ADVERSARIAL EXAMPLES FOR GENERATIVE MODELS

Jernej Kos
National University of Singapore

Ian Fischer
Google Research

Dawn Song
University of California, Berkeley

ABSTRACT

We explore methods of producing adversarial examples on deep generative models such as the variational autoencoder (VAE) and the VAE-GAN. Deep learning architectures are known to be vulnerable to adversarial examples, but previous work has focused on the application of adversarial examples to classification tasks. Deep generative models have recently become popular due to their ability to model input data distributions and generate realistic examples from those distributions. We present three classes of attacks on the VAE and VAE-GAN architectures and demonstrate them against networks trained on MNIST, SVHN and CelebA. Our first attack leverages classification-based adversaries by attaching a classifier to the trained encoder of the target generative model, which can then be used to indirectly manipulate the latent representation. Our second attack directly uses the VAE loss function to generate a target reconstruction image from the adversarial example. Our third attack moves beyond relying on classification or the standard loss for the gradient and directly optimizes against differences in source and target latent representations. We also motivate why an attacker might be interested in deploying such techniques against a target generative network.

1 INTRODUCTION

Adversarial examples have been shown to exist for a variety of deep learning architectures.[1] They are small perturbations of the original inputs, often barely visible to a human observer, but carefully crafted to misguide the network into producing incorrect outputs. Seminal work by Szegedy et al. (2013) and Goodfellow et al. (2014), as well as much recent work, has shown that adversarial examples are abundant and finding them is easy.

Most previous work focuses on the application of adversarial examples to the task of classification, where the deep network assigns classes to input images. The attack adds small adversarial perturbations to the original input image. These perturbations cause the network to change its classification of the input, from the correct class to some other incorrect class (possibly chosen by the attacker). Critically, the perturbed input must still be recognizable to a human observer as belonging to the original input class.[2]

Deep generative models, such as Kingma & Welling (2013), learn to generate a variety of outputs, ranging from handwritten digits to faces (Kulkarni et al., 2015), realistic scenes (Oord et al., 2016), videos (Kalchbrenner et al., 2016), 3D objects (Dosovitskiy et al., 2016), and audio (van den Oord et al., 2016). These models learn an approximation of the input data distribution in different ways, and then sample from this distribution to generate previously unseen but plausible outputs.

To the best of our knowledge, no prior work has explored using adversarial inputs to attack generative models. There are two main requirements for such work: describing a plausible scenario in which an attacker might want to attack a generative model; and designing and demonstrating an attack that succeeds against generative models. We address both of these requirements in this work.

One of the most basic applications of generative models is input reconstruction. Given an input image, the model first encodes it into a lower-dimensional latent representation, and then uses that representation to generate a reconstruction of the original input image. Since the latent representation

[1] Adversarial examples are even easier to produce against most other machine learning architectures, as shown in Papernot et al. (2016), but we are focused on deep networks.

[2] Random noise images and "fooling" images (Nguyen et al., 2014) do not belong to this strict definition of an adversarial input, although they do highlight other limitations of current classifiers.

usually has much fewer dimensions than the original input, it can be used as a form of compression. The latent representation can also be used to remove some types of noise from inputs, even when the network has not been explicitly trained for denoising, due to the lower dimensionality of the latent representation restricting what information the trained network is able to represent. Many generative models also allow manipulation of the generated output by sampling different latent values or modifying individual dimensions of the latent vectors without needing to pass through the encoding step.

These properties of input reconstruction generative networks suggest a variety of different attacks that would be enabled by effective adversaries against generative networks. Any attack that targets the compression bottleneck of the latent representation can exploit natural security vulnerabilities in applications built to use that latent representation. Specifically, if the person doing the encoding step is separated from the person doing the decoding step, the attacker may be able to cause the encoding party to believe they have encoded a particular message for the decoding party, but in reality they have encoded a different message of the attacker's choosing. We explore this idea in more detail as it applies to the application of compressing images using a VAE or VAE-GAN architecture.

2 Related work and background

This work focuses on adversaries for variational autoencoders (VAEs, proposed in Kingma & Welling (2013)) and VAE-GANs (VAEs composed with a generative adversarial network, proposed in Larsen et al. (2015)).

2.1 Related work on adversaries

Many adversarial attacks on classification models have been described in existing literature (Goodfellow et al., 2014; Szegedy et al., 2013). These attacks can be untargeted, where the adversary's goal is to cause any misclassification, or the least likely misclassification (Goodfellow et al., 2014; Kurakin et al., 2016); or they can be targeted, where the attacker desires a specific misclassification. Moosavi-Dezfooli et al. (2016) gives a recent example of a strong targeted adversarial attack. Some adversarial attacks allow for a threat model where the adversary does not have access to the target model (Szegedy et al., 2013; Papernot et al., 2016), but commonly it is assumed that the attacker does have that access, in an online or offline setting (Goodfellow et al., 2014; Kurakin et al., 2016).[3]

Given a classifier $f(\mathbf{x}) : \mathbf{x} \in \mathcal{X} \rightarrow y \in \mathcal{Y}$ and original inputs $\mathbf{x} \in \mathcal{X}$, the problem of generating *untargeted* adversarial examples can be expressed as the following optimization: $\text{argmin}_{\mathbf{x}^*} L(\mathbf{x}, \mathbf{x}^*) \; s.t. \; f(\mathbf{x}^*) \neq f(\mathbf{x})$, where $L(\cdot)$ is a chosen distance measure between examples from the input space (e.g., the L_2 norm). Similarly, generating a *targeted* adversarial attack on a classifier can be expressed as $\text{argmin}_{\mathbf{x}^*} L(\mathbf{x}, \mathbf{x}^*) \; s.t. \; f(\mathbf{x}^*) = y_t$, where $y_t \in \mathcal{Y}$ is some target label chosen by the attacker.

These optimization problems can often be solved with optimizers like L-BFGS or Adam (Kingma & Ba, 2015), as done in Szegedy et al. (2013) and Carlini & Wagner (2016). They can also be approximated with single-step gradient-based techniques like fast gradient sign (Goodfellow et al., 2014), fast gradient L_2 (Huang et al., 2015), or fast least likely class (Kurakin et al., 2016); or they can be approximated with iterative variants of those and other gradient-based techniques (Kurakin et al., 2016; Moosavi-Dezfooli et al., 2016).

An interesting variation of this type of attack can be found in Sabour et al. (2015). In that work, they attack the hidden state of the target network directly by taking an input image $\mathbf{x}$ and a target image $\mathbf{x}_t$ and searching for a perturbed variant of $\mathbf{x}$ that generates similar hidden state at layer l of the target network to the hidden state at the same layer generated by $\mathbf{x}_t$. This approach can also be applied directly to attacking the latent vector of a generative model.

A variant of this attack has also been applied to VAE models in the concurrent work of Tabacof et al. (2016)[4], which uses the KL divergence between the latent representation of the source and target images to generate the adversarial example. However in their paper, the authors mention that they tried attacking the output directly and that this only managed to make the reconstructions more

[3] See Papernot et al. (2015) for an overview of different adversarial threat models.

[4] This work was made public shortly after we published our early drafts.

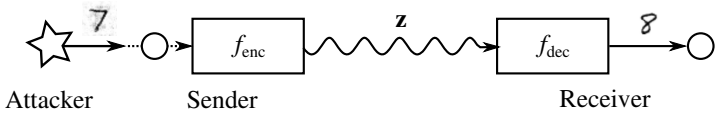

Figure 1: Depiction of the attack scenario. The VAE is used as a compression scheme to transmit a latent representation of the image from the sender (left) to the receiver (right). The attacker convinces the sender to compress a particular image into its latent vector, which is sent to the receiver, where the decoder reconstructs the latent vector into some other image chosen by the attacker.

blurry. While they do not explain the exact experimental setting, the attack sounds similar to our $\mathcal{L}_{\text{VAE}}$ attack, which we find very successful. Also, in their paper the authors do not consider the more advanced VAE-GAN models and more complex datasets like CelebA.

2.2 Background on VAEs and VAE-GANs

The general architecture of a variational autoencoder consists of three components, as shown in Figure 8. The **encoder** $f_{\text{enc}}(\mathbf{x})$ is a neural network mapping a high-dimensional input representation $\mathbf{x}$ into a lower-dimensional (compressed) **latent representation** $\mathbf{z}$. All possible values of $\mathbf{z}$ form a latent space. Similar values in the latent space should produce similar outputs from the decoder in a well-trained VAE. And finally, the **decoder/generator** $f_{\text{dec}}(\mathbf{z})$, which is a neural network mapping the compressed latent representation back to a high-dimensional output $\hat{\mathbf{x}}$. Composing these networks allows basic input reconstruction $\hat{\mathbf{x}} = f_{\text{dec}}(f_{\text{enc}}(\mathbf{x}))$. This composed architecture is used during training to backpropagate errors from the loss function.

The variational autoencoder's loss function $\mathcal{L}_{\text{VAE}}$ enables the network to learn a latent representation that approximates the intractable posterior distribution $p(\mathbf{z}|\mathbf{x})$:

$$\mathcal{L}_{\text{VAE}} = -D_{\text{KL}}[q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})] + E_q[\log p(\mathbf{x}|\mathbf{z})]. \tag{1}$$

$q(\mathbf{z}|\mathbf{x})$ is the learned approximation of the posterior distribution $p(\mathbf{z}|\mathbf{x})$. $p(\mathbf{z})$ is the prior distribution of the latent representation $\mathbf{z}$. D_{KL} denotes the Kullback–Leibler divergence. $E_q[\log p(\mathbf{x}|\mathbf{z})]$ is the variational lower bound, which in the case of input reconstruction is the cross-entropy $H[\mathbf{x}, \hat{\mathbf{x}}]$ between the inputs $\mathbf{x}$ and their reconstructions $\hat{\mathbf{x}}$. In order to generate $\hat{\mathbf{x}}$ the VAE needs to sample $q(\mathbf{z}|\mathbf{x})$ and then compute $f_{\text{dec}}(\mathbf{z})$.

For the VAE to be fully differentiable while sampling from $q(\mathbf{z}|\mathbf{x})$, the reparametrization trick (Kingma & Welling, 2013) extracts the random sampling step from the network and turns it into an input, $\boldsymbol{\varepsilon}$. VAEs are often parameterized with Gaussian distributions. In this case, $f_{\text{enc}}(\mathbf{x})$ outputs the distribution parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$. That distribution is then sampled by computing $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\varepsilon}\sqrt{\boldsymbol{\sigma}^2}$ where $\boldsymbol{\varepsilon} \sim N(0, 1)$ is the input random sample, which does not depend on any parameters of f_{enc}, and thus does not impact differentiation of the network.

The VAE-GAN architecture of Larsen et al. (2015) has the same f_{enc} and f_{dec} pair as in the VAE. It also adds a discriminator f_{disc} that is used during training, as in standard generative adversarial networks (Goodfellow et al., 2014). The loss function of f_{dec} uses the disciminator loss instead of cross-entropy for estimating the reconstruction error.

3 Problem definition

We provide a motivating attack scenario for adversaries against generative models, as well as a formal definition of an adversary in the generative setting.

