MIRROR: Model Inversion for Deep Learning Network with High Fidelity

Shengwei An[†], Guanhong Tao[†], Qiuling Xu[†], Yingqi Liu[†],
Guangyu Shen[†], Yuan Yao[‡], Jingwei Xu[‡], Xiangyu Zhang[†]
[†]Department of Computer Science, Purdue University, USA
[‡]State Key Laboratory for Novel Software Technology, Nanjing University, China
Email: [†]{an93, taog, xu1230, liu1751, shen447, xyzhang}@cs.purdue.edu, [‡]{y.yao, jingweix}@nju.edu.cn

***Abstract*—Model inversion reverse-engineers input samples from a given model, and hence poses serious threats to information confidentiality. We propose a novel inversion technique based on StyleGAN, whose generator has a special architecture that forces the decomposition of an input to styles of various granularities such that the model can learn them separately in training. During sample generation, the generator transforms a latent value to parameters controlling these styles to compose a sample. In our inversion, given a target label of some subject model to invert (e.g., a private face based identity recognition model), our technique leverages a StyleGAN trained on public data from the same domain (e.g., a public human face dataset), uses the gradient descent or genetic search algorithm, together with distribution based clipping, to find a proper parameterization of the styles such that the generated sample is correctly classified to the target label (by the subject model) and recognized by humans. The results show that our inverted samples have high fidelity, substantially better than those by existing state-of-the-art techniques.**

I. Introduction

Given a deep learning (DL) model, model inversion aims to generate samples for a label of interest. It often assumes knowledge of the subject model's application domain, for example, face recognition, but no information of individual labels or no availability of training samples. *White-box* model inversion assumes the model internals, e.g., gradients, are accessible, whereas *black-box* inversion considers the subject model a black-box, e.g., when it is hosted online as a service. Model inversion may disclose critical information of the target model and labels, causing serious privacy leakage. For example, if the attacker successfully inverts a face recognition model used in access control, he could disguise himself as the inverted person, causing devastating security breach.

While model inversion for general machine learning models has a long history [23], [22], [61], [28], [73], [69], DL model inversion is of particular importance due to the rapidly growing applications of these models in security-critical tasks. Early model inversion work such as [22] directly leverages gradient back-propagation to mutate a random input such that the subject model classifies the mutated input to the target label. The leftmost set in Figure 1 shows two sample inversion results from the original paper [22] for a neural network without hidden layers. Many different methods have been proposed ever since. The two state-of-the-art white-box methods are DeepInversion [69] and GMI [73]. The former is based on the observation that many models have *batch normalization* (BN) layers that record the means and variances of internal feature maps (for the training dataset), which are leveraged to normalize activation values for better training/classification performance. DeepInversion enforces a mutated input to follow the recorded distribution in all the BN layers. This provides strong regulation during inversion. The rightmost set in Figure 1 shows two inverted samples from their official GitHub repository for a ResNet50v1.5 model trained on ImageNet [46]. Observe that the samples have many high-fidelity details of the target label. In contrast, GMI leverages a GAN to invert models. Instead of mutating input, it mutates the latent value that is fed to the generator of the GAN to generate the sample. Gradients are back-propagated from the subject model to the generated input (by the GAN) and further to the latent value. Since GAN can ensure the naturalness of generated samples, GMI produces good quality inversion results as demonstrated by the third set in Figure 1, which shows the inverted samples from the original paper. AMI [65] is the state-of-the-art black-box method. It leverages an auxiliary dataset to learn a generator that can generate samples based on prediction scores of the target model. The second set in Figure 1 shows two samples from the original paper. Observe that the inversion quality degrades a bit due to its black-box nature. More discussion of related work can be found in Section VI.

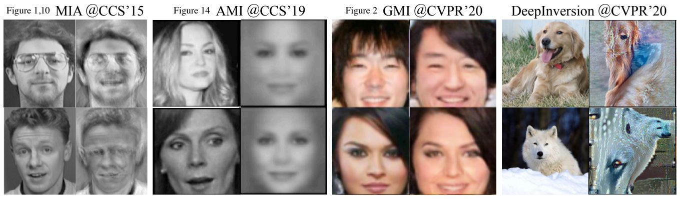


Fig. 1: Inversion results of existing methods from their original papers or the official GitHub repository. In each set, the left image is the target person and the right one is the inverted image.

While existing techniques have made substantial progress, there is still lots of room to improve. Specifically, from the samples in Figure 1, one can observe that while DeepInversion can have high fidelity features, these features are often out

Network and Distributed Systems Security (NDSS) Symposium 2022
27 February - 3 March 2022, San Diego, CA, USA
ISBN 1-891562-74-6
https://dx.doi.org/10.14722/ndss.2022.24335
www.ndss-symposium.org

of place. This is because although BN layers record statistics for individual feature maps, they do not provide enough constraints across features, e.g., their relative locations. In addition, some of the details may not be that realistic as it is not like GAN, which is trained to capture natural distributions of features in all aspects. GMI shows great potential. However, it only supports low-resolution inversion. According to our experience, when the resolution becomes higher, the latent space is too sparse such that direct optimization in the space becomes under-constrained. In addition, the latent space of regular GANs is known to have substantial entanglement [34], meaning features do not have locality. As such, it becomes difficult for the optimization to find a good minimal. In Section III, we will discuss in details the limitations of the state-of-the-art solutions.

We propose a new GAN based inversion technique. We observe there are a number of prominent challenges in using GANs for model inversion. We find the entanglement in the latent space makes it not an ideal space for optimization, especially when the space is sparse. A plausible idea of using the discriminator of GAN, which is supposed to distinguish generated samples and natural samples, to ensure naturalness of inversion results is problematic as the discriminator is closely coupled with the corresponding generator and cannot serve as a general classifier of naturalness. We hence propose to use a StyleGAN generator, which generates a realistic sample from a latent value controlling the parameterized re-composition of a set of nicely decomposed features/styles. The decomposition is enforced in training by the structure of StyleGAN (to enable re-composition in generation). During inversion, given a subject model (e.g., a private face based identity recognition system) and a StyleGAN pre-trained on public data from the domain of the subject model (e.g., public face dataset), gradient back-propagation or a genetic algorithm can be used to search for the best parameterization for the StyleGAN generator that can lead to a generated sample causing the subject model and humans to classify to the target label. To improve the image quality, we further propose to regulate the optimization/search in an auxiliary space in the StyleGAN that follows a multi-variate Gaussian (MVG) distribution. To improve fidelity (i.e., the inverted images should resemble the target label), we utilize random drop-out to mitigate feature overfitting that commonly exists in subject models.

Our contributions are summarized as follows.

- We study challenges in GAN-based model inversion.
- We propose to use StyleGAN for model inversion. Our inversion method features additional regulation in an auxiliary space and the support for both white-box and black-box settings. In the black-box setting, instead of using gradients, we use a genetic algorithm to search for minimals.
- We develop a novel random drop-out method to mitigate feature outfitting in the subject model that could degrade inversion fidelity.
- We explore a large design space of GAN-based and StyleGAN-based inversion and report our findings.
- We build a prototype MIRROR (Model InveRsion for deep leaRning NetwORk) in PyTorch. Our evaluation on 6 models and 3 widely used face datasets shows our approach can invert models with high fidelity, generating samples that closely resemble the target labels. Our user study shows that on average 88.4% users prefer our inversion results to those by existing techniques. When mixing the inverted image of a person with a set of images that closely resemble the person, 89.29% users can correctly select the inverted image from the others. Our quantitative analysis shows that the inverted samples have on average 15x better generalizability compared to existing state-of-the-art methods, meaning that the inverted samples have 15x times better classification accuracy (to the target label) by other models different from the subject models. We also test our black-box approach on two commercial services: the Azure face recognition service [1] and Clarifai [2]. Our approach achieves 100% effectiveness (i.e., the inverted samples are 100% classified to the target labels by the subject model) and 50%+ generalizability (50%+ by another model), while the state-of-the-art black-box method has close to 0 effectiveness and generalizability. Moreover, MIRROR outperforms existing methods in a number of image quality metrics. Besides face recognition models, we also show MIRROR can produce high quality inversion results on other datasets (e.g., cars and cats) and models. Our approach can successfully attack models trained with several existing defense methods including adversarial training against adversarial samples, information regularization against model inversion and differential privacy against membership inference. MIRROR will be open-sourced upon publication on our project page [7].

Threat Model. Our threat model is consistent with that in existing works [73], [65], [69]. We assume attackers know the application domain of the subject model and there is public data available for the domain. Note that the public data may not overlap at all with the private data used to train the subject model. The assumption is reasonable as real-world applications of DL models are likely rooted at data from the physical world. Attackers can hence sample from the physical world to support their StyleGAN training. In the white-box setting, we assume attackers have access to model internals. But they have no other information such as label specific information or specific training data. In the black-box setting, they do not have the access to model. The subject models are normally trained without specific hardening (*e.g.*, to prevent inversion). Although we demonstrate that MIRROR is resilient to existing hardening methods, how to better defend model inversion will be our future work.

II. Background: StyleGANs

StyleGAN was originally proposed in [34] and later improved in [35]. While generators in traditional GANs directly map latent vectors to samples, generators in StyleGANs first map an initial latent vector to an intermediate latent vector which is further transformed (via simple functions) to styles at different granularities. These styles shape an average sample denoted by a constant input to the convolutional layers (e.g., an average face) to a specific sample (e.g., a face with

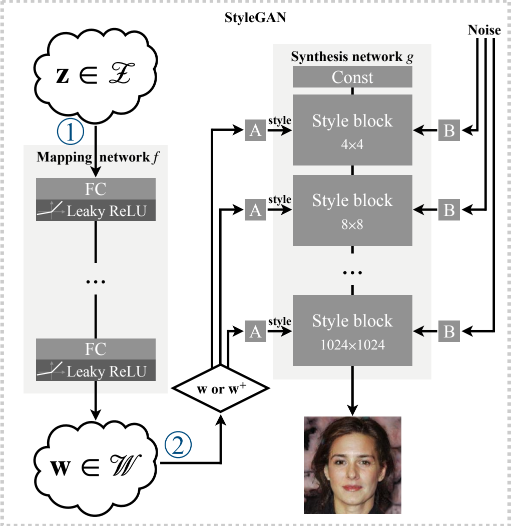


Fig. 2: Architecture of StyleGAN

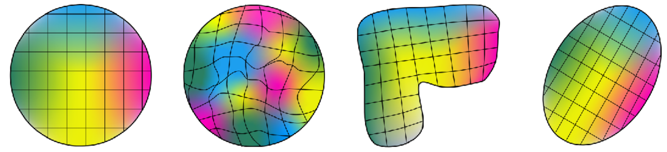

Distribution of features (a)
Distribution of $\mathscr{Z}$ (b)
Distribution of $\mathscr{W}$ (c)
Distribution of $\mathscr{P}$ (d)

Fig. 3: Real-world Feature Space in (a) and Different GAN/StyleGAN Spaces in (b)-(d). They have different levels of entanglement and different distributions.

personalized features). The intermediate latent space is known to be less entangled [34], [35].

Figure 2 shows the architecture of a StyleGAN generator. Its two main components are the mapping network f (on the left) and the synthesis network g (on the right). Network f consists of several fully connected layers with the leaky ReLU activation functions; g is composed of multiple style blocks with various resolution scales (*e.g.*, 4^2 to 1024^2). Each style block has two convolutional layers like that in a typical CNN. The difference is that the styles of the feature maps in each block are controlled (by the input from a square A), and the values are perturbed by random noises (from the input from a square B). In the literature, the means and variances of feature maps are believed to encode styles [30]. As such, style application is achieved by mutating the means and variances of the feature maps.

Specifically, in step ①, a latent vector $\mathbf{z}$ is sampled from the $\mathcal{Z}$ space (*e.g.*, Gaussian distribution) and provided to the mapping network f. Function $f(\mathbf{z})$ produces an intermediate latent vector $\mathbf{w}$ (all $\mathbf{w}$'s form the $\mathcal{W}$ space). In step ②, $\mathbf{w}$ is duplicated and sent to every style block in g to synthesize the sample. The square A at each style block denotes a linear layer that transforms $\mathbf{w}$ to styles and the square B adjusts the strength of random noises. Formally, the style-based generator returns an image $x = g(f(\mathbf{z}))$. During training, the parameters in f, g, A, and B are updated.

