Abstract

Adversarial examples for image classifiers are typically created by searching for a suitable norm-constrained perturbation to the pixels of an image. However, such perturbations represent only a small and rather contrived subset of possible adversarial inputs; robustness to norm-constrained pixel perturbations alone is insufficient. We introduce a novel method for the construction of a rich new class of semantic adversarial examples. Leveraging the hierarchical feature representations learnt by generative models, our procedure makes adversarial but realistic changes at different levels of semantic granularity. Unlike prior work, this is not an ad-hoc algorithm targeting a fixed category of semantic property. For instance, our approach perturbs the pose, location, size, shape, colour and texture of the objects in an image without manual encoding of these concepts. We demonstrate this new attack by creating semantic adversarial examples that fool state-of-the-art classifiers on the MNIST and ImageNet datasets.

1. Introduction

Despite their many successes, deep neural networks have been found to be vulnerable to adversarial examples: inputs designed to deliberately fool a model. In this paper, we introduce a new method that performs adversarial perturbations in the space of feature representations learnt by a generator network. By performing these perturbations at different layers in the generator, we can alter the full range of feature granularities from macro (e.g. the shape of a mountain) to micro (e.g. the texture of a segment of a dog’s ear).

In contrast, most adversarial examples research focuses on perturbations at the level of individual pixels. Although when originally proposed, robustness against such \( l_p \) norm-constrained perturbations was intended only as a toy problem from which a solution could be generalised [1], adversarial robustness research has remained preoccupied with this rather artificial paradigm. While it has led to some useful insights, there is simply no realistic threat which is well modelled by norm-constrained pixel perturbations. Adversarial pixel perturbations can be alternatively motivated as modelling a worst-case failure of the i.i.d. assumption. But robustness to pixel perturbations does not imply robustness to other kinds of distribution shift [2], so a stronger, more thoughtful threat model is needed to provide robustness to realistic distribution shifts.

This work belongs to the burgeoning movement considering unrestricted adversarial examples [3]: inputs which fool a target network yet are not constrained to be within a certain distance of a given input. This paradigm is clearly strong enough to include any realistic adversarial or distribution-shift threat to correctness.

Our method is not the first to use semantic changes to images to create unrestricted adversarial examples. However, while prior works present ad-hoc techniques which are tailored to a certain kind of semantic change (such as colour [4]), our method is general: the features that can be perturbed are not hand-crafted but those that have been learnt by a generative model. This allows for a much richer space of possible manipulations, which will only improve as generative machine learning continues to develop. Note also that our method is not specific to a certain kind dataset (or even the domain of images): if a generator network can be trained on a dataset, then our method can be used to find unrestricted adversarial examples for it. Finally, our method is able to leverage any perturbation attack algorithm in its search for semantic adversarial perturbations; advances in this field can therefore also be readily applied.

2. Background

Adversarial Perturbation Attacks Since the galvanising discovery that imperceptible changes to the pixel values of an image could fool state-of-the-art classifier networks [5], many attack procedures have been proposed. Customarily, these entail a white- or black-box greedy search for a perturbation (with constrained magnitude under some \( l_p \) norm) which, when applied to the given input, results in the network’s output being incorrect. Similarly many defence techniques have been proposed [6], the inadequacy of which [7, 8, 9] has prompted a smaller literature of more successful approaches to defence which are able to prove the extent of their robustness to perturbation attacks on a test set [10].
3. Construction of Adversarial Semantic Perturbations

Suppose we have a trained generator neural network that has learnt to map a standard distribution in latent space $\mathbb{H}_0$ to the distribution of training images in pixel space, $\mathbb{X}$. We will view it as a fixed function $g : \mathbb{H}_0 \rightarrow \mathbb{X}$. In this paper, we use the generator network from a GAN, but $g$ could also be the decoder network of a Variational Auto-Encoder, or any other generative model. Since neural networks are composed of layers, we think of our generator function as a composition $g = g_n \circ g_{n-1} \circ \cdots \circ g_1$, where $g_i : \mathbb{H}_{i-1} \rightarrow \mathbb{H}_i$ is a function between latent (activation) spaces, with $\mathbb{H}_n = \mathbb{X}$. Note that the elements of each space are tensors.