3.1 Motivating attack scenario

To motivate the attacks presented below, we describe the attack scenario depicted in Figure 1. In this scenario, there are two parties, the sender and the receiver, who wish to share images with each other over a computer network. In order to conserve bandwidth, they share a VAE trained on the input distribution of interest, which will allow them to send only latent vectors $\mathbf{z}$.

Figure 2: Results for the L_2 optimization latent attack (see Section 4.3) on the VAE-GAN, targeting a specific image from the class 0. Shown are the first 12 non-zero images from the test SVHN data set. The columns are, in order: the original image, the reconstruction of the original image, the adversarial example, the predicted class of the adversarial example, the reconstruction of the adversarial example, the predicted class of the reconstructed adversarial example, the reconstruction of the reconstructed adversarial example (see Section 4.5), and the predicted class of that reconstruction.

The attacker's goal is to convince the sender to send an image of the attacker's choosing to the receiver, but the attacker has no direct control over the bytes sent between the two parties. However, the attacker has a copy of the shared VAE. The attacker presents an image $\mathbf{x}^*$ to the sender which resembles an image $\mathbf{x}$ that the sender wants to share with the receiver. For example, the sender wants to share pictures of kittens with the receiver, so the attacker presents a web page to the sender with a picture of a kitten, which is $\mathbf{x}^*$. The sender chooses $\mathbf{x}^*$ and sends its corresponding $\mathbf{z}$ to the receiver, who reconstructs it. However, because the attacker controlled the chosen image, when the receiver reconstructs it, instead of getting a faithful reproduction $\hat{\mathbf{x}}$ of $\mathbf{x}$ (e.g., a kitten), the receiver sees some other image of the attacker's choosing, $\hat{\mathbf{x}}_{\text{adv}}$, which has a different meaning from $\mathbf{x}$ (e.g., a request to send money to the attacker's bank account).

There are other attacks of this general form, where the sender and the receiver may be separated by distance, as in this example, or by time, in the case of storing compressed images to disk for later retrieval. In the time-separated attack, the sender and the receiver may be the same person or multiple different people. In either case, if they are using the insecure channel of the VAE's latent space, the messages they share may be under the control of an attacker. For example, an attacker may be able to fool an automatic surveillance system if the system uses this type of compression to store the video signal before it is processed by other systems. In this case, the subsequent analysis of the video signal could be on compromised data showing what the attacker wants to show.

While we do not specifically attack their models, viable compression schemes based on deep neural networks have already been proposed in the literature, showing promising results Toderici et al. (2015; 2016).

3.2 Defining adversarial examples against generative models

We make the following assumptions about generating adversarial examples on a target generative model, $G_{\text{targ}}(\mathbf{x}) = f_{\text{dec}}(f_{\text{enc}}(\mathbf{x}))$. G_{targ} is trained on inputs $\mathcal{X}$ that can naturally be labeled with semantically meaningful classes $\mathcal{Y}$, although there may be no such labels at training time, or the labels may not have been used during training. G_{targ} normally succeeds at generating an output $\hat{\mathbf{x}} = G_{\text{targ}}(\mathbf{x})$ in class y when presented with an input $\mathbf{x}$ from class y. In other words, whatever target output class the attacker is interested in, we assume that G_{targ} successfully captures it in the latent representation such that it can generate examples of that class from the decoder. This target output class does not need to be from the most salient classes in the training dataset. For example, on models trained on MNIST, the attacker may not care about generating different target digits (which are the most salient classes). The attacker may prefer to generate the same input digits in a different style (perhaps to aid forgery). We also assume that the attacker has access to G_{targ}. Finally, the attacker has access to a set of examples from the same distribution as $\mathcal{X}$ that have the target label

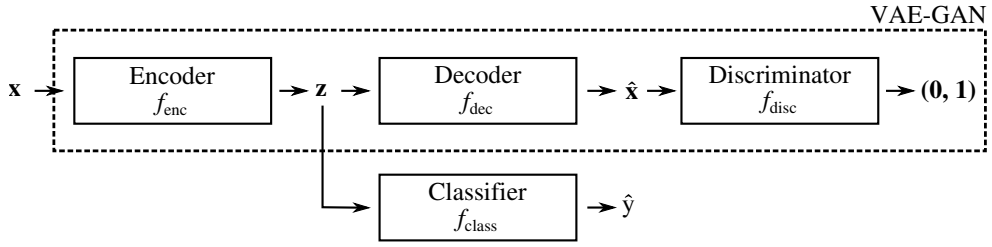

Figure 3: The VAE-GAN classifier architecture used to generate classifier-based adversarial examples on the VAE-GAN. The VAE-GAN in the dashed box is the target network and is frozen while training the classifier. The path $\mathbf{x} \rightarrow f_{\text{enc}} \rightarrow \mathbf{z} \rightarrow f_{\text{class}} \rightarrow \hat{y}$ is used to generate adversarial examples in $\mathbf{z}$, which can then be reconstructed by f_{dec}.

y_t the attacker wants to generate. This does not mean that the attacker needs access to the labeled training dataset (which may not exist), or to an appropriate labeled dataset with large numbers of examples labeled for each class $y \in \mathcal{Y}$ (which may be hard or expensive to collect). The attacks described here may be successful with only a small amount of data labeled for a single target class of interest.

One way to generate such adversaries is by solving the optimization problem $\text{argmin}_{\mathbf{x}^*} L(\mathbf{x}, \mathbf{x}^*)$ $s.t.$ $\text{ORACLE}(G_{\text{targ}}(\mathbf{x}^*)) = y_t$, where ORACLE reliably discriminates between inputs of class y_t and inputs of other classes. In practice, a classifier trained by the attacker may server as ORACLE. Other types of adversaries from Section 2.1 can also be used to approximate this optimization in natural ways, some of which we describe in Section 4.

If the attacker only needs to generate one successful attack, the problem of determining if an attack is successful can be solved by manually reviewing the $\mathbf{x}^*$ and $\hat{\mathbf{x}}_{\text{adv}}$ pairs and choosing whichever the attacker considers best. However, if the attacker wants to generate many successful attacks, an automated method of evaluating the success of an attack is necessary. We show in Section 4.5 how to measure the effectiveness of an attack automatically using a classifier trained on $\mathbf{z} = f_{\text{enc}}(\mathbf{x})$.

4 ATTACK METHODOLOGY

The attacker would like to construct an adversarially-perturbed input to influence the latent representation in a way that will cause the reconstruction process to reconstruct an output for a different class. We propose three approaches to attacking generative models: a classifier-based attack, where we train a new classifier on top of the latent space $\mathbf{z}$ and use that classifier to find adversarial examples in the latent space; an attack using $\mathcal{L}_{\text{VAE}}$ to target the output directly; and an attack on the latent space, $\mathbf{z}$. All three methods are technically applicable to any generative architecture that relies on a learned latent representation $\mathbf{z}$. Without loss of generality, we focus on the VAE-GAN architecture.

4.1 CLASSIFIER ATTACK

By adding a classifier f_{class} to the pre-trained generative model[5], we can turn the problem of generating adversaries for generative models back into the previously solved problem of generating adversarial examples for classifiers. This approach allows us to apply all of the existing attacks on classifiers in the literature. However, as discussed below, using this classifier tends to produce lower-quality reconstructions from the adversarial examples than the other two attacks due to the inaccuracies of the classifier.

Step 1. The weights of the target generative model are frozen, and a new classifier $f_{\text{class}}(\mathbf{z}) \rightarrow \hat{y}$ is trained on top of f_{enc} using a standard classification loss $\mathcal{L}_{\text{classifier}}$ such as cross-entropy, as shown in Figure 3. This process is independent of how the original model is trained, but it requires a

[5] This is similar to the process of semi-supervised learning in Kingma et al. (2014), although the goal is different.

training corpus pulled from approximately the same input distribution as was used to train G_{targ}, with ground truth labels for at least two classes: y_t and $y_{\tilde{t}}$, the negative class.

Step 2. With the trained classifier, the attacker finds adversarial examples $\mathbf{x}^*$ using the methods described in Section 4.4.

Using f_{class} to generate adversarial examples does not always result in high-quality reconstructions, as can be seen in the middle column of Figure 5 and in Figure 11. This appears to be due to the fact that f_{class} adds additional noise to the process. For example, f_{class} sometimes confidently misclassifies latent vectors $\mathbf{z}$ that represent inputs that are far from the training data distribution, resulting in f_{dec} failing to reconstruct a plausible output from the adversarial example.

4.2 $\mathcal{L}_{\text{VAE}}$ ATTACK

Our second approach generates adversarial perturbations using the VAE loss function. The attacker chooses two inputs, $\mathbf{x}_s$ (the source) and $\mathbf{x}_t$ (the target), and uses one of the standard adversarial methods to perturb $\mathbf{x}_s$ into $\mathbf{x}^*$ such that its reconstruction $\hat{\mathbf{x}}^*$ matches the reconstruction of $\mathbf{x}_t$, using the methods described in Section 4.4.

The adversary precomputes the reconstruction $\hat{\mathbf{x}}_t$ by evaluating $f_{\text{dec}}(f_{\text{enc}}(\mathbf{x}_t))$ once before performing optimization. In order to use $\mathcal{L}_{\text{VAE}}$ in an attack, the second term (the reconstruction loss) of $\mathcal{L}_{\text{VAE}}$ (see Equation 1) is changed so that instead of computing the reconstruction loss between $\mathbf{x}$ and $\hat{\mathbf{x}}$, the loss is computed between $\hat{\mathbf{x}}^*$ and $\hat{\mathbf{x}}_t$. This means that during each optimization iteration, the adversary needs to compute $\hat{\mathbf{x}}^*$, which requires the full $f_{\text{dec}}(f_{\text{enc}}(\mathbf{x}^*))$ to be evaluated.

4.3 LATENT ATTACK

Our third approach attacks the latent space of the generative model.

Single latent vector target. This attack is similar to the work of Sabour et al. (2015), in which they use a pair of source image $\mathbf{x}_s$ and target image $\mathbf{x}_t$ to generate $\mathbf{x}^*$ that induces the target network to produce similar activations at some hidden layer l as are produced by $\mathbf{x}_t$, while maintaining similarity between $\mathbf{x}_s$ and $\mathbf{x}^*$.

For this attack to work on latent generative models, it is sufficient to compute $\mathbf{z}_t = f_{\text{enc}}(\mathbf{x}_t)$ and then use the following loss function to generate adversarial examples from different source images $\mathbf{x}_s$, using the methods described in Section 4.4:

$$\mathcal{L}_{\text{latent}} = L(\mathbf{z}_t, f_{\text{enc}}(\mathbf{x}^*)). \tag{2}$$

$L(\cdot)$ is a distance measure between two vectors. We use the L_2 norm, under the assumption that the latent space is approximately euclidean.

We also explored a variation on the single latent vector target attack, which we describe in Section A.1 in the Appendix.

4.4 METHODS FOR SOLVING THE ADVERSARIAL OPTIMIZATION PROBLEM

We can use a number of different methods to generate the adversarial examples. We initially evaluated both the fast gradient sign Goodfellow et al. (2014) method and an L_2 optimization method. As the latter produces much better results we focus on the L_2 optimization method, while we include some FGS results in the Appendix. The attack can be used either in targeted mode (where we want a specific class, y_t, to be reconstructed) or untargeted mode (where we just want an incorrect class to be reconstructed). In this paper, we focus on the targeted mode of the attacks.

