CLASSIFIER-TO-GENERATOR ATTACK: ESTIMATION OF TRAINING DATA DISTRIBUTION FROM CLASSIFIER

Anonymous authors
Paper under double-blind review

ABSTRACT

Suppose a deep classification model is trained with samples that need to be kept private for privacy or confidentiality reasons. In this setting, can an adversary obtain the private samples if the classification model is given to the adversary? We call this reverse engineering against the classification model the *Classifier-to-Generator (C2G) Attack*. This situation arises when the classification model is embedded into mobile devices for offline prediction (e.g., object recognition for the automatic driving car and face recognition for mobile phone authentication).

For C2G attack, we introduce a novel GAN, *PreImageGAN*. In PreImageGAN, the generator is designed to estimate the the sample distribution conditioned by the preimage of classification model f, $P(X|f(X) = y)$, where X is the random variable on the sample space and y is the probability vector representing the target label arbitrary specified by the adversary. In experiments, we demonstrate PreImageGAN works successfully with hand-written character recognition and face recognition. In character recognition, we show that, given a recognition model of hand-written digits, PreImageGAN allows the adversary to extract numeric images without knowing that the model is built for numeric images. In face recognition, we show that, when an adversary obtains a face recognition model for a set of individuals, PreImageGAN allows the adversary to extract face images of specific individuals contained in the set, even when the adversary has no knowledge of the face of the individuals.

1 INTRODUCTION

Recent rapid advances in deep learning technologies are expected to promote the application of deep learning to online services with recognition of complex objects. Let us consider the face recognition task as an example. The probabilistic classification model f takes a face image $\mathbf{x}$ and the model predicts the probability of which the given face image is associated with an individual t, $f(\mathbf{x}) \simeq \Pr[T = t|X = \mathbf{x}]$.

The following three scenarios pose situations that probabilistic classification models need to revealed in public for online services in real applications:

Prediction with cloud environment: Suppose an enterprise provides an online prediction service with a cloud environment, in which the service takes input from a user and returns predictions to the user in an online manner. The enterprise needs to deploy the model f into the cloud to achieve this.

Prediction with private information: Suppose an enterprise develops a prediction model f (e.g., disease risk prediction) and a user wishes to have a prediction of the model with private input (e.g., personal genetic information). The most straightforward way to preserve the user's privacy entirely is to let the user download the entire model and perform prediction on the user side locally.

Offline prediction: Automatic driving cars or laptops with face authentication contain face/object recognition systems in the device. Since these devices are for mobile use and need to work standalone, the full model f needs to be embedded in the device.

In such situations that classification model f is revealed, we consider a reverse-engineering problem of models with deep architectures. Let $\mathcal{D}_{\text{tr}}$ and $d_{X,T}$ be a set of training samples and its underlying distribution, respectively. Let f be a model trained with $\mathcal{D}_{\text{tr}}$. In this situation, is it possible for an

adversary to obtain the training samples $\mathcal{D}_{\text{tr}}$ (or its underlying distribution $d_{X,T}$) if the classification model is given to the adversary?. If this is possible, this can cause serious problems, particularly when $\mathcal{D}_{\text{tr}}$ or $d_{X,T}$ is private or confidential information.

Privacy violation by releasing face authentication: Let us consider the face authentication task as an example again. Suppose an adversary is given the classification model f. The adversary aims to estimate the data (face) distribution of a target individual $t^*, d_{X|T=t^*}$. If this kind of reverse-engineering works successfully, serious privacy violation arises because individual faces are private information. Furthermore, once $d_{X|T=t^*}$ is revealed, the adversary can draw samples from $d_{X|T=t^*}$, which would cause another privacy violation (say, the adversary can draw an arbitrary number of the target's face images).

Confidential information leakage by releasing object recognizer: Let us consider an object recognition system for automatic driving cars. Suppose a model f takes as input images from car-mounted cameras and detect various objects such as traffic signs or traffic lights. Given f, the reverse engineering reveals the sample distribution of the training samples, which might help adversaries having malicious intentions. For example, generation of adversarial examples that make the recognition system confuse without being detected would be possible. Also, this kind of attack allows exposure of hidden functionalities for privileged users or unexpected vulnerabilities of the system.

If this kind of attack is possible, it indicates that careful treatment is needed before releasing model f in public considering that publication of f might cause serious problems as listed above. We name this type of reverse engineering *classifier-to-generator (C2G) attack* . In principle, estimation of labeled sample distributions from a classification/recognition model of complex objects (e.g., face images) is a difficult task because of the following two reasons. First, estimation of generative models of complex objects is believed to be a challenging problem itself. Second, model f often does not contain sufficient information to estimate the generative model of samples. In supervised classification, the label space is always much more abstract than the sample space. The classification model thus makes use of only a limited amount of information in the sample space that is sufficient to classify objects into the abstract label space. In this sense, it is difficult to estimate the sample distribution given only classification model f.

To resolve the first difficulty, we employ Generative Adversarial Networks (GANs). GANs are a neural network architecture for generative models which has developed dramatically in the field of deep learning. Also, we exploit one remarkable property of GANs, the ability to interpolate latent variables of inputs. With this interpolation, GANs can generate samples (say, images) that are not included in the training samples, but realistic samples[1].

Even with this powerful generation ability of GANs, it is difficult to resolve the second difficulty. To overcome this for the C2G attack, we assume that the adversary can make use of *unlabeled* auxiliary samples $\mathcal{D}_{\text{aux}}$ as background knowledge. Suppose f be a face recognition model that recognizes Alice and Bob, and the adversary tries to extract Alice's face image from f. It is natural to suppose that the adversary can use public face image samples that do not contain Alice's and Bob's face images as $\mathcal{D}_{\text{aux}}$. PreImageGAN exploits *unlabeled* auxiliary samples to complement knowledge extracted from the model f.

1.1 Our Contribution

The contribution of this study is summarized as follows.