Bau et al. [12] showed that individual tensor elements (neurons) in these latent spaces $\mathbb{H}_i$ can represent semantic features of the generated image. For instance, some neurons may represent the presence of clouds. Our key idea is to perform adversarial perturbations in these activation spaces $\mathbb{H}_i$ as an image is being generated rather than in pixel space $\mathbb{X}$. Our main hypothesis is that this results in semantic changes of different kinds (as opposed to meaningless pixel-level adjustments) that successfully fool classifiers without changing the true class of the data.

3.1. Single-Depth Attack

Consider using a trained network $f : \mathbb{X} \rightarrow \mathbb{Y}$ to classify the output of the generator $g$. In effect, this is using the function composition $f \circ g$ to predict the class of the image generated by inputs $h_0 \in \mathbb{H}_0$ to the generator. Noting that $g$ itself is a composition of functions allows us to now take a new perspective: pick some depth index $0 \leq i \leq n$, and now define a new pair of neural networks, $g' = g_i \circ g_{i-1} \circ \cdots \circ g_1$ and $f' = f \circ g_n \circ \cdots \circ g_{i+1}$. Note that $f' \circ g' = f \circ g$, so the computation is identical but we now have a new view of these. We can consider $g'$ to be a generator of realistic activation tensors $h_i \in \mathbb{H}_i$, while $f'$ is able to classify these activation tensors according to the label of the images that would be induced in $\mathbb{X}$ if each $h_i$ were passed through the rest of the generator, $g_0 \circ \cdots \circ g_{i+1}$. Figure 1 may help the reader to visualise this.

Suppose we have an adversarial perturbation algorithm $A$ that, given a classifier network $c$ and an input point $x$, searches for a nearby adversarial example $\hat{x}$ such that $c(x) \neq c(\hat{x})$ and $\|x - \hat{x}\| < \epsilon$ for some distance metric $\|\cdot\|$ and bound $\epsilon$. Under these definitions, traditional pixel-space perturbations could be applied to generated images by running algorithm $A$ with $c = f$, $x = g(h_0)$ for some appropriately sampled $h_0 \in \mathbb{H}_0$, $\|\cdot\|$ as an $l_p$ norm, and $\epsilon$ chosen to be suitably small.

Our method is to instead run perturbation algorithm $A$ with input $x = g'(h_0)$ and classifier $c = f'$, as defined above. This finds an adversarially perturbed activation vector $\hat{x} \in \mathbb{H}_i$, which is nearby $x = g'(h_0)$. Consider passing $\hat{x}$ through the remainder of the original generator $g$. This image, $(g_n \circ \cdots \circ g_{i+1})(\hat{x})$, is classified differently to the unperturbed image $(g_n \circ \cdots \circ g_i)(x)$ since the output of $A$ is such that $f'(x) \neq f'(\hat{x})$. Our claim is that the correct classification of both images ought to be the same, since a sufficiently small perturbation in an activation space leads to only small changes visible in the resulting image. This claim is empirically evaluated in Section 4.

Performing such mid-generator perturbations is possible because there is no fundamental difference between an image classifier $f$ and an activation vector classifier $f' = f \circ g_n \circ \cdots \circ g_{i+1}$: they are both simply neural networks composed of a number of layers. Any attack (or indeed defence) algorithm that can therefore be used for pixel-space perturbations against $f$ can be used for semantic feature-space perturbations against $f'$.

3.2. Multiple-Depth Attack

The previous section described how adversarial perturbations can be made to activations in a latent space $\mathbb{H}_i$. A strictly stronger attack would be to perform such a perturbation to activations at every layer during the generation of an image. This section describes how such an attack can be framed as another $l_p$ norm-constrained perturbation to an input of a constructed classifier $F$. See Figure 4 for an illustration.

