries manipulate a target model's training dataset (original dataset) to embed an auxiliary, and possibly malicious, task. The adversary has a **hijackee dataset** which follows a similar distribution to the original dataset. The original task of the target model is termed the **hijackee task**, while the adversary's added dataset/task is termed the **hijacking dataset/task**. To carry out the model hijacking attack covertly, first, the hijacking dataset is camouflaged. This camouflaged dataset visually mirrors the hijackee dataset yet incorporates the features of the hijacking dataset. The camouflaged dataset is then used to poison the target model's training, allowing it to perform its original and hijacking tasks simultaneously. We summarize the related terms in Table I in the appendix.

However, the techniques in previous works [1], [2] do not work in the generative models. For image generative models, the generated synthetic images must be recognizable to human beings. The technique in [1] targets the image classifiers and can only conceal certain features of the hijacking samples in the latent space. Such features are recognizable for classifier models but **not** recognizable for human vision. Another technique in [2] focuses on the natural language processing (NLP) domain classification tasks, making their methodologies unsuitable for direct application in the field of GANs.

In this work, we broaden the applicable scope of model hijacking to generative models, specifically GANs. Given a hijackee image, human vision can only recognize its pixels and is unable to discern its features in the latent space. Therefore, the key to our approach lies in manipulating the pixels of the hijackee image rather than the features in its latent space. We achieve this by proposing Neeko, which uses image scaling attacks to manipulate the pixels instead of the features in latent space. We also explore different approaches to enhance the efficiency of Neeko, including the U-Net-based Disguiser.

C. Threat Model

Our approach adopts a very classical threat model, **which is the same as the previous data poisoning attacks and model hijacking attacks** [1]–[4], assuming no prior knowledge about the target GAN and only the ability to poison its training set. We assume the presence of a hijackee dataset, which follows a similar distribution to the GAN's training dataset. The hijacking dataset is embedded in the hijakee one. After successfully hijacking a GAN using Neeko, the adversary can easily obtain the hijacking synthetic images by down-sampling the camouflaged synthetic images with corresponding scaling methods. Note that, not all synthetic images can be downscaled to resemble the hijacking ones, as some will appear as downscaled versions of the original images.

III. Methodology

A. General Attack Pipeline

To hijack the target GAN, the adversary first selects a *hijacking dataset* to implement in the target GAN. They also obtain a *hijackee dataset* to embed the hijacking dataset, enhancing the stealthiness of the Neeko. The hijackee dataset should obey the similar distribution of the original training dataset of the target GAN, or even be part of the original dataset. The hijacking dataset is then camouflaged within the hijackee dataset, creating a *camouflaged dataset*. This camouflaged dataset is then concatenated with the clean original dataset to create a poisoned dataset. Once the training of the target GAN is completed, the target GAN is hijacked.

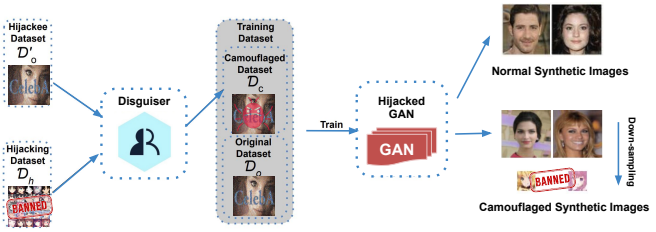


Fig. 1: Overview of Neeko using Disguiser. Neeko can stealthily poison the training dataset of a target GAN using banned images from an adversary, thereby causing the GAN to generate banned synthetic images.

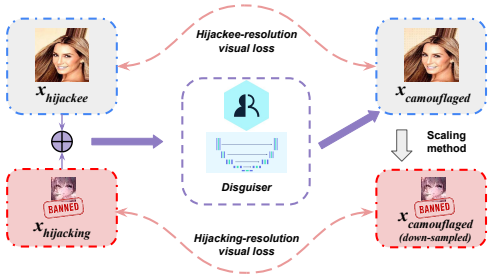


Fig. 2: Overview of Disguiser training.

