Variational Model Inversion Attacks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Given the ubiquity of deep neural networks, it is important that these models do not reveal information about sensitive data that they have been trained on. In model inversion attacks, a malicious user attempts to recover the private dataset used to train a supervised neural network. A successful model inversion attack should generate realistic and diverse samples that accurately describe each of the classes in the private dataset. In this work, we provide a probabilistic interpretation of model inversion attacks, and formulate a variational objective that accounts for both diversity and accuracy. In order to optimize this variational objective, we choose a variational family defined in the code space of a deep generative model, trained on a public auxiliary dataset that shares some structural similarity with the target dataset. Empirically, our method substantially improves performance in terms of target attack accuracy, sample realism, and diversity on datasets of faces and chest X-ray images.

1 Introduction

Thanks to recent advances, deep neural networks are now widely used in applications including facial recognition [41], intelligent automated assistants, and personalized medicine [9, 46]. Powerful deep neural networks are trained on large datasets that could contain sensitive information about individuals. Publishers release only the trained models (e.g., on TensorFlow Hub [54]), but not the original dataset to protect the privacy of individuals whose information is in the training set. Regrettably, the published model can leak information about the original training set [18, 38, 48]. Attacks that exploit such properties of machine learning models are privacy attacks [44]. Other real-world scenarios where an attacker can access models trained on private datasets include interaction with APIs (e.g., Amazon Rekognition [2]), or in the growing paradigm of federated learning [5, 58] where a malicious insider can try to steal data from other data centers [23]. Kaissis et al. [26] provides an overview of how these privacy concerns are hindering the widespread application of deep learning in the medical domain.

This paper focuses on the model inversion (MI) attack, a type of privacy attack that tries to recover the training set given access only to a trained classifier [6, 15, 16, 22, 29, 59, 62]. The classifier under attack is referred to as the "target classifier" (see Figure 1). An important relevant application concerns identity recognition from face images, since facial recognition is a common approach to biometric identification [17, 43]. Once an attacker successfully reconstructs images of the faces of individuals in the private training set, they can use the stolen identity to break into otherwise secure systems.

Early studies of MI attacks focused on attacking classifiers whose inputs are tabular data [15, 16, 22] by directly performing inference in the input space. However, the same methods fail when the target classifier processes high-dimensional inputs, e.g., in the above example of a face recognizer. Recent methods cope with this high-dimensional search space by performing the attack in the latent space of a neural network generator [29, 59, 62]. Introducing a generator works well in practice, but this

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

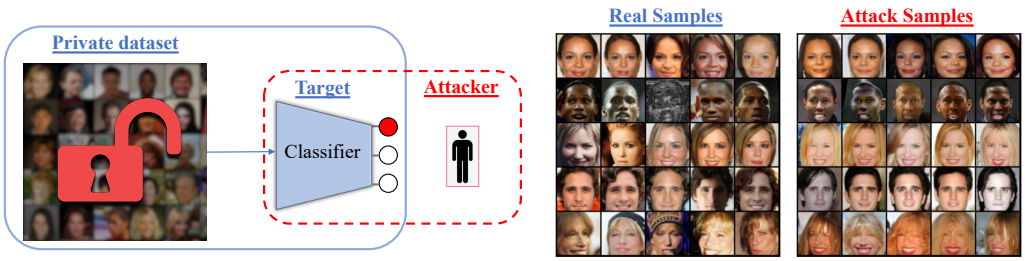

Figure 1: In model inversion attacks, the attacker tries to recover the original dataset used to train a classifier with access to only the trained classifier. In the two figures on the right, each row corresponds to a private identity – an individual whose images are not available to the attacker. The first five identities (not cherry-picked) were visualized. On the left are real data samples, and on the right are generated samples from our proposed method.

success is not theoretically understood. In this paper, we frame MI attacks as a variational inference (VI) [25, 4, 30] problem, and derive a practical objective. This view allowed us to justify existing approaches based on a deep generative model, and identify missing ingredients in them. Empirically, we find that our proposed objective leads to significant improvements: more accurate and diverse attack images, which are quantified using standard sample quality metrics (i.e., FID [21], precision, and recall [32, 37]).

In summary, our contributions and findings are:

- We view the model inversion (MI) attack problem as a variational inference (VI) problem. This view allows us to derive a practical objective based on a statistical divergence, and provides a unified framework for analyzing existing methods.
- We provide an implementation of our framework using a mixture of deep flow [12, 13, 31] models in the extended latent space of a StyleGAN [28]. This implementation can leverage the hierarchy of learned representations, and perform targeted attacks while preserving diversity.
- Empirically, on the CelebA [35], and ChestX-ray [56] datasets , our method results in higher attack accuracy, and more diverse generation compared to existing methods. We provide thorough comparative experiments to understand which components contribute to improved target accuracy, sample realism, and sample diversity, and detail the benefits of the VI perspective.

Code to reproduce the reported results is available in the supplementary materials, and will be released publicly after the anonymous phase.

2 Task Setup: Model Inversion Attack

In the problem of model inversion (MI) attack, the attacker has access to a "target classifier":

$$\overline{p}^{\text{tar}}(y|\boldsymbol{x}) : \mathbb{R}^d \to \Delta^{C-1}$$

where Δ^{C-1} to denote the (C-1)-simplex, representing the C-dimensional probability. This target classifier is trained on the private target dataset, $\mathcal{D}^{\text{tar}} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{N^{\text{tar}}}$ where $\boldsymbol{x} \in \mathbb{R}^d$ and $y \in [1, 2, ..., C]$. We use $\overline{p}^{\text{tar}}(y|\boldsymbol{x})$ to denote the given target classifier, which is an approximation of the true conditional probability $p^{\text{tar}}(y|\boldsymbol{x})$ of the underlying data generating distribution.