- We formulate the Classifier-to-Generator (C2G) Attack, which estimates the training sample distribution when a classification model and auxiliary samples are given(Section 3)
- We propose PreImageGAN as an algorithm for the C2G attack. The proposed method estimates the sample generation model using the interpolation ability of GANs even when the auxiliary samples used by the adversary is not drawn from the same distribution as the training sample distribution (Section 4)

[1]Radford et al. (2015) reported GANs could generate intermediate images between two different images. Also, Radford et al. (2015) realizes the operation of latent vectors. For example, by subtracting a latent vector of a man's face from a face image of a man wearing glasses, and then adding a latent vector of a female's face, then the GAN can generate the woman's face image wearing glasses.

- We demonstrate the performance of C2G attack with PreImageGAN using EMNIST (alphanumeric image dataset) and FaceScrub (face image dataset). Experimental results show that the adversary can estimate the sample distribution even when the adversary has no samples associated with the target label at all (Section 5)

2 GENERATIVE ADVERSARIAL NETWORKS

Generative Adversarial Networks (GANs) is a recently developed methodology for designing generative models proposed by Goodfellow et al. (2014). Given a set of samples, GANs is an algorithm with deep architectures that estimates the sample-generating distribution. One significant property of GANs is that it is expected to be able to accurately estimate the sample distribution even when the sample space is in the high dimensional space, and the target distribution is highly complex, such as face images or natural images. In this section, we introduce the basic concept of GANs and its variants.

The learning algorithm of GANs is formulated by minimax games consisting of two players, *generator* and *discriminator* (Goodfellow et al. (2014)). Generator G generates a fake sample $G(\mathbf{z})$ using a random number $\mathbf{z} \sim d_Z$ drawn from any distribution (say, uniform distribution). Discriminator D is a supervised model and is trained so that it outputs 1 if the input is a real sample $\mathbf{x} \sim d_X$ drawn from the sample generating distribution d_X; it outputs 0 or -1 if the input is a fake sample $G(\mathbf{z})$. The generator is trained so that the discriminator determines a fake sample as a real sample.

By training the generator under the setting above, we can expect that samples generated from $G(\mathbf{z})$ for arbitrary $\mathbf{z}$ are indistinguishable from real samples $\mathbf{x} \sim d_X$. Letting Z be the random variable of d_Z, $G(Z)$ can be regarded as the distribution of samples generated by the generator. Training of GANs is known to be reduced to optimization of G so that the distribution between $G(Z)$ and the data generating distribution d_X is minimized in a certain type of divergence (Goodfellow et al. (2014)).

Training of GAN proposed by Goodfellow et al. (2014) (VanillaGAN) is shown to be reduced to minimization e of Jensen Shannon (JS) divergence of $G(Z)$ and d_X. Minimization of JS-divergence often suffers gradient explosion and mode collapse (Goodfellow et al. (2014), Arjovsky & Bottou (2017)). To overcome these problems, Wasserstein-GAN (WGAN), GAN that minimizes Wasserstein distance between $G(Z)$ and d_X, was proposed (Arjovsky et al. (2017), Arjovsky & Bottou (2017)). As a method to stabilize convergence behavior of WGAN, a method to add a regularization term called *Gradient Penalty (GP)* to the loss function of the discriminator was introduced (Gulrajani et al. (2017)).

Given a set of labeled samples $\{(\mathbf{x}, \mathbf{c}), \cdots\}$ where $\mathbf{c}$ denotes the label, Auxiliary Classifier GAN (ACGAN) was proposed as a GAN to estimate $d_{X|C=\mathbf{c}}$, sample distribution conditioned by label $\mathbf{c}$ (Odena et al. (2016)). Differently from VanillaGAN, the generator of ACGAN takes as input a random noise $\mathbf{z}$ and a label $\mathbf{c}$. Also, the discriminator of ACGAN is trained to predict a label of sample in addition to estimation of real or fake samples. In the learning process of ACGAN, generator is trained so that discriminator predicts correctly the label of generated sample in addition. The generator of ACGAN can generate samples with a label specified arbitrarily. For example, when x corresponds to face images and c corresponds to age or gender, ACGAN can generate images with specifying the age or gender (Mirza & Osindero (2014), Gauthier (2014)). In our proposed algorithm introduced in the latter sections, we employ WGAN and ACGAN as building blocks.

3 PROBLEM FORMULATION

3.1 PROBABILISTIC DISCRIMINATION MODEL

We consider a supervised learning setting. Let $\mathbb{T}$ be the label set, and $\mathbb{X} \subseteq \mathbb{R}^d$ be the sample domain where d denotes the sample dimension. Let ρ_t be the distribution of samples in $\mathbb{X}$ with label t. In face recognition, $\mathbf{x} \in \mathbb{X}$ and $t \in \mathbb{T}$ correspond to a (face) image and an individual, respectively. ρ_t thus denotes the distribution of face images of individual t.

We suppose the images contained in the training dataset are associated with a label subset $\mathbf{T}_{\text{tr}} \subset \mathbb{T}$. Then, the training dataset is defined as $\mathcal{D}_{\text{tr}} = \{(\mathbf{x}, t) | \mathbf{x} \in \mathbb{X}, t \in \mathbf{T}_{\text{tr}}\}$. We denote the random

variables associated with $(\mathbf{x}, t)$ by $(X_{\text{tr}}, T_{\text{tr}})$. Then, the distribution of X_{tr} is given by a mixture distribution

$$d_{X_{\text{tr}}} = \sum_{t \in \mathbf{T}_{\text{tr}}} \alpha_t \rho_t \tag{1}$$

where $\sum_{t \in \mathbf{T}_{\text{tr}}} \alpha_t = 1, \alpha_t > 0$ for all $t \in \mathbf{T}_{\text{tr}}$. In the face recognition task example again, a training sample consists of a pair of an individual t and his/her face image $\mathbf{x}$, $(\mathbf{x}, t)$ where $\mathbf{x} \sim \rho_t$.

Next, we define the probabilistic discrimination model we consider in our problem. Let $\mathbb{Y}$ be a set of $|\mathbf{T}_{\text{tr}}|$-dimension probability vector, $\Delta^{|\mathbf{T}_{\text{tr}}|}$. Given a training dataset $\mathcal{D}_{\text{tr}}$, a learning algorithm $\mathcal{L}$ gives a probabilistic discrimination model $f : \mathbb{X} \to \mathbb{Y}$ as $f = \mathcal{L}(\mathcal{D}_{\text{tr}})$. Here the tth element of the output $(f(\mathbf{x}))_t$ of f corresponds to the probability with which $\mathbf{x}$ has label t. Letting T_{tr} and X_{tr} represents the random variable of $\mathbf{T}_{\text{tr}}$ and $d_{X_{\text{tr}}}$, f is the approximation of $\Pr[T_{\text{tr}}|X_{\text{tr}}]$.