The hijacked GAN could generate two kinds of synthetic images, including clean synthetic images and camouflaged synthetic images. The adversary can obtain hijacking synthetic images by downsizing the camouflaged synthetic images generated by the hijacked GAN.

B. Neeko Using Quadratic Programming (QP)

Directly blending the hijacking images with hijackee images is not covert and could lead to the mode collapse or failed training of GAN. To overcome the limitations, we employ the typical QP-based image scaling attack [5], [6] to embed the hijacking images into the hijackee dataset.

The adversary seeks a minimal perturbation Δ of the hijackee sample S, such that the downscaling $\mathcal{S}(\cdot)$ of the camouflaged sample $C = \Delta + S$ produces an output similar to the hijacking sample T. Goals are summarized as the following optimization:

$$min(\|\Delta\|_2^2),\ s.t. \|\mathcal{S}(S+\Delta) - T\|_\infty \ \leq \varepsilon$$

Additionally, each pixel value of C needs to remain within the fixed range (e.g., $[0, 255]$ for 8-bit images). All variables are known except for Δ and ε. Δ represents the desired output, while ε is the predetermined threshold. This problem can be solved with quadratic programming [6]. When successful, the adversary can obtain a camouflaged image C that bears a resemblance to the hijackee sample but matches the hijacking sample after downscaling operations.

While this methodology enhances the stealthiness of the adversarial attack, it requires lots of computational time. For example, the adversary needs to spend about 4,985 minutes processing 36,000 hijacking images as shown in Figure 3a. This long processing time constitutes a significant hindrance for potential adversaries, particularly those operating under time-sensitive conditions.

C. Neeko Using Disguiser

To improve the efficiency of Neeko, we propose a novel U-Net-based model, namely Disguiser, to replace QP. Inspired by recent advancements in diffusion models [7]–[9], the Disguiser utilizes a U-Net architecture to efficiently camouflage the hijacking dataset using the hijackee dataset.

Overview: We present the overview of Neeko using Disguiser in Figure 1. In the initial phase, the adversary focuses on training a Disguiser. This Disguiser takes images from both the hijackee dataset $\mathcal{D}_o^{'}$ and hijacking dataset $\mathcal{D}_h$ as two inputs. It is designed to subtly modify the hijackee images by embedding pixels from the smaller hijacking images into them. The output is a camouflaged image that, when viewed at high resolution, remains visually indistinguishable from the original hijackee image. However, when the camouflaged image undergoes down-sampling, it has a visual similarity to the hijacking sample. After producing a sufficient number of camouflaged images with the Disguiser, the adversary compiles these into a camouflaged dataset, denoted as $\mathcal{D}c$. This dataset is then merged with the original dataset, $\mathcal{D}o$, to craft a poisoned training set to poison the training of the target GAN. Once the GAN is trained on this poisoned dataset, it becomes successfully hijacked.

Hijacked GANs exhibit the ability to generate two unique kinds of synthetic images. The first kind consists of clean synthetic images that closely mirror the original and hijackee datasets distributions. The second kind involves camouflaged synthetic images. Although they closely match the hijackee dataset in high resolution, their visual appearance changes to align more with the hijacking dataset distribution when downscaled.

Disguiser ($\mathcal{M}_d$): The Disguiser is a U-Net-based model designed to integrate images from the hijacking dataset and the hijackee dataset. It takes two images as input: one from the hijacking dataset ($x_{hijacking} \sim \mathcal{D}_{hijacking}$) and the other from the hijackee dataset ($x_{hijackee} \sim \mathcal{D}_{hijackee}$). These images are concatenated, resulting in a single input with six channels. To handle differing dimensions, an upscaling function $\mathcal{F}_{up}(\cdot)$ is used to match the size of the hijacking dataset. The concatenated image is then passed through the Disguiser, which scales it down to a camouflaged image $x_{camouflaged}$ with three channels. The goal is for $x_{camouflaged}$ to visually resemble the hijackee image while exhibiting similarity to the hijacking image when downscaling.