Goal: Given a target classifier $\overline{p}^{\text{tar}}(y|\boldsymbol{x})$, we wish to approximate the class conditional distribution $p^{\text{tar}}(\boldsymbol{x}|y)$ without having access to the private training set $\mathcal{D}^{\text{tar}}$.

A good model inversion attack should approximate $p^{\text{tar}}(\boldsymbol{x}|y) \propto p^{\text{tar}}(y|\boldsymbol{x})p^{\text{tar}}(\boldsymbol{x})$ well, and allow the attacker to generate realistic, accurate, and diverse samples. Let's consider a scenario where the attacker intends to fake a victim's identity. A good security system might test the realism of a sequence

of inputs [24]. More generally, faking someone's identity requires passing a two-sample test [24, 34], which means the ability to generate samples from $p^{\text{tar}}(\boldsymbol{x}|y)$ is necessary. Another scenario can be an attacker trying to steal private/proprietary information. For example, a malicious insider in a federated learning setting can try to steal information from another datacenter by accessing the shared model [23]. In such a scenario, a good $p^{\text{tar}}(\boldsymbol{x}|y)$ model can reveal not only the distinguishing features of the data, but also its variations.

3 Related Work

In this section, we first discuss related model inversion attack studies, and then discuss applications of inverting a classifier outside of the privacy attack setting. Lastly, since our attack method relies on using a pretrained generator, we discuss existing studies that used a pretrained generator for improving other applications.

Model Inversion Attacks. MI attacks are one type of privacy attack where the attacker tries to reconstruct the training examples given access only to a target classifier. Other privacy attacks include membership attacks, attribute inference, and model extraction attacks (see Rigaki and Garcia [44] for a general discussion). Fredrikson et al. [16, 15] were the first to study the MI attack and demonstrated successful attacks on low-capacity models (e.g., logistic regression, and a shallow MLP network) when partial input information was available to the attacker. The setting where partial input information was not available was studied by Hidano et al. [22]. They termed performing gradient ascent w.r.t. the target classifier in the input space a *General Model Inversion* attack. Though effective on tabular data, this approach failed on deep image classifiers. It is well-known that directly performing gradient ascent in the image space results in imperceptible and unnatural changes [52].

To handle high-dimensional observations such as images, MI attacks based on deep generators were proposed [62, 6, 59]. By training a deep generator on an *auxiliary dataset*, these methods effectively reduce the search space to only the manifold of relevant images. For example, if the target classifier is a facial recognition system, then an auxiliary dataset of generic faces can be easily obtained. Zhang et al. [62] pretrain a Generative Adversarial Network (GAN) [19] on the auxiliary dataset and performed gradient-based attack in the latent space of the generator. Yang et al. [59] trained a generator that inverts the prediction of the target classifier, using the architecture of an autoencoder. Our work is the first to formalize the assumption of a pretrained generator. In Section 4 we show that a practical objective can be derived from the variational inference perspective using this assumption, and use this as a framework to analyze existing methods.

Other Applications of Classifier Inversion. Inverting the predictions and activations of a classifier has been studied for applications such as model explanation, and model distillation Motivated by using saliency maps to explain the decision of a classifier [50], Mahendran and Vedaldi [36] proposed to maximize the activation of a target neuron in a classifier for synthesizing images that best explain that neuron. Their method is based on gradient ascent in the pixel-space. The same idea was recently extended to enable knowledge distillation in a setting where the original dataset was too large to be transferred [61]. Methods above have the advantage of not requiring the original dataset, but the generated images still appear unnatural. One effective way to improve the realism of the generated images is to optimize in the latent code space of a deep generator [39, 40].

Pretrained Generators. Pretrained generators from GANs have other applications. Image inpainting/restoration was one of the early successful applications of using a pretrained GAN as image prior. Yeh et al. [60] optimized the latent code of GANs to minimize the reconstruction error of a partially occluded image for realistic inpainting. The same method formed the basis of AnoGAN, a popular technique for anomaly detection in medical images [45]. A combination of reconstruction error and discriminator loss was used as the anomaly score. A potential future direction is to apply our proposed method to these adjacent applications.

4 Model Inversion Attack as Variational Inference

For each class label y, we wish to find a $q(\boldsymbol{x})$ such that it is close to $p^{\text{tar}}(\boldsymbol{x}|y)$, the true target posterior distribution. This can be achieved by minimizing the KL-divergence from p to q, $D_{\text{KL}}(q||p)$, as

follows:

$$
\begin{aligned}
q^* &= \arg\min_q \mathcal{L}(q), \\
\mathcal{L}(q) &= D_{\mathrm{KL}}(q(\boldsymbol{x})||p^{\text{tar}}(\boldsymbol{x}|y)) \\
&= \log p(y^{\text{tar}}) + \mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x})}[-\log p^{\text{tar}}(y|\boldsymbol{x})] + D_{\mathrm{KL}}(q(\boldsymbol{x})||p^{\text{tar}}(\boldsymbol{x})), \\
\mathcal{J}(q) &= \mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x})}[-\log p^{\text{tar}}(y|\boldsymbol{x})] + D_{\mathrm{KL}}(q(\boldsymbol{x})||p^{\text{tar}}(\boldsymbol{x}))
\end{aligned} \tag{1}
$$

The derivation of $\mathcal{L}(q)$ can be found in Appendix A.1. Notice the first term in $\mathcal{L}(q)$ is independent of q, so it can be dropped to obtain $\mathcal{J}(q)$ while keeping the minimizer the same, i.e. $q^* = \arg\min_q \mathcal{J}(q)$. Since $\mathcal{J}(q)$ is derived from a statistical divergence, it implies that $q^*(\boldsymbol{x}) = p^{\text{tar}}(\boldsymbol{x}|y)$. Hence, optimizing $\mathcal{J}(q)$ indeed results in our desired solution.