3.2 Classifier-to-Generator Attack

We define the *Classifier-to-Generator Attack (C2G Attack)* in this section. We consider two stakeholders, *publisher* and *adversary* in this attack. The publisher holds training dataset $\mathcal{D}_{\text{tr}}$ drawn from $d_{X_{\text{tr}}}$ and a learning algorithm $\mathcal{L}$. She trains model $f = \mathcal{L}(\mathcal{D}_{\text{tr}})$ and publishes f to the adversary. We suppose training dataset $\mathcal{D}_{\text{tr}}$ and data generating distribution ρ_t for any $t \in \mathbf{T}_{\text{tr}}$ is private or confidential information of the publisher and the publisher does not wish to leak them by publishing f.

Given f and $\mathbf{T}_{\text{tr}}$, the adversary aims to obtain ρ_{t^*} for any label $t^* \in \mathbf{T}_{\text{tr}}$ specified by the adversary. We suppose the adversary can make use of an auxiliary dataset $\mathcal{D}_{\text{aux}}$ drawn from underlying distribution $d_{X_{\text{aux}}}$ as background knowledge. $\mathcal{D}_{\text{aux}}$ is a set of samples associated with labels in $\mathbf{T}_{\text{aux}} \subset \mathbb{T}$. We remark that $\mathcal{D}_{\text{aux}}$ is defined as a set of samples associated with a specific set of labels, however, in our algorithm described in the following sections, we do not require that samples in $\mathcal{D}_{\text{aux}}$ are labeled. Then, the underlying distribution $d_{X_{\text{aux}}}$ is defined as follows:

$$d_{X_{\text{aux}}} = \sum_{t \in \mathbf{T}_{\text{aux}}} \alpha'_t \rho_t \tag{2}$$

where $\sum_{t \in \mathbf{T}_{\text{aux}}} \alpha'_t = 1, \alpha'_t > 0$ for all $t \in \mathbf{T}_{\text{aux}}$.

The richness of the background knowledge can be determined by the relation between $\mathbf{T}_{\text{tr}}$ and $\mathbf{T}_{\text{aux}}$. When $\mathbf{T}_{\text{tr}} = \mathbf{T}_{\text{aux}}$, $d_{X_{\text{tr}}} = d_{X_{\text{aux}}}$ holds. That is, the adversary can make use of samples drawn from the distribution that is exactly same as that of the publisher. In this sense, this setting is the most advantageous to the adversary. If $t^* \notin \mathbf{T}_{\text{aux}}$, the adversary cannot make use of samples with the target label t^*; this setting is more advantageous to the publisher. As the overlap between $\mathbf{T}_{\text{tr}}$ and $\mathbf{T}_{\text{aux}}$ increases, the situation becomes more advantageous to the adversary. Discussions on the background knowledge of the adversary are given in 3.4 in detail.

The goal of the adversary is to learn the publisher's private distribution ρ_{t^*} for any $t^* \in \mathbf{T}_{\text{tr}}$ specified by the adversary provided model f, target label t^* and auxiliary (unlabeled) samples $\mathcal{D}_{\text{aux}}$. Let $\mathcal{A}$ be the adversary's attack algorithm. Then, the attack by the adversary can be formulated by

$$\hat{\mathbf{x}}^{(t^*)} \sim \mathcal{A}(f, \mathcal{D}_{\text{aux}}, t^*)$$

where the output of $\mathcal{A}$ is a distribution over $\mathbb{X}$. In the face recognition example of Alice and Bob again, when the target label of the adversary is t^* =Alice, the objective of the adversary is to estimate the distribution of face images of Alice by $\mathcal{A}(f, \mathcal{D}_{\text{aux}}, t^*)$.

3.3 Evaluation of the C2G Attack

The objective of the C2G attack to estimate ρ_{t^*}, the private data generating distribution of the publisher. In principle, the measure of the success of the C2G attack is evaluated with the quasi-distance between the underlying distribution ρ_{t^*} and the estimated generative model $\mathcal{A}(f, \mathcal{D}_{\text{aux}}, t^*)$. If the two distributions are close, we can confirm that the adversary successfully estimates ρ_{t^*}. However, ρ_{t^*} is unknown, and we cannot evaluate this quasi-distance directly.

Instead of evaluating the distance of the two distributions directly, we evaluate the attack algorithm empirically. We first prepare a classifier f' that is trained with $\mathcal{D}_{\text{tr}}$ using a learning algorithm different from f. We then give samples drawn from $\mathcal{A}(f, \mathcal{D}_{\text{aux}}, t^*)$ to f' and evaluate the probability of which the label of the given samples are predicted as t^*. We expect that the classifier f' would label samples drawn from $\mathcal{A}(f, \mathcal{D}_{\text{aux}}, t^*)$ as t^* with high probability if $\mathcal{A}(f, \mathcal{D}_{\text{aux}}, t^*)$ successfully estimates ρ_{t^*}. Considering the possibility that $\mathcal{A}(f, \mathcal{D}_{\text{aux}}, t^*)$ overfits to f, we employ another classifier f' for this evaluation. This evaluation criterion is the same as the *inception accuracy* introduced for ACGAN by Odena et al. (2016). In our setting, since our objective is to estimate the distribution concerning a specific label t^*, we employ the following inception accuracy:

$$\Pr_{\mathbf{x} \sim \mathcal{A}(f, \mathcal{D}_{\text{aux}}, t^*)} \left[\arg\max_{t \in \mathbf{T}_{\text{tr}}} (f'(x))_t = t^* \right]$$

We remark that the generated model with a high inception accuracy is not always a reasonable estimation of ρ_{t^*}. Discrimination models with a deep architecture are often fooled with artifacts. For example, Nguyen et al. (2014) reported that images look like white noise for humans can be classified as a specific object with high probability. For this reason, we cannot conclude that a model with a high inception accuracy always generates meaningful images. To avoid this, the quality of generated images should be subjectively checked by humans.