Consider, as above, a generator network $g = g_n \circ \cdots \circ g_1$ and a classifier $f$. We will construct a generator $G : (\mathbb{H}_0 \times \mathbb{H}_1 \times \cdots \times \mathbb{H}_n) \rightarrow \mathbb{H}_n$ that takes as input the usual initial seed $h_0$ followed by a series of perturbations to apply at each latent space: $G(h_0, p_1, p_2, \ldots, p_n) = (g_n \circ \cdots \circ G_1)(h_0)$, where $G_1(h_{i-1}) = g_i(h_{i-1}) + p_i$. Now consider $F = f \circ G$, which predicts the classes of images resulting from the inputs to $G$. Given an input $h_0$ to original generator $g$, we can use any adversarial attack algorithm $A$ with $c = F$ and $x = (h_0, 0, 0, \ldots, 0)$ to find an adversarial example constructed by performing adversarial perturbations at every space $\mathbb{H}_i$ in the generator. The magnitude of a perturbed input $(h_1, p_1, \ldots, p_n)$ can be found by flattening and concatenating each tensor before applying the distance metric of choice.

3.2.1 Scaling of Perturbation Magnitudes

Finding the magnitude of a traditional adversarial perturbation is straightforward: normalise each pixel from its original range, typically $[0, 255]$, to the range $[0, 1]$; then compute
Semantic perturbation
(Pixel perturbation)

Figure 1: Illustration of semantic adversarial perturbation attacks for depth $i = 3$ to aid understanding of Section 3.1.

| Depth: | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|-------|---|---|---|---|---|---|---|
| Original: | ![MNIST Original](image) | ![MNIST Original](image) | ![MNIST Original](image) | ![MNIST Original](image) | ![MNIST Original](image) | ![MNIST Original](image) | ![MNIST Original](image) |
| Perturbation: | ![MNIST Perturbation](image) | ![MNIST Perturbation](image) | ![MNIST Perturbation](image) | ![MNIST Perturbation](image) | ![MNIST Perturbation](image) | ![MNIST Perturbation](image) | ![MNIST Perturbation](image) |
| Perturbed: | ![MNIST Perturbed](image) | ![MNIST Perturbed](image) | ![MNIST Perturbed](image) | ![MNIST Perturbed](image) | ![MNIST Perturbed](image) | ![MNIST Perturbed](image) | ![MNIST Perturbed](image) |

Figure 2: Untargeted single-depth semantic adversarial perturbation attacks for MNIST. Green pixels in the perturbation represent an increase in value; red represent a decrease.

| Depth: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|-------|---|---|---|---|---|---|---|---|
| Original: | ![ImageNet Original](image) | ![ImageNet Original](image) | ![ImageNet Original](image) | ![ImageNet Original](image) | ![ImageNet Original](image) | ![ImageNet Original](image) | ![ImageNet Original](image) | ![ImageNet Original](image) |
| Perturbation: | ![ImageNet Perturbation](image) | ![ImageNet Perturbation](image) | ![ImageNet Perturbation](image) | ![ImageNet Perturbation](image) | ![ImageNet Perturbation](image) | ![ImageNet Perturbation](image) | ![ImageNet Perturbation](image) | ![ImageNet Perturbation](image) |
| Perturbed: | ![ImageNet Perturbed](image) | ![ImageNet Perturbed](image) | ![ImageNet Perturbed](image) | ![ImageNet Perturbed](image) | ![ImageNet Perturbed](image) | ![ImageNet Perturbed](image) | ![ImageNet Perturbed](image) | ![ImageNet Perturbed](image) |

Figure 3: Untargeted single-depth semantic adversarial perturbation attacks for ImageNet. Note how the granularity of the features altered by the perturbation varies with depth.
\[ \|x\|_p = \left( \sum_i |x_i|^p \right)^{\frac{1}{p}} \] However, finding the magnitude of a semantic perturbation is more challenging. The elements of an activation vector do not share a well-defined range; to normalise the vector to \([0, 1]\), we empirically measure the maximum and minimum values seen for each tensor element (neuron) over 256 runs, then perform linear scaling for each.

There is a further element of nuance: perturbations of the same magnitude may have different visual effect sizes at different depths in the generator. For instance, we found that an \(l_\infty\) perturbation of magnitude 0.01 at some layers could very significantly change the classifier output, yet be almost imperceptible in pixel space. If not compensated for, this could result in poor performance: using the same perturbation magnitude uniformly across all depths would result in the effects from some depths being dominated by the effects from others.