The U-Net architecture's interpolation capabilities make it more effective in reproducing intricate details while incorporating relevant information from both datasets, compared to a simple encoder-decoder model used in previous works [1].

Hijacking-resolution Visual Loss ($\mathcal{L}_{hijacking}$): Similarly, to ensure visual similarity between the downscaled output of the Disguiser and the hijacking sample, we utilize the hijacking-resolution visual loss. This loss is calculated by measuring the

$L1$ distance between the hijacking samples and the downscaled version of the Disguiser's output ($\mathcal{S}(x_{camouflaged})$):

$$\mathcal{L}_{hijacking} = \|\mathcal{S}[x_{camouflaged}] - x_{hijacking}\|_1$$

Hijackee-resolution Visual Loss ($\mathcal{L}_{hijackee}$): To ensure visual resemblance between the output of the Disguiser ($x_{camouflaged}$) and the hijackee sample, we use the hijackee-resolution visual loss. This loss is also computed using the $L1$ distance metric:

$$\mathcal{L}_{hijackee} = \|x_{camouflaged} - x_{hijackee}\|_1$$

Disguiser Training: The training process is shown in Figure 2. To train the Disguiser, we use a weighted combination of two visual loss components: hijacking-resolution and hijackee-resolution visual losses.

$$\mathcal{L}_s = \lambda\mathcal{L}_{hijackee} + (1-\lambda)\mathcal{L}_{hijacking}$$

Here, λ controls the weight assigned to each loss term, determining their relative importance in the overall optimization objective. During training, in each epoch, random pairs of samples are created by pairing images from the hijacking dataset with those from the hijackee dataset. This random pairing strategy ensures the generalization of the Disguiser. To recap, prior to pairing, the hijacking samples are upsampled to match the dimensions of the hijackee samples. The upsampled hijacking sample is then concatenated with the hijackee sample, forming the input for the Disguiser. The output of the Disguiser, along with its corresponding hijackee and hijacking samples, are used to compute the $\mathcal{L}_s$. Note that the loss of the Disguiser is independent of the target model, making it applicable to various settings.

Neeko Execution: After training the Disguiser, the adversary uses it to create a camouflaged dataset by camouflaging the hijacking dataset with the hijackee dataset. This camouflaged dataset is then used to poison the training dataset of the target GAN, resulting in a hijacked GAN. The adversary can query the hijacked GAN to obtain synthetic images resembling samples from the hijackee dataset or the original datasets.

To obtain hijacking fake images, the adversary downscales the generated images. However, since the GAN is trained on both clean and camouflaged datasets, the generated fake images consist of a mixture of clean fake images and clean hijacked data. As a result, not all downscaled fake images align with the hijacking dataset distribution.

IV. Evaluation

A. Evaluation Settings

Datasets: The hijackee datasets and original datasets used in experiments are derived from diverse benchmark datasets, including `CelebA` [10], `FFHQ` [11], and `NIH Chest X-ray 14` [12], for face and medical images. For the hijacking datasets, we use `MNIST` [13] for handwritten digits, `Konachan Avatar` [14], and `Anime Faces` [15] for anime faces. The training sizes are 70k for `FFHQ` and 120k for `CelebA` and `NIH Chest X-ray 14`. We set the scaling factor $\alpha = 8.0$ from image scaling attacks as recommended [5], [6], which means the length and width of the hijacking image are both 1/8 of the original. The resolutions of hijacking datasets are 16^2 and 32^2 while those of original/hijackee datasets are 128^2 and 256^2. Under this setting, the corresponding artifacts in the camouflaged images are considered to be invisible. The poisoning rate is 30%.