The evaluation criterion of the C2G Attack we employed for this study is similar to those for GANs. Since the objective of GANs and the C2G attack is to estimate unknown generative models, we cannot employ the pseudo distance between the underlying generating distribution and the estimated distribution. The evaluation criterion of GANs is still an open problem, and subjective evaluation is needed for evaluation of GANs (Goodfellow (2017)). In this study, we employ both the inception accuracy and subjective evaluation for performance evaluation of the C2G attack.

3.4 Background Knowledge of Adversary

The richness of the background knowledge of the adversary affects the performance of the C2G attack significantly. We consider the following three levels of the background knowledge. In the following, let $\mathbf{T}_{\text{aux}}$ be the set of labels of samples generated by the underlying distribution of the auxiliary data, $d_{X_{\text{aux}}}$. Also, let $\mathbf{T}_{\text{tr}}$ be the set of labels of samples generated by the underlying distribution of the training data.

- **Exact same:** $\mathbf{T}_{\text{tr}} = \mathbf{T}_{\text{aux}}$
  In this setting, we suppose $\mathbf{T}_{\text{tr}}$ is exactly same as the $\mathbf{T}_{\text{aux}}$. Since $\mathcal{D}_{\text{aux}}$ follows $d_{X_{\text{tr}}}$, $\mathcal{D}_{\text{aux}}$ contains samples with the target label. That is, the adversary can obtain samples labeled with the target label. The background knowledge of the adversary in this setting is the most powerful among the three settings.
- **Partly same:** $t^* \notin \mathbf{T}_{\text{aux}}, \mathbf{T}_{\text{aux}} \subset \mathbf{T}_{\text{tr}}$
  In this setting, $\mathbf{T}_{\text{aux}}$ and $\mathbf{T}_{\text{tr}}$ are overlapping. However, $\mathbf{T}_{\text{aux}}$ does not contain the target label. That is, the adversary cannot obtain samples labeled with the target label. In this sense, the background knowledge of the adversary in this setting is not as precise as that in the former setting.
- **Mutually exclusive:** $\mathbf{T}_{\text{aux}} \cap \mathbf{T}_{\text{tr}} = \emptyset$
  In this setting, we suppose $\mathbf{T}_{\text{aux}}$ and $\mathbf{T}_{\text{tr}}$ are mutually exclusive, and the adversary cannot obtain samples labeled with the target label. In this setting, the adversary cannot obtain any samples with labels used for training of model f. In this sense, the background knowledge of the adversary in this setting is the poorest among the three settings.

4 PreImageGAN

4.1 Outline of PreImageGAN

In this section, we propose PreImageGAN that works as an algorithm that achieves C2G attack. Given the model $f \simeq d_{Y_{\text{tr}}|X_{\text{tr}}}$, the goal of PreImageGAN is to estimate $\left(d_{X_{\text{tr}}|Y_{\text{tr}}=\mathbf{y}^{(t^*)}}\right) = \rho_{t^*}$ for

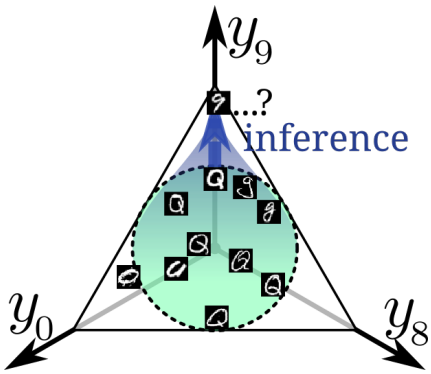

Figure 1: Inference on the space of $\mathbf{y}$. Here, we suppose the adversary has auxiliary samples labeled with alphabets only and a probabilistic classification model that takes an image of a number and outputs the corresponding number. The axis corresponds to an element of the probabilistic vector outputted by the classification model. For example, $y_9 = (f(\mathbf{x}))_9$ denotes the probability that the model discriminates the input image as "9". The green region in the figure describes the spanned by auxiliary samples in $\mathcal{D}_{\text{aux}}$. $\mathcal{D}_{\text{aux}}$ does not contain images of numbers classified so that $y_9 = 1$ or $y_8 = 1$ whereas PreImageGAN generates samples close to "9" by interpolating latent variables of images that are close to "9" such as "Q" and "g".

target label t^* specified by the adversary where $\mathbf{y}^{(t^*)}$ is the one-hot vector in which the element corresponds to t^* is activated.

In the adversary can draw samples from $d_{X_{\text{tr}}|Y_{\text{tr}}=\mathbf{y}^{(t^*)}}$, this generative model can be straightforwardly estimated with existing GANs. However, in the setting of the C2G attack, what the adversary can utilize for the C2G attack is the probabilistic classification model f that is expected to be similar to $d_{Y_{\text{tr}}|X_{\text{tr}}}$ and auxiliary samples that are drawn from auxiliary distribution $d_{X_{\text{aux}}}$ only. So the target distribution cannot be estimated in a straightforward manner. PreImageGAN first gives labels to auxiliary samples with the given model as $\mathbf{y} = f(\mathbf{x})$ and estimates $d_{X_{\text{aux}}|Y_{\text{aux}}}$ using $\{(\mathbf{x}, \mathbf{y})|\mathbf{x} \in \mathcal{D}_{\text{aux}}\}$ by optimizing the objective function defined below.

If the sample distribution of the auxiliary samples is close to that of the true underlying distribution, we can expect that estimation of $d_{X_{\text{aux}}|Y_{\text{aux}}}$ can be used as an approximation of $d_{X_{\text{tr}}|Y_{\text{tr}}}$. More specifically, we can obtain the generative model of the target label $d_{X_{\text{aux}}|Y_{\text{aux}}=\mathbf{y}^{(t^*)}}$ by specifying the one-hot vector of the target label as the condition.