Figure 3 (a)-(c) illustrate the conceptual differences between traditional GAN and StyleGAN, by showing the distribution of samples that share similar features (e.g., faces sharing similar nose shapes) in different designs. Each color represents a feature. A pixel with a specific color denotes a sample with a specific feature. In (a), natural samples sharing similar features are close to each other. However in (b), i.e., the $\mathcal{Z}$ space in classic GANs, they are distributed in numerous small local regions. In other words, features are *entangled*. In StyleGAN, after the f transformation, the features are largely *disentangled* in the intermediate $\mathcal{W}$ space as shown in (c). The disentanglement is achieved by the special architecture of StyleGAN during training. In particular, the A blocks are linear functions whose outputs directly control the means and variances of feature maps. The separation of the style blocks with different resolutions structurally decomposes a sample to styles, which are essentially features of different levels of complexity. If the network converges in training, a $\mathbf{w}$ value must be disentangled in a way that its linear projections (through the A blocks) control parameterizations of the various features. Please refer to the original papers [34], [35] for more details.

III. Motivation

This section shows and analyzes the limitations of existing state-of-the-art inversion methods, and thus motivates our work. We will leverage the sample results in Figure 4 to provide an intuitive comparison of different methods and demonstrate that our method outperforms the existing ones. Each row in the figure shows the results of a person inverted by different methods. Each inverted image is annotated with the classification confidence to the target person by the model. Two real images of each target person are provided as the ground truth for comparison. For brevity, we use M and t to denote the model and the target label to invert, respectively. Given M and t, the goal of model inversion is to generate an image x such that humans or machines consider x an image of t.

A. Existing White-box Inversion Methods

Two current state-of-the-art white-box approaches are DeepInversion [69] and GMI [73].

DeepInversion. DeepInversion leverages a loss to constrain internal features in *batch normalization* (BN) layers. During training, the parameters of prior layers keep updating and therefore the input distributions of subsequent layers keep changing. BN layers were proposed to address this so-called "internal covariate shift" to accelerate training [31]. A BN layer maintains the running mean and variance observed over the batches of training data. It normalizes the output distribution of a prior layer to obtain an internal distribution with the mean of 0 and variance of 1 so that the following layer has a more stable input distribution. If we feed a batch of real input data to the network, the statistics of the internal features should be similar to those stored in the BN layers. These means and variances are used to regularize the feature distributions of inverted inputs such that they are close to the distributions of real inputs. Specifically, in addition to the cross-entropy (CE) loss which enforces the inverted images to be correctly classified as the target label, the internal constraints minimize the ℓ_2 distance between the variance (resp. mean) calculated by a batch of inputs to optimize and the running variance (resp.

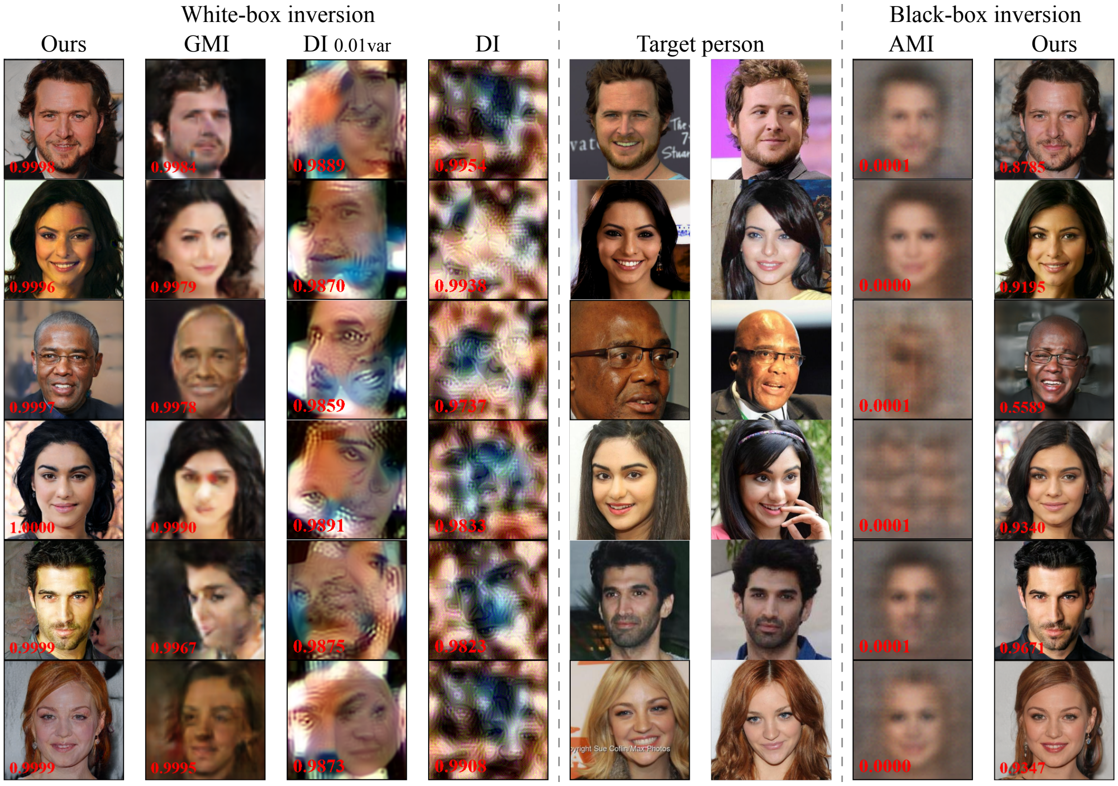


Fig. 4: Comparison with existing state-of-the-art methods under white-box and black-box settings. Each row corresponds to a person. The target people in first and second rows exist in the training data of inversion methods, while the others are not. The first to fourth columns show the white-box inversion results of our method, existing GMI and DeepInversion. The fifth and sixth columns show two real images for each target person. The seventh and eighth columns show the black-box inversion results of existing AMI and our method. The number in each inverted image denotes the classification confidence.

mean) stored in each BN layer. The goal of DeepInversion is to find $\arg\min_x Loss(M(x), t)$.

DeepInversion cannot be directly applied to models without BN layers (*e.g.*, VGG16 and SphereFace). Inverted images are usually not as natural as generator-based methods like GMI and MIRROR as shown by Figure 4. The fourth column shows the images inverted by DeepInversion using its default parameters. They look like random noises. After tuning the parameters, the best results we can get are achieved by multiplying 0.01 with the ℓ_2 distance of variances and the corresponding results are shown in the third column. Although some facial features can be inverted, there are also multiple overlapping faces and misplaced eyes. Such results are consistent with what DeepInversion originally generated for a ResNet50v1.5 pre-trained on ImageNet shown in Figure 1. This phenomenon is not beyond expectation. Models are usually trained to learn correlations between features and outputs, and may not learn the relative constraints across features (e.g., their relative positions). As such, the use of normalization statistics allows inverting individual *normal* features but does not sufficiently constrain their compositions. This is sufficient for the original goal of DeepInversion, which is data augmentation, but not for inverting human-recognizable faces. We have explored many ways to enhance DeepInversion, by adding more constraints to enforce various properties of inverted faces, such as exactly two eyes and one nose at the correct position of a face. However, these results (in our online Appendix XI [7]) are still not comparable to those based on generators.

GMI. GANs are able to generate in-distribution inputs. For instance, a GAN trained on a human face dataset is capable of generating natural human faces. Hence, an idea is to use a pre-trained GAN to regulate the inverted results, in addition to the subject model. A GAN contains a pair of a generative network G and a discriminative network D. G and D are trained in an alternative and competitive way. The goal of D is to distinguish the fake images generated by G from the real images. The goal of G is to generate images from a pre-defined latent space to fool D. After convergence, G usually is able to generate in-distribution random inputs (*e.g.*, random human faces). In contrast to DeepInversion which directly optimizes the input image, GMI optimizes the latent vector $\mathbf{z}$ to generate an image $G(\mathbf{z})$ so that M classifies $G(\mathbf{z})$ as label t and D considers it a real image. That is, the goal of GMI is to find $x = G(z)$ where $z = \arg\min_z Loss(M(G(z)), t)$. The latent space has fewer dimensions than the image space (*e.g.*, 100 vs. 3×64^2). Observe in Figure 4 that the faces inverted by GMI are more human-recognizable than those by DeepInversion and some of them do possess features of the target persons. However, there are also two limitations of GMI caused by its traditional GAN architecture: 1) inverted images are of low resolution 64×64 and thus the facial characteristics are too blurry to disclose needed information (for the attacker); 2) the latent space is entangled [34], meaning that features do not have locality in the space such that exploration of the space is very likely stuck in numerous local optimals, leading to low-quality images that do not resemble the target person as shown by Figure 4 and Figure 13. A straight-forward idea to solve the first problem is employing a more capable GAN such as PGGAN (3×1024^2) [32]. However, as shown by Figure 13 and Figure 26 (in online Appendix [7]), without solving the second problem, the high-resolution inverted images are still unnatural and unlike the target persons.

B. Existing Black-box Inversion Methods

The existing state-of-the-art black-box inversion method is AMI [65], which is based on a generative network. To invert a label t, AMI feeds a one-hot vector (*i.e.*, a vector with all 0 but 1 at the t-th element) to a generative model. Different from the generative model in GMI which learns a mapping from latent values to images, here it learns a mapping from the prediction scores to the training images from an auxiliary dataset. The prediction scores are acquired by querying the target model with the training images. The training goal is to minimize the *mean squared error* (MSE) between the generated images and the training images in the auxiliary dataset. The images inverted by AMI in Figure 4 reveal its limitations: 1) inverted images are blurry average faces and disclose little information of the target labels; 2) the low classification confidence means the target model does not recognize the inverted images as the target persons; 3) many target labels cannot be inverted. The first limitation is due to the low capability of its generative model (3×64^2) and the use of MSE loss, which focuses on smoothing the detailed areas of generated images to achieve more pixel-wise correctness [42]. Since the technique aims to use a mapping learned from an auxiliary dataset to invert labels from an unseen dataset, it is critical for the model to learn human face features that can be decomposed and generalized. However, the simple MSE loss on the auxiliary dataset does not seem to enforce that.

C. Our Solution

To overcome the issues of weak generative networks and entangled latent spaces, we propose to use StyleGANs (3×1024^2) [34], [35]. StyleGAN has a special architecture that substantially mitigates entanglement and decomposes features to *styles* that can be controlled separately to generate different outputs. To leverage StyleGAN for inversion, a simple idea is to optimize in the (disentangled) $\mathcal{W}$ space to generate a natural input that is classified to the target label, and avoid optimizing in the entangled $\mathcal{Z}$ space. However, although the $\mathcal{W}$ space is less entangled, its distribution is difficult to model, as demonstrated by the irregular shape of the space in Figure 3 (c). As such, it is difficult to prevent the optimization from going beyond the distribution, leading to undesirable inversion results. We hence propose to regulate at an internal space that is very close to $\mathcal{W}$ but having a multi-variate Gaussian distribution. As such, on one hand, we can benefit from the $\mathcal{W}$ space's capabilities of controlling features/styles; on the other hand, we can easily ensure such controls are within distribution. In the black-box setting, we apply a search algorithm (*e.g.*, genetic algorithm) to explore the $\mathcal{W}$ space with similar regularization. Directly using StyleGANs sometimes generates inverted samples that do not resemble the target label (e.g., wrong gender) even though the subject model classifies the samples to the target label with high confidence. This is often due to low level feature biases in the original training set of the subject model which causes the model to undesirably learn strong connections between the target label t and some low level features. As such, during inversion StyleGAN may generate a face that possesses the features but does not resemble t in humans' eyes. To address the problem, we randomly drop out neurons in the subject model during inversion. As such, the inverted samples that resemble t stand out over time (see Section IV-B and V-C6).

From Figure 4, the images inverted by our method have finer-grained facial characteristics and are much more similar to the target people. Consider the person in the first row. Observe that his unique features include the eyes (and eye brows), nose, and hair style. His smile lines are kind of special too. They are faithfully captured by our inversion results in both settings. In contrast, the GMI result catches the eye shape and the beard. But the rest is blurry and non-distinguishing. The DeepInversion result catches the nose but the face is very scrambled. The females in the second and the fourth rows are similar to each other (by looking at the ground-truth samples). The differences lie in that the former has a relatively more pointy nose while the latter has a relatively round one, and the latter has very obvious double eyelids whereas the former does not. Their smile lines are also different. Our inversion results clearly capture these differences. In contrast, none of the other techniques can disclose such details. Our results for the black male in the third row faithfully reveal his skin color, nose shape, lip shape, cheek lines, eyeglasses, and bald hair. In contrast, the best result from other methods (i.e., the one by GMI) lacks a lot of details. Another observation is that the subject model produces close to 1.0 classification confidence for the results by most white-box methods despite the obvious quality difference. It indicates that cross-entropy loss is not a deciding factor for the quality.

As we will show in Section V-D, our technique can invert models on various kinds of data (in addition to human faces) with high fidelity. These results suggest that if DL models are operating on sensitive data, privacy leakage is indeed a serious and practical concern. A large set of inverted samples are provided at [7] for interested readers.