When performing a multiple-depth perturbation attack, we want to scale the perturbation for each depth so that a change of magnitude \(\epsilon\) to the input vector results in a similar perceptual change when this is applied at each depth. This requires a rescaling of each perturbation \(p_i\). To determine the weights for these rescalings, we find by inspection the greatest magnitude perturbation at each depth for which the induced perturbed images remain recognisable. We then multiply each perturbation part \(p_i\) in the input to \(G\) with a scalar weight so that \(\|p_i\| = 1\) in the input vector scales to this largest allowable perturbation size.

This tuning procedure was fairly crude; we suspect that a more careful finetuning of the magnitude weightings could produce better results for our multiple-depth attack.

### 3.3. Semantic Perturbations of Arbitrary Inputs

The previous sections have described how to find a semantic adversarial perturbation for a generated image. One might ask whether our method can be adapted to find a semantic adversarial perturbation to an arbitrary image. It can, to the extent that the generator is capable of generating the given image: procedures exist which determine the input to a generator which produces an image most similar to the one that is given [13]. This can then be semantically perturbed in the usual way.

However, we argue that this question and answer is not relevant. From a security perspective, there are no realistic threat models in which an adversary has the ability to perform perturbation to existing data yet cannot simply present new data of their own construction; and from a machine learning perspective a generated image is equivalent to a ‘real’ image assuming that the generator has converged to the training distribution.

### 4. Experiments

We report experimental results for two datasets, representing two extremes: MNIST and ImageNet.

MNIST is notoriously small and simple, making its classification very easy; this is the only dataset for which classifiers somewhat robust against adversarial \(l_p\)-perturbations are possible. This makes MNIST the most challenging dataset by far for successfully finding adversarial examples. We target Wong and Kolter’s network [14] in particular, which is provably robust to perturbations of magnitude \(\epsilon = 0.1\) under \(l_\infty\) in the sense that such attacks are provably unsuccessful on 94.2% of the MNIST test set. Other state-of-the-art adversarially-robust classifiers give similar results.

Having tried three different GANs for MNIST, we find that our method works equally well with each. However, we
also find that even small perturbations early in the generator can result in what could be called ‘class smudging’: as shown in Figure 5, even small changes can result in images failing to maintain their correct true label. The simple solution is to use a generator trained to generate one class only (we use class 9 for our experiments). Because such a generator does not learn representations for the other classes, it is much less likely that a small perturbation to an early-layer representation could result in a change of true label. If the computational cost of training such a GAN is a concern, note that a pretrained multi-class generator can still be used, albeit with class smudging leading to a lower success rate.

ImageNet, in contrast to MNIST, is notoriously large and complex. While this makes fooling its classifiers less challenging, some techniques struggle with the challenge of functioning at ImageNet scale. We target the ResNeXt-101-32x8d network\cite{15}, with top-1 accuracy of 79.3% and top-5 accuracy of 94.5%. BigGAN \cite{16} is state of the art in ImageNet generation; we use use the author’s ‘officially unofficial’ implementation and checkpoints \cite{17}.

Full details of our experimental setup can be found in Appendix A.

\subsection{4.1. Single-Depth Attack}

\subsubsection{4.1.1 Efficacy of Attacks}

We first evaluate how successful our method is when applied at different depths in the generator. We make use of the Foolbox library \cite{18}, which implements many standard adversarial perturbation algorithms. We allow it to select a perturbation magnitude which is sufficiently high for the resulting image to change classification; human judges then label the perturbed images. Each perturbation attack is successful if the perturbed image is classified incorrectly as desired, with its true label remains unchanged.

Table 1 shows the success rate of performing feature-level semantic perturbations at each depth in the generator, using standard $l_\infty$ projected gradient descent (PGD) as the attack algorithm. The success rates are consistently high. Note that the pixel-space (depth 6) attacks are successful despite the network being defended against such perturbations because we increase the perturbation magnitude until misclassification occurs. Targeted attacks, for which a certain misclassification label must be achieved, are more challenging. This is reflected in the lower success rates for this case.

For ImageNet, we find that all reasonable adversarial perturbation algorithms succeed 100% of the time at all depths with perturbations so small as to leave the correct image label unchanged with certainty. This is as expected, since robust ImageNet classification is so difficult, but serves as proof that our method scales well. All ImageNet results in this paper use the Carlini and Wagner attack \cite{19} unless stated otherwise. Attacks typically take between one and ten seconds for single-depth attacks (or between ten and one hundred seconds for multiple-depth attacks), depending on the attack algorithm used.