Models: We use the cutting-edge StyleGAN v3 as our target GAN model for the main experiments. GANs are trained from scratch with the basic configurations. In the main experiment, the Disguiser architecture is based on U-Net with `ResNet18` encoder and decoder layers, taking inputs with 6 input channels and producing output images with 3 channels. We also add a 4-layer autoencoder-based Disguiser, which has the same architecture as the camouflager [1] but different loss functions, to compare. We train the Disguiser on a randomly sampled set of $5k$ hijacking-hijackee image pairs, following Section III-C. The weight λ is set to 0.5.

Metrics: We apply the peak signal-to-noise ratio (PSNR) to measure the similarity between camouflaged and original samples, with PSNR values above 25 dB indicating strong similarity, as suggested by previous research [6]. We calculate the mean and standard deviation of PSNR across the entire training dataset to assess Neeko's stealthiness by evaluating artifact visibility in camouflaged samples. For attack efficiency, we measure the time to generate the camouflaged dataset.

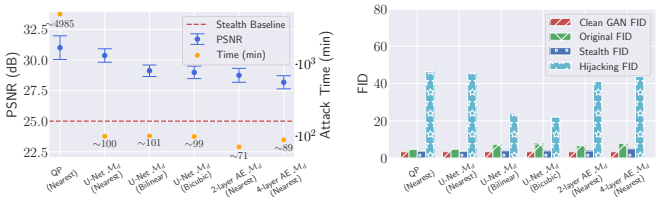
We use the Fréchet Inception Distance (FID) metric for evaluating hijacked GANs' performance [16], despite its known sensitivity to scaling [17], due to the absence of a better alternative. To distinguish between clean and camouflaged fake images produced by the hijacked GANs, we use a classifier with a slightly adjusted score threshold of 0.60 for greater accuracy, reducing the impact of misclassification. We compute three types of FID scores for hijacked GANs: original FID (clean training images vs. synthetic images), stealth FID (camouflaged and clean training images vs. synthetic images), and hijacking FID (down-sampled camouflaged training images vs. down-sampled camouflaged synthetic images). These metrics provide insights into the utility, stealthiness, and efficacy of the hijacked GANs during their training phase.

Other Settings: We use scaling methods in `PyTorch` and apply the nearest neighbor method in main experiments. Compute resources are detailed in Table II in the appendix.

B. Evaluation Results

The attack time and PSNR of camouflaged samples are shown in Figure 3a. The target GAN is trained on the poisoned dataset, with FID metrics reported in Figure 3b, and visual examples in Figure 4.

Experimental results confirm high stealthiness for both QP-based and Disguiser-based Neeko. For example, hijacking `CelebA` 256^2 with `Konachan` 32^2 using QP achieves PSNRs above 25 dB (Figure 3a) and a stealth FID of 3.79 (Figure 3b). The clean fake image FID increases by only 1.30, preserving high-quality camouflaged images. Similar trends are observed with Disguiser-based Neeko. While the hijacking FID is higher due to previously discussed factors, qualitative



(a) PSNRs and attack time. (b) FIDs of clean and hijacked GANs.

Fig. 3: Quantitative results of Neeko with different scaling and attack methods: the hijacking dataset is `Konachan` 32^2 and the hijackee dataset is `CelebA` 256^2 (size: 120k, 30% camouflaged). Methods outside the bracket are attack methods, and those inside are the corresponding scaling methods.

Fig. 4: Samples of camouflaged synthetic images output by hijacked GANs with different attack methods and scaling methods: the hijacking dataset and hijackee dataset are `Konachan` 32^2 and `CelebA` 256^2, respectively; the methods outside and inside the bracket are the attack methods and the corresponding scaling methods, respectively.

results (Figure 4) indicate good image quality for the hijacking task. More samples are in Figure 8 of the appendix.