In order to minimize $\mathcal{J}(q)$, we need to compute $D_{\mathrm{KL}}(q(\boldsymbol{x})||p^{\text{tar}}(\boldsymbol{x}))$, which requires the true image prior $p^{\text{tar}}(\boldsymbol{x})$ that is not accessible. Similar to recent works on MI attacks [59, 62, 29], we assume that a public auxiliary dataset, $\mathcal{D}^{\text{aux}}$, which shares some structural similarity with the target dataset, is available and can be used to learn a relevant image prior. We parametrize this image prior $p^{\text{aux}}(\boldsymbol{x})$ by a deep latent variable model, $p^{\text{aux}}(\boldsymbol{x}) = \int_{\boldsymbol{z}} p^{\text{aux}}(\boldsymbol{z}) p_G(\boldsymbol{x}|\boldsymbol{z}) d\boldsymbol{z}$ with a generator $G : \boldsymbol{z} \to \boldsymbol{x}$ that is trained as a generative adversarial network (GAN) on the auxiliary dataset, and $p^{\text{aux}}(\boldsymbol{z}) = \mathcal{N}(0, I)$ which is the probability density function of a Gaussian. The learned generator defines a low dimensional manifold in the observation space $\boldsymbol{x}$, when $\boldsymbol{z}$ is chosen to have fewer dimensions than $\boldsymbol{x}$. We further limit the search space of $q(\boldsymbol{x})$, to all distributions that lie on this manifold by assuming that $q(\boldsymbol{x})$ is of the form $q(\boldsymbol{x}) = \int_{\boldsymbol{z}} q(\boldsymbol{z}) p_G(\boldsymbol{x}|\boldsymbol{z}) d\boldsymbol{z}$. We define our assumption as follows:

Assumption 1 (Common Generator) *There exists a common generator $G : \boldsymbol{z} \to \boldsymbol{x}$ that captures a shared conditional distribution, i.e. $p_G(\boldsymbol{x}|\boldsymbol{z}) = p^{tar}(\boldsymbol{x}|\boldsymbol{z}) = q(\boldsymbol{x}|\boldsymbol{z})$. Further, we assume the parameters of G can be learned from the auxiliary dataset $\mathcal{D}^{aux}$.*

The name Common Generator is inspired by recent progress in theoretical analyses of few-shot and transfer learning, where assumptions about an underlying shared representation have been used to provide error bounds; this representation is referred to as the "common" representation [14, 55].

Given the common generator G, the variational distribution can be rewritten as $q(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}[q(\boldsymbol{x}|\boldsymbol{z})] = \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}[p_G(\boldsymbol{x}|\boldsymbol{z})]$. The problem of optimizing Equation 1 over $q(\boldsymbol{x})$ distributions can be reduced to optimization over $q(\boldsymbol{z})$. Under Assumption 1, $\mathcal{J}(q)$ can be rewritten as:

$$
\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}[-\log p^{\text{tar}}(y|G(\boldsymbol{z}))] + D_{\mathrm{KL}}(q(\boldsymbol{z})||p^{\text{tar}}(\boldsymbol{z})) \geq \mathcal{J}(q) \tag{2}
$$

which is an upper bound of the ideal objective. Thanks to Assumption 1, the KL-divergence is now computed in the latent code space. Because the target distribution is unknown, we replace $p^{\text{tar}}(\boldsymbol{z})$ with $p^{\text{aux}}(\boldsymbol{z})$, arriving at an objective that can be optimized in practice. While this is not always an upper bound for every $q(\boldsymbol{z})$, empirically this KL-divergence term still led to good attack results as shown in Section 5. Intuitively, since our attack optimizes $q(\boldsymbol{z})$ to approach $p^{\text{tar}}(\boldsymbol{z})$, samples from $q(\boldsymbol{z})$ are likely assigned higher probability by $p^{\text{tar}}(\boldsymbol{z})$ than by $p^{\text{aux}}(\boldsymbol{z})$. When this holds, this objective remains an upper bound of $\mathcal{J}(q)$. Detailed derivations and discussion can be found in Appendix A.2.

We make one additional modification to handle classifier miscalibration. Miscalibration is a known problem in neural network classifiers [20]; namely, the classifier predictions can be over- or under-confident. We adjust for miscalibration by letting the given "target classifier" $\overline{p}^{\text{tar}}(y|\boldsymbol{x})$ be equal to the true target conditional $p^{\text{tar}}(y|\boldsymbol{x})$ up to a temperature scaling term γ:

$$
p^{\text{tar}}(y|\boldsymbol{x}) = \frac{1}{Z}\exp(\frac{1}{\gamma}\log \overline{p}^{\text{tar}}(y|\boldsymbol{x})), \quad Z = \sum_y \exp(\frac{1}{\gamma}\log \overline{p}^{\text{tar}}(y|\boldsymbol{x}))
$$

Plugging in these adjustments to Equation 2, and multiplying by γ, we obtain our VMI learning objective $\mathcal{J}^{CG}$ where CG stands for "common generator":

$$
\mathcal{J}^{CG}(q) = \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}[-\log \overline{p}^{\text{tar}}(y|G(\boldsymbol{z}))] + \gamma D_{\mathrm{KL}}(q(\boldsymbol{z})||p^{\text{aux}}(\boldsymbol{z})) \tag{3}
$$

The first term in Equation 3 can be computed given both the target classifier and the common generator. The second KL term can be computed when $q(z)$ is a prescribed density model (see Section 4.2). We study the effect of γ in Section 5.

Figure 2: Learned features at different layers of a generator learned with StyleGAN [27]. Not all factors of variation are relevant to MI attacks. In face identity attacks, factors governing facial features should be modified (layers 0,1,2 and 3,4,5,6), but not low-level image statistics like lighting and overall shades (layers 7,8,9). Given the learned layer-wise decomposition,our VMI attack can directly change latent codes of the intermediate layers (3,4,5,6), and manipulate features relevant to an individual's identity. This is not facilitated by the generator of a DCGAN [42], as found in existing MI attack methods [62, 6] Our experimental results (Figure 7) show that the VMI attack discovers the right layers to optimize, aligning with the qualitative observations here.