L_2 optimization. The optimization-based approach, explored in Szegedy et al. (2013) and Carlini & Wagner (2016), poses the adversarial generation problem as the following optimization problem:

$$\text{argmin}_{\mathbf{x}^*} \lambda L(\mathbf{x}, \mathbf{x}^*) + \mathcal{L}(\mathbf{x}^*, y_t). \tag{3}$$

As above, $L(\cdot)$ is a distance measure, and $\mathcal{L}$ is one of $\mathcal{L}_{\text{classifier}}$, $\mathcal{L}_{\text{VAE}}$, or $\mathcal{L}_{\text{latent}}$. The constant λ is used to balance the two loss contributions. For the $\mathcal{L}_{\text{VAE}}$ attack, the optimizer must do a full

reconstruction at each step of the optimizer. The other two attacks do not need to do reconstructions while the optimizer is running, so they generate adversarial examples much more quickly, as shown in Table 1.

4.5 MEASURING ATTACK EFFECTIVENESS

To generate a large number of adversarial examples automatically against a generative model, the attacker needs a way to judge the quality of the adversarial examples. We leverage f_{class} to estimate whether a particular attack was successful.[6]

Reconstruction feedback loop. The architecture is the same as shown in Figure 3. We use the generative model to reconstruct the attempted adversarial inputs $\mathbf{x}^*$ by computing:

$$\hat{\mathbf{x}}^* = f_{\text{dec}}(f_{\text{enc}}(\mathbf{x}^*)). \tag{4}$$

Then, f_{class} is used to compute:

$$\hat{y} = f_{\text{class}}(f_{\text{enc}}(\hat{\mathbf{x}}^*)). \tag{5}$$

The input adversarial examples $\mathbf{x}^*$ are not classified directly, but are first fed to the generative model for reconstruction. This reconstruction loop improves the accuracy of the classifier by 60% on average against the adversarial attacks we examined. The predicted label $\hat{y}$ after the reconstruction feedback loop is compared with the attack target y_t to determine if the adversarial example successfully reconstructed to the target class. If the precision and recall of f_{class} are sufficiently high on y_t, f_{class} can be used to filter out most of the failed adversarial examples while keeping most of the good ones.

We derive two metrics from classifier predictions after one reconstruction feedback loop. The first metric is $AS_{ignore-target}$, the attack success rate ignoring targeting, i.e., without requiring the output class of the adversarial example to match the target class:

$$AS_{ignore-target} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{\hat{y}^i \neq y^i} \tag{6}$$

N is the total number of reconstructed adversarial examples; $\mathbf{1}_{\hat{y}^i \neq y^i}$ is 1 when $\hat{y}^i$, the classification of the reconstruction for image i, does not equal y^i, the ground truth classification of the original image, and 0 otherwise. The second metric is AS_{target}, the attack success rate including targeting (i.e., requiring the output class of the adversarial example to match the target class), which we define similarly as:

$$AS_{target} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{\hat{y}^i = y_t^i}. \tag{7}$$

Both metrics are expected to be higher for more successful attacks. Note that $AS_{target} \leq AS_{ignore-target}$. When computing these metrics, we exclude input examples that have the same ground truth class as the target class.

5 EVALUATION

We evaluate the three attacks on MNIST (LeCun et al., 1998), SVHN (Netzer et al., 2011) and CelebA (Liu et al., 2015), using the standard training and validation set splits. The VAE and VAE-GAN architectures are implemented in TensorFlow (Abadi & et al., 2015). We optimized using Adam with learning rate 0.001 and other parameters set to default values for both the generative model and the classifier. For the VAE, we use two architectures: a simple architecture with a single fully-connected hidden layer with 512 units and ReLU activation function; and a convolutional architecture taken from the original VAE-GAN paper Larsen et al. (2015) (but trained with only the VAE loss). We use the same architecture trained with the additional GAN loss for the VAE-GAN model, as described in that work. For both VAE and VAE-GAN we use a 50-dimensional latent representation on MNIST, a 1024-dimensional latent representation on SVHN and 2048-dimensional latent representation on CelebA.

[6] Note that f_{class} here is being used in a different manner than when we use it to generate adversarial examples. However, the network itself is identical, so we don't distinguish between the two uses in the notation.

Figure 4: Results for the L_2 optimization latent attack on the VAE-GAN, targeting the mean latent vector for 0. Shown are the first 12 non-zero images from the test MNIST data set. The columns are, in order: the original image, the reconstruction of the original image, the adversarial example, the predicted class of the adversarial example, the reconstruction of the adversarial example, the predicted class of the reconstructed adversarial example, the reconstruction of the reconstructed adversarial example (see Section 4.5), and the predicted class of that reconstruction.

In this section we only show results where no sampling from latent space has been performed. Instead we use the mean vector $\boldsymbol{\mu}$ as the latent representation $\mathbf{z}$. As sampling can have an effect on the resulting reconstructions, we evaluated it separately. We show the results with different number of samples in Figure 22 in the Appendix. On most examples, the visible change is small and in general the attack is still successful.

5.1 MNIST

Both VAE and VAE-GAN by themselves reconstruct the original inputs well as show in Figure 9, although the quality from the VAE-GAN is noticeably better. As a control, we also generate random noise of the same magnitude as used for the adversarial examples (see Figure 13), to show that random noise does not cause the reconstructed noisy images to change in any significant way. Although we ran experiments on both VAEs and VAE-GANs, we only show results for the VAE-GAN as it generates much higher quality reconstructions than the corresponding VAE.

5.1.1 Classifier attack

We use a simple classifier architecture to help generate attacks on the VAE and VAE-GAN models. The classifier consists of two fully-connected hidden layers with 512 units each, using the ReLU activation function. The output layer is a 10 dimensional softmax. The input to the classifier is the 50 dimensional latent representation produced by the VAE/VAE-GAN encoder. The classifier achieves 98.05% accuracy on the validation set after training for 100 epochs.

To see if there are differences between classes, we generate targeted adversarial examples for each MNIST class and present the results per-class. For the targeted attacks we used the optimization method with lambda 0.001, where Adam-based optimization was performed for 1000 epochs with a learning rate of 0.1. The mean L_2 norm of the difference between original images and generated adversarial examples using the classifier attack is 3.36, while the mean RMSD is 0.120.

Numerical results in Table 2 show that the targeted classifier attack successfully fools the classifier. Classifier accuracy is reduced to 0%, while the matching rate (the ratio between the number of predictions matching the target class and the number of incorrectly classified images) is 100%, which means that all incorrect predictions match the target class. However, what we are interested in (as per the attack definition from Section 3.2) is how the generative model reconstructs the adversarial examples. If we look at the images generated by the VAE-GAN for class 0, shown in Figure 4, the targeted attack is successful on some reconstructed images (e.g. one, four, five, six and nine are reconstructed as zeroes). But even when the classifier accuracy is 0% and matching rate is 100%, an incorrect classification does not always result in a reconstruction to the target class, which shows that the classifier is fooled by an adversarial example more easily than the generative model.

Reconstruction feedback loop. The reconstruction feedback loop described in Section 4.5 can be used to measure how well a targeted attack succeeds in making the generative model change the

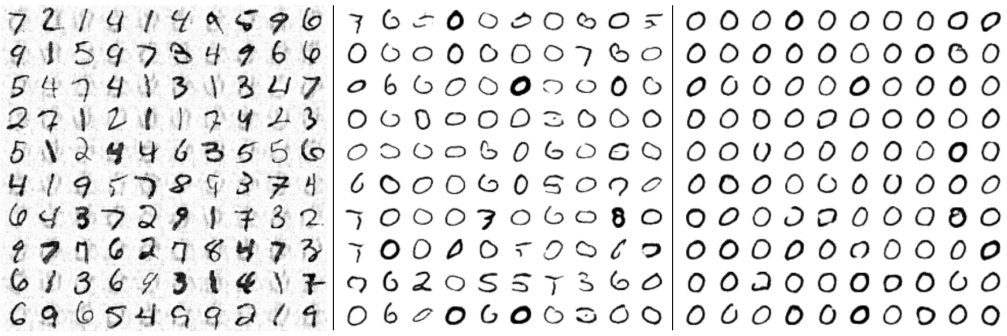

Figure 5: *Left:* representative adversarial examples with a target class of 0 on the first 100 non-zero images from the MNIST validation set. These were produced using the L_2 optimization latent attack (Section 4.3). *Middle:* VAE-GAN reconstructions from adversarial examples produced using the L_2 optimization classifier attack on the same set of 100 validation images (those adversaries are not shown, but are qualitatively similiar, see Section 4.1). *Right:* VAE-GAN reconstructions from the adversarial examples in the left column. Many of the classifier adversarial examples fail to reconstruct as zeros, whereas almost every adversarial example from the latent attack reconstructs as zero.

reconstructed classes. Table 4 in the Appendix shows $AS_{ignore-target}$ and AS_{target} for all source and target class pairs. A higher value signifies a more successful attack for that pair of classes. It is interesting to observe that attacking some source/target pairs is much easier than others (e.g. pair $(4, 0)$ vs. $(0, 8)$) and that the results are not symmetric over source/target pairs. Also, some pairs do well in $AS_{ignore-target}$, but do poorly in AS_{target} (e.g., all source digits when targeting 4). As can be seen in Figure 11, the classifier adversarial examples targeting 4 consistently fail to reconstruct into something easily recognizable as a 4. Most of the reconstructions look like 5, but the adversarial example reconstructions of source 5s instead look like 0 or 3.

5.1.2 $\mathcal{L}_{\text{VAE}}$ ATTACK

For generating adversarial examples using the $\mathcal{L}_{\text{VAE}}$ attack, we used the optimization method with $\lambda = 1.0$, where Adam-based optimization was performed for 1000 epochs with a learning rate of 0.1. The mean L_2 norm of the difference between original images and generated adversarial examples with this approach is 3.68, while the mean RMSD is 0.131.

We show $AS_{ignore-target}$ and AS_{target} of the $\mathcal{L}_{\text{VAE}}$ attack in Table 5 in the Appendix. Comparing with the numerical evaluation results of the latent attack (below), we can see that both methods achieve similar results on MNIST.

5.1.3 LATENT ATTACK

To generate adversarial examples using the latent attack, we used the optimization method with $\lambda = 1.0$, where Adam-based optimization was performed for 1000 epochs with a learning rate of 0.1. The mean L_2 norm of the difference between original images and generated adversarial examples using this approach is 2.96, while the mean RMSD is 0.105.

Table 3 shows $AS_{ignore-target}$ and AS_{target} for all source and target class pairs. Comparing with the numerical evaluation results of the classifier attack we can see that the latent attack performs much better. This result remains true when visually comparing the reconstructed images, shown in Figure 5.

We also tried an untargeted version of the latent attack, where we change Equation 2 to maximize the distance in latent space between the encoding of the original image and the encoding of the adversarial example. In this case the loss we are trying to minimize is unbounded, since the L_2 distance can always grow larger, so the attack normally fails to generate a reasonable adversarial example.