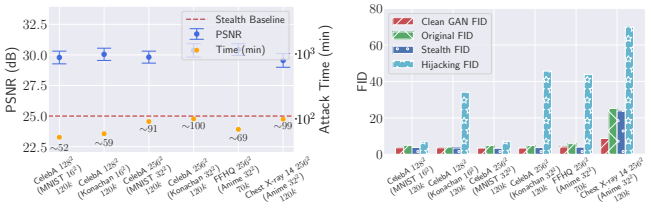
Our results also show that the novel Disguiser-based attack is much more efficient than the QP-based. Detailedly, training a U-Net-based Disguiser usually requires less than 300 minutes, and camouflaging 36k images usually takes less than 110 minutes. Compared with the QP-based attack (approximately 4,985 minutes), Disguiser could save over 90% less time to generate camouflaged samples.

C. Hyperparameters

We omit image scaling attack hyperparameters, which are extensively discussed in prior works [5], [6], and focus only on those relevant to Neeko.

Dataset Pairs: We test Neeko using Disguiser (U-Net-based) on various dataset pairs, with PSNRs and FIDs detailed in Figure 5a and Figure 5b. Lower bounds of PSNR always exceed 25 dB, showing good stealthiness during training. Hijacked GAN performance varies by dataset pair. Minimal original FID increases ($\leq$ 1.79) occur with simple hijacking tasks (e.g., `MNIST`) or similar datasets (e.g., `Konachan` hijacking `CelebA`). For complex, dissimilar tasks, metrics degrade, with original and stealth FIDs exceeding 20 and hijacking FID reaching 69.93 (e.g., `NIH Chest X-ray 14`).

Disguiser Architectures: We compare the U-Net with the autoencoder (AE) from [1]. During training, the PSNR drops by 1.62 dB for the two-layer AE and 2.19 dB for the four-layer AE compared to the U-Net (Figure 3a). The two-layer



(a) PSNRs and attack time. (b) FIDs of clean and hijacked GANs.

Fig. 5: Quantitative results of Neeko using Disguiser on different hijackee and hijacking dataset pairs are reported. The U-Net-based Disguiser and the Nearest scaling method are used. The datasets in and outside the brackets represent the hijackee and hijacking datasets, respectively.

AE preserves more details but adds noise, while the four-layer AE reduces noise but loses detail. The U-Net balances detail preservation and low noise, ensuring better stealth and equivalent attack efficiency. In inference, the two-layer AE improves the hijacking FID by 4.44 but increases the original FID by 1.87. The four-layer AE shows no FID advantage. Overall, the U-Net introduces smaller perturbations, making the attack more covert than AE-based methods.

Scaling Methods: We evaluate the impact of scaling methods (Nearest, Bilinear, Bicubic) using a U-Net-based Disguiser to hijack a `CelebA` GAN with the `Konachan` dataset. Full results are in Figure 3 and Figure 4.

Bilinear and Bicubic scaling reduce the PSNR lower bound by 1.17 dB and 1.34 dB, respectively, compared to Nearest. They also lower hijacking FID values by 21.88 and 23.61 but increase the original FID by 2.65 and 2.79, reducing clean fake image quality. Scaling methods with larger perturbations (lower PSNR) enhance hijacking but harm GAN utility.

Target Models: We first evaluate Neeko on simpler GANs, DCGAN [18] and WGAN [19], using `MNIST` 16^2 to hijack `CelebA` 256^2 GANs. DCGAN shows FIDs of 91.03 (original), 103.22 (stealth), and 47.51 (hijacking); WGAN shows 69.33, 72.76, and 38.86, respectively. Note that these large FID values are due to the weak performance of the GANs compared to the state-of-the-art ones, e.g., a clean DCGAN still results in 87.63. Despite their simplicity, Neeko succeeds. We then apply Neeko to another advanced GAN, StyleGAN v2 [20], achieving good results. Using `MNIST` to hijack `CelebA`, the FID scores are: original 3.98, stealth 3.92, and hijacking 6.01. With the `Konachan` dataset, the FIDs are: hijackee 4.39, stealth 3.95, and hijacking 42.61. Visual samples are shown in Figure 7.