4.1 The Common Generator

For the common generator G we use a GAN due to their ability to synthesize high quality samples [1, 3, 27, 47]. The generator is trained on the "auxiliary" dataset. A common choice is the DCGAN architecture [42]. Though optimizing in the latent space results in more realistic samples, this search space is also more restrictive. Early investigation showed that optimizing at some intermediate layers of a pretrained generator was better than either optimizing in the code space, or the pixel space. This motivated the use of a generator architecture where activations of intermediate layers can be manipulated.

StyleGAN. A framework that naturally allows for optimization in the intermediate layers is the StyleGAN [27]. It consists of a mapping network $f : \boldsymbol{z} \rightarrow \boldsymbol{w}$, and a synthesis network S. The synthesis network takes as input one $\boldsymbol{w}$ at each layer, $S : \{\boldsymbol{w}_l\}_{l=1}^{L} \rightarrow \boldsymbol{x}$, where L is the number of layers in S. During training of a StyleGAN, the output from the mapping network is copied L times, $\{\boldsymbol{w}\}_{l=1}^{L} = \text{COPY}(\boldsymbol{w}, L)$ before being fed to the synthesis network. This input space of the synthesis network S has been referred to as the "extended W space" [1]. The full generator is $G_{\text{style}}(\boldsymbol{z}) = S(\text{COPY}(f(\boldsymbol{z}), L)$.

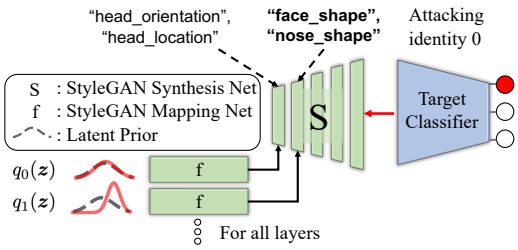

Figure 3: Utilizing the StyleGAN architecture, our VMI attack tends to focus on layers whose representation are relevant for the attack, such as the layer for "face shape".

StyleGAN is known for its surprising ability to perform "style mixing". Namely, the synthesis network can generate a "mixed" image when being fed a mixture of two w's in the extended W space, i.e. $S([\{w_1\}_{l=1}^{L_0}; \{w_2\}_{l=L_0+1}^{L}])$ where $[\cdot;\cdot]$ denotes concatenation, and L_0 is some intermediate layer. Figure 2 visualizes the effect of manipulating different layers of the synthesis network. Performing an attack in this extended W space allows us to achieve the desired effect of optimizing the intermediate layers. Because of the lack of a explicit prior for w, we instead optimize the extended Z space, i.e. using a different latent code z for each layer. Using a separate approximate distribution at each layer $q_l(\boldsymbol{z})$ for $l = 1...L$, our formulation becomes:

$$\mathcal{J}^{CGS}(q) = \mathbb{E}_{\{\boldsymbol{z}_l \sim q_l(\boldsymbol{z})\}_{l=0}^{L}}[-\log \overline{p}^{\text{tar}}(y|S(\{f(\boldsymbol{z}_l)\}_{l=1}^{L}))] + \frac{\gamma}{L}\sum_{l=1}^{L} D_{\text{KL}}(q_l(\boldsymbol{z})||p^{\text{aux}}(\boldsymbol{z})) \quad (4)$$

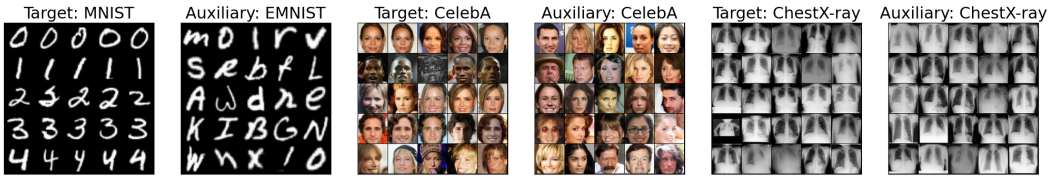

Figure 4: Visualization of real samples for all 3 tasks: MNIST, CelebA, and ChestX-ray. For the target datasets, each row corresponds to one class/identity. Best viewed zoomed in.

4.2 Model Class of $q(\boldsymbol{z})$

In the VMI formulations (Equation 3 and 4), the approximate distribution $q(\boldsymbol{z})$ can be seen as a generator, or "miner" network [57] for the latent space of the common generator. It must belong to a model class whose distribution can be sampled from, and can be evaluated in order to optimize the KL-divergence. In this work, we experiment with a powerful class of tractable generative models: deep flow models. When optimizing $\mathcal{J}^{CGS}(q)$ (Equation 4), we use L such models, one for each layer in the synthesis network. In our experiments, we used an architecture similar to that of Glow [31] where we treated the latent vector as 1x1 images. We removed the original squeezing layers that reduced the image size to account for this change. We also experimented with using a Gaussian distribution for $q(\boldsymbol{z})$.

4.3 Relationships with existing methods

VMI provides a unified framework that encompasses existing attack methods. Through the lens of VMI, we can obtain insights on the conceptual differences between the existing methods and their pros and cons. "General MI" attack [22] performs optimization in the observation space directly as follows:

$$\boldsymbol{x}' = \arg\max_{\boldsymbol{x}} \overline{p}^{\text{tar}}(y|\boldsymbol{x}) \tag{5}$$

which corresponds to the first term in $\mathcal{J}^{CG}(q)$, except that the optimization is not performed in the latent space of a generator. It is arguably more expressive, but this extra expressivity hurts in the case of high-dimensional observations like images.