As we mentioned in Section 3.4, the sample generating distribution of the auxiliary samples is not necessarily equal or close to the true sample generating distribution. In the "partly same" setting or "mutually exclusive" setting, $\mathcal{D}_{\text{aux}}$ does not contain samples labeled with t^* at all. It is well known that GANs can generate samples with interpolating latent variables (Berthelot et al. (2017), Radford et al. (2015)). We expect that PreImageGAN generates samples with the target label by interpolation of latent variables of given samples without having samples of the target label. More specifically, if latent variables of given auxiliary samples are diverse enough and $d_{X_{\text{aux}}|Y_{\text{aux}}}$ well approximates the true sample generating distribution, we expect that GAN can generate samples with the target label by interpolating obtained latent variables of auxiliary samples without having samples with the target label.

Figure 1 describes a conceptual illustration of generation of image "9" when PreImageGAN takes images of alphabets (A-Z,a-z) as auxiliary samples and $f(\mathbf{x}) = \Pr[\text{number}|\mathbf{x}]$ as a probabilistic classification model. The auxiliary samples do not contain images of numbers whereas PreImageGAN generates images close to "9" by interpolating latent variables of alphabets close to "9", such as "g" or "Q".

4.2 OBJECTIVE FUNCTION

In PreImageGAN, a conditional attribute $\mathbf{y} \sim d_{Y_{\text{aux}}}$ in addition to noise $\mathbf{z} \sim d_Z$ is used to generate samples. Similarly to other GANs, d_Z can be arbitrarily determined by the learner (adversary). Given f and $\mathcal{D}_{\text{aux}}$, $d_{Y_{\text{aux}}}$ can be empirically estimated from the set of predictions $\{ f(\mathbf{x}) | \mathbf{x} \in \mathcal{D}_{\text{aux}} \}$.

Generator $G : (\mathbb{Z}, \mathbb{Y}) \to \mathbb{X}$ of PreImageGAN generates fake samples $\mathbf{x}_{\text{fake}} = G(\mathbf{z}, \mathbf{y})$ using random draws of $\mathbf{y}$ and $\mathbf{z}$. After the learning process is completed, we expect generated fake samples $\mathbf{x}_{\text{fake}}$ satisfy $f(\mathbf{x}_{\text{fake}}) = \mathbf{y}$. On the other hand, discriminator $D : \mathbb{X} \to \mathbb{R}$ takes as input a sample $\mathbf{x}$ and discriminates whether it is a generated fake sample $\mathbf{x}_{\text{fake}}$ or a real sample $\mathbf{x}_{\text{real}} \in \mathcal{D}_{\text{aux}}$.

With these requirements, the objective function of G and D is formulated as follows

$$\begin{aligned} \min_{G} \max_{\|D\|_{\text{L}} \leq 1} & \mathrm{E}_{\mathbf{x} \sim d_X} \left[D(\mathbf{x}) \right] \\ & - \mathrm{E}_{\mathbf{z} \sim d_Z, \mathbf{y} \sim d_{Y_{\text{aux}}}} \left[D(G(\mathbf{z}, \mathbf{y})) \right] \\ & - \gamma \mathrm{E}_{\mathbf{z} \sim d_Z, \mathbf{y} \sim d_{Y_{\text{aux}}}} \left[\text{similarity}(f(G(\mathbf{z}, \mathbf{y})), \mathbf{y}) \right] \end{aligned} \tag{3}$$

where $\| \cdot \|_{\text{L}} \leq 1$ denotes α-Lipschitz functions with $\alpha \leq 1$.

By maximizing the first and second term concerning D, Wasserstein distance between the marginal of the generator $\int G(Z, Y_{\text{aux}}) dY_{\text{aux}}$ and the generative distribution of auxiliary samples $d_{X_{\text{aux}}}$ is minimized. By maximizing the similarity between $\mathbf{y}$ and $f(G(\mathbf{z}, \mathbf{y}))$, G is trained so that samples generated from $G(\mathbf{z}, \mathbf{y})$ satisfy $f(G(\mathbf{z}, \mathbf{y})) = \mathbf{y}$. $\gamma \geq 0$ works as a parameter adjusts the effect of this term.

For sample generation with PreImageGAN, $G(Z, \mathbf{y}^{(t^*)})$ is utilized as the estimation of ρ_{t^*}. We here remark that model f is regarded as a constant in the learning process of GAN and used as it is.

5 EXPERIMENTS

In this section, we show that the proposed method enables to perform C2G attack with experiments. We experimentally demonstrate that the adversary can successfully estimate ρ_{t^*} given classifier f and the set of unlabeled auxiliary samples $\mathcal{D}_{\text{aux}} = \{\mathbf{x} | \mathbf{x} \in \mathbb{X}\}$ even in the partly same setting and the mutually exclusive setting. .

5.1 EXPERIMENTAL SETUP

For demonstration, we consider a hand-written character classification problem (MNIST/EMNIST) and a face recognition problem (FaceScrub). We used the Adam optimizer ($\alpha = 2e + 4$, $\beta_1 = 0.5$, $\beta_2 = 0.999$) for the training of the generator and discriminator. The batch size was set as 64. We set the number of discriminator (critic) iterations per each generator iteration $n_{\text{cric}} = 5$. To enforce the 1-Lipschitz continuity of the discriminator, we add a gradient penalty (GP) term to the loss function of the discriminator (Gulrajani et al. (2017)) and set the strength parameter of GP as $\lambda = 10$. We used 128-dim uniform random distribution $[-1, 1]^{128}$ as d_Z. We estimated $d_{Y_{\text{aux}}}$ empirically from $\{f(\mathbf{x}) | \mathbf{x} \in \mathcal{D}_{\text{aux}}\}$ using kernel density estimation where the bandwidth is 0.01, and the Gaussian kernel was employed.

5.2 EMNIST: HAND-WRITTEN CHARACTERS CLASSIFIER

EMNIST consists of grayscale 28x28 pixel images from 62 alphanumeric characters (0-9A-Za-z). We evaluated C2G attack with changing the richness of the adversary's background knowledge as discussed in Section 3.4.

Table 1 represents the settings used for the C2G attack to alphanumeric classification. In all cases, the target label t^* was set as numeric characters ($t^* \in \{0, 1, \ldots, 9\}$). The alphanumeric classifier given to the adversary in the exact same setting is trained on 10k samples for ten epochs and achieved test accuracy 0.8443. The numeric classifier given to the adversary in the partly same/mutually exclusive setting is trained on 116k samples for ten epochs and achieved test accuracy 0.9911. In the training process of PreImageGAN, we trained the generator for 20k iterations. We set the initial value of γ to 0, incremented gamma by 0.001 per generator iteration while γ is less than 10.