D. Possible Defenses

Defenders are typically unaware of the adversary's poisoning methods, and camouflaged samples resemble clean ones, thus relying on unsupervised detection methods. We test three popular unsupervised clustering algorithms (DBSCAN [21], Agglomerative Clustering [22], and K-Means [23]) on a test set of 200 `CelebA` images, 30% of which are camouflaged. Experimental results show that DBSCAN fails to detect cam-

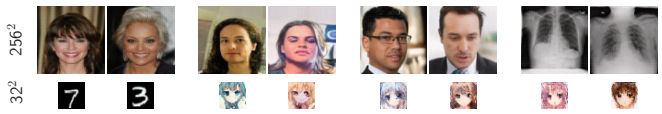


Fig. 6: Samples of camouflaged synthetic images generated by hijacked GANs on different hijackee and hijacking dataset pairs: U-Net-based Disguiser and Nearest scaling method are used; the hijackee dataset is indicated outside the brackets, while the corresponding hijacking dataset is indicated inside.

ouflaged images, grouping all 200 images together. Agglomerative Clustering divides images into two clusters, but one cluster has 30 out of 61 camouflaged images, and the other has 70 out of 139. K-Means produces similar results. We believe the small perturbation from Neeko makes unsupervised methods remain clustering based on other features, like hair color in `CelebA`, rather than those manipulated pixels.

Another possible defense against Neeko is applying denoising techniques to remove hijacking signals. We evaluate the Non-local Means Filter, a highly effective blind denoising method. Figure 9 in the appendix shows the impact of this denoising on hijacked GAN images. The results indicate reduced adversarial effects but also a significant loss of image detail, as reflected by an FID of 43.03 for the denoised images. This highlights the trade-off between defending against the attack and maintaining image quality.

V. Conclusion

We present a novel model hijacking attack against GANs, namely Neeko. **This attack shares the same threat model as the data poisoning attack.** Neeko achieves the hijacking generation task by merely manipulating the samples in the training dataset, without altering the GAN's inherent architecture or loss function. We explore various techniques for the effective and covert execution of Neeko, including QP and Disguiser. Notably, the U-Net-based Disguiser we propose acts as a particularly efficacious methodology in terms of both performance and efficiency. Comprehensive experimentation demonstrates the capacity of Neeko to compromise GANs. Our work highlights critical ethical and safety concerns, particularly vulnerabilities in dataset governance. Adversaries could exploit these flaws to subvert public GANs using illegal or copyrighted data, posing risks of intellectual property violations. Additionally, we reveal the risks of parasitic computing, where adversaries exploit GAN training processes to offload their computational costs, potentially draining resources from benign users.

References

[1] Ahmed Salem, Michael Backes, and Yang Zhang, "Get a Model! Model Hijacking Attack Against Machine Learning Models," in *Network and Distributed System Security Symposium (NDSS)*. 2022, Internet Society.

[2] Wai Man Si, Michael Backes, Yang Zhang, and Ahmed Salem, "Two-in-One: A Model Hijacking Attack Against Text Generation Models," *CoRR abs/2305.07406*, 2023.

[3] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li, "Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning," in *IEEE Symposium on Security and Privacy (S&P)*. 2018, pp. 19–35, IEEE.

[4] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein, "Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*. 2018, pp. 6103–6113, NeurIPS.

[5] Qixue Xiao, Yufei Chen, Chao Shen, Yu Chen, and Kang Li, "Seeing is Not Believing: Camouflage Attacks on Image Scaling Algorithms," in *USENIX Security Symposium (USENIX Security)*. 2019, pp. 443–460, USENIX.