"Generative MI" attack [62] also uses a generator trained on a "auxiliary" dataset. The attack optimizes the following:

$$\arg\min_{\boldsymbol{z}} -\lambda \log(\overline{p}^{\text{tar}}(y|(G(\boldsymbol{z}))) - \log \sigma(D(G(\boldsymbol{z}))) \tag{6}$$

where G, D denotes the generator and discriminator respectively, and $\sigma(\cdot)$ is the sigmoid function. The first term coincides with $\mathcal{J}^{CG}(q)$. There are two key differences: (1) Instead of optimizing for a distribution $q(\boldsymbol{z})$, they optimized individual latent codes independently; and (2) in order to preserve realism, they proposed to use the discriminator loss as a regularizer. The theoretical justification is unclear. Suppose the GAN has converged to the global optimum, where $p_g = p_d$ (Theorem 1 in Goodfellow et al. [19]), then D outputs 0.5 everywhere, making $D(\cdot)$ completely uninformative. In contrast, our formulation optimizes a variational distribution q which allows us to compute the KL-divergence from the prior. This term naturally encourages realism and diversity. Lastly, having a distribution q allows us to draw new attack samples after one optimization, whereas the Generative MI attack requires re-running optimization for new samples.

Recently, [6] also proposed optimizing parameters of a Gaussian distribution in the latent space of the pretrained generator, which can be seen as a special case of our VMI framework: the variant using a DCGAN generator, and a Gaussian $q(\boldsymbol{z})$ when γ=0. Additionally, they applied extra regularization terms during the pretraining of the GAN. In our experience, these additional regularization terms did not lead to significant improvements.

5 Experiments

In this section, we demonstrate the effectiveness of the proposed VMI attack method on three separate tasks. We will refer to each task by their target dataset: MNIST [33], CelebA [35], and ChestX-ray

	Accuracy↑			FID ↓		
	MNIST	**CelebA**	**CXR**	**MNIST**	**CelebA**	**CXR**
General MI [22]	0 ± 0.00	0± 0.00	0.23± 0.29	376.7	421.21	499.54
Generative MI [62]	0.54 ± 0.11	0.06± 0.02	0.28± 0.24	114.47	57.61	152.91
+ Rejection Sampling	0.92 ± 0.02	0.07± 0.02	0.28± 0.25	88.91	43.21	142.66
VMI w/ DCGAN (Eqn 3)	**0.95**± 0.02	0.37± 0.07	0.42± 0.28	**77.73**	40.89	265.14
VMI w/ StyleGAN (Eqn 4)	-	**0.55**± 0.06	**0.69**± 0.23	-	**17.41**	**123.17**

Table 1: MI attack comparison across tasks. Both VMI methods here used the Flow model for $q(\boldsymbol{z})$. Best values are in ***bold***, and values within the 95% confidence interval (±) of the best are *underlined*. Our VMI attack results in better accuracy and FID compared to baselines on all three tasks.

(CXR) [56]. Then, we showcase the robustness of the VMI attack to different target classifiers on CelebA. More experimental details can be found in Appendix B.

Data. For the MNIST task, we used the 'letters' split (i.e., handwritten English alphabets) of the EMNIST [8] dataset as the auxiliary dataset. For CelebA, we used the 1000 most frequent identities as the target dataset, and the rest as the auxiliary dataset. For ChestX-ray, we used the 8 diseases outlined by Wang et al. [56] as the target dataset, and randomly selected 50,000 images from the remaining as the auxiliary dataset. Real samples from these six datasets are visualized in Figure 4.

Target Classifiers. The target classifiers used in all tasks were ResNets trained using SGD with momentum. Optimization hyperparameters were selected to maximize accuracy on a validation set of the private data.

Evaluation Metrics. As discussed in Section 2, an ideal evaluation should reflect how well the attack samples approximate $p^{\text{tar}}(\boldsymbol{x}|y) \propto p^{\text{tar}}(y|\boldsymbol{x})p^{\text{tar}}(\boldsymbol{x})$. To capture all aspects, a suite of metrics was used to measure *target accuracy*, *sample realism*, and *sample diversity*.

Target Accuracy: To automate the process of judging the accuracy of an attack, we use an "evaluation" classifier. This metrics measures how well the generated samples resemble the target class/identity, and can be seen as a surrogate measure for $p^{\text{tar}}(y|\boldsymbol{x})$. Given a target identity i, an attack produces a set of samples $\{\boldsymbol{x}'_n\}_{n=1}^N$. Each assigned a predicted label by the evaluation classifier: $i'_n = \max_i p(y = i|\boldsymbol{x}'_n)$. Accuracy is computed as follows: $\frac{1}{N}\sum_{i'_n} \mathbb{1}[i'_n = i]$, where $\mathbb{1}[\cdot]$ is the indicator function.

If a good evaluation classifier only generalized to realistic inputs, then this metric would also penalize unrealistic attack samples. To some extent, this was the case in our experiments as evidenced by the extremely low accuracy assigned to the General MI attack baseline in Table 1. In Figure 5, the General MI attack samples were clearly unrealistic. To ensure a fair and informative evaluation, the evaluation classifier should be as accurate and generalizable as possible (details in Appendix B.3).

Sample Realism: We used FID [21] to directly measure the quality of overall sample quality, which can be seen as a surrogate measure for $p^{\text{tar}}(\boldsymbol{x})$. To compute FID, we aggregated attack samples across classes and compared them to aggregated real images. Following standard practice, the feature extractor used for computing FID was the pretrained Inception network [53].

Sample Diversity: Though FID captures realism and overall diversity, it is known to neglect properties such as intra-class diversity, which is an important feature of a good MI attack. To overcome this limitation, we computed the improved precision & recall [32], coverage & density [37] on a per class basis, capturing the intra-class diversity of the generated samples. The same Inception network was used as the feature extractor. Recall proposed in Kynkäänniemi et al. [32] can be problematic when the generated samples contain outliers. Coverage proposed in Naeem et al. [37] is less susceptible to outliers, but also less sensitive. For convenience of comparison, we take the average of the two quantities and call the resulting metric diversity, i.e. "diversity" = (coverage + recall) / 2.