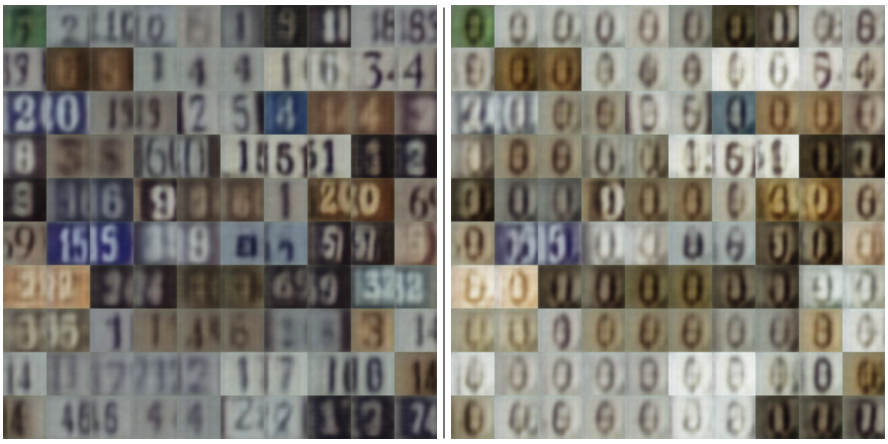

Figure 6: *Left:* VAE-GAN reconstructions of adversarial examples generated using the L_2 optimization $\mathcal{L}_{\text{VAE}}$ attack (single image target). *Right:* VAE-GAN reconstructions of adversarial examples generated using the L_2 optimization latent attack (single image target). Approximately 85 out of 100 images are convincing zeros for the L_2 latent attack, whereas only about 5 out of 100 could be mistaken for zeros with the $\mathcal{L}_{\text{VAE}}$ attack.

Additionally, we also experimented with targeting latent representations of specific images from the training set instead of taking the mean, as described in Section 4.3. We show the numerical results in Table 3 and the generated reconstructions in Figure 15 (in the Appendix). It is also interesting to compare the results with $\mathcal{L}_{\text{VAE}}$, by choosing the same image as the target. Results for $\mathcal{L}_{\text{VAE}}$ for the same target images as in Table 3 are shown in Table 6 in the Appendix. The results are identical between the two attacks, which is expected as the target image is the same – only the loss function differs between the methods.

5.2 SVHN

The SVHN dataset consists of cropped street number images and is much less clean than MNIST. Due to the way the images have been processed, each image may contain more than one digit; the target digit is roughly in the center. VAE-GAN produces high-quality reconstructions of the original images as shown in Figure 17 in the Appendix.

For the classifier attack, we set $\lambda = 10^{-5}$ after testing a range of values, although we were unable to find an effective value for this attack against SVHN. For the latent and $\mathcal{L}_{\text{VAE}}$ attacks we set $\lambda = 10$.

In Table 10 we show $AS_{ignore-target}$ and AS_{target} for the L_2 optimization latent attack. The evaluation metrics are less strong on SVHN than on MNIST, but it is still straightforward for an attacker to find a successful attack for almost all source/target pairs. Figure 2 supports this evaluation. Visual inspection shows that 11 out of the 12 adversarial examples reconstructed as 0, the target digit. It is worth noting that 2 out of the 12 adversarial examples look like zeros (rows 1 and 11), and two others look like both the original digit and zero, depending on whether the viewer focuses on the light or dark areas of the image (rows 4 and 7). The L_2 optimization latent attack achieves much better results than the $\mathcal{L}_{\text{VAE}}$ attack (see Table 11 and Figure 6) on SVHN, while both attacks work equally well on MNIST.

5.3 CelebA

The CelebA dataset consists of more than 200,000 cropped faces of celebrities, each annotated with 40 different attributes. For our experiments, we further scale the images to 64x64 and ignore the attribute annotations. VAE-GAN reconstructions of original images after training are shown in Figure 19 in the Appendix.

Since faces don't have natural classes, we only evaluated the latent and $\mathcal{L}_{\text{VAE}}$ attacks. We tried lambdas ranging from 0.1 to 0.75 for both attacks. Figure 20 shows adversarial examples generated

Method	MNIST			SVHN		
	Mean L_2	Mean RMSD	Time to attack	Mean L_2	Mean RMSD	Time to attack
L_2 Optimization Classifier Attack	3.36	0.120	277	1.77	0.032	274
L_2 Optimization $\mathcal{L}_{\text{VAE}}$ Attack	3.68	0.131	734	2.36	0.043	895
L_2 Optimization Latent Attack	2.96	0.105	236	2.80	0.051	242

Table 1: Comparison of mean L_2 norm and RMSD between the original images and the generated adversarial examples for the different attacks. *Time to attack* is the mean number of seconds it takes to generate 1000 adversarial examples using the given attack method (with the same number of optimization iterations for each attack).

using the latent attack and a lambda value of 0.5 (L_2 norm between original images and generated adversarial examples 9.78, RMSD 0.088) and the corresponding VAE-GAN reconstructions. Most of the reconstructions reflect the target image very well. We get even better results with the $\mathcal{L}_{\text{VAE}}$ attack, using a lambda value of 0.75 (L_2 norm between original images and generated adversarial examples 8.98, RMSD 0.081) as shown in Figure 21.

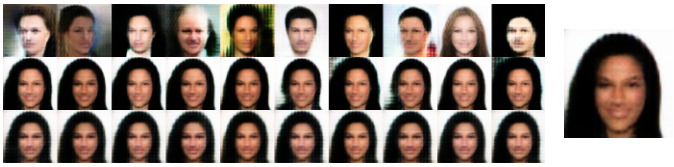

Figure 7: Summary of different attacks on CelebA dataset: reconstructions of original images (top), reconstructions of adversarial examples generated using the latent attack (middle) and $\mathcal{L}_{\text{VAE}}$ attack (bottom). Target reconstruction is shown on the right. Full results are in the Appendix.

5.4 Summary of different attack methods

Table 1 shows a comparison of the mean distances between original images and generated adversarial examples for the three different attack methods. The larger the distance between the original image and the adversarial perturbation, the more noticeable the perturbation will tend to be, and the more likely a human observer will no longer recognize the original input, so effective attacks keep these distances small while still achieving their goal. The latent attack consistently gives the best results in our experiments, and the classifier attack performs the worst.

We also measure the time it takes to generate 1000 adversarial examples using the given attack method. The $\mathcal{L}_{\text{VAE}}$ attack is by far the slowest of the three, due to the fact that it requires computing full reconstructions at each step of the optimizer when generating the adversarial examples. The other two attacks do not need to run the reconstruction step during optimization of the adversarial examples.

6 Conclusion

We explored generating adversarial examples against generative models such as VAEs and VAE-GANs. These models are also vulnerable to adversaries that convince them to turn inputs into surprisingly different outputs. We have also motivated why an attacker might want to attack generative models. Our work adds further support to the hypothesis that adversarial examples are a general phenomenon for current neural network architectures, given our successful application of adversarial attacks to popular generative models. In this work, we are helping to lay the foundations for understanding how to build more robust networks. Future work will explore defense and robustification in greater depth as well as attacks on generative models trained using natural image datasets such as CIFAR-10 and ImageNet.

Acknowledgments

This material is in part based upon work supported by the National Science Foundation under Grant No. TWC-1409915. Any opinions, findings, and conclusions or recommendations expressed in this

material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

Martín Abadi and Ashish Agarwal et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. *arXiv preprint arXiv:1608.04644*, 2016.

Alexey Dosovitskiy, Jost Springenberg, Maxim Tatarchenko, and Thomas Brox. Learning to generate chairs, tables and cars with convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99):1–1, 2016. ISSN 0162-8828. doi: 10.1109/TPAMI.2016.2567384.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *ArXiv e-prints*, June 2014.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. Learning with a strong adversary. *CoRR*, abs/1511.03034, 2015.

Nal Kalchbrenner, Aaron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. *arXiv preprint arXiv:1610.00527*, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015.

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

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pp. 3581–3589, 2014.

Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pp. 2539–2547, 2015.

Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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)*, 2015.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. 2016.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

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.

Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *arXiv preprint arXiv:1606.05328*, 2016.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Proceedings of the 1st IEEE European Symposium on Security and Privacy*, 2015.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint arXiv:1602.02697*, 2016.

Sara Sabour, Yanshuai Cao, Fartash Faghri, and David J. Fleet. Adversarial manipulation of deep representations. *CoRR*, abs/1511.05122, 2015. URL `http://arxiv.org/abs/1511.05122`.

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.

P. Tabacof, J. Tavares, and E. Valle. Adversarial Images for Variational Autoencoders. *ArXiv e-prints*, December 2016.

George Toderici, Sean M O'Malley, Sung Jin Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell, and Rahul Sukthankar. Variable rate image compression with recurrent neural networks. *arXiv preprint arXiv:1511.06085*, 2015.

George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. Full resolution image compression with recurrent neural networks. *arXiv preprint arXiv:1608.05148*, 2016.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016. URL `http://arxiv.org/abs/1609.03499`.

A Appendix

A.1 Mean latent vector targeted attack

A variant of the single latent vector targeted attack described in Section 4.3, that was not explored in previous work to our knowledge is to take the mean latent vector of many target images and use that vector as $\mathbf{x}_t$. This variant is more flexible, in that the attacker can choose different latent properties to target without needing to find the ideal input. For example, in MNIST, the attacker may wish to have a particular line thickness or slant in the reconstructed digit, but may not have such an image available. In that case, by choosing some images of the target class with thinner lines or less slant, and some with thicker lines or more slant, the attacker can find a target latent vector that closely matches the desired properties.

In this case, the attack starts by using f_{enc} to produce the target latent vector, $\mathbf{z}_t$, from the chosen target images, $\mathbf{x}_{(t)}$.

$$\mathbf{z}_t = \frac{1}{|\mathbf{x}_{(t)}|} \sum_{i=0}^{|\mathbf{x}_{(t)}|} f_{\text{enc}}(\mathbf{x}^i_{(t)}). \tag{8}$$

In this work, we choose to reconstruct "ideal" MNIST digits by taking the mean latent vector of all of the training digits of each class, and using those vectors as $\mathbf{x}_t$. Given a target class y_t, a set of examples $\mathcal{X}$ and their corresponding ground truth labels $\mathbf{y}$, we create a subset $\mathbf{x}_{(t)}$ as follows:

$$\mathbf{x}_{(t)} = \{\mathbf{x}_i | \mathbf{x}_i \in \mathcal{X} \wedge y_i = y_t\}. \tag{9}$$

Both variants of this attack appear to be similarly effective, as shown in Figure 15 and Figure 5. The trade-off between the two in these experiments is between the simplicity of the first attack and the flexibility of the second attack.

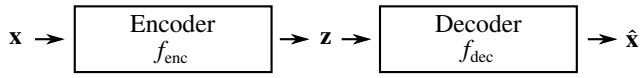

Figure 8: Variational autoencoder architecture.