IV. DESIGN

Figure 5 presents an overview of MIRROR. The top dash frame illustrates a StyleGAN that transforms an initial latent space $\mathcal{Z}$ (at ①) to an intermediate space $\mathcal{W}$ (at ②) which is used to parameterize styles and synthesize images (at ③). The top and middle dash frames together illustrate white-box inversion. Starting from the initial $\mathbf{w}$ (at ②), StyleGAN generates a sample (at ③) which is fed to the target model to get the output (at ④). MIRROR randomly dropouts neurons (denoted by the white squares) in the target model during classification to mitigate low level feature overfitting. It then computes the classification loss and uses a gradient-decent method to optimize $\mathbf{w}$ to minimize the loss (⑤). To ensure the quality of inverted images, it regularizes $\mathbf{w}$ at step ⑥, which is a detour from the $\mathcal{W}$ space to a $\mathcal{P}$ space (which will be defined in Section IV-B) and then back to $\mathcal{W}$. The loop ②③④⑤⑥ terminates when the inversion succeeds or the maximal epochs are reached. Note that the $\mathcal{Z}$ space and the mapping network f (at ①) are excluded in the process.

The top and the bottom frames in Figure 5 denote the pipeline for black-box inversion. At ④′, the attacker can only

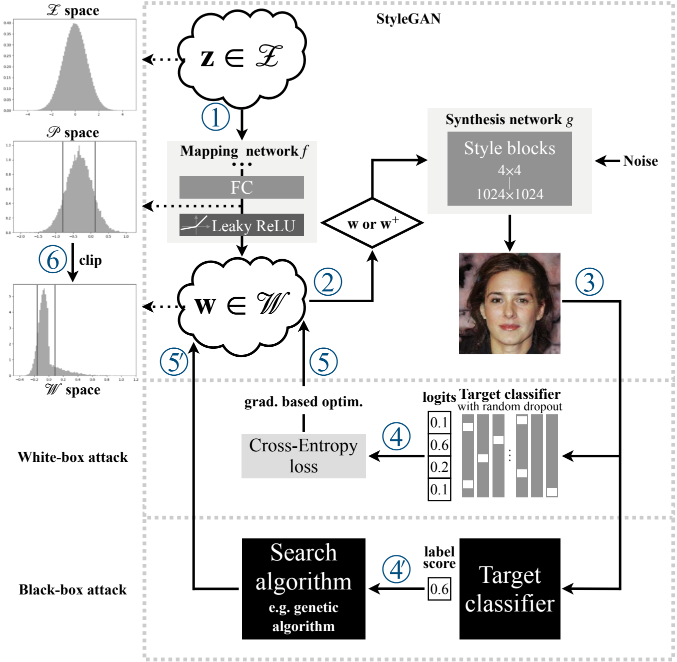


Fig. 5: Architecture of Our Approach

access the target label score instead of values of the whole output layer and gradients. Step ⑤′ utilizes a genetic algorithm to find $\mathbf{w}$ that can improve the score.

A. *Observations in Using GANs for Model Inversion*

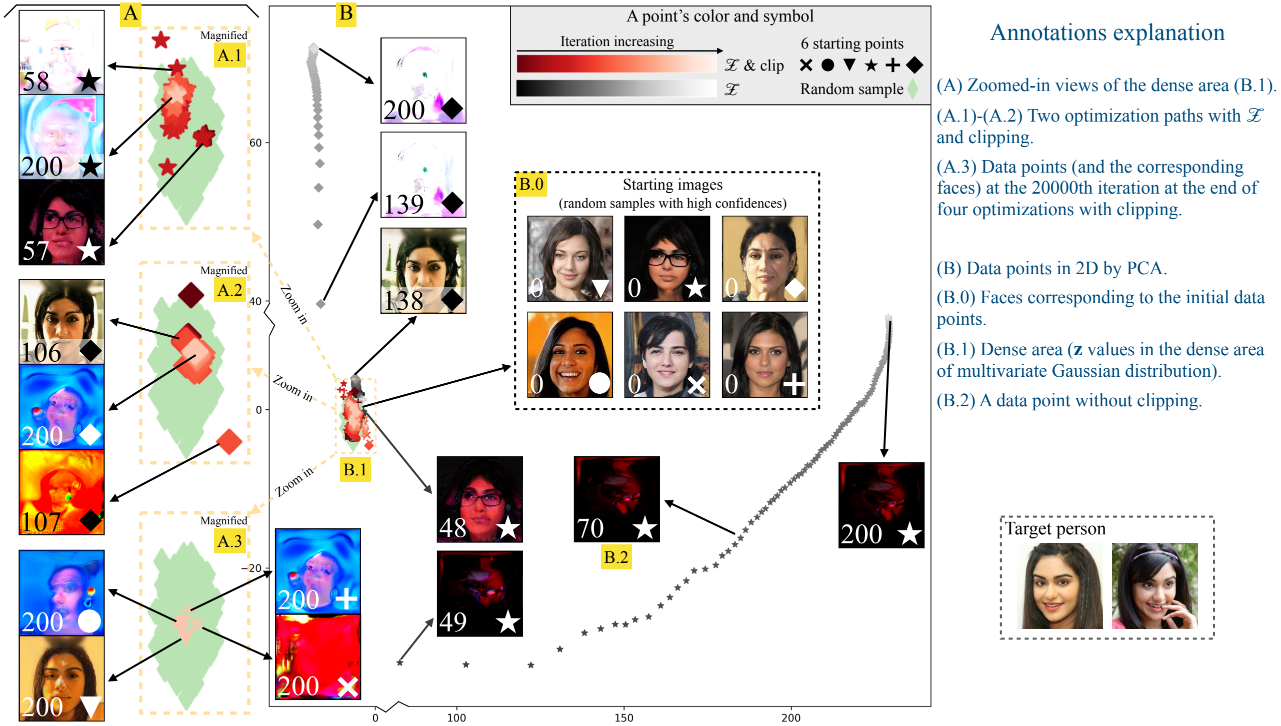
Our inversion method is based on GANs. While the idea of using GAN to invert class labels is not new, e.g., GMI [73], we find that if not done properly, it produces results of poor quality. We have explored various design choices and some results are presented in online Appendix VII and VIII [7]. In this subsection, we discuss a number of observations we have made in the exploration. They lead to our final design.

1) Optimization in the $\mathcal{Z}$ Space is Ineffective: A basic assumption of GAN is that samples in a natural domain follow a Gaussian distribution such that the generator is essentially a mapping from $\mathbf{z}$ values with a Gaussian distribution to the natural domain. Therefore, given a class label, a straightforward idea followed by existing GAN-based inversion methods is to optimize the $\mathbf{z}$ vector to yield the minimum loss. The inverted image is hence the one produced by the GAN generator from the $\mathbf{z}$ vector. In order to enforce the naturalness of generated images, they clip $\mathbf{z}$ vectors based on the (Gaussian) distribution of $\mathcal{Z}$ space. However, we find that the nature of $\mathcal{Z}$ space makes it difficult to achieve good results.

Figure 6 illustrates an example inversion procedure searching the $\mathcal{Z}$ space of StyleGAN [34]. The goal is to invert label 38 of a ResNet50 model pre-trained on VGGFace2. Two images of the target person are shown on the right-bottom. We first randomly sample 2000 $\mathbf{z}$ vectors following the Gaussian distribution. They correspond to a set of normal images generated by the GAN. They are the green data points in the dense area (B.1). We then select 6 images that the target model has the highest confidence for label 38 as the starting images for optimization. They are denoted by the unique symbols as shown in the legend at the top-right corner of (B). For each starting image, we run two kinds of optimization: 1) with clipping $\mathbf{z}$ to 0±1; 2) without clipping. Each data point generated at an optimization iteration is colored based on the strategy (red for with clipping and gray for without clipping) and has the symbol of its origin. The color is gradually lightened as the number of iterations grows. As such, the data points of the same symbol form an optimization path. We also show the corresponding (face) images for a set of data points. Each image is annotated with a number denoting the hundredth step (in optimization) and its symbol. For example, the image at (B.2) denotes a face generated after 70000 optimization iterations, starting from the initial face with the same star symbol in (B.0). As the dense area is too crowded, we provide three zoom-in views on the left. For example, the (A.1) view presents the inversion results with clipping after 5700, 5800, and 20000 steps.

From Figure 6, we have a number of observations. 1) By comparing the greyish data points (of different symbols), which denote the optimization results without clipping and are distant from the dense area (B.1), with their reddish counterparts with clipping in (A.1) and (A.2), we can see clipping helps confine vectors to the dense area. Specifically, observe the optimization path of the grey star symbol, i.e., the path containing (B.2). The images become scrambled when the optimization goes beyond the normal zone (e.g., at step 4900). 2) Even the confined images corresponding to the reddish data points in (A.1) and (A.2) may become unrecognizable too. Observe the 4 images in (A.3). They fall well within the dense area but three of them are not human recognizable. In other words, not all data points in the distribution denote natural faces. Just like adversarial samples close to an input of a certain label can be classified to other labels, the latent space of a GAN is not robust either, namely, the close-by values of a latent value denoting a natural face may not represent natural faces. As such, $\mathcal{Z}$ space clipping is a weak constraint for generating human recognizable faces. GMI uses a strategy of having a small number of optimization steps, trying to mitigate the problem. However, 3) a small number of optimization steps do not sufficiently search the space such that the resulting images do not deviate from the starting images much and do not resemble the target person. 4) The regions denoting the target label's features are scattering in the entire $\mathcal{Z}$ space. Note that all the optimization results (with different symbols) are for the inversion of the same person although they are distributed in the whole space.

The literature has pointed out that the $\mathcal{Z}$ space is highly entangled [34]. In our context of model inversion, it means that *similar features do not have locality and distribute in numerous and separated areas in the space.* As such, it is very difficult for the optimization to find the global minimal. Figure 3 provides a conceptual illustration. The circle in (a) denotes the natural domain in a Gaussian distribution. Each color denotes a kind of feature. Observe that they have locality. The transitions between colors are smooth, denoting samples possessing combinations of multiple features. In other words, naturally similar samples are close to each other. In contrast, figure (b) denotes the $\mathcal{Z}$ space of a standard GAN. Observe that there is no locality of colors. Instead, they scatter in numerous small areas. While such an entangled space does not affect generation, it indeed makes inversion highly challenging.



(A) Zoomed-in views of the dense area (B.1).

(A.1)-(A.2) Two optimization paths with $\mathcal{Z}$ and clipping.

(A.3) Data points (and the corresponding faces) at the 20000th iteration at the end of four optimizations with clipping.

(B) Data points in 2D by PCA.

(B.0) Faces corresponding to the initial data points.

(B.1) Dense area (**z** values in the dense area of multivariate Gaussian distribution).

(B.2) A data point without clipping.

Fig. 6: Model inversion in the $\mathcal{Z}$ space. We invert images for a target person (as shown in the bottom-right) with 2 different strategies (i.e., with and without clipping), starting from 6 random $\mathbf{z}$ values denoting 6 random faces. The optimization derives a $\mathbf{z}$ value at each iteration. We present these values inside **B** after reducing them to 2-dimensional data points. As shown by the shaded legend in the top-right corner, data points originating from the different starting points are denoted with six unique symbols. The data points generated by the two optimization strategies have red and gray colors, respectively, with the different color scales denoting the optimization iterations. Therefore, all the data points starting from the same initial value form an *optimization path* in the space with the same symbol of a gradually lightened color. We also display the corresponding faces for a selected subset of data points. Each image is tagged with a number denoting the hundredth iterations (e.g., 70 meaning the 7000th iteration) and the symbol denoting the initial data point. The dense area **B.1** is enlarged to **A.1**, **A.2** and **A.3** on the left for better visibility.

2) Pre-trained Discriminator Cannot Enforce Naturalness: During a GAN's training, a discriminative network D competes with a generative network G. The optimization goal is to find a fixed point of a two-player min-max game:

$$\min_{G} \max_{D} \mathbb{E}_x[\log D(x)] + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))] \ ,$$

where x is sampled from real images and $\mathbf{z}$ is sampled from the latent space (*e.g.*, a Gaussian distribution).

After a pair of G and D is trained, a plausible idea of ensuring generation of high-quality faces is to use D, like in a number of GAN-based inversion methods (*e.g.*, GMI). Particularly, the inverted images and a set of random natural images are fed to D to compute a *discriminative loss* $\log(1 - D(G(\mathbf{z})))$, which is supposed to gauge the differences between the sets. However, it's not effective based on our study. The GMI results in Figure 4 are not particularly of good quality. In addition, removing the discriminative loss from GMI or enlarging its weight has little effect on the result (online Appendix VIII [7]). These indicate *pre-trained discriminators can hardly ensure quality during inversion despite its conceptual feasibility*.