Table 1: Summary of the settings

	Training samples $\mathcal{D}_{\text{tr}}$ and classifier f	Auxiliary samples $\mathcal{D}_{\text{aux}}$	Adversary's target t^*
Exact same setting	alphanumeric	numeric	numeric
Partially same setting	alphanumeric	alphabet	numeric
Mutually exclusive setting	numeric	alphabet	numeric

Fig. 2 represents the results of the C2G attack. In the exact same setting, we can confirm that PreImageGAN generated images that look like numeric images corresponding to each target label quite well. In the partially same setting, some generated images are disfigured compared to images generated in the exact same setting (especially when $t^* = 3, 7$) while we can recognize most of the generated images as numeric images corresponding to each target label as well. From these results, we can conclude that the C2G attack against character recognition works successfully in the exact same and partially same setting.

We note that the adversary could generate numeric images even when the adversary has no numeric images from the result in the partially same setting. In the mutually exclusive setting, some digit-like images are generated while some samples still look like alphabets. When the classifier cannot distinguish alphabet images and numeric images well, PreImageGAN tends to generate disfigure images. This observation indicates that when classification models can classify samples without exploiting the feature of entire images, the attack does not necessarily work successfully.

We evaluate inception accuracy using f' where we employed ResNet-based network architecture (Table A). As a baseline, we trained ACGAN with the same training samples $\mathcal{D}_{\text{tr}}$ and evaluated the inception accuracy. In the exact same setting, the inception accuracy is almost equal to ACGAN. From the results, we can see that the inception scores drop as the background knowledge of the adversary becomes poorer. This result indicates that the background knowledge of the adversary affects the result of the C2G attack significantly.

5.3 FaceScrub: Face Recognition

FaceScrub dataset consists of color face images (530 persons). We resized images to 64x64 pixel images for experiments and evaluated the C2G attack in the mutually exclusive setting. In detail, we picked up 100 persons as $\mathbf{T}_{\text{tr}}$ and used remaining 430 persons as $\mathbf{T}_{\text{aux}}$. If the adversary can generate faces of 100 persons in $\mathbf{T}_{\text{tr}}$ by utilizing model f recognizing $\mathbf{T}_{\text{tr}}$ and face images with labels in $\mathbf{T}_{\text{aux}}$, we can confirm that the C2G attack works successfully. $\mathcal{D}_{\text{tr}}$ consisted of 13,650 images and $\mathcal{D}_{\text{aux}}$ consists of 59,764 images. f is trained on $\mathcal{D}_{\text{tr}}$ for ten epochs and achieved test accuracy 0.8395.

In training PreImageGAN, we train generator for 130k iterations. We set the initial value of γ to 0, incremented gamma by 0.0001 per generator iteration while γ is less than 10.

Fig. 3 represents the results of the C2G attack against face recognition. Those samples are randomly generated without human selection. The generated face images well capture the features of the face images in the training samples, and we can conclude that the C2G attack successfully extracts training samples from face recognition model without having training samples in the mutually exclusive setting.

As byproducts, we can learn what kind of features the model used for face recognition with the results of the C2G attack. For example, all generated face images of Keanu Reeves wear the mustache. This result implies that f exploits his mustache to recognize Keanu Reeves.

6 Related Work

Fredrikson et al. (2014) proposed the model inversion attack against machine learning algorithms that extract private input attributes from published predicted values. Through a case study of personalized adjustment of Warfaline dosage, Fredrikson et al. (2014) showed that publishing predicted dosage amount can cause leakage of private input attributes (e.g., personal genetic information) in generalized linear regression. Fredrikson et al. (2015) presented a model inversion attack that re-