A.2 Evaluation results

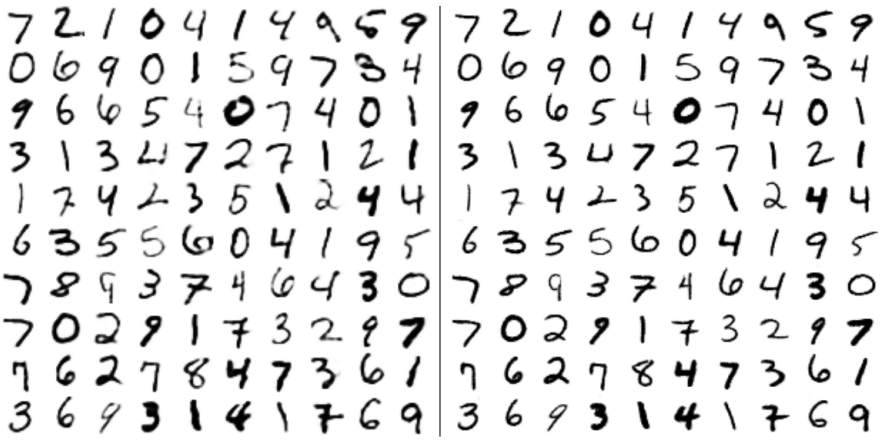

Figure 9: **Original Inputs and Reconstructions:** The first 100 images from the validation set reconstructed by the VAE (left) and the VAE-GAN (right).

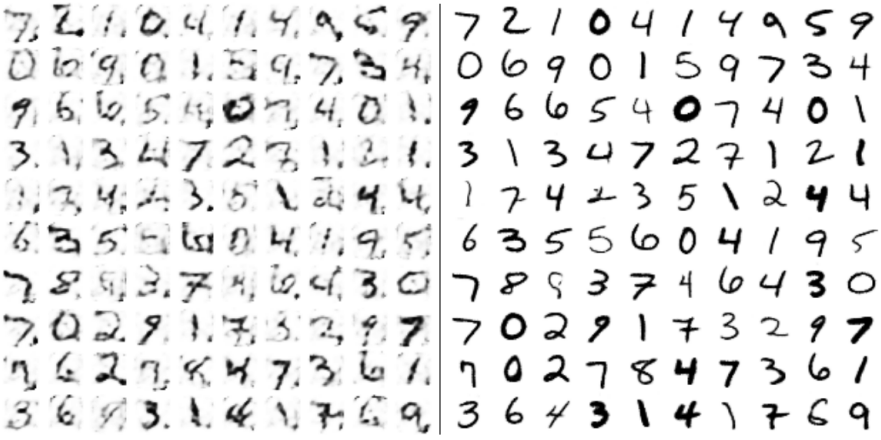

Figure 10: **Untargeted FGS $\mathcal{L}_{\text{VAE}}$ Attack:** VAE reconstructions (left) and VAE-GAN reconstructions (right). Note the difference in reconstructions compared to Figure 9. Careful visual inspection reveals that none of the VAE reconstructions change class, and only two of the VAE-GAN reconstructions change class (a 6 to a 0 in the next-to-last row, and a 9 to a 4 in the last row). Combining FGS with $\mathcal{L}_{\text{VAE}}$ does not seem to give an effective attack.

Target	0	1	2	3	4	5	6	7	8	9
Classifier accuracy	1.98%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Matching rate	95.06%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	99.89%

Table 2: L_2 **Optimization Classifier Attack on MNIST:** f_{class} accuracy on adversarial examples against the VAE-GAN for each target class (middle row) and the matching rate between the predictions f_{class} made and the adversarial target class (bottom row). The adversarial examples successfully fool f_{class} into predicting the target class almost 100% of the time, which makes this attack seem like a strong attack, but the attack actually fails to generate good reconstructions in many cases. Reconstructions for target classes 0 and 4 can be seen in Figure 4 and Figure 11.

Source	Target 0	Target 1	Target 2	Target 3	Target 4	Target 5	Target 6	Target 7	Target 8	Target 9
0	-	85.54% (34.94%)	100.00% (100.00%)	100.00% (13.25%)	75.90% (75.90%)	96.39% (92.77%)	100.00% (100.00%)	96.39% (91.57%)	0.00% (0.00%)	100.00% (83.13%)
1	100.00% (100.00%)	-	100.00% (100.00%)	100.00% (0.00%)	100.00% (93.60%)	100.00% (100.00%)	100.00% (100.00%)	100.00% (100.00%)	100.00% (0.00%)	100.00% (98.40%)
2	100.00% (100.00%)	97.37% (55.26%)	-	100.00% (55.26%)	97.37% (88.60%)	95.61% (74.56%)	100.00% (100.00%)	99.12% (94.74%)	100.00% (0.00%)	100.00% (92.98%)
3	100.00% (100.00%)	90.65% (89.72%)	100.00% (100.00%)	-	100.00% (91.59%)	94.39% (94.39%)	100.00% (100.00%)	85.05% (84.11%)	100.00% (0.00%)	90.65% (88.79%)
4	100.00% (100.00%)	97.27% (67.27%)	100.00% (100.00%)	100.00% (18.18%)	-	100.00% (100.00%)	100.00% (100.00%)	100.00% (100.00%)	100.00% (0.00%)	100.00% (100.00%)
5	100.00% (100.00%)	96.55% (80.46%)	100.00% (100.00%)	2.30% (2.30%)	100.00% (96.55%)	-	100.00% (100.00%)	98.85% (89.66%)	100.00% (0.00%)	95.40% (94.25%)
6	100.00% (100.00%)	87.36% (80.46%)	100.00% (100.00%)	100.00% (11.49%)	100.00% (97.70%)	100.00% (100.00%)	-	100.00% (98.85%)	100.00% (0.00%)	100.00% (96.55%)
7	100.00% (100.00%)	90.91% (82.83%)	100.00% (100.00%)	100.00% (16.16%)	100.00% (79.80%)	100.00% (98.99%)	100.00% (100.00%)	-	100.00% (0.00%)	100.00% (100.00%)
8	100.00% (100.00%)	89.77% (71.59%)	100.00% (100.00%)	100.00% (35.23%)	100.00% (97.73%)	89.77% (62.50%)	100.00% (100.00%)	98.86% (92.05%)	-	98.86% (96.59%)
9	100.00% (100.00%)	95.65% (75.00%)	100.00% (100.00%)	100.00% (18.48%)	100.00% (97.83%)	100.00% (95.65%)	100.00% (100.00%)	100.00% (100.00%)	100.00% (0.00%)	-

Table 3: L_2 **Optimization Latent Attack on MNIST (single latent vector target):** $AS_{ignore-target}$ (AS_{target} in parentheses) after one reconstruction loop for different source and target class pairs on the VAE-GAN model. The latent representation of a random image from the target class is used to generate the target latent vector. Higher values indicate more successful attacks against the generative model.

Source	Target 0	Target 1	Target 2	Target 3	Target 4	Target 5	Target 6	Target 7	Target 8	Target 9
0	-	40.96% (1.20%)	6.02% (4.82%)	10.84% (7.23%)	75.90% (0.00%)	6.02% (3.61%)	28.92% (28.92%)	37.35% (20.48%)	6.02% (1.20%)	10.84% (3.61%)
1	99.20% (77.60%)	-	7.20% (5.60%)	1.60% (1.60%)	85.60% (0.00%)	8.00% (5.60%)	28.80% (28.00%)	8.80% (7.20%)	3.20% (1.60%)	69.60% (0.80%)
2	85.96% (80.70%)	3.51% (2.63%)	-	29.82% (23.68%)	78.95% (0.00%)	72.81% (20.18%)	72.81% (46.49%)	35.09% (8.77%)	41.23% (12.28%)	68.42% (2.63%)
3	93.46% (83.18%)	26.17% (12.15%)	27.10% (16.82%)	-	67.29% (0.00%)	66.36% (62.62%)	87.85% (22.43%)	50.47% (27.10%)	23.36% (8.41%)	33.64% (8.41%)
4	100.00% (82.73%)	70.00% (48.18%)	28.18% (10.91%)	84.55% (17.27%)	-	66.36% (31.82%)	95.45% (71.82%)	62.73% (37.27%)	20.91% (0.91%)	51.82% (44.55%)
5	93.10% (89.66%)	21.84% (1.15%)	68.97% (11.49%)	28.74% (18.39%)	3.45% (0.00%)	-	20.69% (19.54%)	80.46% (41.38%)	22.99% (2.30%)	44.83% (12.64%)
6	29.89% (28.74%)	44.83% (1.15%)	24.14% (3.45%)	59.77% (11.49%)	77.01% (0.00%)	10.34% (8.05%)	-	62.07% (8.05%)	8.05% (0.00%)	75.86% (4.60%)
7	79.80% (65.66%)	77.78% (26.26%)	20.20% (8.08%)	8.08% (4.04%)	100.00% (0.00%)	56.57% (23.23%)	97.98% (17.17%)	-	38.38% (1.01%)	17.17% (10.10%)
8	94.32% (84.09%)	96.59% (18.18%)	60.23% (42.05%)	57.95% (43.18%)	100.00% (0.00%)	93.18% (80.68%)	100.00% (57.95%)	100.00% (34.09%)	-	87.50% (26.14%)
9	98.91% (79.35%)	97.83% (33.70%)	26.09% (1.09%)	17.39% (2.17%)	100.00% (0.00%)	22.83% (21.74%)	100.00% (30.43%)	47.83% (43.48%)	31.52% (4.35%)	-

Table 4: L_2 **Optimization Classifier Attack on MNIST:** $AS_{ignore-target}$ (AS_{target} in parentheses) for all source and target class pairs using adversarial examples generated on the VAE-GAN model. Higher values indicate more successful attacks against the generative model.

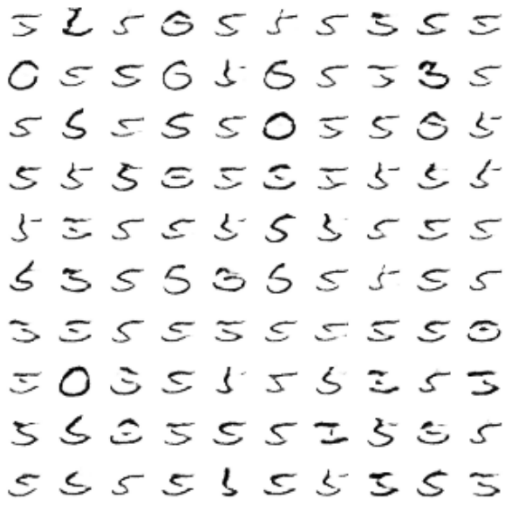

Figure 11: L_2 **Optimization Classifier Attack:** Reconstructions of the first 100 adversarial examples targeting 4, demonstrating why the AS_{target} metric is 0 for all source digits.