[6] Erwin Quiring, David Klein, Daniel Arp, Martin Johns, and Konrad Rieck, "Adversarial preprocessing: Understanding and preventing image-scaling attacks in machine learning," in *USENIX Security Symposium (USENIX Security)*. 2020, pp. 1363–1380, USENIX.

[7] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli, "Deep Unsupervised Learning using Nonequilibrium Thermodynamics," in *International Conference on Machine Learning (ICML)*. 2015, pp. 2256–2265, PMLR.

[8] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 10684–10695, IEEE.

[9] Prafulla Dhariwal and Alexander Quinn Nichol, "Diffusion Models Beat GANs on Image Synthesis," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*. 2021, pp. 8780–8794, NeurIPS.

[10] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang, "Deep Learning Face Attributes in the Wild," in *IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 3730–3738, IEEE.

[11] Tero Karras, Samuli Laine, and Timo Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 4401–4410, IEEE.

[12] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M. Summers, "Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2097–2106, IEEE.

[13] "MNIST," http://yann.lecun.com/exdb/mnist/.

[14] "Konachan," https://aistudio.baidu.com/aistudio/datasetdetail/110820/0.

[15] "Anime," https://github.com/bchao1/Anime-Face-Dataset.

[16] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter, "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium," in *Annual Conference on Neural Information Processing Systems (NIPS)*. 2017, pp. 6626–6637, NIPS.

[17] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu, "On aliased resizing and surprising subtleties in GAN evaluation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 11400–11410, IEEE.

[18] Alec Radford, Luke Metz, and Soumith Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," in *International Conference on Learning Representations (ICLR)*, 2016.

[19] Martin Arjovsky, Soumith Chintala, and Léon Bottou, "Wasserstein Generative Adversarial Networks," in *International Conference on Machine Learning (ICML)*. 2017, pp. 214–223, PMLR.

[20] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila, "Analyzing and Improving the Image Quality of StyleGAN," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 8107–8116, IEEE.

[21] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *International Conference on Knowledge Discovery and Data Mining (KDD)*. 1996, pp. 226–231, AAAI.

[22] Daniel Müllner, "Modern hierarchical, agglomerative clustering algorithms," *CoRR abs/1109.2378*, 2011.

[23] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu, "An Efficient k-Means Clustering Algorithm: Analysis and Implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

APPENDIX

A. Limitations and Future Work

Neeko shows promising prospects but also has limitations. First, the hijacking task must be smaller than the original, as larger hijacking images compromise the concealment of Neeko. Second, generative tasks require more training data than classification tasks, leading to a higher data poisoning rate for effective hijacking. We set the data poisoning rate to 0.30, compared to 0.17 for classifier attacks [1]. Attackers have to balance hijacking performance and poisoning rate. Finally, the FID metric may not fully reflect qualitative success, suggesting the need for alternative metrics in future research.

B. Description of Terms

Here we detailedly describe the terms used in Neeko.

TABLE I: Description of different terms used in Neeko.

Term	Definition
Neeko	Neeko is a champion in League of Legends. She can blend into any crowd by borrowing the others' appearances, which match the function of our Disguiser. We borrow the champion's name "Neeko" to describe our proposed attacks.
Original Dataset	The training dataset of the target GAN's original task.
Hijackee Dataset	The dataset from the same distribution as the target GAN's training dataset.
Hijackee Samples	Samples from the hijackee dataset. Usually, they are at a big resolution.
Hijacking Dataset	The training dataset of the adversary's hijacking task.
Hijacking Samples	Samples from the hijacking dataset. Usually, they are at a small resolution.
Camouflaged Dataset	The modified hijacking dataset after being stealthily embedded in a hijackee dataset.
Camouflaged Samples	Samples from the camouflaged dataset. They look similar to the corresponding hijacking samples at the large resolution but look similar to hijacked samples at the small resolution.
Poisoned Dataset	The dataset the model will be trained on, i.e., the concatenation of the camouflaged and the original datasets.
Hijackee Resolution	The resolution of the hijackee dataset. Usually, it is larger than the hijacking resolution.
Hijacking Resolution	The resolution of the hijacking dataset.