Generators. Two generator models were considered: DCGAN [42], and StyleGAN [27]. For StyleGAN, we used the original implementations.[1] There were 10 and 12 layers in the synthesis networks for the 64x64 CelebA data and 128x128 ChestX-ray data respectively. We do not use StyleGAN for the MNIST task since the images are simple.

[1]Code from: `https://github.com/NVlabs/stylegan2-ada`

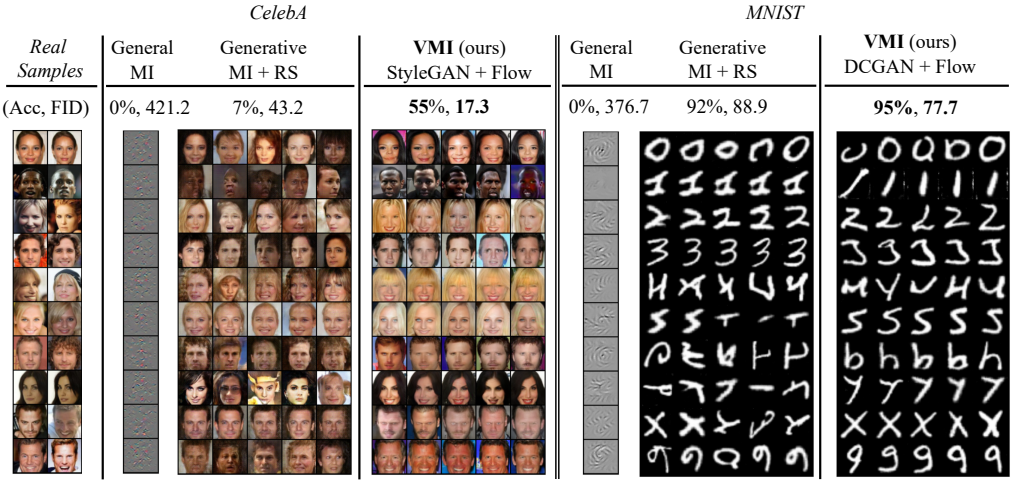

Figure 5: MI attack samples on the first ten identities of CelebA and MNIST. Each row corresponds to a different identity/digit. Qualitative differences are best viewed zoomed in.

	Generative MI + RS [62]	VMI (ours) DCGAN $q(\boldsymbol{z})$ = Gaussian γ=0	DCGAN Flow 0	DCGAN Flow 1e-3	DCGAN Flow 1e-1	StyleGAN Gaussian 0	StyleGAN Gaussian 1e-3	StyleGAN Gaussian 1e-1	StyleGAN Flow 0	StyleGAN Flow 1e-3	StyleGAN Flow 1e-1
Acc	0.07 ± 0.02	0.24 ± 0.05	0.33 ± 0.09	0.37 ± 0.07	0.13 ± 0.03	0.57 ± 0.06	0.56 ± 0.05	0.23 ± 0.03	**0.58** ± 0.06	0.55 ± 0.06	0.39 ± 0.07
Precision	0.51 ± 0.04	0.64 ± 0.05	0.48 ± 0.08	0.52 ± 0.06	0.40 ± 0.06	0.87 ± 0.02	0.88 ± 0.02	0.82 ± 0.02	0.87 ± 0.03	0.87 ± 0.03	**0.89** ± 0.03
Density	0.41 ± 0.04	0.67 ± 0.08	0.49 ± 0.11	0.52 ± 0.08	0.38 ± 0.06	1.26 ± 0.07	1.28 ± 0.07	1.14 ± 0.06	1.22 ± 0.08	1.22 ± 0.08	**1.31** ± 0.10
Recall	0.21 ± 0.04	0.03 ± 0.01	0.00 ± 0.00	0.01 ± 0.01	0.13 ± 0.03	0.22 ± 0.03	0.25 ± 0.03	**0.42** ± 0.03	0.11 ± 0.02	0.15 ± 0.03	0.21 ± 0.04
Coverage	0.83 ± 0.03	0.79 ± 0.04	0.37 ± 0.06	0.67 ± 0.06	0.70 ± 0.06	0.98 ± 0.01	0.98 ± 0.01	**0.98** ± 0.01	0.96 ± 0.02	0.97 ± 0.02	0.98 ± 0.02
Diversity	0.52 ± 0.05	0.41 ± 0.04	0.19 ± 0.06	0.34 ± 0.06	0.41 ± 0.07	0.60 ± 0.03	0.61 ± 0.03	**0.70** ± 0.03	0.54 ± 0.02	0.56 ± 0.04	0.59 ± 0.05
FID	43.21	28.98	58.39	40.89	40.74	16.69	16.11	**13.49**	17.28	17.41	21.35

Table 2: Detailed comparison between attack methods on CelebA.

5.1 Results

Comparison with Baselines Across all three tasks, our VMI formulations outperformed baselines in terms of target accuracy and sample realism (see Table 1), even when using the same generator architecture (DCGAN). Using a flow model for $q(\boldsymbol{z})$ led to improvements in terms of accuracy. Using the formulation in Equation 4 and a StyleGAN further improved performance on all metrics. Detailed evaluation using sample diversity metrics are reported in Table 2. In general, increasing γ increased diversity and realism while sacrificing accuracy. Our VMI attack outperformed the baseline on all metrics at multiple settings. The reported improvements in the quantitative metrics can be verified qualitatively in Figure 5. Detailed results for MNIST and ChestX-ray (Table 4, and 5) can be found in the Appendix. Using StyleGAN with a Gaussian $q(\boldsymbol{z})$ resulted in a slightly lower accuracy, but better diversity.