The targeted case for ImageNet is much harder, however, since the task becomes crafting an adversarial image that is classified into the single target class, rather than just any one of the 999 incorrect classes. As a result, the variance in the success of single-depth attacks is much higher, with a dependency on perturbation algorithm used and the perturbation depth. For example, the Carlini & Wagner attack has near-zero success rate at depth 3 but near-100% success rate at depth 1. The possible reasons for this discrepancy are discussed later.

\subsubsection{4.1.2 Visual Effect of Attacks}

Figures 2 and 3 show the effect of performing semantic adversarial perturbations at varying depths in the generator. Appendix B contains more extensive examples.

As the adversarial perturbations are made closer to the input of the generator, we observe that the changes:

1. are grouped into increasingly large contiguous regions,
2. have increasing magnitude in pixel space, and
3. coincide with increasingly high-level semantic features.

For ImageNet, perturbations performed near the beginning of the generator result in changes to the shape, size, location and orientation of macro-level objects; perturbation in the middle stages induce small adversarial changes in micro-level features; and perturbations in the final stages result in the usual pixel-level adversarial ‘fuzz’.

For MNIST, perturbations close to the beginning of the generator result in small adversarial changes to the shape and orientation of the characters; perturbations in middle stages typically result in changes to the thickness or length of small
line segments; and perturbations in the last stages result in pixel-level changes to the edges of the characters.

Visual inspection of the perturbations (see Appendix B for more examples) is strong evidence in support of our main claim that generator networks’ learnt representations can be leveraged to make adversarial changes to different kinds of semantic features.

Understanding Failures It is informative to examine the cases for which our attacks fail, which happens much more often in the more challenging targeted misclassification scenario. For MNIST, there are two clear failure modes: class smudging (as previously described, see Fig. 5), and transformation into a meaningless blob. We conjecture that this latter failure mode may occur when the activation vectors are perturbed into regions well beyond the usual distribution of activation vectors at that layer, and so the generator is unable to use them to construct a suitable image. For ImageNet, the failures are often intriguing – see the examples in Figure 6. We again speculate that these may be caused by perturbations resulting in out-of-distribution activation vectors which the remainder of the generator has not learned to handle. Of particular interest is that the nature of the distortions is at the same level of granularity as we expect for that depth.

4.1.3 Conspicuousness of Attacks

One motivation of this work is the ‘non-suspicious input’ threat model [20]: we would like to develop models which are robust to any adversarial input which is inconspicuous in the sense that it is not visually identifiable as being adversarial. That many of the visual effects of our semantic perturbations could be realised in the real world makes this threat model more relevant. We therefore evaluate the conspicuousness of our semantic perturbation attacks by measuring the proportion of the time that human judges are able to correctly identify the semantically-perturbed image when presented alongside a unperturbed dataset example. The results are shown in Table 2.

Note that for MNIST the least conspicuous single-depth attacks are those for which the perturbation occurs in the middle layers of the generator. For perturbations near the beginning of the generator, there can be noticeable macro-level artefacts such as unusually twisted shapes or extra marks. For perturbations near the end of the generator, the adversarial ‘fuzz’ becomes noticeable: this is always absent on real examples. It seems that the middle level of granularity – alterations to micro-features – is small enough not to attract attention but large enough as to have been plausibly created by a human pen.

Unsurprisingly, targeted attacks are more conspicuous since the task is harder and so larger perturbations are needed.

![Figure 6: Examples of failed targeted semantic perturbations for ImageNet, using a single-depth perturbation attack. Note how the granularity of the distortions correspond to the depth of the perturbation.](image)

4.2. Multiple-Depth Attack

4.2.1 Efficacy of Attacks

As described in Section 3.2, we perform adversarial perturbations at every stage of the generator network. On average, the multi-depth perturbation attack is successful 81% of the time in the untargeted case, and 74% of the time in the targeted case. Ignoring pixel-space perturbations (depth 6), this is comparable to the typical single-depth success rates for the untargeted case and well above average for the targeted case (see Table 2). We conjecture that this improvement in success rate for the targeted case is because it is able to perturb features at all levels of granularity, so a smaller perturbation is required at each depth; a large perturbation at a particular depth may cause an attack failure by distorting the image so its true label is not maintained.