Source	Target 0	Target 1	Target 2	Target 3	Target 4	Target 5	Target 6	Target 7	Target 8	Target 9
0	-	90.36% (14.46%)	100.00% (100.00%)	100.00% (98.80%)	100.00% (61.45%)	91.57% (90.36%)	100.00% (96.39%)	68.67% (50.60%)	100.00% (91.57%)	98.80% (37.35%)
1	100.00% (100.00%)	-	100.00% (100.00%)	100.00% (100.00%)	100.00% (99.20%)	100.00% (100.00%)	100.00% (97.60%)	100.00% (96.00%)	100.00% (100.00%)	100.00% (96.00%)
2	100.00% (100.00%)	84.21% (60.53%)	-	100.00% (100.00%)	90.35% (71.93%)	100.00% (85.96%)	88.60% (88.60%)	97.37% (76.32%)	94.74% (94.74%)	97.37% (35.09%)
3	100.00% (100.00%)	75.70% (66.36%)	100.00% (100.00%)	-	94.39% (52.34%)	99.07% (99.07%)	98.13% (82.24%)	64.49% (53.27%)	100.00% (96.26%)	67.29% (31.78%)
4	100.00% (100.00%)	100.00% (52.73%)	100.00% (100.00%)	100.00% (100.00%)	-	100.00% (97.27%)	100.00% (100.00%)	100.00% (99.09%)	100.00% (100.00%)	85.45% (83.64%)
5	100.00% (100.00%)	96.55% (40.23%)	100.00% (100.00%)	100.00% (100.00%)	93.10% (59.77%)	-	100.00% (95.40%)	93.10% (71.26%)	96.55% (96.55%)	83.91% (51.72%)
6	100.00% (100.00%)	97.70% (70.11%)	100.00% (100.00%)	100.00% (100.00%)	100.00% (91.95%)	100.00% (100.00%)	-	97.70% (67.82%)	100.00% (98.85%)	95.40% (50.57%)
7	100.00% (100.00%)	85.86% (58.59%)	100.00% (100.00%)	100.00% (100.00%)	100.00% (98.99%)	100.00% (97.98%)	100.00% (79.80%)	-	100.00% (98.99%)	100.00% (96.97%)
8	100.00% (100.00%)	69.32% (44.32%)	100.00% (100.00%)	100.00% (100.00%)	54.55% (53.41%)	96.59% (96.59%)	95.45% (92.05%)	73.86% (52.27%)	-	42.05% (29.55%)
9	100.00% (100.00%)	100.00% (44.57%)	100.00% (100.00%)	100.00% (100.00%)	96.74% (95.65%)	100.00% (97.83%)	100.00% (100.00%)	100.00% (97.83%)	100.00% (100.00%)	-

Table 5: L_2 **Optimization** $\mathcal{L}_{\text{VAE}}$ **Attack on MNIST (single image target):** $AS_{ignore-target}$ (AS_{target} in parentheses) for different source and target class pairs using adversarial examples generated on the VAE-GAN model. Higher values indicate more successful attacks against the generative model.

Figure 12: **Untargeted FGS Classifer Attack:** Adversarial examples (left) and their reconstructions by the generative model (right) for the first 100 images from the MNIST validation set. Top results are for VAE, while bottom results are for VAE-GAN. Note the difference in quality of the reconstructed adversarial examples.

Figure 13: Original images with random noise added (top) and their reconstructions by VAE (bottom left) and VAE-GAN (bottom right). The magnitude of the random noise is the same as for the generated adversarial noise shown in Figure 12. Random noise does not cause the reconstructed images to change in a significant way.

Source	Target 0	Target 1	Target 2	Target 3	Target 4	Target 5	Target 6	Target 7	Target 8	Target 9
0	-	85.54% (34.94%)	100.00% (100.00%)	100.00% (13.25%)	75.90% (75.90%)	96.39% (92.77%)	100.00% (100.00%)	96.39% (91.57%)	0.00% (0.00%)	100.00% (83.13%)
1	100.00% (100.00%)	-	100.00% (100.00%)	100.00% (0.00%)	100.00% (93.60%)	100.00% (100.00%)	100.00% (100.00%)	100.00% (100.00%)	100.00% (0.00%)	100.00% (98.40%)
2	100.00% (100.00%)	97.37% (55.26%)	-	100.00% (55.26%)	97.37% (88.60%)	95.61% (74.56%)	100.00% (100.00%)	99.12% (94.74%)	100.00% (0.00%)	100.00% (92.98%)
3	100.00% (100.00%)	90.65% (89.72%)	100.00% (100.00%)	-	100.00% (91.59%)	94.39% (94.39%)	100.00% (100.00%)	85.05% (84.11%)	100.00% (0.00%)	90.65% (88.79%)
4	100.00% (100.00%)	97.27% (67.27%)	100.00% (100.00%)	100.00% (18.18%)	-	100.00% (100.00%)	100.00% (100.00%)	100.00% (100.00%)	100.00% (0.00%)	100.00% (100.00%)
5	100.00% (100.00%)	96.55% (80.46%)	100.00% (100.00%)	2.30% (2.30%)	100.00% (96.55%)	-	100.00% (100.00%)	98.85% (89.66%)	100.00% (0.00%)	95.40% (94.25%)
6	100.00% (100.00%)	87.36% (80.46%)	100.00% (100.00%)	100.00% (11.49%)	100.00% (97.70%)	100.00% (100.00%)	-	100.00% (98.85%)	100.00% (0.00%)	100.00% (96.55%)
7	100.00% (100.00%)	90.91% (82.83%)	100.00% (100.00%)	100.00% (16.16%)	100.00% (79.80%)	100.00% (98.99%)	100.00% (100.00%)	-	100.00% (0.00%)	100.00% (100.00%)
8	100.00% (100.00%)	89.77% (71.59%)	100.00% (100.00%)	100.00% (35.23%)	100.00% (97.73%)	89.77% (62.50%)	100.00% (100.00%)	98.86% (92.05%)	-	98.86% (96.59%)
9	100.00% (100.00%)	95.65% (75.00%)	100.00% (100.00%)	100.00% (18.48%)	100.00% (97.83%)	100.00% (95.65%)	100.00% (100.00%)	100.00% (100.00%)	100.00% (0.00%)	-

Table 6: L_2 **Optimization** $\mathcal{L}_{\text{VAE}}$ **Attack (mean reconstruction target):** $AS_{ignore-target}$ (AS_{target} in parentheses) for all source and target class pairs using adversarial examples generated on the VAE-GAN model. The mean reconstruction image for each target class (over all of the images of that class in the training set) is used as the target reconstruction. Higher values indicate more successful attacks against the generative model.

Source	Target 0	Target 1	Target 2	Target 3	Target 4	Target 5	Target 6	Target 7	Target 8	Target 9
0	-	40.96% (10.84%)	65.06% (65.06%)	53.01% (46.99%)	62.65% (54.22%)	36.14% (36.14%)	59.04% (59.04%)	46.99% (46.99%)	13.25% (12.05%)	44.58% (27.71%)
1	100.00% (100.00%)	-	100.00% (100.00%)	100.00% (100.00%)	100.00% (100.00%)	100.00% (100.00%)	100.00% (100.00%)	100.00% (100.00%)	100.00% (100.00%)	100.00% (96.80%)
2	96.49% (96.49%)	60.53% (59.65%)	-	95.61% (95.61%)	78.07% (75.44%)	98.25% (71.05%)	94.74% (90.35%)	71.05% (69.30%)	52.63% (50.88%)	75.44% (42.98%)
3	100.00% (100.00%)	87.85% (66.36%)	90.65% (90.65%)	-	85.98% (73.83%)	95.33% (95.33%)	79.44% (53.27%)	65.42% (64.49%)	59.81% (46.73%)	70.09% (58.88%)
4	99.09% (99.09%)	67.27% (66.36%)	96.36% (96.36%)	100.00% (81.82%)	-	100.00% (98.18%)	93.64% (93.64%)	98.18% (95.45%)	97.27% (92.73%)	39.09% (39.09%)
5	100.00% (100.00%)	79.31% (51.72%)	100.00% (83.91%)	70.11% (70.11%)	80.46% (72.41%)	-	73.56% (73.56%)	87.36% (73.56%)	55.17% (52.87%)	75.86% (65.52%)
6	97.70% (97.70%)	68.97% (50.57%)	96.55% (96.55%)	95.40% (71.26%)	73.56% (73.56%)	87.36% (77.01%)	-	88.51% (72.41%)	90.80% (55.17%)	91.95% (35.63%)
7	100.00% (97.98%)	83.84% (83.84%)	100.00% (100.00%)	100.00% (100.00%)	93.94% (90.91%)	98.99% (96.97%)	88.89% (81.82%)	-	100.00% (86.87%)	50.51% (50.51%)
8	100.00% (100.00%)	96.59% (78.41%)	100.00% (100.00%)	98.86% (95.45%)	94.32% (86.36%)	98.86% (98.86%)	98.86% (93.18%)	98.86% (73.86%)	-	87.50% (78.41%)
9	100.00% (100.00%)	100.00% (76.09%)	100.00% (100.00%)	98.91% (96.74%)	100.00% (100.00%)	100.00% (98.91%)	97.83% (97.83%)	98.91% (98.91%)	97.83% (94.57%)	-

Table 7: L_2 **Optimization Latent Attack (mean latent vector target):** $AS_{ignore-target}$ (AS_{target} in parentheses) for all source and target class pairs using adversarial examples generated on the VAE-GAN model. The mean latent vector for each target class (over all of the images of that class in the training set) is used as the target latent vector. Higher values indicate more successful attacks against the generative model.

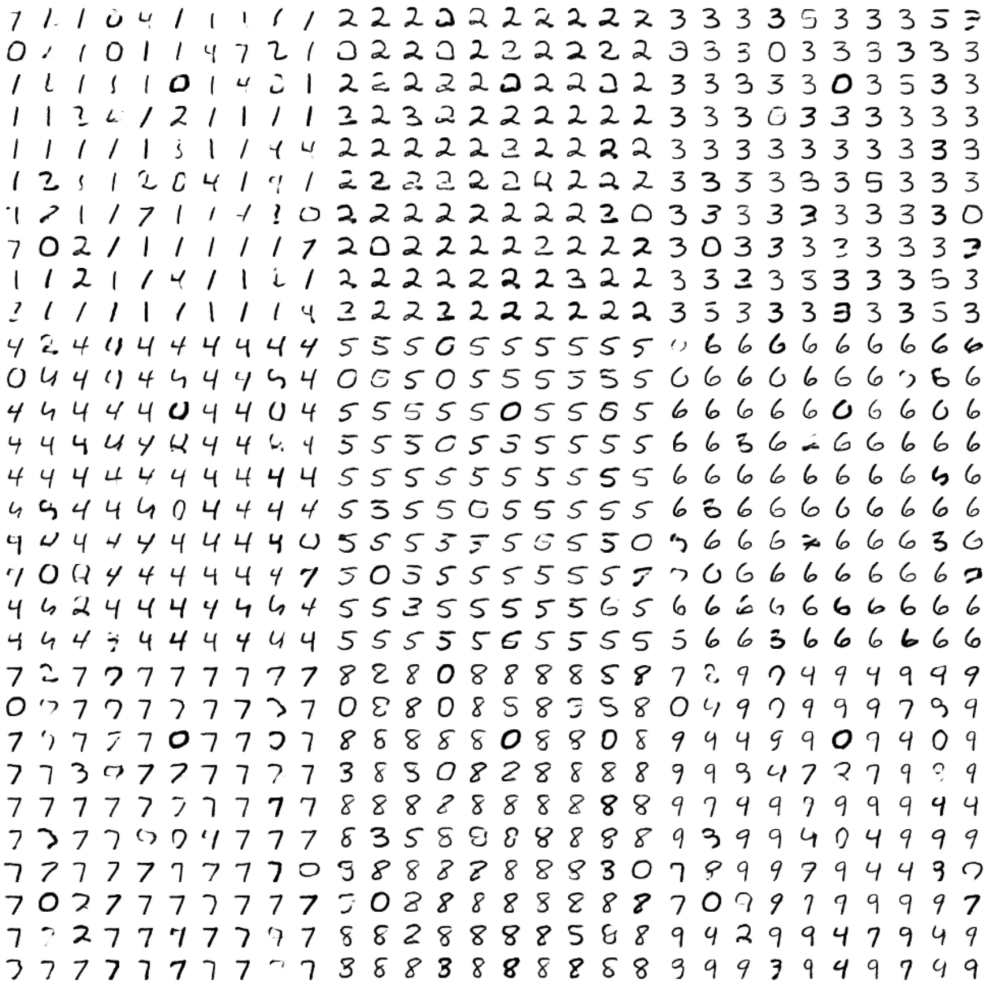

Figure 14: L_2 **Optimization Latent Attack (mean latent vector targets):** VAE-GAN reconstructions of adversarial examples with target classes from 1 through 9. Original examples which already belong to the target class are excluded.