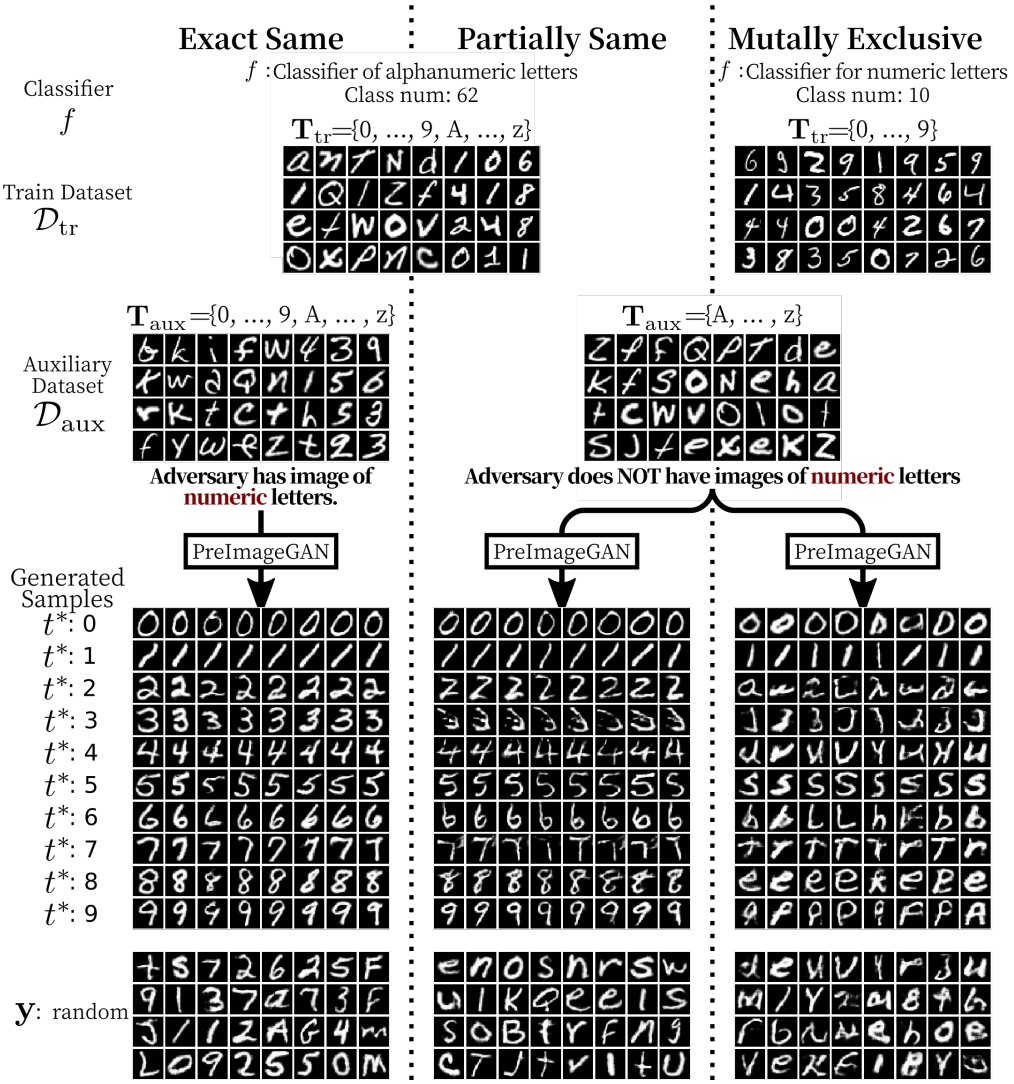

Figure 2: **EMNIST: The results of C2G attack against hand-written alphanumeric character classifier with changing the richness of the background knowledge of the adversary.** The samples in the bottom row ("$\mathbf{y}$: random") are generated when $\mathbf{y}$ is randomly drawn from empirically estimated $d_{Y_{\text{aux}}}$.

Table 2: **Inception Accuracy with changing background knowledge of the adversary.** As a baseline, ACGAN [Odena et al. (2016)] is trained on $\mathcal{D}_{\text{tr}}$. As the background knowledge of the attacker degrades, the inception accuracy decreases.

	target label: t^*									
Setting	0	1	2	3	4	5	6	7	8	9
Baseline (ACGAN)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Exact same	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00
Partially same	1.00	1.00	0.89	0.71	0.99	1.00	0.90	0.53	0.79	0.99
Mutually exclusive	0.97	1.00	0.59	0.39	0.86	0.98	0.97	0.61	0.72	0.48

constructs face images from a face recognition model. The significant difference between the C2G attack and the model inversion attack is the goal of the adversary. In the model inversion attack, the adversary tries to estimate a private input (or input attributes) $\mathbf{x}$ from predicted values $\mathbf{y} = f(\mathbf{x})$

Figure 3: **FaceScrub: The results of the C2G attack against face recognition in the mutually exclusive setting.** We trained a face recognition model of 100 people (including Brad Pitt, Keanu Reeves, Nicolas Cage and Marg Helgenberger), and evaluated the C2G attack where the classification model for the 100 people is given to the adversary while no face images of the 100 people are not given. Generated samples are randomly generated, and we did not cherry-pick "good" samples. We can recognize the generated face images as those of the target label. This indicates that the C2G attack works successfully for the face recognition model.

using the predictor f. Thus, the adversary's goal is the private input $\mathbf{x}$ itself in the model inversion attack. By contrast, in the C2G attack, the adversary's goal is to obtain the training sample distribution. Another difference is that the target network model. The target model of Fredrikson et al. (2015) was a shallow neural network model while ours is deep neural networks. As the network ar-

chitecture becomes deeper, it becomes more difficult to extract information about the input because the output of the model tends to be more abstract (Hitaj et al. (2017)).

Hitaj et al. (2017) discussed leakage of training samples in collaborative learning based on the model inversion attack using the IcGAN (Perarnau et al. (2016)). In their setting, the adversary's goal is not to estimate training sample distribution but to extract training samples. Also, their demonstration is limited to small-scale datasets, such as MNIST dataset (hand-written digit grayscale images, 10 labels) and AT&T dataset (400 face grayscale images with 40 labels). By contrast, our experiments are demonstrated with larger datasets, such as EMNIST dataset (62 labels) and FaceScrub dataset (530 labels, 100,000+ color images).

Hayes et al. (2017) discussed the membership inference attack against a generative model trained by BEGAN (Berthelot et al. (2017)) or DCGAN (Radford et al. (2015)). In the membership inference attack, the adversary's goal is to determine whether the sample is contained in the private training dataset; the problem and the goal are apparently different from ours.

Song et al. (2017) discussed malicious regularizer to memorize private training dataset when the adversary can specify the learning algorithm and obtain the classifier trained on the private training data. Their experiments showed that the adversary can estimate training data samples from the classifier when the classifier is trained with malicious regularizer. Since our setting does not assume that the adversary can specify the learning algorithm, the problem setting is apparently different from ours.

Mahendran & Vedaldi (2015) and Mahendran & Vedaldi (2016) consider the understanding representation of deep neural networks through reconstruction of input images from intermediate features. Their studies are related to ours in the sense that the algorithm exploits intermediate features to attain the goal (correct?). To the best of our knowledge, no attack algorithm has been presented to estimate private training sample distribution as the C2G attack achieves.

7 Conclusion

As described in this paper, we formulated the Classifier-to-Generator (C2G) Attack, which estimates the training sample distribution ρ_{t^*} from given classification model f and auxiliary dataset $\mathcal{D}_{\mathrm{tr}}$. As an algorithm for C2G attack, we proposed PreImageGAN which is based on ACGAN and WGAN. The proposed method can estimate the sample generation model using the interpolation ability of GANs even if the adversary does not have samples with training data labels. In experiments, we demonstrated the performance of C2G attack against handwritten character classifier and face recognition model. Experimental results show that the adversary can estimate the training sample distribution ρ_{t^*} even when the adversary does not have samples with training data labels.

Acknowledgments

To be written.

References

M. Arjovsky and L. Bottou. Towards Principled Methods for Training Generative Adversarial Networks. *ArXiv e-prints*, January 2017.

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *ArXiv e-prints*, January 2017.

David Berthelot, Tom Schumm, and Luke Metz. BEGAN: boundary equilibrium generative adversarial networks. *CoRR*, abs/1703.10717, 2017. URL `http://arxiv.org/abs/1703.10717`.