Resilience to Single-Depth Defence We note that our multiple-depth attack cannot be mitigated by any defence against a single-depth attack. Suppose a multiple-depth perturbation is used to attack a network which is robust to pixel-level adversarial perturbations: the attack will succeed, since it will simply use its perturbation ‘budget’ to instead attack at earlier layers, which result in coarse-grained semantic changes which have a large magnitude in pixel space. We experimentally verify this by increasing the relative weighting from 0 to 1 of the pixel-space component of a multiple-depth attack against Wong and Kolter’s classifier robust to pixel perturbations [14]. That the success rate of the attack is a monotonically increasing function of this weighting demonstrates that defending against a multiple-depth attack requires defending against the conjunction of the relevant single-depth attacks.
4.2.2 Visual Effect of Multiple-Depth Attacks

Figure 8 gives examples of targeted multiple-depth attacks; more can be found Appendix B. The visual effect is, unsurprisingly, a combination of the effects seen at each single depth. That is, the multiple-depth attack makes changes at every level of granularity from changes to location, shape, orientation and colour of high-level objects, through texture and adjustments to micro-level to pixel-level perturbations.

Since the attack now has a larger range of kinds of adversarial change it can make to the image, the change at each level of granularity needs only to be much smaller. This is analogous to the decrease in pixel perturbation magnitude required as the number of perturbed pixels increases. As a result, the failure cases shown in Figure 6 do not occur, and ‘class smudging’ on MNIST also decreases because the perturbation is less concentrated at one level of granularity.

4.2.3 Conspicuousness of Attacks

For multiple-depth attack, humans are able to correctly identify the adversarial examples 94% and 96% of the time in the untargeted and targeted cases respectively. This is considerably better than the pixel-perturbation attacks, which can be identified almost 100% of the time, but not as good as other single-depth attacks.

We conjecture that this is due to the procedure we used to determine the relative magnitude weightings for the perturbations at different depths in the multiple-depth attack. In particular, our procedure was not focused on indistinguishability, but rather on distortion to the point of unrecognisability. These are quite distinct goals: pixel-space perturbations, for instance, are very unlikely to distort an image to the extent that it is no longer recognisable, but the presence of visible pixel perturbations immediately rules out an image from being from the test dataset.

4.3. Choice of Perturbation Algorithm

As well as being able to take any trained generator network as a source of semantic feature representations, our procedure is able to use any standard pixel-perturbation attack algorithm to search in these semantic feature spaces. The choice of this attack algorithm does matter (see Appendix B for relevant results). For instance, the Fast Gradient Sign Method (FGSM)
PGD ([21]) is usually unable to find suitable perturbations for ImageNet examples, and has a low success rate for MNIST examples. Conversely, the Carlini and Wagner attack ([19]) using the $L_2$ norm has a high rate of success, as does the projected gradient descent (PGD) under $l_p$. Figure 9 compares the visual effects of these two algorithms when used to find multiple-depth perturbations for ImageNet with source class 'ambulance'.

In short, the Carlini and Wagner attack makes larger changes to high-level features than PGD does. We conjecture that this is because the $L_2$ norm encourages sparsity, and if fewer features are to be perturbed, then higher-level features offer the biggest change per feature.

5. Related Work

Jain et al. [22] and Liu et al. [23] each propose a method performing norm-constrained perturbations to latent variables encoding semantic image properties such as lighting, weather and foliage. Unfortunately, these semantic representations are hard-coded rather than learnt: the approaches require a hand-crafted invertible differentiable renderer to be built, capable of rendering any possible scene for the dataset of interest. This is likely to be prohibitively expensive for all but simple domains. In contrast, our approach automatically leverage learnt semantic representations, requiring only a dataset (even labels are optional) from which a GAN can be trained. Furthermore, while we are able to demonstrate high success rates against state-of-the-art classifiers on MNIST (the most challenging dataset to attack), renderer-based attacks have not been evaluated on adversarially-robust networks, with the success rate of Jain et al.’s [22] method being rather disappointing even against non-robust networks (at best a 30 percentage point reduction in precision).