Source	Target 0	Target 1	Target 2	Target 3	Target 4	Target 5	Target 6	Target 7	Target 8	Target 9
0	-	95.18% (9.64%)	100.00% (100.00%)	98.80% (93.98%)	100.00% (48.19%)	91.57% (89.16%)	100.00% (89.16%)	73.49% (43.37%)	100.00% (87.95%)	100.00% (25.30%)
1	100.00% (100.00%)	-	100.00% (100.00%)	100.00% (100.00%)	100.00% (92.80%)	100.00% (97.60%)	100.00% (98.40%)	100.00% (76.00%)	100.00% (100.00%)	100.00% (90.40%)
2	98.25% (98.25%)	83.33% (48.25%)	-	100.00% (100.00%)	88.60% (43.86%)	99.12% (63.16%)	74.56% (71.93%)	99.12% (63.16%)	93.86% (92.98%)	99.12% (21.05%)
3	99.07% (98.13%)	57.01% (42.99%)	99.07% (99.07%)	-	82.24% (36.45%)	89.72% (88.79%)	99.07% (61.68%)	57.01% (37.38%)	98.13% (92.52%)	67.29% (18.69%)
4	100.00% (100.00%)	100.00% (37.27%)	100.00% (100.00%)	100.00% (99.09%)	-	100.00% (80.00%)	98.18% (93.64%)	100.00% (94.55%)	100.00% (99.09%)	86.36% (80.00%)
5	100.00% (100.00%)	97.70% (19.54%)	100.00% (98.85%)	98.85% (98.85%)	85.06% (44.83%)	-	95.40% (88.51%)	93.10% (45.98%)	96.55% (96.55%)	87.36% (34.48%)
6	100.00% (100.00%)	96.55% (58.62%)	100.00% (98.85%)	100.00% (98.85%)	100.00% (86.21%)	100.00% (97.70%)	-	100.00% (56.32%)	100.00% (96.55%)	95.40% (43.68%)
7	100.00% (100.00%)	80.81% (40.40%)	100.00% (100.00%)	100.00% (98.99%)	100.00% (92.93%)	100.00% (87.88%)	100.00% (62.63%)	-	100.00% (97.98%)	100.00% (88.89%)
8	100.00% (100.00%)	44.32% (18.18%)	100.00% (100.00%)	100.00% (100.00%)	30.68% (28.41%)	78.41% (76.14%)	89.77% (81.82%)	75.00% (38.64%)	-	22.73% (15.91%)
9	100.00% (100.00%)	98.91% (17.39%)	100.00% (100.00%)	100.00% (100.00%)	97.83% (92.39%)	100.00% (89.13%)	100.00% (92.39%)	98.91% (94.57%)	100.00% (100.00%)	-

Table 8: L_2 **Optimization** $\mathcal{L}_{\text{VAE}}$ **Attack (mean reconstruction target):** $AS_{ignore-target}$ (AS_{target} in parentheses) for all source and target class pairs using adversarial examples generated on the VAE-GAN model. The mean image for each target class (over all of the images of that class in the training set) is used as the target. Higher values indicate more successful attacks against the generative model.

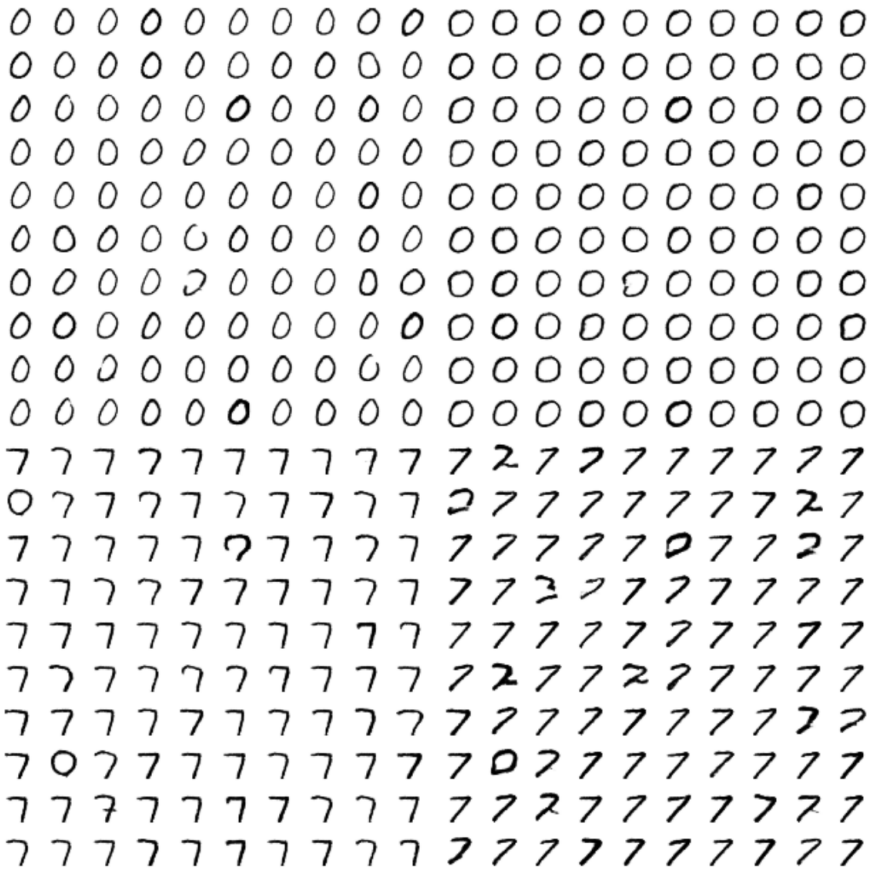

Figure 15: L_2 **Optimization Latent Attack (single latent vector target):** VAE-GAN reconstructions of adversarial examples generated using the latent attack with target classes 0 and 7 using two random targets in latent space per target class. Original examples which already belong to the target class are excluded. The stylistic differences in the reconstructions are clearly visible.

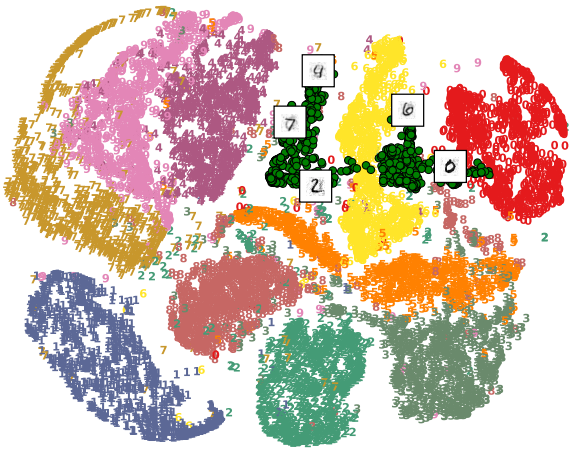

Figure 16: L_2 **Optimization Latent Attack (single latent vector target):** t-SNE plot of the latent space, with the addition of green circles representing the adversarial examples for target class 0. In this plot, it appears that the adversarial examples cluster around 6 (yellow) and 0 (red).

Source	Target 0	Target 1	Target 2	Target 3	Target 4	Target 5	Target 6	Target 7	Target 8	Target 9
0	-	92.77% (38.55%)	100.00% (100.00%)	100.00% (66.27%)	100.00% (34.94%)	100.00% (22.89%)	100.00% (100.00%)	79.52% (63.86%)	97.59% (90.36%)	100.00% (62.65%)
1	100.00% (100.00%)	-	100.00% (100.00%)	100.00% (99.20%)	100.00% (90.40%)	100.00% (0.80%)	100.00% (100.00%)	100.00% (100.00%)	100.00% (100.00%)	100.00% (100.00%)
2	97.37% (97.37%)	97.37% (57.02%)	-	100.00% (87.72%)	98.25% (42.11%)	100.00% (50.88%)	100.00% (99.12%)	97.37% (89.47%)	89.47% (89.47%)	100.00% (81.58%)
3	100.00% (100.00%)	89.72% (85.05%)	100.00% (100.00%)	-	62.62% (48.60%)	91.59% (45.79%)	100.00% (99.07%)	95.33% (90.65%)	97.20% (94.39%)	90.65% (79.44%)
4	100.00% (100.00%)	95.45% (67.27%)	100.00% (100.00%)	100.00% (73.64%)	-	100.00% (30.00%)	100.00% (100.00%)	100.00% (99.09%)	100.00% (99.09%)	99.09% (99.09%)
5	100.00% (100.00%)	98.85% (79.31%)	100.00% (100.00%)	73.56% (73.56%)	83.91% (34.48%)	-	100.00% (100.00%)	90.80% (87.36%)	100.00% (100.00%)	87.36% (82.76%)
6	100.00% (100.00%)	86.21% (79.31%)	100.00% (100.00%)	100.00% (88.51%)	95.40% (71.26%)	10.34% (10.34%)	-	100.00% (83.91%)	100.00% (97.70%)	100.00% (70.11%)
7	100.00% (100.00%)	91.92% (79.80%)	100.00% (100.00%)	100.00% (87.88%)	100.00% (63.64%)	100.00% (58.59%)	100.00% (100.00%)	-	100.00% (100.00%)	100.00% (100.00%)
8	100.00% (100.00%)	88.64% (73.86%)	100.00% (100.00%)	100.00% (46.59%)	95.45% (44.32%)	96.59% (31.82%)	100.00% (100.00%)	96.59% (94.32%)	-	95.45% (79.55%)
9	100.00% (100.00%)	96.74% (72.83%)	100.00% (100.00%)	100.00% (59.78%)	66.30% (63.04%)	100.00% (28.26%)	100.00% (100.00%)	98.91% (98.91%)	100.00% (100.00%)	-

Table 9: L_2 **Optimization** $\mathcal{L}_{\text{VAE}}$ **Attack on MNIST (single image target):** $AS_{ignore-target}$ (AS_{target} in parentheses) for different source and target class pairs using adversarial examples generated on the VAE-GAN model. Higher values indicate more successful attacks against the generative model.

