# 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.**

## I. 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 [2] and [3], 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 [5], learn to generate a variety of outputs, including handwritten digits, faces [6], realistic scenes [7], videos [8], 3D objects [9], and audio [10]. 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 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.

## II. Related work and background

This work focuses on adversaries for variational autoencoders (VAEs, proposed in [5]) and VAE-GANs (VAEs composed with a generative adversarial network, proposed in [11]).

### A. Related work on adversaries

Many adversarial attacks on classification models have been described in existing literature [2], [3]. These attacks can be untargeted, where the adversary's goal is to cause any misclassification, or the least likely misclassification [3], [12]; or they can be targeted, where the attacker desires a specific

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

[2] Random noise images and "fooling" images [4] do not belong to this strict definition of an adversarial input, although they do highlight other limitations of current classifiers.

DOI 10.1109/SPW.2018.00014

misclassification. [13] 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 [1], [2], but commonly it is assumed that the attacker does have that access, in an online or offline setting [3], [12].[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: $\operatorname{argmin}_{\mathbf{x}^*} L(\mathbf{x}, \mathbf{x}^*)\ s.t.\ f(\mathbf{x}^*) \neq f(\mathbf{x})$, 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 $\operatorname{argmin}_{\mathbf{x}^*} L(\mathbf{x}, \mathbf{x}^*)\ s.t.\ f(\mathbf{x}^*) = y_t$, 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 [15], as done in [2] and [16]. They can also be approximated with single-step gradient-based techniques like fast gradient sign [3], fast gradient $L_2$ [17], or fast least likely class [12]; or they can be approximated with iterative variants of those and other gradient-based techniques [12], [13].

An interesting variation of this type of attack can be found in [18]. 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 [19][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 blurry. While they do not explain the exact experimental setting, the attack sounds similar to our $\mathcal{L}_{\mathrm{VAE}}$ 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.

### B. Background on VAEs and VAE-GANs

The general architecture of a variational autoencoder consists of three components. The **encoder** $f_{\mathrm{enc}}(\mathbf{x})$ 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** $f_{\mathrm{dec}}(\mathbf{z})$, 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 $\hat{\mathbf{x}} = f_{\mathrm{dec}}(f_{\mathrm{enc}}(\mathbf{x}))$. This composed architecture is used during training to backpropagate errors from the loss function.

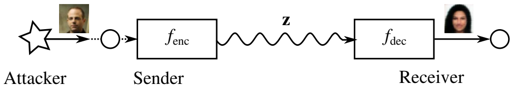

Fig. 1. Depiction of the attack scenario. The VAE is used as a compression scheme to transmit a latent representation of the image from the sender (left) to the receiver (right). The attacker convinces the sender to compress a particular image into its latent vector, which is sent to the receiver, where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

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

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

$q(\mathbf{z}|\mathbf{x})$ is the learned approximation of the posterior distribution $p(\mathbf{z}|\mathbf{x})$. $p(\mathbf{z})$ is the prior distribution of the latent representation $\mathbf{z}$. $D_{\mathrm{KL}}$ 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 $f_{\mathrm{dec}}(\mathbf{z})$.

For the VAE to be fully differentiable while sampling from $q(\mathbf{z}|\mathbf{x})$, the reparametrization trick [5] 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, $f_{\mathrm{enc}}(\mathbf{x})$ 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 $f_{\mathrm{enc}}$, and thus does not impact differentiation of the network.

The VAE-GAN architecture of [11] has the same $f_{\mathrm{enc}}$ and $f_{\mathrm{dec}}$ pair as in the VAE. It also adds a discriminator $f_{\mathrm{disc}}$ that is used during training, as in standard generative adversarial networks [20]. The loss function of $f_{\mathrm{dec}}$ uses the disciminator loss instead of cross-entropy for estimating the reconstruction error.

## III. 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.

### A. 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 $\mathbf{z}$.

The attacker's goal is to convince the sender to send an image of the attacker's choosing to the receiver, but the

[3] See [14] for an overview of different adversarial threat models.

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

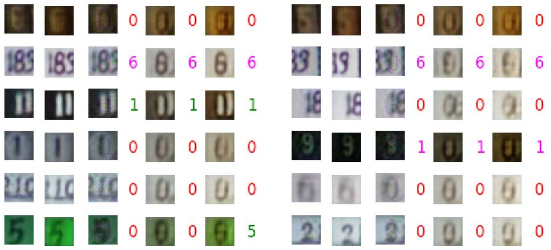

Fig. 2. Results for the $L_2$ optimization latent attack (see Section IV-C) on the VAE-GAN, targeting a specific image from the class 0. Shown are the first 12 non-zero images from the test SVHN data set. The columns are, in order: the original image, the reconstruction of the original image, the adversarial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

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, $\hat{\mathbf{x}}_{\mathrm{adv}}$, 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 [21], [22].