We conduct an experiment to further study the underlying reasons. Figure 7 shows a set of images with various levels of quality (in humans' eyes) in an ascending order of discriminative loss based on the discriminator in a pre-trained StyleGAN. The number in each image shows its discriminative loss and the text denotes its source. CelebA means the StyleGAN's training data. VGGFACE denotes the VGGFACE dataset. Images annotated with GMI/AMI/Ours are inverted by GMI/AMI/Ours. People usually think images with a lower discriminative loss are more real. However, it's not the case as shown in this figure. For example, the blurry images inverted by AMI and GMI have much smaller discriminative loss than the CelebA data. We speculate that although conceptually D is to distinguish in-distribution (natural) and out-of-distribution (unnatural) samples, it is indeed to distinguish two sets of

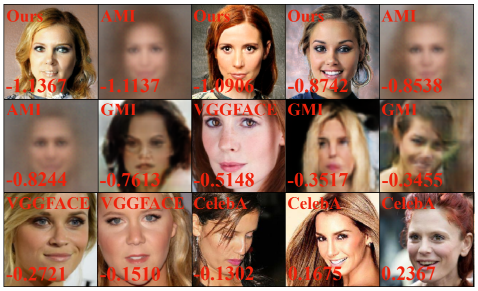


Fig. 7: Discriminative loss on images of various qualities. Lower discriminative losses are usually believed to be related to more natural images. However, it is not true.

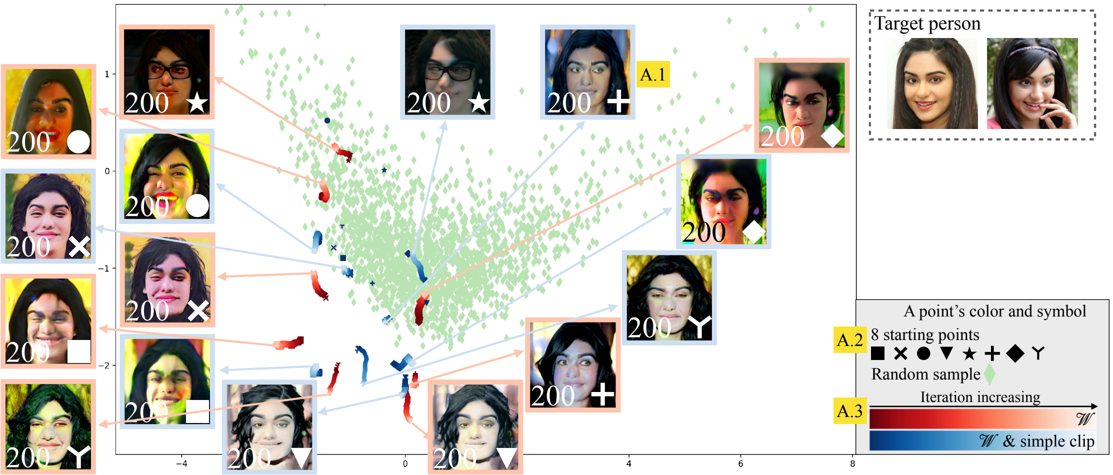


Fig. 8: Model inversion in the $\mathcal{W}$ space. The $\mathbf{w}$ vectors are reduced to 2D by Principal component analysis (PCA) and denoted as data points in the figure. Inversion starts from 8 initial values denoted by 8 different symbols as shown in **A.2**. The data points in green denote 2,000 $\mathbf{w}$ latent vectors obtained by sampling in the $\mathcal{Z}$ space. We use red to denote optimization without clipping and blue with simple clipping which clips $\mathbf{w}$'s dimensions based on their range profiles (see **A.3**). Images corresponding to latent vectors at the last 20000th iteration are displayed. The number and symbol in each image denote the hundredth step (of optimization) and its origin, respectively. The optimization goal and starting points are the same as Figure 6 but here we use $\mathcal{W}$ space. The paths with different scales of blue/red color denote the optimization paths.

samples: one from the generator and the other from the natural domain. Close to the end of training, G already learns to generate highly realistic samples such that D tends to learn low level feature differences between G's outputs and the natural samples. That is, it is refining the details. As such, D does not serve well in making a high level decision if a sample is natural. We have also tried to use a D from a half-way trained GAN. The results are not better. We speculate that the design of standard GAN training would only produce D's that can improve the corresponding G's. These D's are not classifiers of naturalness in general.

The two observations render a naive application of GAN in model inversion ineffective and motivate our design.

B. White-box Inversion in MIRROR

$\mathcal{W}$ Space Is Disentangled but Not Normal. Since the $\mathcal{W}$ space is disentangled, a straightforward method is to directly optimize $\mathbf{w}$ to invert a label. However, we find that such a simple design hardly works. Figure 8 shows the $\mathcal{W}$ space and the inversion procedure of a target person in the space. The V shape of the green samples (i.e., the $\mathbf{w}$ values of 2000 Gaussian samples acquired from the $\mathcal{Z}$ space denoting normal persons) indicates that these samples do not follow a Gaussian distribution in $\mathcal{W}$. As such, while the optimization is able to explore localized areas (due to disentanglement), the resulting samples (after 20000 steps) have substantially degraded quality as they are out of distribution. Due to the same reason, range based clippings do not work either because such clippings may introduce substantial bias when the underlying distribution is not isotropic Gaussian [16]. More results can be found in Figure 31 in online Appendix X [7].

Regulation by Clipping in Multi-variate Gaussian $\mathcal{P}$ Space. $\mathcal{W}$ does not follow a Gaussian distribution. Neither are the individual dimensions of a $\mathbf{w}$ vector. The third distribution from the top in the left of Figure 5 shows a typical distribution of $\mathbf{w}$ dimension. The irregular distribution is due to the leaky ReLU operation [63]. In contrast, the values of individual dimensions before the leaky ReLU follow Gaussian distributions. It is called the $\mathcal{P}$ space and it has a multi-variate Gaussian distribution. Figure 9 shows the histograms and Quantile-Quantile (Q-Q) plots [45] of a random dimension of the spaces before/after the last two LeakyReLU layers of StyleGAN's mapping network. A Q-Q plot compares two distributions by plotting their quantiles. If the plotted dots fall approximately along the 45-degree reference line, then the two distributions are similar. Here we compare the distribution of the selected dimension against a Gaussian distribution with its computed mean and standard deviation. We can see the spaces before LeakyReLU are Gaussian (the histograms being bell-shaped and the dots aligning well with the reference line) while those after LeakyRLU are not. We use $\mathcal{P}$ to denote the space before the last LeakyReLU. We regularize latent vectors $\mathbf{w}$ in $\mathcal{P}$ because it's the closest to the $\mathcal{W}$ space and it's the deepest space satisfying Gaussian distribution. The StyleGAN paper claimed deeper mapping networks perform better and have a more disentangled space [34]. Note that although the output of the LeakyReLU preceding the $\mathcal{P}$ space is not a Gaussian distribution, there is a fully-connected layer between it and

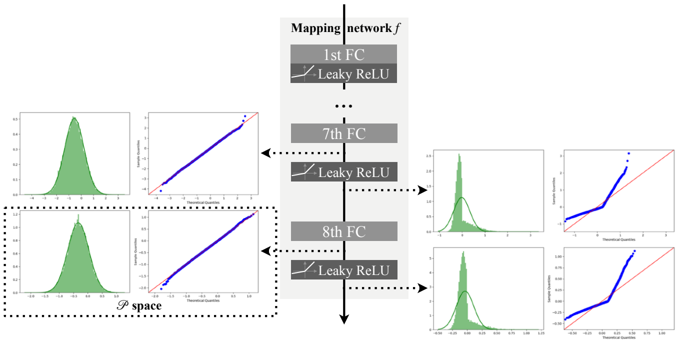


Fig. 9: Histogram and Q-Q plot.

Algorithm 1 White-box inversion

```
1:  function INVERSION(model $M$, target $t$, epoch $e$, lr $\eta$)
2:      $w_1$ = INITWSPACE($M$, $t$)
3:      $w_{best}, \ell_{best} = w_1, \infty$
4:      for $i \in \{1, \ldots, e\}$ do
5:          $img_i$ = SYNTHESIZE($w_i$)                ▷ i.e., $g(w_i)$
6:          $\ell_i$ = LOSS($M(img_i)$, $t$)             ▷ e.g., CE Loss
7:          if $\ell_i < \ell_{best}$ then
8:              $w_{best}, \ell_{best} = w_i, \ell_i$
9:          end if
10:         $w_{i+1} = w_i - \eta \nabla_w \ell_i$
11:         $p_{i+1}$ = TOPSPACE($w_{i+1}$)
12:         $p_{i+1}$ = CLIPPSPACE($p_{i+1}$)
13:         $w_{i+1}$ = TOWSPACE($p_{i+1}$)
14:     end for
15:     return SYNTHESIZE($w_{best}$)                  ▷ i.e., $g(w_i)$
16: end function
```

the $\mathcal{P}$ space. The output of a fully-connected layer can be considered as the weighted average for each input dimension. According to the central limit theorem [14], it tends toward a normal distribution. In the literature, as a result of the central limit theorem, similar Gaussian distributions of outputs before ReLU layers are observed in ResNet152 [64], while others also observe Gaussian distributions for network weights [21], [57].

Therefore, we ensure in-distribution optimization as follows. We take a large number of Gaussian samples from the $\mathcal{Z}$ space and feed them through the mapping network f and collect their activation values in the $\mathcal{P}$ space. From these activation values, we establish the Gaussian distributions for individual $\mathbf{p}$ dimensions. The following equation denotes the leaky ReLU operation that transforms a $\mathbf{p}$ to the corresponding $\mathbf{w}$ and its inverse function.

$$\text{LeakyReLU}(x) = \max(0, x) + 0.2 \times \min(0, x)$$
$$\text{LeakyReLU}^{-1}(x) = \max(0, x) + 5 \times \min(0, x)$$

Intuitively, it retains the value if it is positive otherwise multiplies it with a small weight 0.2. In the inverse direction, it retains the value if it is positive otherwise multiplies it with a weight value 5, which is 1/0.2. Given an updated $\mathbf{w}$ by the optimization, we regulate it by first projecting it to the $\mathcal{P}$ space (using the inverse leaky ReLU function), clipping its individual dimensions based on the profiled $\mathcal{P}$ space distributions, and then projecting it back to the $\mathcal{W}$ space (using leaky ReLU). Formally, let $\mathbf{w}[i]$ and $\mathbf{p}[i]$ denote the i-th dimension of $\mathbf{w}$ and $\mathbf{p}$, with the latter's distribution being $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$.

$$\mathbf{p}[i] = \text{LeakyReLU}^{-1}(\mathbf{w}[i]) \quad (1)$$
$$\mathbf{p}[i] = \max(\min(\mathbf{p}[i], \boldsymbol{\mu}[i] + \boldsymbol{\sigma}[i]), \boldsymbol{\mu}[i] - \boldsymbol{\sigma}[i]) \quad (2)$$
$$\mathbf{w}[i] = \text{LeakyReLU}(\mathbf{p}[i]) \quad (3)$$

Algorithm. The overall inversion procedure is the one described at the beginning of this section. It is formally described by Algorithm 1. Line 2 first initializes $\mathbf{w}_1$ by selecting those with the highest target confidence from the samples pre-generated to compute the $\mathcal{P}$ space statistics, by calling a function INITWSPACE(). The function's definition is elided due to simplicity. Lines 4-14 execute the inversion loop for e times. In each iteration i, line 5 calls SYNTHESIZE() to generate candidate images img_i from w_i using the g function of the generator (step ② in Figure 5). Line 6 feeds the images to the target model and computes the loss according to the target label (steps ③④). Line 7 records the results with the smallest loss. Line 10 updates $\mathbf{w}$ guided by the gradient decent (step ⑤). Lines 11-13 regulate $\mathbf{w}$ with TOPSPACE, CLIPPSPACE, TOWSPACE defined in Equations (1), (2), (3), respectively (step ⑥). Line 15 produces the image with the best $\mathbf{w}$.

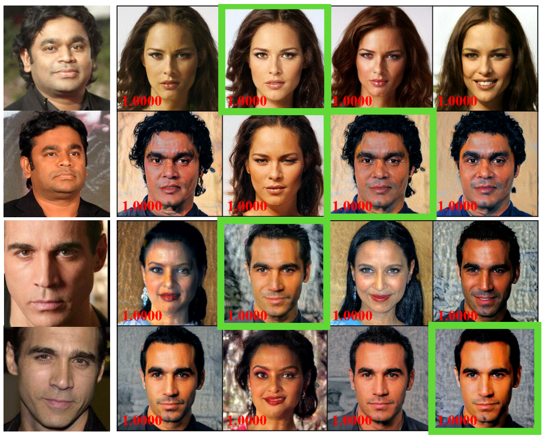


Fig. 10: Effects of dropout and consistent selection strategy. The first column shows two images of each target person. The odd/even row shows the inversion results without/with dropout strategy. The final images selected by our consistent selection strategy are highlighted by green squares. Numbers denote the target confidence.