Attacking different target classifiers. After establishing the effectivenss of our VMI attack on different datasets, we conducted experiments to test the effectiveness of our VMI attack against different target classifiers. We further tested two target classifier on CelebA: a VGG network [49] trained from scratch on our target dataset, and an adapted ArcFace classifier [10], which we denote as ArcFace-NC. The ArcFace is a specialized face identity recognizer pretrained on a large collection of datasets. We adapt it by using the pretrained network as a feature encoder,[2] and adding a nearest cluster classifier in the feature space. This approach of adapting a powerful feature extractor is popular and effective in few-shot classification [51, 7]. The resulting accuracy on held-out test examples in the target dataset were 62.1% (VGG) and 96.1% (ArcFace-NC). Results in Table 3 show that the same observations hold. The VMI attack that used a StyleGAN and a flow $q(\boldsymbol{z})$ was the most accurate, and scored well in terms of the precision and density metrics. Using a flow $q(\boldsymbol{z})$ has a better accuracy-diversity trade-off (also visualized in Figure 6). We conjecture that the robustness to different target classifiers and better accuracy-diversity trade-off for the StyleGAN+flow model is due to its increased expressiveness, making optimization more likely to succeed.

[2]Code and model from: `https://github.com/TreB1eN/InsightFace_Pytorch`

	VGG					ArcFace-NC				
		VMI (ours)					VMI (ours)			
	Generative	DCGAN		StyleGAN		Generative	DCGAN		StyleGAN	
	MI + RS [62]	Gaussian	Flow	Gaussian	Flow	MI + RS [62]	Gaussian	Flow	Gaussian	Flow
Acc	0.04 ± 0.03	0.16 ± 0.09	0.12 ± 0.08	0.17 ± 0.10	**0.33** ± 0.18	0.21 ± 0.08	0.63 ± 0.17	0.55 ± 0.18	0.52 ± 0.13	**0.90** ± 0.07
Precision	0.51 ± 0.09	0.60 ± 0.10	0.74 ± 0.07	**0.80** ± 0.08	0.79 ± 0.11	0.59 ± 0.09	0.52 ± 0.13	0.53 ± 0.14	**0.86** ± 0.04	0.84 ± 0.07
Density	0.42 ± 0.11	0.63 ± 0.16	0.76 ± 0.20	**1.04** ± 0.19	1.02 ± 0.24	0.46 ± 0.13	0.55 ± 0.18	0.64 ± 0.21	1.26 ± 0.21	**1.26** ± 0.23
Recall	0.23 ± 0.10	0.09 ± 0.04	0.02 ± 0.02	**0.28** ± 0.06	0.04 ± 0.03	0.11 ± 0.04	0.01 ± 0.01	0.00 ± 0.01	**0.23** ± 0.07	0.04 ± 0.03
Coverage	0.79 ± 0.13	0.75 ± 0.13	0.77 ± 0.12	**0.93** ± 0.08	0.88 ± 0.10	0.81 ± 0.07	0.69 ± 0.11	0.69 ± 0.12	**0.98** ± 0.03	0.96 ± 0.05
Diversity	0.51 ± 0.16	0.42 ± 0.13	0.39 ± 0.12	**0.61** ± 0.10	0.46 ± 0.11	0.46 ± 0.08	0.35 ± 0.12	0.35 ± 0.12	**0.60** ± 0.08	0.50 ± 0.05
FID*	59.27	53.85	57.88	**32.51**	46.13	63.63	59.23	63.37	**32.94**	40.22

Table 3: Detailed comparison on CelebA for two other target classifiers: VGG, and ArcFace-NC. FID* reported here was computed over 20 identities instead of 100, so the values can not be directly compared with those in Table 2.

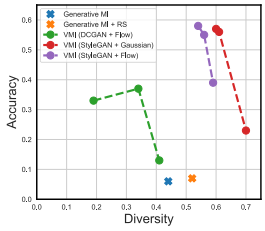

Figure 6: Accuracy and diversity tradeoff on CelebA.

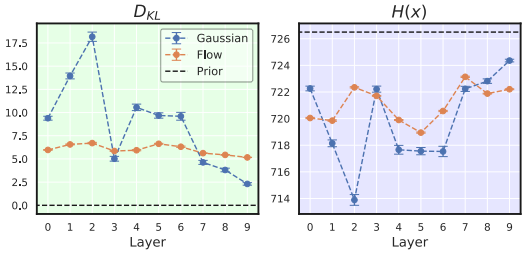

Figure 7: KL and entropy of $q_l(\boldsymbol{z})$ after VMI attack on CelebA at different layers.

Analysis: Effect of individual layers. As motivated in Section 4.1, the extended latent space in Equation 4 allowed our attack to separately modify layers which controlled different aspects of an image. Figure 7 shows values for the KL-divergence and entropy for $q_l(\boldsymbol{z})$ at all layers after VMI optimization. $q(\boldsymbol{z})$ at intermediate layers were modified more by the optimization, reflected in the higher values of the KL-divergence, and lower values of entropy. Echoing the qualitative observation in Figure 2, these intermediate layers had more control over features relevant to the identity of a face.

6 Limitation & Ethical Concerns

While our VMI attack was effective, it relied on the same assumption made in previous works: the attacker had access to a relevant auxiliary dataset [59, 62]. A worthwhile future direction is to develop methods that do not depend on this assumption. Methodologically, our proposed VMI attack used a GAN to narrow down the search space of the variational objective, but GANs might not be equally effective for other data modalities such as text, and tabular data. Lastly, our study focused on the white-box setting where the attacker had full access to the model, extensions of the current VMI attack to the grey-box or black-box settings are also worthy of further investigation.

In this work, we focused on improving model inversion attacks, which could have negative societal impacts if fallen into the wrong hands. As discussed in Section 1, malicious users can use attacks like this to break into secured systems and steal information from private data-centers. On the other hand, by pointing out the extent of the vulnerability of current models, we hope to facilitate research that builds better privacy-preserving algorithms, and raise awareness about privacy concerns.