### B. *Defining adversarial examples against generative models*

We make the following assumptions about generating adversarial examples on a target generative model, $G_{\mathrm{targ}}(\mathbf{x}) = f_{\mathrm{dec}}(f_{\mathrm{enc}}(\mathbf{x}))$. $G_{\mathrm{targ}}$ 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. $G_{\mathrm{targ}}$ normally succeeds at generating an output $\hat{\mathbf{x}} = G_{\mathrm{targ}}(\mathbf{x})$ 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 $G_{\mathrm{targ}}$ 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 $G_{\mathrm{targ}}$. Finally, the attacker has access to a set of examples from the same distribution as $\mathcal{X}$ that have the target label $y_t$ 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

$$\mathrm{argmin}_{\mathbf{x}^*}\ L(\mathbf{x}, \mathbf{x}^*)\ s.t.\ \textsc{Oracle}(G_{\mathrm{targ}}(\mathbf{x}^*)) = y_t,$$

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 II-A can also be used to approximate this optimization in natural ways, some of which we describe in Section IV.

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 $\hat{\mathbf{x}}_{\mathrm{adv}}$ 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 IV-E how to measure the effectiveness of an attack automatically using a classifier trained on $\mathbf{z} = f_{\mathrm{enc}}(\mathbf{x})$.

## IV. 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 $\mathcal{L}_{\mathrm{VAE}}$ 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.

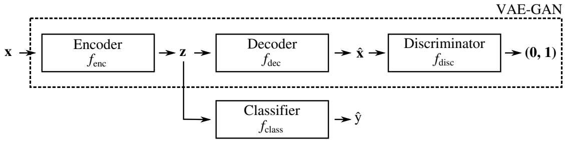

Fig. 3. The VAE-GAN classifier architecture used to generate classifier-based adversarial examples on the VAE-GAN. The VAE-GAN in the dashed box is the target network and is frozen while training the classifier. The path $\mathbf{x} \rightarrow f_{\text{enc}} \rightarrow \mathbf{z} \rightarrow f_{\text{class}}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

### A. Classifier attack

By adding a classifier $f_{\text{class}}$ to the pre-trained generative model[5] as shown in Figure 3, 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, 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. We omit further details due to lack of space.

### B. $\mathcal{L}_{\text{VAE}}$ 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 IV-D.

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.

### C. Latent attack

Our third approach attacks the latent space of the generative model. It is similar to the work of [18], 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 IV-D:

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

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

### D. 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 [3] method and an $L_2$ optimization method. As the latter produces much better results we focus on the $L_2$ optimization method. 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.

The optimization-based approach, explored in [2] and [16], 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 $\lambda$ is used to balance the two loss contributions.

### E. 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 $f_{\text{class}}$ to estimate whether a particular attack was successful.[6]

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, $f_{\text{class}}$ 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 $f_{\text{class}}$ are sufficiently high on $y_t$, $f_{\text{class}}$ 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:

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

[5] This is similar to the process of semi-supervised learning in [23], although the goal is different.

[6] Note that $f_{\text{class}}$ 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.

$N$ 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:

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

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.

## V. Evaluation

We evaluate the three attacks on MNIST [24], SVHN [25] and CelebA [26], using the standard training and validation set splits. Due to space constraints, we only report the most interesting results. The VAE and VAE-GAN architectures are implemented in TensorFlow [27]. 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 [11] (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.

In this section we only show results where no sampling from latent space was 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 8 in Section V-B2. On most examples, the visible change is small and in general the attack is still successful.

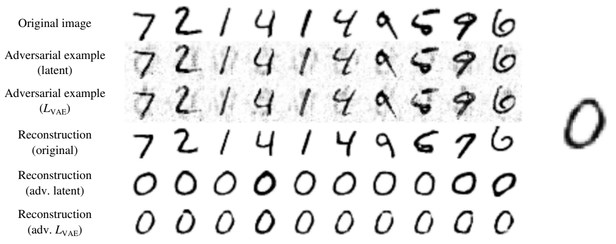
Original image
Adversarial example (latent)
Adversarial example ($L_{\mathrm{VAE}}$)
Reconstruction (original)
Reconstruction (adv. latent)
Reconstruction (adv. $L_{\mathrm{VAE}}$)

Fig. 4. Summary of different attacks on MNIST dataset and VAE-GAN model: original images, adversarial examples for both methods (latent and $\mathcal{L}_{\mathrm{VAE}}$) and reconstructions of original images and adversarial examples. Target reconstruction is shown on the right.