Random Dropout to Mitigate Feature Overfitting. We find that using the aforementioned inversion algorithm sometimes may generate samples that largely differ from the target label (e.g., wrong gender). These samples look natural and are classified to the target label by the subject model. We speculate that the subject model overfits on some low-level features such that the optimization falls into some local optima. We propose to use random dropout during inversion to mitigate the effects of overfitting. The idea is that if the neurons denoting those undesirably strong neurons (for the target label) are dropped, i.e., their activations set to 0, the optimization can escape from the local optima. Specifically, in each iteration, MIRROR randomly selects layers of M and append each of them with a dropout layer that randomly sets a subset of neurons in the layer to 0. Figure 10 shows some samples before and after random dropout. The odd rows are inverted samples of a target person without dropout while the even ones show those with dropout. For example, without dropout, the inversion samples for the first person all have the wrong gender and do not resemble the target person. With dropout, MIRROR generates more accurate inversion results.

While dropout allows generating more accurate inversion results, it cannot exclude the wrong ones. For example, there are still a few female inverted images in the second row of Figure 10. Since MIRROR does not have any prior knowledge of the target label, it needs to determine which image is more likely the target person. We observe that images corresponding to local optima (i.e., the wrong inverted images) have more diverse top-5 labels compared to the correct ones. Note that both the correct and wrong ones often have the same (high) top-1 confidence. We hence propose to select the result images by identifying the largest subset with consistent top-5 labels.

Algorithm 2 Black-box genetic inversion

```
 1: function GenInv(model $M$, target $t$, size $n$, epoch $e$, conf $c$)
 2:     $generation_1 = \text{InitGeneration}(n)$
 3:     for $i \in \{1, \ldots, e\}$ do
 4:         $scores_i = \text{ComputeScore}(M,\ generation_i)$
 5:         $elite_i = \text{FindElite}(generation_i,\ scores_i)$
 6:         if $scores_i[elite_i] \geqslant c$ then
 7:             return $elite_i$
 8:         end if
 9:         $generation_{i+1} = \{elite_i\}$
10:         for $j \in \{1, \ldots, n-1\}$ do
11:             $parent_1 = \text{Sample}(generation_i,\ scores_i)$
12:             $parent_2 = \text{Sample}(generation_i,\ scores_i)$
13:             $child_j = \text{Crossover}(parent_1,\ parent_2)$
14:             $child_j = \text{Mutate}(child_j)$
15:             $p_j = \text{ToPSpace}(child_j)$
16:             $p_j = \text{ClipPSpace}(p_j)$
17:             $child_j = \text{ToWSpace}(p_j)$
18:             $generation_{i+1} = generation_{i+1} \cup \{child_i\}$
19:         end for
20:     end for
21:     return $\text{FindElite}(generation_e,\ scores_e)$
22: end function
```

In particular, given a batch of n inverted images for the target label of model M, we use M to predict top-5 labels for each inverted image. Given a predefined parameter $k <= 5$, we identify the largest subset of the n inverted images that share at least k common labels in their top-5 labels. We then return the image with the largest confidence within the subset. We use $k = 2$ in this paper. The final images selected by our consistent selection strategy in Figure 10 are highlighted by bold squares. Observe that it returns the correct images (when used together with dropout).

Other Design Choices. Besides the $\mathcal{Z}$, $\mathcal{W}$, and $\mathcal{P}$ spaces we have described, there are other spaces that can be leveraged for inversion. Discussion and results can be found in online Appendix I [7].

C. Black-box Inversion

In the black-box setting, gradients are not accessible. However, with the disentangled $\mathcal{W}$ space and the regularization in the $\mathcal{P}$ space, discrete search algorithms can be used to look for a minimal of the same loss function. Here, we employ a genetic algorithm [15]. Starting from an initial *population* (i.e., a set of initial samples in $\mathcal{W}$), the algorithm derives the next generation by *cross-over* and *mutation*. The former is by exchanging parts of two arbitrary samples. The latter is by randomly perturbing individual dimensions of a sample from the population. The new samples are regulated (in $\mathcal{P}$) and then fed to the target model to determine their health level, which is the prediction score. A set of *healthy* samples are selected to form the next generation.

Algorithm 2 formally describes the process. Line 2 is the initialization that provides the first generation of size n. We sample $\mathcal{Z}$ to get a set of latent vectors in $\mathcal{W}$ and the corresponding synthesized images. We query the target model to find images with the n highest prediction scores and use the corresponding $\mathbf{w}$ vectors as the initial generation. Line 4 describes the fitness function which is the prediction score (a single value denoting the target label's confidence). At line 9, it employs the elitism strategy [15] and adds the elite sample with the highest fitness score in the current generation (by function FindElite() at line 5) to the next. Lines 10-19 produce (healthy) offsprings. To produce a better next generation, the algorithm independently and randomly selects parents from the current generation with probabilities proportional to their fitness scores (by function Sample() at lines 11-12). Line 13 denotes the Crossover operation. A child's dimensions are composed of dimensions from either $parent_1$ or $parent_2$ based on the probability $(p, 1-p)$ where p is defined as follows so that the better ones have higher chances to be inherited.

$$p = \frac{scores[parent_1]}{scores[parent_1] + scores[parent_2]}$$

Line 14 denotes the Mutate operation. To diversify the next generation and better explore the search space, each dimension of a child is mutated with a selected probability. Random perturbations uniformly sampled in the range $(-c \cdot \sigma_p, c \cdot \sigma_p)$ are performed, where c is a coefficient to adjust the mutation strength and σ_p is the standard deviation in this dimension of $\mathcal{P}$. Most functions have self-explaining names and their detailed definitions are omitted.

V. Evaluation

We evaluate MIRROR and compare with state-of-the-art approaches on various datasets and models. We also evaluate it on two commercial face recognition services provided by Microsoft Azure and Clarifai. An ablation study is conducted to justify the parameters for our approach. We further show MIRROR is resilient to several existing defense methods. We implement MIRROR in PyTorch [51]. Most experiments are carried out on a server equipped with 256 GB RAM, two Intel Xeon Silver 4214 2.20GHz 12-core CPUs and eight NVIDIA Quadro RTX 6000 GPU. We have the full set of inversion results on Github [7] and we will release our system upon publication.

A. Experimental Setup

1) Datasets and Models: Table I shows the basic information about the 5 datasets and 8 models we use. We use 3 popular face recognition datasets. On each dataset, we select 2 pre-trained models with different architectures. The input image sizes are $3 \times 224 \times 224$ (Channel×Height×Weight), $3 \times 160 \times 160$ and $3 \times 112 \times 96$. They are much larger than what can be handled by previous methods: $3 \times 64 \times 64$ for GMI and AMI. In addition, we follow a similar setup as AMI [65] to evaluate our black-box approach on two commercial facial recognition services [1], [2]. The services let the user upload images of different people and later allow the user to call the API to classify a given sample. We randomly select 8 people from the VGGFace2 dataset to upload and then use the API to evaluate the black-box methods. Moreover, we show inversion results for car brand and model classification with ResNet34 trained on the Stanford Cars dataset [36] and cat/tiger breed classification with ResNet18 trained on cats from the Oxford-IIIT Pet dataset [50] and images of Amur tigers [37] and white tigers. Although car model and cat breed in general do not constitute sensitive information, our goal is to show the inversion results have high fidelity such that DL models processing private information have serious privacy leakage concerns.

TABLE I: Datasets and models on which our approach is evaluated. The two numbers in parentheses after each dataset denote the number of different identities and the number of images in that dataset. The models below a dataset means they are pre-trained on the dataset (except the last two columns). For example, "ResNet50" and "InceptionV1" under "VGGFace2" are pre-trained on the "VGGFace2" dataset and later each of them will be used as the target model while the other is the reference model.

Dataset	VGGFace (2,622/2.6M) [49]		VGGFace2 (9,131/3.3M) [17]		CASIA (10,575/0.5M) [67]		Cat+tiger (14/2.8K) [50], [37]	Cars (196/16.2K) [36]
Model	VGG16 [6]	VGG16BN [6]	ResNet50 [6]	InceptionV1 [3]	InceptionV1 [3]	SphereFace [4]	ResNet18 [26]	ResNet34 [26]
Accuracy	97.22	96.29	99.88	99.65	99.05	99.22	93.05	90.30
Parameters	145M	110M	41M	28M	29M	28M	11M	21M
Image size	3×224×224	3×224×224	3×224×224	3×160×160	3×160×160	3×112×96	3×224×224	3×400×400

2) StyleGANs: We mainly use StyleGANs trained by third parties as our generators. We use a pre-trained StyleGAN from [55] based on 103,706 images selected from the CelebA datset [38] for face-recognition tasks, StyleGAN and StyleGAN2 pre-trained on the FFHQ dataset [34] in our ablation study, and a StyleGAN pre-tained on portrait art [11] for additional results. Moreover, we use a pre-trained StyleGAN from [33] on the the LSUN Car dataset [71] for car model classification, and a pre-trained model from [33] on the LSUN Cat dataset [71] for cat breed classification.

3) Non-overlapping Inversion: We only invert the labels that are *not* in the StyleGANs' training datasets unless otherwise stated. That is, our approach uses StyleGANs to invert *unseen* identities. In this section, the results are aggregated by inverting the first 100 non-overlapping labels of each target model by default.

4) Baselines: We compare our methods to the state-of-the-art inversion methods GMI [73] and DeepInversion [69] in the white-box setting and AMI [65] in the black-box setting. For DeepInversion, we use three different initialization settings: random noise/cartoon face/average face denoted by DIR/DIC/DIA. The last two are proposed by us to enhance DeepInversion's effectiveness. To make fair comparison, we train all the generators using the same dataset as StyleGAN. We also propose another three baselines: a) optimizing latent vectors in a traditional but high-resolution PGGAN [32]; b) simply choosing the images in GAN's training data with the highest target confidence; c) sampling $\mathbf{z}$ of StyleGAN and choosing the generated images with the highest target confidence. We emphasize the last two baselines are necessary yet neglected by many existing works.

B. Evaluation Metrics

In addition to qualitative evaluation by showcasing images inverted by different methods, we also employ the following quantitative metrics as well as human studies.

Inversion Effectiveness (Accuracy). We report the classification accuracy of the target model on the inverted images.

Inversion Generalizability (Transferable Accuracy). If the inverted images indeed reveal the target person's characteristics rather than overfit to the target model, they should be recognized as the target label by other models with different architectures pre-trained on the same dataset. We hence report accuracy on other models as well. Besides the (top-1) transferable classification accuracy, we also collect the top-5 accuracy. If the target label appears in the top-5 labels sorted by prediction confidences, we consider it correct.

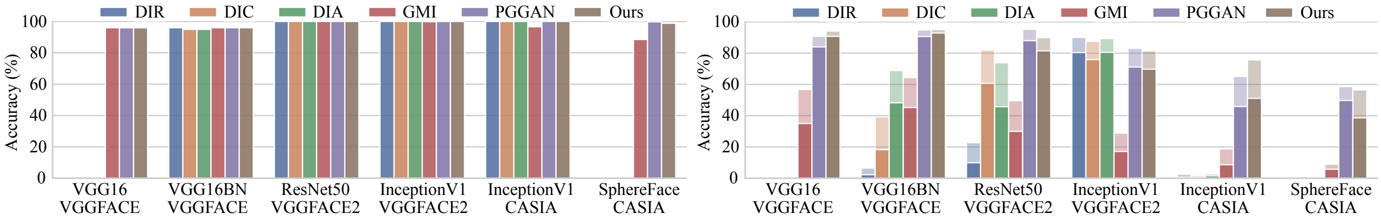
Feature Distances. We compute the ℓ_2 feature distance between inverted images and the centroid of target class. Features are extracted from the penultimate layer of the model, except for SphereFace for which we use the angle distance because it's trained to increase the angular margin between different features. We also report the feature distance on another model with different architectures pre-trained on the same dataset as the target model. The metric is the lower the better.

Natural Image Quality Evaluator (NIQE). NIQE was proposed as a non-reference metric for the perceptual quality evaluation [43]. It constructs "quality-aware" features from a natural scene statistic model and fits a multi-variate Gaussian (MVG) model. Given a test image, it fits another MVG model and the quality is described by the distance between the two MVG models. A smaller score means the better perceptual quality.