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*, CCS '15, pp. 1322–1333, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3832-5. doi: 10.1145/2810103.2813677. URL `http://doi.acm.org/10.1145/2810103.2813677`.

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)*, pp. 17–32, San Diego, CA, 2014. USENIX Association. ISBN 978-1-931971-15-7.

Jon Gauthier. Conditional generative adversarial nets for convolutional face generation. *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester*, 2014.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Ian J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160, 2017. URL http://arxiv.org/abs/1701.00160.

Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. *CoRR*, abs/1704.00028, 2017. URL http://arxiv.org/abs/1704.00028.

Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. LOGAN: evaluating privacy leakage of generative models using generative adversarial networks. *CoRR*, abs/1705.07663, 2017. URL http://arxiv.org/abs/1705.07663.

Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-Cruz. Deep models under the GAN: information leakage from collaborative deep learning. *CoRR*, abs/1702.07464, 2017. URL http://arxiv.org/abs/1702.07464.

Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 5188–5196, 2015. doi: 10.1109/CVPR.2015.7299155. URL https://doi.org/10.1109/CVPR.2015.7299155.

Aravindh Mahendran and Andrea Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision*, 120(3):233–255, 2016. doi: 10.1007/s11263-016-0911-8. URL https://doi.org/10.1007/s11263-016-0911-8.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014. URL http://arxiv.org/abs/1411.1784.

Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *CoRR*, abs/1412.1897, 2014. URL http://arxiv.org/abs/1412.1897.

A. Odena, C. Olah, and J. Shlens. Conditional Image Synthesis With Auxiliary Classifier GANs. *ArXiv e-prints*, October 2016.

G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez. Invertible Conditional GANs for image editing. *ArXiv e-prints*, November 2016.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015. URL http://arxiv.org/abs/1511.06434.

C. Song, T. Ristenpart, and V. Shmatikov. Machine Learning Models that Remember Too Much. *ArXiv e-prints*, September 2017.

A MODEL ARCHITECTURE

A.1 EMNIST

EMNIST: Classifier f
Input Domain: $\mathbb{X} = [0,1]^{28\times28\times1}$
Output Domain: $\mathbb{Y} = \Delta^d$

	Kernel size	Stride	Output channels
Convolution	5x5	2x2	32
BatchNormalization			
Relu			
Convolution	5x5	2x2	64
BatchNormalization			
Relu			
Flatten			
Dropout 0.3			
Linear			2048
Dropout 0.5			
Linear			d
Softmax			

EMNIST: Generator G
Input Domain: $\mathbb{Z} = [-1,1]^{128}, \mathbb{Y} = \Delta^d$
Output Domain: $\mathbb{X} = [0,1]^{28\times28\times1}$

	Kernel size	Stride	Output channels
Concat ($\mathbf{z}$ and $\mathbf{y}$)			
Linear			7x7x256
Reshape			256
Deconvolution	4x4	2x2	128
LeakyReLU			
Deconvolution	4x4	2x2	64
LeakyReLU			
Deconvolution	3x3	1x1	1
Sigmoid			

EMNIST: Discriminator D
Input Domain: $\mathbb{X} = [0,1]^{28\times28\times1}$
Output Domain: $\mathbb{R}$

	Kernel size	Stride	Output channels
Convolution	3x3	1x1	64
LeakyReLU			
Convolution	4x4	2x2	128
LeakyReLU			
Convolution	3x3	1x1	128
LeakyReLU			
Convolution	4x4	2x2	256
LeakyReLU			
Convolution	3x3	1x1	256
LeakyReLU			
Flatten			
Linear			1

EMNIST: Alternative Classifier f'

Input Domain: $\mathbb{X} = [0,1]^{28\times281}$

Output Domain: $\mathbb{Y} = \Delta^d$

	Kernel size	Resample	Output channels
Convolution	3x3	1x1	32
Bottleneck Residual Block	3x3		32
Bottleneck Residual Block	3x3	Down 2x2	64
Bottleneck Residual Block	3x3		64
Bottleneck Residual Block	3x3	Down 2x2	64
BatchNormalization			
Relu			
Flatten			
Dropout 0.3			
Linear			2048
Dropout 0.5			
Linear			d
Softmax			

A.2 FACESCRUB

FaceScrub: Classifier f

Input Domain: $\mathbb{X} = [-1,1]^{64\times64\times3}$

Output Domain: $\mathbb{Y} = \Delta^d$

	Kernel size	Stride	Output channels
Convolution	5x5	2x2	32
BatchNormalization			
Relu			
Convolution	5x5	2x2	64
BatchNormalization			
Relu			
Convolution	5x5	2x2	128
BatchNormalization			
Relu			
Flatten			
Dropout 0.3			
Linear			2048
Dropout 0.5			
Linear			d
Softmax			

EMNIST: Generator G

Input Domain: $\mathbb{Z} = [-1,1]^{128}, \mathbb{Y} = \Delta^d$

Output Domain: $\mathbb{X} = [-1,1]^{64\times64\times3}$

	Kernel size	Stride	Output channels
Concat ($\mathbf{z}$ and $\mathbf{y}$)			
Linear			4x4x512
Reshape			512
Deconvolution	4x4	2x2	256
LeakyReLU			
Deconvolution	4x4	2x2	128
LeakyReLU			
Deconvolution	4x4	2x2	64
LeakyReLU			
Deconvolution	4x4	2x2	32
LeakyReLU			
Deconvolution	3x3	1x1	3
tanh			

EMNIST: Discriminator D
Input Domain: $\mathbb{X} = [-1, 1]^{64\times64\times3}$
Output Domain: $\mathbb{R}$

	Kernel size	Stride	Output channels
Convolution	3x3	1x1	32
LeakyReLU			
Convolution	4x4	2x2	64
LeakyReLU			
Convolution	3x3	1x1	64
LeakyReLU			
Convolution	4x4	2x2	128
LeakyReLU			
Convolution	3x3	1x1	128
LeakyReLU			
Convolution	4x4	2x2	256
LeakyReLU			
Convolution	3x3	1x1	256
LeakyReLU			
Convolution	4x4	2x2	512
LeakyReLU			
Convolution	3x3	1x1	512
LeakyReLU			
Flatten			
Linear			1