Figure 17: **Original Inputs and Reconstructions:** The first 100 images from the SVHN validation set (left) reconstructed by VAE-GAN (right).

Source	Target 0	Target 1	Target 2	Target 3	Target 4	Target 5	Target 6	Target 7	Target 8	Target 9
0	-	64.29% (40.00%)	78.57% (61.43%)	92.86% (80.00%)	84.29% (57.14%)	98.57% (98.57%)	94.29% (38.57%)	88.57% (54.29%)	95.71% (11.43%)	95.71% (25.71%)
1	76.80% (70.72%)	-	74.59% (67.40%)	93.37% (88.95%)	75.69% (65.19%)	98.34% (97.79%)	86.74% (24.86%)	46.96% (36.46%)	96.13% (4.97%)	96.13% (28.73%)
2	82.93% (65.85%)	57.93% (42.68%)	-	90.24% (86.59%)	53.66% (46.34%)	99.39% (98.17%)	82.93% (14.02%)	71.34% (57.32%)	71.34% (6.71%)	24.39% (23.17%)
3	92.17% (64.35%)	58.26% (41.74%)	83.48% (68.70%)	-	84.35% (49.57%)	96.52% (95.65%)	53.91% (23.48%)	90.43% (56.52%)	93.04% (5.22%)	93.91% (33.91%)
4	74.44% (55.56%)	47.78% (43.33%)	70.00% (61.11%)	86.67% (77.78%)	-	100.00% (98.89%)	93.33% (35.56%)	90.00% (36.67%)	85.56% (14.44%)	94.44% (27.78%)
5	75.31% (50.62%)	59.26% (43.21%)	88.89% (58.02%)	97.53% (88.89%)	72.84% (53.09%)	-	37.04% (18.52%)	80.25% (41.98%)	32.10% (6.17%)	92.59% (30.86%)
6	67.44% (47.67%)	56.98% (27.91%)	84.88% (55.81%)	86.05% (79.07%)	65.12% (39.53%)	94.19% (94.19%)	-	90.70% (33.72%)	58.14% (10.47%)	87.21% (22.09%)
7	87.34% (63.29%)	55.70% (48.10%)	79.75% (74.68%)	92.41% (79.75%)	69.62% (41.77%)	97.47% (89.87%)	93.67% (18.99%)	-	91.14% (7.59%)	97.47% (17.72%)
8	98.33% (63.33%)	78.33% (38.33%)	80.00% (63.33%)	100.00% (88.33%)	93.33% (48.33%)	98.33% (96.67%)	96.67% (35.00%)	96.67% (50.00%)	-	95.00% (31.67%)
9	87.88% (66.67%)	72.73% (43.94%)	92.42% (80.30%)	93.94% (86.36%)	80.30% (51.52%)	95.45% (93.94%)	98.48% (27.27%)	92.42% (62.12%)	93.94% (9.09%)	-

Table 10: L_2 **Optimization Latent Attack on SVHN (single latent vector target):** $AS_{ignore-target}$ (AS_{target} in parentheses) after one reconstruction loop for different source and target class pairs on the VAE-GAN model. The latent representation of a random image from the target class is used to generate the target latent vector. Higher values indicate more successful attacks against the generative model.

Source	Target 0	Target 1	Target 2	Target 3	Target 4	Target 5	Target 6	Target 7	Target 8	Target 9
0	-	30.00% (12.86%)	32.86% (5.71%)	34.29% (5.71%)	28.57% (0.00%)	30.00% (1.43%)	30.00% (5.71%)	30.00% (0.00%)	30.00% (1.43%)	31.43% (0.00%)
1	13.26% (1.10%)	-	7.73% (1.66%)	18.78% (4.97%)	13.26% (3.31%)	12.15% (0.00%)	11.60% (0.55%)	9.94% (1.10%)	10.50% (1.10%)	16.02% (0.55%)
2	23.17% (0.61%)	13.41% (3.66%)	-	17.07% (3.05%)	14.63% (1.83%)	14.63% (2.44%)	15.24% (0.00%)	15.24% (1.22%)	14.02% (0.61%)	15.24% (1.22%)
3	30.43% (0.87%)	26.09% (7.83%)	30.43% (2.61%)	-	30.43% (0.00%)	29.57% (6.96%)	27.83% (0.00%)	27.83% (1.74%)	28.70% (2.61%)	33.91% (6.09%)
4	21.11% (0.00%)	15.56% (5.56%)	16.67% (2.22%)	25.56% (4.44%)	-	16.67% (1.11%)	18.89% (0.00%)	16.67% (1.11%)	18.89% (2.22%)	22.22% (0.00%)
5	32.10% (0.00%)	28.40% (3.70%)	27.16% (3.70%)	32.10% (8.64%)	24.69% (2.47%)	-	28.40% (6.17%)	23.46% (0.00%)	27.16% (3.70%)	27.16% (0.00%)
6	27.91% (4.65%)	25.58% (4.65%)	26.74% (0.00%)	33.72% (3.49%)	30.23% (2.33%)	20.93% (4.65%)	-	31.40% (0.00%)	24.42% (3.49%)	32.56% (0.00%)
7	30.38% (0.00%)	27.85% (12.66%)	26.58% (10.13%)	31.65% (5.06%)	31.65% (0.00%)	30.38% (0.00%)	32.91% (0.00%)	-	30.38% (0.00%)	34.18% (1.27%)
8	40.00% (5.00%)	35.00% (0.00%)	33.33% (3.33%)	43.33% (6.67%)	40.00% (3.33%)	35.00% (1.67%)	41.67% (11.67%)	38.33% (0.00%)	-	36.67% (0.00%)
9	34.85% (6.06%)	33.33% (12.12%)	33.33% (9.09%)	40.91% (4.55%)	31.82% (3.03%)	31.82% (0.00%)	33.33% (0.00%)	34.85% (0.00%)	31.82% (1.52%)	-

Table 11: L_2 **Optimization $\mathcal{L}_{\text{VAE}}$ Attack on SVHN (single image target):** $AS_{ignore-target}$ (AS_{target} in parentheses) after one reconstruction loop for different source and target class pairs on the VAE-GAN model. The latent representation of a random image from the target class is used to generate the target latent vector. Higher values indicate more successful attacks against the generative model.

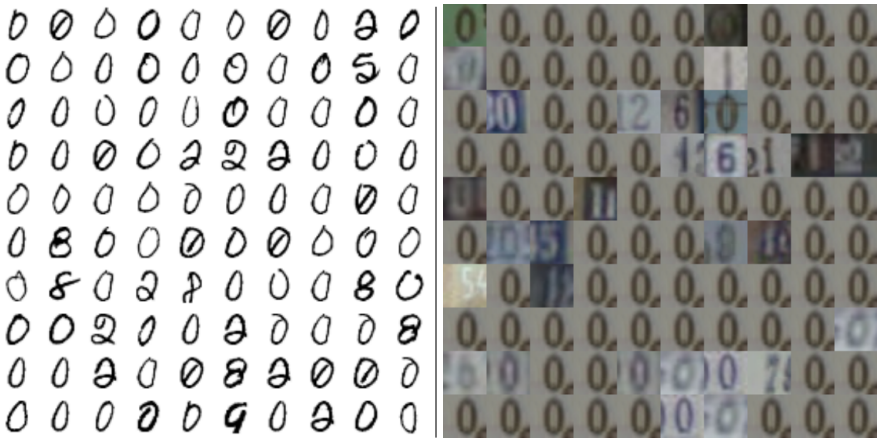

Figure 18: L_2 **Optimization Latent Attack (single latent vector target):** Nearest neighbors in latent space for generated adversarial examples (target class 0) on the first 100 images from the MNIST (left) and SVHN (right) validation sets.

Figure 19: Original images in the CelebA dataset (left) and their VAE-GAN reconstructions (right).

Figure 20: L_2 **Optimization Latent Attack on CelebA Dataset (single latent vector target):** Adversarial examples generated for 100 images from the CelebA dataset (left) and their VAE-GAN reconstructions (right).

Figure 21: L_2 **Optimization $\mathcal{L}_{\text{VAE}}$ Attack on CelebA Dataset (single image target):** Adversarial examples generated for 100 images from the CelebA dataset (left) and their VAE-GAN reconstructions (right).

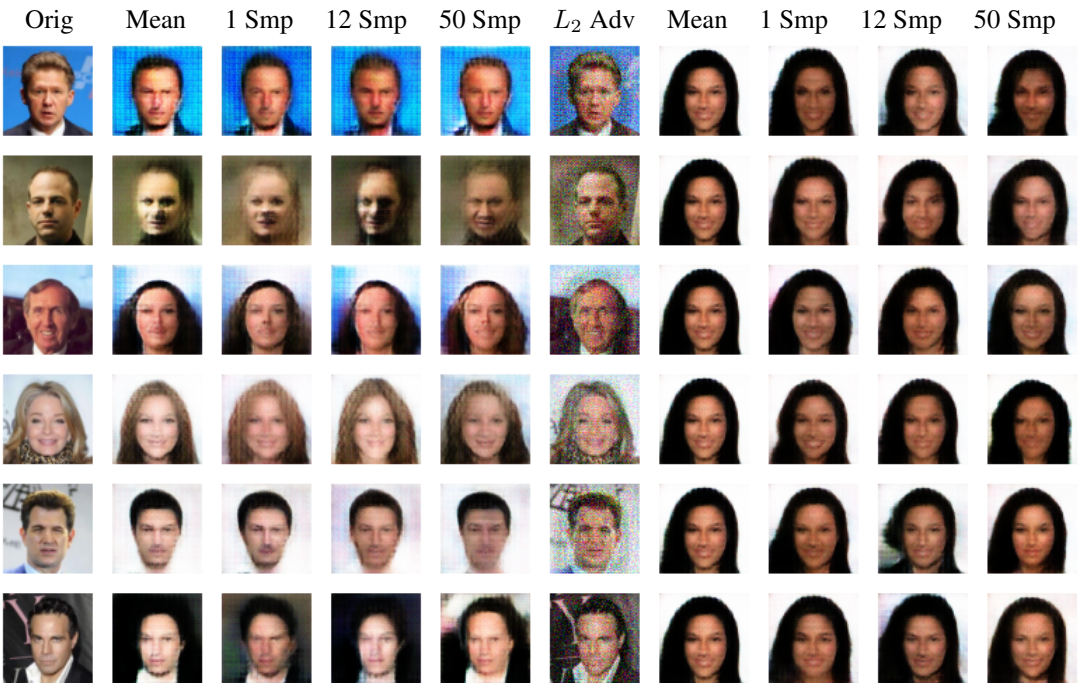

Figure 22: Effect of sampling on adversarial reconstructions. Columns in order: original image, reconstruction of the original image (no sampling, just the mean), reconstruction of the original image (1 sample), reconstruction of the original image (12 samples), reconstruction of the original image (50 samples), adversarial example (latent attack), reconstruction of the adversarial example (no sampling, just the mean), reconstruction of the adversarial example (1 sample), reconstruction of the adversarial example (12 samples), reconstruction of the adversarial example (50 samples).