Human Study. Because natural images may not pose security threats if humans cannot perceive them as the target labels, for the face datasets, we conduct a human study to further quantify the correctness of the inverted identities. We follow a user study design from the literature [72]. In the setting, users are given a real image of the target identity and then asked to pick one from two inverted images that more resembles the real image. The two inverted images include one from our method and the other from a method in comparison. The order is randomized. These three images will flash on screen for five seconds. For each comparison setting between our method and a baseline method, we randomly select 50 identities for evaluation and in total 10900 pairs of identities are evaluated. There are 218 unique users participating in our study, and 170 out of 218 are considered valid after removing data points greater than one standard deviation. Users are given five warm-up practices before the experiments. A typical user study sample can be found in Figure 22 in Appendix III-A [7]. In addition to the above comparative human studies, we also conduct three more human studies to directly evaluate our results by mixing the inverted image of a target person with 1) training images of other randomly selected people, 2) images of other people that most resemble the target person in the original dataset, and 3) images from CelebA classified as the target by the subject model, respectively. Details and examples are in Appendix III-B [7].

C. Inverting Face Recognition Models

1) White-box Inversion Results: Figure 11a shows the inversion effectiveness of different methods. Except GMI, all the methods have similar inversion effectiveness, because white-box inversion methods explicitly consider the classification loss on the target models. Figure 11b presents the transferable accuracy. Our approach outperforms the state-of-the-art method GMI on all cases by a large margin. Except for one model, ours is much superior to vanilla DeepInversion rand.



(a) Accuracy on target models.

(b) Top-1/top-5 (dark/light color) accuracy on reference models.

Fig. 11: White-box inversion effectiveness and generalizability. The higher the better. DeepInversion cannot be applied to models without BatchNorm layers.

TABLE II: Feature distances of images inverted by white-box methods (the lower the better). The two numbers reported for each setting denote the ℓ_2 feature distances on the target model, and the feature distance on the other model pre-trained on the same dataset, respectively. For example, "149.96" and "147.97"in the cell of row "MIRROR" and column "ResNet50" show the distances computed on the target "ResNet50" and the reference model "InceptionV1". DeepInversion cannot be applied to models without BatchNorm layers.

Method	VGGFace				VGGFace2				CASIA			
	VGG16		VGG16BN		ResNet50		InceptionV1		InceptionV1		SphereFace	
DIR	–		102.67	132.41	161.15	201.16	127.76	174.21	172.06	11.51	–	
DIC	–		93.05	131.64	149.10	185.35	129.40	171.26	172.27	11.41	–	
DIA	–		85.26	117.01	150.91	192.95	130.10	172.69	172.50	11.47	–	
GMI	110.20	103.41	98.87	104.49	155.25	198.51	143.47	188.23	153.34	12.09	12.20	187.73
PGGAN	139.57	82.35	85.71	96.41	164.10	150.22	130.64	170.47	181.18	10.13	9.06	159.83
MIRROR	97.73	55.41	72.06	80.65	149.96	147.97	119.82	163.84	154.41	9.81	8.96	156.19

With better initialization methods, DIC/DIA perform better but are still worse than ours on all but one model. Our proposed PGGAN baseline is already stronger than existing state-of-the-art methods and it has comparable transfer-accuracy to ours. Per Figure 13, we suspect PGGAN has comparable transferable accuracy because it generates sharp and distorted features of the target. However, the generated faces have less resemblance to the target (from humans' perspective) as indicated by the NIQE results (Table III) and the user-study (Figure 12) on all the 6 models.

For the (target/transferable) feature distances shown in Table II, images generated by our approach have the smallest ℓ_2 feature distances on both the target model and the reference model except its distance is slightly larger than GMI's for the target InceptionV1 model trained on CASIA and DIC's for the target ResNet50 model trained on VGGFace2. But note that the distances on the two corresponding reference models are much smaller.

For the perceptual quality evaluation, The NIQE scores in Table III for the images inverted by ours are much better. We also display images inverted by different methods in Figure 13. We can observe: 1) our images have better quality and are much more similar to the target people; 2) our approach generates more consistent images in terms of facial characteristics and gender.

Figure 12 shows the user study results. The x axis shows the different datasets. The y axis shows the preference rate for our samples. That is how many times (in percentage) the user prefers our inverted sample over the other. Hence, 50% means that our technique is not better. The center of each point denotes the average preference and its diameter means the standard error. Our inverted images are much more favored by the users during the human study. Note that it is really hard to design a user study to directly measure how much given samples resemble a target person. We hence design the comparative study to show that ours is better than the others. When we mix the inverted image of a target person with the training images of other randomly selected people, 95.71% users can correctly select the target person given the inverted image. When the inverted images are mixed with those of other people that most resemble the target person in the original dataset, the percentage decreases (as expected) to 89.29%, which is still high. When the inverted images are mixed with the images from CelebA misclassified as the target person, the percentage further decreases (as expected) to 78.75%. More details are in Appendix III-B [7]. Interested readers can refer to the samples in Figures 13 and 28 (inverted using an art Style-GAN) in online Appendix and our online repository [7] for further inspection.

TABLE III: NIQE of images inverted by white-box methods (the lower the better). DeepInversion cannot be applied to models without BatchNorm layers.

Method	VGGFace		VGGFace2		CASIA	
	VGG16	VGG16BN	ResNet50	InceptionV1	InceptionV1	SphereFace
DIR	–	25.67	16.84	18.88	18.87	–
DIC	–	6.28	6.38	5.22	5.06	–
DIA	–	6.49	5.68	5.21	5.06	–
GMI	6.82	6.78	6.94	6.57	6.76	6.59
PGGAN	5.32	5.17	4.95	5.00	4.74	5.43
MIRROR	3.56	3.59	3.46	3.58	3.56	3.70

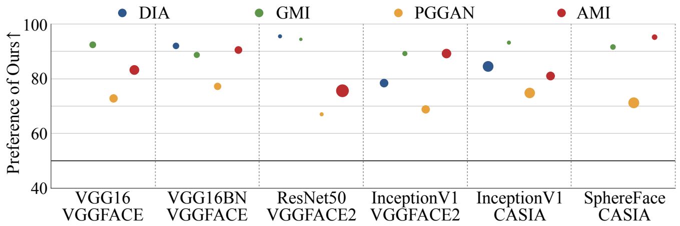


Fig. 12: Human preference of our images over others (the higher the better). DeepInversion cannot be applied to models without BatchNorm layers.

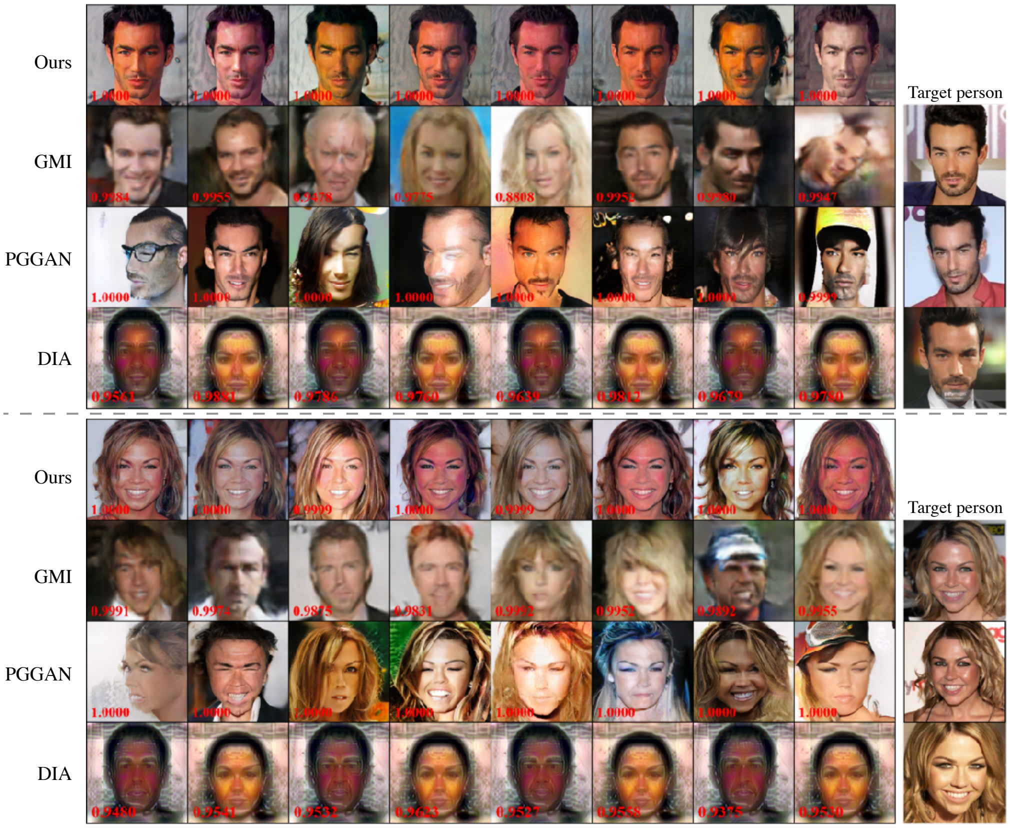


Fig. 13: Images inverted in the white-box setting by our approach MIRROR, GMI, our proposed PGGAN baseline and DeepInversion initialized with average face (DIA). Our images are not only more human-recognizable and similar to the target people, but also more consistent in terms of facial characteristics and gender.

TABLE IV: Feature distances of images inverted by black-box methods (the lower the better). Two numbers in a cell correspond to the ℓ_2 feature distance on the target model and the other model pre-trained on the same dataset (except for the two commercial services whose target feature distances are not computable). For example, “150.27” and “172.03”in the cell of row “MIRROR” and column “ResNet50” show the distances computed on the target “ResNet50” and the reference model “InceptionV1”. For MS Azure, we compute the feature distances based on InceptionResnetV1. For Clarifai, we compute the feature distances based on 24-class VGG16.

Method	VGGFace				VGGFace2				CASIA				8-classes	
	VGG16		VGG16BN		ResNet50		InceptionV1		InceptionV1		SphereFace		MS Azure	Clarifai
AMI	103.10	124.15	125.82	104.73	197.11	234.40	234.95	202.51	188.05	10.89	11.37	198.98	198.51	24.53
MIRROR	90.83	69.05	62.19	85.17	150.25	172.03	130.77	167.86	139.39	10.12	10.09	173.34	176.67	16.66

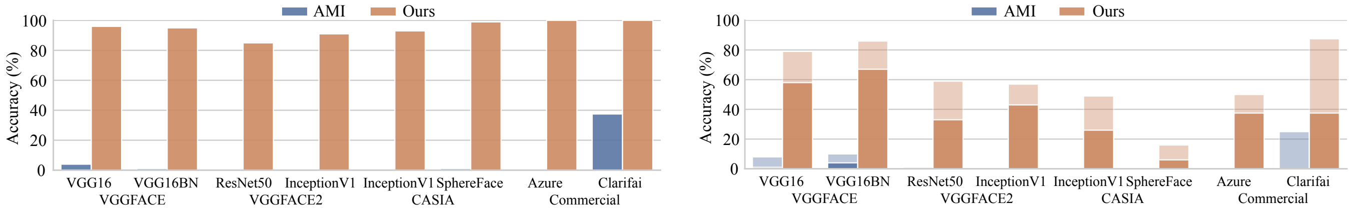
TABLE V: NIQE of images inverted by black-box methods (the lower the better).

Method	VGGFace (4.95)		VGGFace2 (3.09)		CASIA (4.14)	
	VGG16	VGG16BN	ResNet50	InceptionV1	InceptionV1	SphereFace
AMI	4.83	5.10	4.83	4.92	4.84	4.75
MIRROR	3.73	3.78	3.62	3.80	3.72	3.81

2) Black-box Inversion Results: As shown by Figure 14, our black-box inversion approach is quantitatively much better than AMI, which has very low inversion effectiveness and generalizability as we have explained in Section III. Images inverted by our approach have smaller feature distances in all cases as shown by Table IV. It's apparent that our approach can produce images of higher quality than the blurry images of AMI, as shown by the NIQE scores in Table V and the visualized results in Figure 15.

3) Comparison with Choosing Samples with Highest Target Confidence: We compare our approach with two simple baselines which were neglected by many existing works and in fact outperform the existing state-of-the-art methods in most cases. The “Celeba” method simply chooses the images in GAN's training data with the highest target confidence. Similarly, the “Sample” method chooses images from a set of pre-generated images by sampling latent vectors of StyleGAN. As shown by Figure 16-17 and Table VI, our approach is superior to the two baselines. We also conduct a user study with 120 unique users and in total 6000 questions to compare “Celeba”/“Sample” and ours. On average about 69.07%/68.47% users prefer our method over the “Celeba”/“Sample”.

4) Commercial Service Inversion Results: To further prove our approach indeed threatens real-world face recognition service, we evaluate our black-box inversion approach on



(a) Accuracy on target models.

(b) Top-1/top-5 (dark/light color) accuracy on reference models.

Fig. 14: Black-box inversion effectiveness and generalizability. The higher the better.