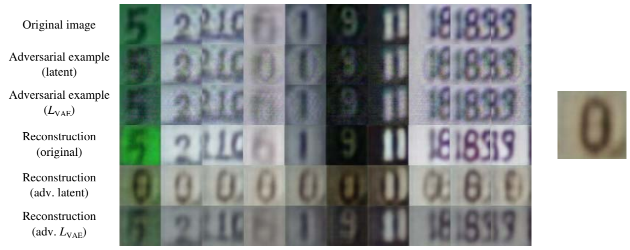
Original image
Adversarial example (latent)
Adversarial example ($L_{\mathrm{VAE}}$)
Reconstruction (original)
Reconstruction (adv. latent)
Reconstruction (adv. $L_{\mathrm{VAE}}$)

Fig. 5. Summary of different attacks on SVHN dataset and VAE-GAN model: original images, adversarial examples for both methods (latent and $\mathcal{L}_{\mathrm{VAE}}$) and reconstructions of original images and adversarial examples. The $\mathcal{L}_{\mathrm{VAE}}$ attack seems ineffective against SVHN in our experiments. Target reconstruction is shown on the right.

### A. 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 5.

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 $\mathcal{L}_{\mathrm{VAE}}$ attacks we set $\lambda = 10$.

The evaluation metrics are less strong on SVHN (mean $AS_{ignore-target} = 0.82$, mean $AS_{target} = 0.51$) than on MNIST (mean $AS_{ignore-target} = 0.96$, mean $AS_{target} = 0.76$), 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 $\mathcal{L}_{\mathrm{VAE}}$ attack (see Figure 5) on SVHN, while both attacks work equally well on MNIST (see Figure 4).

### B. 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 6 (fourth row).

We tried lambdas ranging from 0.1 to 0.75 for both attacks. Figure 6 shows adversarial examples generated 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 $\mathcal{L}_{\mathrm{VAE}}$ attack, using a lambda value of 0.75 ($L_2$ norm between original images and

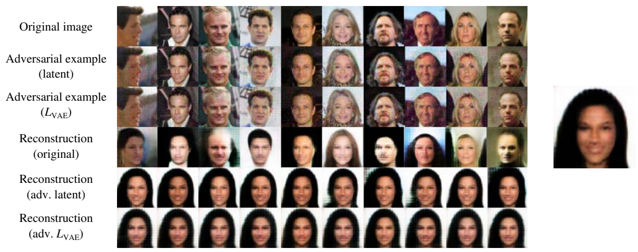
Original image
Adversarial example (latent)
Adversarial example ($L_{\mathrm{VAE}}$)
Reconstruction (original)
Reconstruction (adv. latent)
Reconstruction (adv. $L_{\mathrm{VAE}}$)

Fig. 6. Summary of different attacks on CelebA dataset and VAE-GAN model: original images, adversarial examples for both methods (latent and $\mathcal{L}_{\mathrm{VAE}}$) and reconstructions of original images and adversarial examples. Target reconstruction is shown on the right.

generated adversarial examples $8.98$, RMSD $0.081$) as shown in Figure 6.

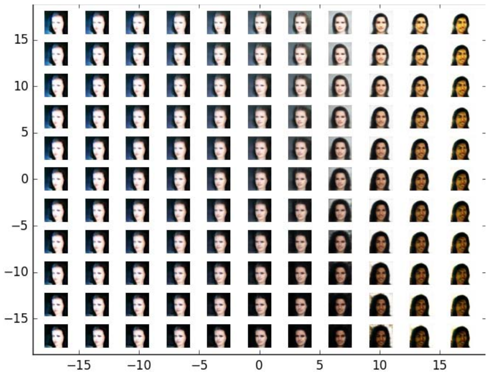

Fig. 7. Visualization of VAE-GAN reconstructions in input image space. The x-axis is the attack direction, while the y-axis is a random orthogonal direction. The reconstruction of the original image is at the center $(0, 0)$.

*1) Visualizing the adversarial example boundary:* We generate a visualization of the reconstructions in input image space, showing that the direction of the generated adversarial example is much more effective than a random direction when generating adversarial examples. Similar in meaning to decision boundary plots [3] for classification models, Figure 7 shows VAE-GAN reconstructions from different points in input image space spanned by the two directions. We generate the plot by defining two normalized vectors, $\mathbf{d}_1$ and $\mathbf{d}_2$, spanning the input image space. The one shown on the x-axis points in the direction of the generated adversarial perturbation ($\mathbf{d}_1$), while the other shown on the y-axis points in a randomly chosen orthogonal direction ($\mathbf{d}_2$). The images in the plane represent reconstructions computed by $f_{\mathrm{dec}}(f_{\mathrm{enc}}(\mathbf{x}+u\mathbf{d}_1+v\mathbf{d}_2))$, where $\mathbf{x}$ is the original image. Values on the axes are the values of constants $u$ and $v$. The target image used for the attack is the same as in Figure 6.