7 Conclusion

An effective model inversion attack method must produce a set of samples that accurately captures the distinguishing features and variations of the underlying identity/class. Our proposed VMI attack optimizes a variational inference objective by using a combination of a StyleGAN generator and a deep flow model as its variational distribution. Our framework also provides insights into existing attack methods. Empirically, we found our VMI attacks were effective on three different tasks, and across different target classifiers.

References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4432–4441, 2019.

[2] Amazon AWS. Amazon rekognition. URL `https://aws.amazon.com/rekognition/`.

[3] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B Tenenbaum, William T Freeman, and Antonio Torralba. Gan dissection: Visualizing and understanding generative adversarial networks. *arXiv preprint arXiv:1811.10597*, 2018.

[4] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[5] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

[6] Si Chen, Ruoxi Jia, and Guo-Jun Qi. Improved techniques for model inversion attack. *arXiv preprint arXiv:2010.04092*, 2020.

[7] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2019.

[8] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[9] International Warfarin Pharmacogenetics Consortium. Estimation of the warfarin dose with clinical and pharmacogenetic data. *New England Journal of Medicine*, 360(8):753–764, 2009.

[10] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019.

[11] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[12] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[13] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[14] Simon S Du, Wei Hu, Sham M Kakade, Jason D Lee, and Qi Lei. Few-shot learning via learning the representation, provably. *arXiv preprint arXiv:2002.09434*, 2020.

[15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.

[16] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 17–32, 2014.

[17] Javier Galbally, Sébastien Marcel, and Julian Fierrez. Biometric antispoofing methods: A survey in face recognition. *IEEE Access*, 2:1530–1552, 2014.

[18] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients–how easy is it to break privacy in federated learning? *arXiv preprint arXiv:2003.14053*, 2020.

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27:2672–2680, 2014.

[20] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017.

[21] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *arXiv preprint arXiv:1706.08500*, 2017.

[22] Seira Hidano, Takao Murakami, Shuichi Katsumata, Shinsaku Kiyomoto, and Goichiro Hanaoka. Model inversion attacks for prediction systems: Without knowledge of non-sensitive attributes. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pages 115–11509. IEEE, 2017.

[23] Briland Hitaj, Giuseppe Ateniese, and Fernando 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*, pages 603–618, 2017.

[24] Corey D Holland and Oleg V Komogortsev. Complex eye movement pattern biometrics: Analyzing fixations and saccades. In *2013 International conference on biometrics (ICB)*, pages 1–8. IEEE, 2013.

[25] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[26] Georgios A Kaissis, Marcus R Makowski, Daniel Rückert, and Rickmer F Braren. Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence*, 2(6):305–311, 2020.

[27] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[28] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020.

[29] Mahdi Khosravy, Kazuaki Nakamura, Naoko Nitta, and Noboru Babaguchi. Deep face recognizer privacy attack: Model inversion initialization by a deep generative adversarial data space discriminator. In *2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1400–1405. IEEE, 2020.

[30] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[31] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018.

[32] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *arXiv preprint arXiv:1904.06991*, 2019.

[33] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database, 2010.

[34] Erich L Lehmann and Joseph P Romano. *Testing statistical hypotheses*. Springer Science & Business Media, 2006.

[35] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[36] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

[37] Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In *International Conference on Machine Learning*, pages 7176–7185. PMLR, 2020.

[38] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE, 2019.

[39] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

[40] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4467–4477, 2017.

[41] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. 2015.

[42] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[43] Christian Rathgeb and Andreas Uhl. A survey on biometric cryptosystems and cancelable biometrics. *EURASIP Journal on Information Security*, 2011(1):1–25, 2011.

[44] Maria Rigaki and Sebastian Garcia. A survey of privacy attacks in machine learning. *arXiv preprint arXiv:2007.07646*, 2020.

[45] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International conference on information processing in medical imaging*, pages 146–157. Springer, 2017.

[46] Elizabeth A Sconce, Tayyaba I Khan, Hilary A Wynne, Peter Avery, Louise Monkhouse, Barry P King, Peter Wood, Patrick Kesteven, Ann K Daly, and Farhad Kamali. The impact of cyp2c9 and vkorc1 genetic polymorphism and patient characteristics upon warfarin dose requirements: proposal for a new dosing regimen. *Blood*, 106(7):2329–2333, 2005.

[47] Yujun Shen, Ceyuan Yang, Xiaoou Tang, and Bolei Zhou. Interfacegan: Interpreting the disentangled face representation learned by gans. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[48] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[49] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[50] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[51] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

[52] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[53] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[54] Google TensorFlow. Tensorflow hub. URL `https://www.tensorflow.org/hub`.

[55] Nilesh Tripuraneni, Michael I Jordan, and Chi Jin. On the theory of transfer learning: The importance of task diversity. *arXiv preprint arXiv:2006.11650*, 2020.

[56] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2097–2106, 2017.

[57] Yaxing Wang, Abel Gonzalez-Garcia, David Berga, Luis Herranz, Fahad Shahbaz Khan, and Joost van de Weijer. Minegan: effective knowledge transfer from gans to target domains with few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9332–9341, 2020.

[58] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[59] Ziqi Yang, Ee-Chien Chang, and Zhenkai Liang. Adversarial neural network inversion via auxiliary knowledge alignment. *arXiv preprint arXiv:1902.08552*, 2019.

[60] Raymond A Yeh, Chen Chen, Teck Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with deep generative models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5485–5493, 2017.

[61] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8715–8724, 2020.

[62] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn 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*, pages 253–261, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] see Section 6
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] see Section 6
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 4.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A.1, and A.2
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] see Appendix B
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] see Appendix B
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A] see original data source of the citations.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]