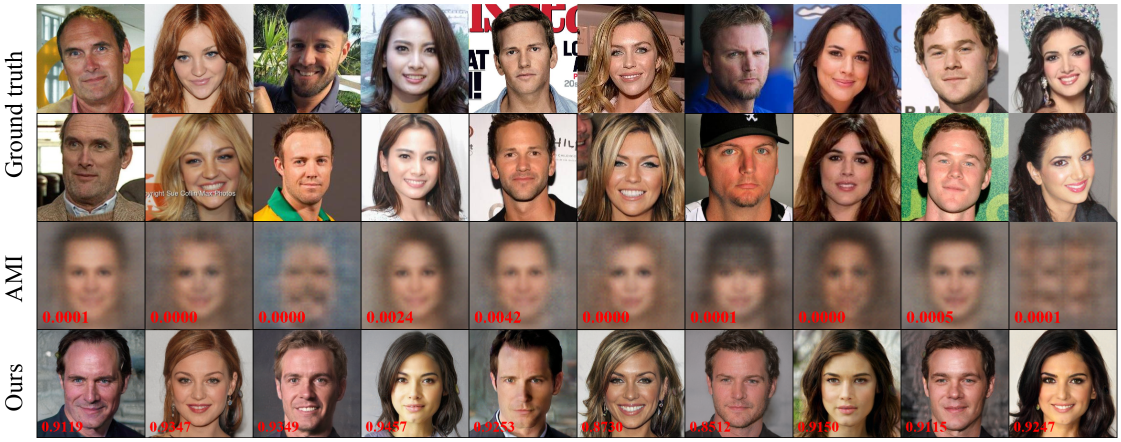


Fig. 15: Images inverted by AMI and ours. Each column corresponds to one person. The first and second rows show the ground truth images. The third displays the images inverted by AMI, while our inverted images are in the fourth row.

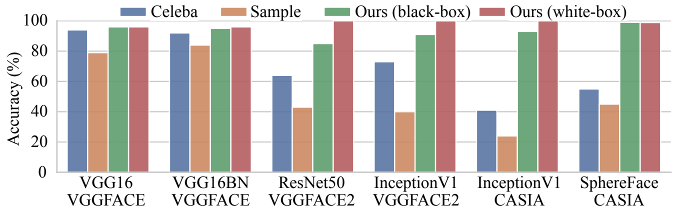


Fig. 16: Inversion effectiveness of sampling/selecting baselines and ours. The higher, the better.

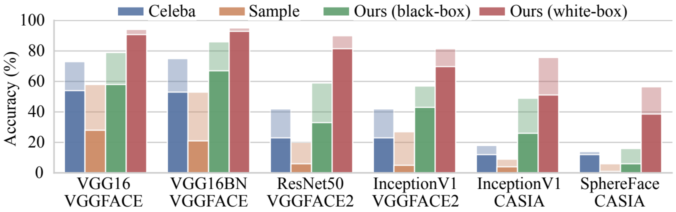


Fig. 17: Inversion generalizability of sampling/selecting baselines and ours. Top-1/top-5 (dark/light color) accuracy. The higher, the better.

the commercial Microsoft Azure Cognitive Services [1]. It provides APIs for users to embed face recognition into their own applications without any requirements on deep learning expertise and thus is widely used. For example, Uber utilizes it to safeguard users against fraud by ensuring a user's selfie is recognized as the person on file [5]. All of our inverted images are predicted as the target people and the top-1/top-5 transferable accuracy is 37.50%/50.00%. Some images are visualized in Figure 18. MIRROR is also evaluated on a model trained on the Clarifi commercial platform [2]. The inversion accuracy and the top-1/top-5 transferable accuracy are 100% and 37.5%/87.5%, respectively.

5) Ablation study: In order to show how different factors impact our approach, we conduct controlled experiments on our approach under various settings. The factors we considered include: 1) GAN's training dataset; 2) the latent space to optimize; 3) the clipping strategy; 4) GAN's architectures; and 5) besides the $\mathcal{Z}$ and $\mathcal{W}$ spaces, we also consider the $\mathcal{Z}+$ and $\mathcal{W}+$ spaces, where separate $\mathbf{z}$ vectors (and respectively $\mathbf{w}$ vectors) are fed to different style blocks. Details of the different factors are explained in online Appendix I [7]. From the NIQE scores in Table VII, we can see all of them are better than the existing state-of-the-art methods and the method with optimization in $\mathcal{W}$ and clipping in $\mathcal{P}$, i.e., w&clip, synthesizes images of the best perceptual quality. Qualitative comparison can be found in Figure 20 in Appendix I [7]. We conduct a human study comparing the default setting, i.e., w&clip with the StyleGAN structure trained on Celeba, with individual other settings to understand their impact. The results are in Table VII. As for the training data, the users prefer the images generated by StyleGAN trained on FFHQ than those generated by StyleGAN trained on Celeba. FFHQ has images of better quality and more diversity so the generated faces may have more natural skin color and better lighting conditions as shown by Figure 21 in Appendix I [7]. However, we chose not to use StyleGAN pre-trained on FFHQ in our major experiments, because FFHQ has only unlabeled images which means we cannot determine the unseen people to invert and therefore the evaluation may be invalid. The remaining results show that the default setting has better human preference rate.

TABLE VI: Feature distances of images inverted by our methods and the sampling/selecting baseline methods (the lower the better). The two numbers for a setting correspond to the ℓ_2 feature distance on the target model and the other model pre-trained on the same dataset.

Method	VGGFace				VGGFace2				CASIA			
	VGG16		VGG16BN		ResNet50		InceptionV1		InceptionV1		SphereFace	
CelebA	105.37	81.27	83.34	106.37	160.71	178.24	159.44	171.90	153.64	14.92	27.33	174.19
Sample	96.27	77.08	77.16	100.05	170.59	200.83	173.82	183.34	153.49	18.89	26.85	174.58
MIRROR (black-box)	90.83	69.05	62.19	85.17	150.25	172.03	130.77	167.86	139.39	10.12	10.09	173.34
MIRROR (white-box)	97.73	55.41	72.06	80.65	149.96	147.97	119.82	163.84	154.41	9.81	8.96	156.19

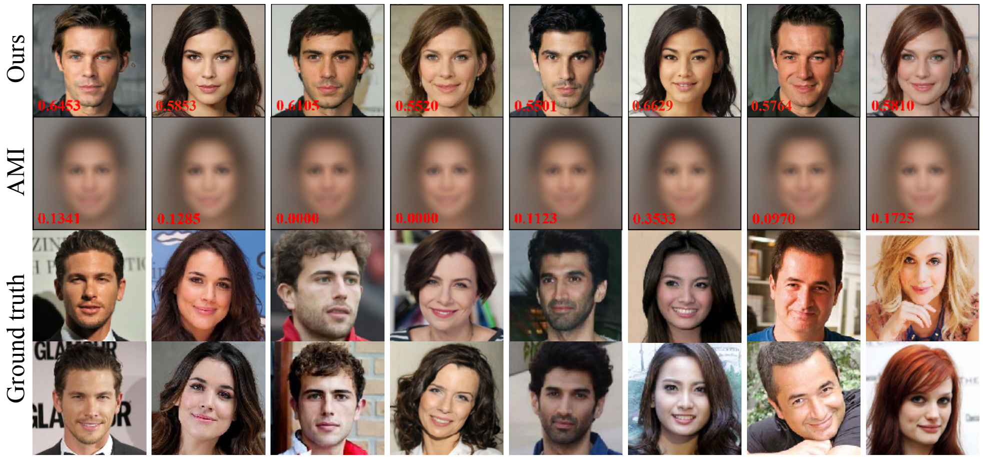


Fig. 18: Inversion results on the commercial Microsoft Azure Cognitive service. The first row shows our inverted images, the second row shows images inverted by AMI, and the last two rows show the ground-truth. Images inverted by AMI are blurry and more like average faces so that they cannot be recognized by humans. Our results give better images and are more similar to the target people.

TABLE VII: Ablation study. We invert 100 randomly selected labels of ResNet50 using our method under various settings and compute the average NIQE (the lower the better). We study the human preference rate between the default setting, i.e., w&clip with StyleGAN on Celeba and individual other settings. For instance, the second row means that 42% prefers results by the default setting over those by using FFHQ. Details of different factors are explained in online Appendix I [7].

Different factors		NIQE ↓	Preference ↑
Training data	CelebA	3.46	–
	FFHQ	4.41	42.0 ±5.7%
Latent space	z	3.77	54.8 ±1.0%
	z&clip	3.87	54.8 ±2.8%
	w	3.80	73.3 ±5.3%
	w&clip	3.46	–
	z+	3.65	55.5 ±1.3%
	z+&clip	3.75	66.0 ±9.6%
	w+	3.99	86.0 ±6.7%
	w+&clip	3.57	78.4 ±7.9%
Architecture	StyleGAN (FFHQ)	4.41	42.0 ±5.7%
	StyleGAN2 (FFHQ)	4.78	54.4 ±0.7%

6) Effects of Dropout and Consistency based Result Selection: We randomly select 50 labels of VGG16 pre-trained on VGGFace to conduct model inversion with and without dropout and consistency based result selection. The accuracy and NIQE are similar. The feature distances on the subject model and the reference model are 98.12/55.90 with the enhancements, and 107.12/65.08 without them.

D. Results on Other Data Domains

To demonstrate the capabilities of MIRROR, we use a StyleGAN trained on art-style face portraits [11] to invert a ResNet50 model. The results can be found in Figure 28 in the online Appendix [7]. They show that using data from a similar (but different) domain can produce good inversion results. We also show MIRROR can perform high quality inversion for models trained on data from other domains, such as cars and cats. Due to the limited space, the results are presented in the online Appendix IX [7].

E. Results on Various Defensive Methods

Besides the original ResNet50 (R_{org}), we attack a robust ResNet50 (R_{adv}) adversarially trained by [60] and a defensive ResNet50 (R_{mid}) by MID [59]. MID tries to regularize the dependency between the input image and the prediction of a network to defend it against the inversion attack. For each of R_{org}, R_{adv} and R_{mid}, we attack it and use the other two as the reference models. The accuracy and the transferable accuracy are both 100%. When using an InceptionV1 as the reference model, the transferable accuracy of attacking R_{org}/R_{adv}/R_{mid} is 74%/76%/73%. Adversarial training cannot make model inversion harder as expected, because it helps model learn robust features against adversarial perturbation. MID was able to defend GMI well (inversion accuracy degraded from 31% to 18% [59]) but not MIRROR (having the same 100% inversion accuracy with and without the defense) likely because MIRROR is a much stronger attack. Some inversion results

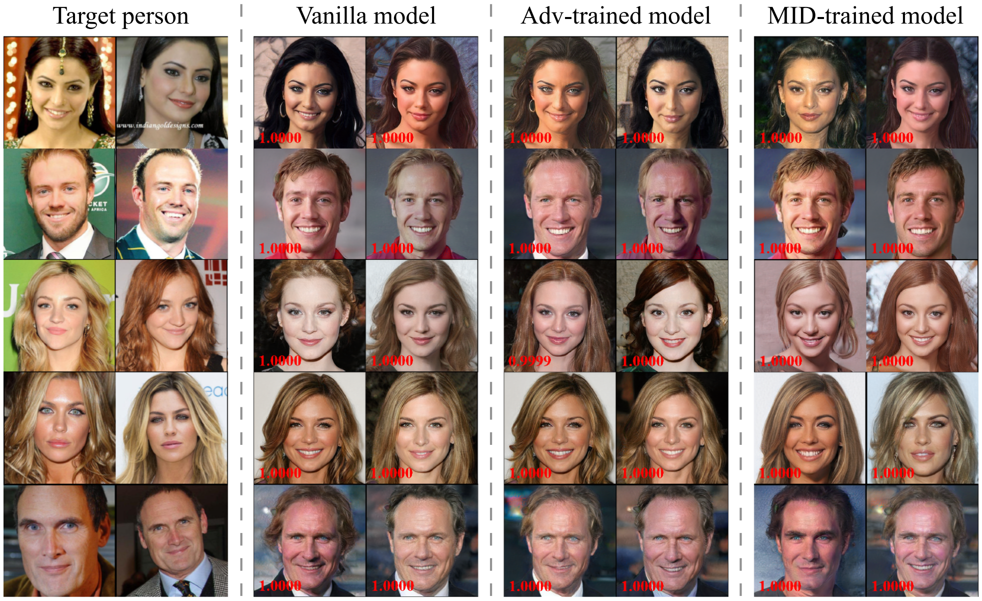


Fig. 19: Inverted images on models trained with different strategies.

are shown by Figure 19. We also evaluate our method against a VGG16 model trained by Differentially Private Stochastic Gradient Descent (DP-SGD) [8], [70], which adds noises to gradients during training to defend membership inference attacks. The inversion accuracy is 100% and the top-1/top-5 transferable inversion accuracy on a normally trained VGG16 and a ResNet18 are 70%/100% and 100%/100%. While DP-SGD can ensure certain privacy for training samples, it cannot defend model inversion attacks. Because different from membership inference attacks where attackers check if a sample is used to train a model, model inversion's goal is to produce a representative image for a target label.