C. Compute Resources

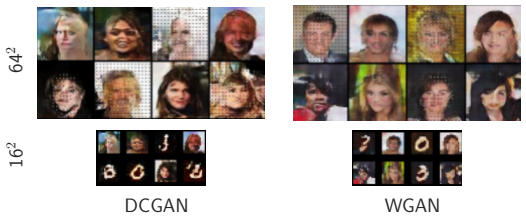
We train all the Disguiser models and poison the images on Server X10DRG-K80 with 1 `Tesla K80` GPU. Then we train all the StyleGANs on Server DGX-A100 with 2 `NVIDIA A100` GPUs. The servers' details are shown in Table II.

TABLE II: Compute resource details.

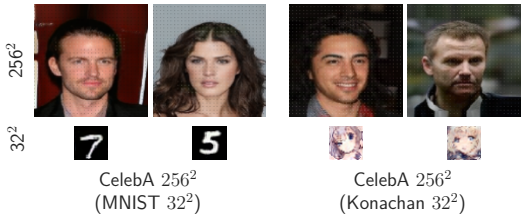
Server Name	Model	CPU	GPU	RAM
X10DRG-K80	Supermicro SYS-4028GR-TRT	Intel Xeon E5-2697	Tesla K80	512 GB
DGX-A100	NVIDIA DGX A100 (40G)	AMD Rome 7742	NVIDIA A100	1 TB

D. Additional Results of Experiments

In this section, we provide additional experiment results to support our conclusions.



(a) Visual samples of poisoned fake images output by the hijacked DCGAN and WGAN: the hijacking dataset and hijackee dataset are `MNIST` 16^2 and `CelebA` 64^2, respectively.



(b) Visual samples of poisoned fake images output by the hijacked StyleGAN v2: the datasets outside and inside the bracket are the hijackee dataset and the corresponding hijacking dataset, respectively.

Fig. 7: Visual results of different target GANs: U-Net-based Disguiser and Nearest scaling method are used; the first row shows the visual appearances of the poisoned fake images at the hijackee resolution and the second row shows those at the corresponding hijacking resolution; the resolution corresponding to each row is labeled on the left of the images.

(a) Quadratic programming and Nearest are used as the attack method and scaling method, respectively.

(b) U-Net-based Disguiser and Nearest are used as the attack method and scaling method, respectively.

(c) U-Net-based Disguiser and Bilinear are used as the attack method and scaling method, respectively.

(d) U-Net-based Disguiser and Bicubic are used as the attack method and scaling method, respectively.

Fig. 8: Visual samples of downsampled poisoned fake images: the hijacking dataset and hijackee dataset are `Konachan` 32^2 and `CelebA` 256^2, respectively.

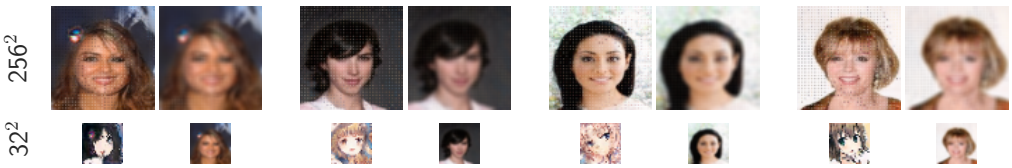


Fig. 9: Visual appearances of poisoned fake images and the corresponding de-noised versions: the hijacking dataset and hijackee dataset are `Konachan` 32^2 and `CelebA` 256^2, respectively; the first row shows the visual appearances of the poisoned fake images at the hijackee resolution and the second row shows those at the corresponding hijacking resolution; the resolution corresponding to each row is labeled on the left of the images.