This visualization shows that if you move in the direction of the generated adversarial example, you quickly bump into adversarial examples, while moving in random directions in image space has no major effect on changing the reconstruction.

*2) Effect of sampling:* Since VAE and VAE-GAN are stohastic models, we also evaluate the effect of sampling on the success of adversarial attacks on the CelebA dataset. Evaluation is performed using different amount of samples taken: no samples (only the mean is used), a single sample, 12 samples and 50 samples. Figure 8 shows the reconstructions of original images and adversarial examples with different number of samples taken. The results show that sampling only has a limited effect on targeted adversarial attacks on VAE-GAN using the CelebA dataset.

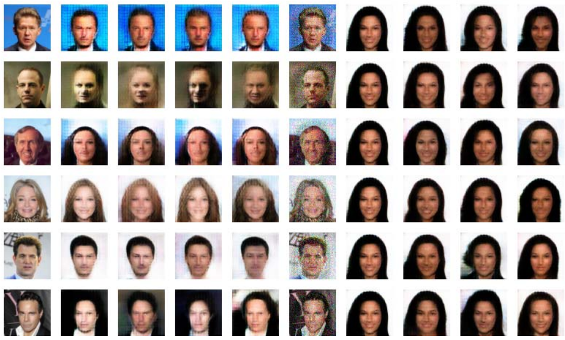

Fig. 8. Effect of sampling on adversarial reconstructions. Columns in order: original image, reconstruction of the original image (no sampling), reconstruction of the original image (1 sample), reconstruction of the original image (12 samples), reconstruction of the original image (50 samples), adversarial example (latent attack), reconstruction of the adversarial example (no sampling), reconstruction of the adversarial example (1 sample), reconstruction of the adversarial example (12 samples), reconstruction of the adversarial example (50 samples).

## VI. Conclusion

We explored generating adversarial examples against generative models such as VAEs and VAE-GANs. These models are 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.

## References

[1] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, “Practical black-box attacks against machine learning,” in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS ’17. New York, NY, USA: ACM, 2017, pp. 506–519. [Online]. Available: http://doi.acm.org/10.1145/3052973.3053009

[2] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” *arXiv preprint arXiv:1312.6199*, 2013.

[3] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[4] A. M. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," *CoRR*, vol. abs/1412.1897, 2014.

[5] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[6] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum, "Deep convolutional inverse graphics network," in *Advances in Neural Information Processing Systems*, 2015, pp. 2539–2547.

[7] A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional image generation with pixelcnn decoders," *arXiv preprint arXiv:1606.05328*, 2016.

[8] N. Kalchbrenner, A. v. d. Oord, K. Simonyan, I. Danihelka, O. Vinyals, A. Graves, and K. Kavukcuoglu, "Video pixel networks," *arXiv preprint arXiv:1610.00527*, 2016.

[9] A. Dosovitskiy, J. Springenberg, M. Tatarchenko, and T. Brox, "Learning to generate chairs, tables and cars with convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, no. 99, pp. 1–1, 2016.

[10] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *CoRR*, vol. abs/1609.03499, 2016. [Online]. Available: http://arxiv.org/abs/1609.03499

[11] A. B. L. Larsen, S. K. Snderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1558–1566. [Online]. Available: http://proceedings.mlr.press/v48/larsen16.html

[12] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *CoRR*, vol. abs/1607.02533, 2016.

[13] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," 2016.

[14] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proceedings of the 1st IEEE European Symposium on Security and Privacy*, 2015.

[15] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2015.

[16] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, May 2017, pp. 39–57.

[17] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvári, "Learning with a strong adversary," *CoRR*, vol. abs/1511.03034, 2015.

[18] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet, "Adversarial manipulation of deep representations," *CoRR*, vol. abs/1511.05122, 2015. [Online]. Available: http://arxiv.org/abs/1511.05122

[19] P. Tabacof, J. Tavares, and E. Valle, "Adversarial Images for Variational Autoencoders," *ArXiv e-prints*, Dec. 2016.

[20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[21] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar, "Variable rate image compression with recurrent neural networks," *arXiv preprint arXiv:1511.06085*, 2015.

[22] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell, "Full resolution image compression with recurrent neural networks," *arXiv preprint arXiv:1608.05148*, 2016.

[23] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Advances in Neural Information Processing Systems*, 2014, pp. 3581–3589.

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[25] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

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

[27] M. Abadi and A. A. et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/