VI. Related Work

Machine Learning Privacy. Researchers have studied various privacy issues in machine learning [48]. Shokri *et al.* [56] proposed membership inference attacks which aim to infer whether a given data point belongs to the training set. Such attacks are followed up by several works [39], [66], [54], [18]. Song *et al.* [58] built an alternative model that performs as accurate as the original model while memorizes information about the training set. Following [13], Ganju *et al.* [24] proposed property inference attacks that extract data properties of the training data (*e.g.*, the ratio of minorities). Salem *et al.* [53] considered the online learning scenario and proposed to infer the data used to update a trained model. There are also recent works that studied information leakage problem in collaborative learning [27], [29], [41], [68], where the training or inference tasks are distributed to multiple participants. For example, GradInversion [68] is a method based on DeepInversion to recover private training data used by clients. It requires the gradient from a set of private training images w.r.t the model's weights during training which is not available in model inversion. Also, it can only work for models with BatchNorm layers in the white-box setting. Vec2Face [20] focuses on recovering an image from its feature embedding. It is not applicable when only the label information is available in model inversion. Different from the above, our focus is on the model inversion attack that aims to reconstruct representative data points for target identities.

Model Inversion Attack. Fredrikson first introduced such attack in *et al.* [23], [22], whose goal includes inferring the private user attributes from a linear regression model, recovering a person's face image from shallow neural networks, etc. Later, Mahendran and Vedaldi [40] put special focus on inverting images and proposed to add natural image priors, and Wu *et al.* [61] further provided formalization for model inversion attacks. Overall, these earlier works mainly targeted at shallow models and the inverted images do not have high fidelity. Recently, several state-of-the-art model inversion methods have been proposed [65], [69], [73]. For example, by querying the target model with an auxiliary dataset, AMI [65] trains an extra generative model to map truncated predictions to grayscale images. DeepInversion [69] extends the optimization framework in DeepDream [44] by incorporating an additional regularization term to constrain feature similarities over batch normalization layers. GMI [73] proposed to use GAN for inversion. It optimizes over the latent space and could generate more natural images. Following [73], Chen *et al.* [19] further proposed to involve the target model into GAN. It inherits limitations of GMI. We have done thorough comparison with these techniques and MIRROR outperforms.

GANs and Image Inversion. Since the first GAN [25], lots of GANs have been proposed [52], [12], [47]. Recently, a series of improved GANs [32], [34], [35] was devised to generate high fidelity natural images, including StyleGANs [34], [35]. StyleGANs are widely used in image super-resolution [42], image edition [9], [10], [62], [63]. A common technique used in these image applications is called image inversion. Different from model inversion, image inversion aims to find the latent vector which can precisely reconstruct a given image. It has much stronger constraints compared to model inversion and existing image inversion methods work only in the white-box setting. In contrast, MIRROR takes a label and generates images for that label and works for both white-box and black-box settings. For example, [63] is a StyleGAN based image inversion method. It adds the co-variance matrix and the mean of the Gaussian distribution as an energy term which is actually the Mahalanobis Distance between the latent vector and the mean, whereas MIRROR truncates latent vectors. While its optimization method can be adapted for mode inversion, our experiments show our optimization outperforms [63]'s in the context of model inversion by 25.5%. Please see details in online Appendix V [7].

VII. Conclusion

We study the challenges in GAN based model inversion. We develop a new inversion technique MIRROR based on StyleGAN. It leverages StyleGAN's capabilities of disentangling the latent space. It supports both white-box and black-box inversions, and regulates the image quality in an auxiliary space of StyleGAN with a multi-variate Gaussian distribution. Our results show that the samples inverted by MIRROR have substantially better quality and fidelity compared to the state-of-the-art methods.

Acknowledgment

We thank the anonymous reviewers for their constructive comments. This research was supported, in part by NSF 1901242 and 1910300, ONR N000141712045, N000141410468 and N000141712947, IARPA TrojAI W911NF-19-S-0012, DARPA VSPELLS (under subcontract 1119482 with BAE Systems), and the National Natural Science Foundation of China 62172199. Yuan Yao and Jingwei Xu are partially supported by the Collaborative Innovation Center of Novel Software Technology and Industrialization. We thank Clarifai (especially Ian Kelk and Jérémy Faret) for providing a free account to run experiments. Any opinions, findings, and conclusions in this paper are those of the authors only and do not necessarily reflect the views of our sponsors.

References

[1] "Azure cognitive services," https://azure.microsoft.com/en-us/services/cognitive-services/.

[2] "Clarifai," https://www.clarifai.com/.

[3] "Inception resnet (v1) models in pytorch," https://github.com/timesler/facenet-pytorch.

[4] "Sphereface model in pytorch," https://github.com/clcarwin/sphereface_pytorch.

[5] "Uber with azure cognitive services," https://customers.microsoft.com/en-us/story/731196-uber.

[6] "Vggface/vggface2 models in pytorch," https://www.robots.ox.ac.uk/~albanie/pytorch-models.html.

[7] "Our system MIRROR," https://model-inversion.github.io/mirror/, 2021.

[8] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 308–318.

[9] R. Abdal, Y. Qin, and P. Wonka, "Image2stylegan: How to embed images into the stylegan latent space?" in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 4432–4441.

[10] R. Abdal, Y. Qin, and P. Wonka, "Image2stylegan++: How to edit the embedded images?" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 8296–8305.

[11] ak9250, "Stylegan art," https://github.com/ak9250/stylegan-art, 2019.

[12] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017.

[13] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers," *International Journal of Security and Networks*, vol. 10, no. 3, pp. 137–150, 2015.

[14] I. Bárány and V. Vu, "Central limit theorems for gaussian polytopes," *The Annals of Probability*, vol. 35, no. 4, pp. 1593–1621, 2007.

[15] D. Bhandari, C. Murthy, and S. K. Pal, "Genetic algorithm with elitist model and its convergence," *International journal of pattern recognition and artificial intelligence*, vol. 10, no. 06, pp. 731–747, 1996.

[16] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[17] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: A dataset for recognising faces across pose and age," in *2018 13th IEEE international conference on automatic face & gesture recognition (FG)*. IEEE, 2018, pp. 67–74.

[18] D. Chen, N. Yu, Y. Zhang, and M. Fritz, "Gan-leaks: A taxonomy of membership inference attacks against generative models," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2020, pp. 343–362.

[19] S. Chen, R. Jia, and G.-J. Qi, "Improved techniques for model inversion attack," *arXiv preprint arXiv:2010.04092*, 2020.

[20] C. N. Duong, T.-D. Truong, K. Luu, K. G. Quach, H. Bui, and K. Roy, "Vec2face: Unveil human faces from their blackbox features in face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 6132–6141.

[21] G. Franchi, A. Bursuc, E. Aldea, S. Dubuisson, and I. Bloch, "Tradi: Tracking deep neural network weight distributions," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 105–121.

[22] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2015, pp. 1322–1333.

[23] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *23rd {USENIX} Security Symposium ({USENIX} Security)*, 2014, pp. 17–32.

[24] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov, "Property inference attacks on fully connected neural networks using permutation invariant representations," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2018, pp. 619–633.

[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[27] Z. He, T. Zhang, and R. B. Lee, "Model inversion attacks against collaborative inference," in *Proceedings of the 35th Annual Computer Security Applications Conference (ACSAC)*, 2019, pp. 148–162.

[28] S. Hidano, T. Murakami, S. Katsumata, S. Kiyomoto, and G. Hanaoka, "Model inversion attacks for prediction systems: Without knowledge of non-sensitive attributes," in *2017 15th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2017, pp. 115–124.

[29] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017, pp. 603–618.

[30] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 1501–1510.

[31] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning (ICML)*. PMLR, 2015, pp. 448–456.

[32] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing

of gans for improved quality, stability, and variation," in *International Conference on Learning Representations (ICLR)*, 2018.

[33] T. Karras, S. Laine, and T. Aila, "Nvidia stylegan," https://github.com/NVlabs/stylegan, 2019.

[34] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4401–4410.

[35] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 8110–8119.

[36] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," in *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[37] S. Li, J. Li, H. Tang, R. Qian, and W. Lin, "Atrw: A benchmark for amur tiger re-identification in the wild," in *Proceedings of the 28th ACM International Conference on Multimedia (MM)*, 2020, pp. 2590–2598.

[38] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[39] Y. Long, V. Bindschaedler, L. Wang, D. Bu, X. Wang, H. Tang, C. A. Gunter, and K. Chen, "Understanding membership inferences on well-generalized learning models," *arXiv preprint arXiv:1802.04889*, 2018.

[40] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2015, pp. 5188–5196.

[41] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 691–706.

[42] S. Menon, A. Damian, S. Hu, N. Ravi, and C. Rudin, "Pulse: Self-supervised photo upsampling via latent space exploration of generative models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2437–2445.

[43] A. Mittal, R. Soundararajan, and A. C. Bovik, "Making a "completely blind" image quality analyzer," *IEEE Signal Processing Letters*, vol. 20, no. 3, pp. 209–212, 2012.

[44] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going deeper into neural networks," 2015.

[45] NIST/SEMATECH, "e-handbook of statistical methods," https://www.itl.nist.gov/div898/handbook/eda/section3/qqplot.htm, 2012.

[46] NVIDIA, "Resnet50 v1.5 for pytorch," https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/Classification/ConvNets/resnet50v1.5.

[47] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," in *International Conference on Machine Learning (ICML)*. PMLR, 2017, pp. 2642–2651.

[48] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman, "Sok: Security and privacy in machine learning," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2018, pp. 399–414.

[49] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," 2015.

[50] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, "Cats and dogs," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[51] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32 (NeurIPS)*. Curran Associates, Inc., 2019, pp. 8024–8035.

[52] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015.

[53] A. Salem, A. Bhattacharya, M. Backes, M. Fritz, and Y. Zhang, "Updates-leak: Data set inference and reconstruction attacks in online learning," in *29th {USENIX} Security Symposium ({USENIX} Security)*, 2020, pp. 1291–1308.

[54] A. Salem, Y. Zhang, M. Humbert, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," in *Network and Distributed Systems Security Symposium (NDSS)*. Internet Society, 2019.

[55] Y. Shen, Y. Xu, C. Yang, J. Zhu, and B. Zhou, "Genforce," https://github.com/genforce/genforce, 2020.

[56] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 3–18.

[57] J. Sirignano and K. Spiliopoulos, "Mean field analysis of neural networks: A central limit theorem," *Stochastic Processes and their Applications*, vol. 130, no. 3, pp. 1820–1852, 2020.

[58] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017, pp. 587–601.

[59] T. Wang, Y. Zhang, and R. Jia, "Improving robustness to model inversion attacks via mutual information regularization," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 35, no. 13, 2021, pp. 11 666–11 673.

[60] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," in *International Conference on Learning Representations (ICLR)*, 2020.

[61] X. Wu, M. Fredrikson, S. Jha, and J. F. Naughton, "A methodology for formalizing model-inversion attacks," in *IEEE 29th Computer Security Foundations Symposium (CSF)*. IEEE, 2016, pp. 355–370.

[62] Z. Wu, D. Lischinski, and E. Shechtman, "Stylespace analysis: Disentangled controls for stylegan image generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 12 863–12 872.

[63] J. Wulff and A. Torralba, "Improving inversion and generation diversity in stylegan using a gaussianized latent space," *arXiv preprint arXiv:2009.06529*, 2020.

[64] Q. Xu, G. Tao, and X. Zhang, "D-square-b: Deep distribution bound for natural-looking adversarial attack," 2020.

[65] Z. Yang, J. Zhang, E.-C. Chang, and Z. Liang, "Neural network inversion in adversarial setting via background knowledge alignment," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2019, pp. 225–240.

[66] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*. IEEE, 2018, pp. 268–282.

[67] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," *arXiv preprint arXiv:1411.7923*, 2014.

[68] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, "See through gradients: Image batch recovery via gradinversion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 16 337–16 346.

[69] H. Yin, P. Molchanov, J. M. Alvarez, Z. Li, A. Mallya, D. Hoiem, N. K. Jha, and J. Kautz, "Dreaming to distill: Data-free knowledge transfer via deepinversion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 8715–8724.

[70] A. Yousefpour, I. Shilov, A. Sablayrolles, D. Testuggine, K. Prasad, M. Malek, J. Nguyen, S. Ghosh, A. Bharadwaj, J. Zhao, G. Cormode, and I. Mironov, "Opacus: User-friendly differential privacy library in pytorch," *arXiv preprint arXiv:2109.12298*, 2021.

[71] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015.

[72] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *European Conference on Computer Vision*. Springer, 2016, pp. 649–666.

[73] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song, "The secret revealer: Generative model-inversion attacks against deep neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 253–261.