Song et al. [24] create unrestricted adversarial examples by searching for an adversarial input to the generator, somewhat analogously to our single-depth attack at depth 0. Our work can be viewed as a generalisation of this in three dimensions simultaneously: rather than using an ad-hoc search procedure, we are able to leverage all existing adversarial perturbation algorithms; rather than attacking only in the input space $\mathbb{H}_0$, we demonstrate attacks for all latent spaces $\mathbb{H}_i$, individually and jointly, thereby creating an attack space incorporating a wealth of semantic transformations in addition to the standard pixel perturbations; and rather than require an auxiliary-classifier GAN, any generator that can be decomposed is sufficient for our method. Since perturbations at depth 0 are relatively easy for a human to identify as being suspicious, our method offers a stronger attack under the ‘non-suspicious input’ threat model [20].

In previous work [25], we train a generator to generate unrestricted adversarial examples. This approach is orthogonal to the present paper: while training a generator is a search for generator weights that represent feature instances which are adversarial, semantic perturbations search in the existing generator feature space for adversarial examples. The flexibility of the former approach to learn new representations allows adaptation of the the attack more readily to mitigate defences, while the present use of representations designed solely for realism results in more realistic adversarial examples. If a pretrained GAN can be used, the present approach also does not require any further GAN training at all.

A number of other methods for semantic adversarial example creation consist of ad-hoc adversarially-tuned manipulations such as colouring [4], rotations and translations [26], and corruptions such as blurring, Gaussian noise and fogging [27]. Unlike these approaches, our work is not tailor to one specific method of adversarially-tuning one specific image property.

Qiu et al. [28] utilise the learnt domain-to-domain mappings of StarGAN [29] to find adversarial linear interpolations between modified and unmodified faces. A key limitation of this approach is the need for a dataset with labels for each of the semantic domains to be interpolated over, using which the StarGAN must then be trained; our approach, in contrast, uses ordinary generative GANs to learn the semantic features of interest.

Bhattad et al. [30] also repurpose existing techniques, introducing a texture-transfer attack which modifies the style transfer [31] to make it adversarial and minimally-perceptible, and a colouring attack which adversarially adjusts the parameters of a colourisation network [32]. Again, although these attacks show promise, they are ad-hoc, and untested against state-of-the-art robust classifiers (since these currently exist for MNIST only, for which the techniques do not apply).

6. Conclusion

We introduce the first method that allows the feature representations learnt by a pretrained generator network to be adversarially manipulated to create semantic adversarial examples. Significantly, this automatically includes a rich space of label-preserving transformations: there is no need for ad-hoc procedures targeting particular feature types such as colour, texture or object position. Experiments on ImageNet have demonstrated that the depth at which the adversarial perturbations occur directly affects the granularity of the features which are altered in the resulting image: performing perturbations at all layers simultaneously therefore allows adversarial changes
to be made to features at every scale. Experiments on MNIST have shown that this approach is able to fool state-of-the-art robust classifiers with high success rates; use of the multiple-depth procedure is more successful for targeted attacks since its perturbations to each feature can remain relatively small.

References

[1] Justin Gilmer, Ryan P Adams, Ian J Goodfellow, David Andersen, and George E Dahl. “Motivating the Rules of the Game for Adversarial Example Research”. In: CoRR abs/1807.0 (2018). arXiv: 1807.06732.

[2] Daniel Kang, Yi Sun, Dan Hendrycks, Tom Brown, and Jacob Steinhardt. “Testing Robustness Against Unforeseen Adversaries”. In: (Aug. 21, 2019). arXiv: 1908.08016 [cs, stat].

[3] Tom B Brown, Nicholas Carlini, Chiyuan Zhang, Catherine Olsson, Paul Francis Christiano, and Ian J Goodfellow. “Unrestricted Adversarial Examples”. In: CoRR abs/1809.0 (2018). arXiv: 1809.08352.

[4] Hossein Hosseini and Radha Poovendran. “Semantic Adversarial Examples”. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2018, Salt Lake City, UT, USA, June 18-22, 2018. IEEE Computer Society, 2018, pp. 1614–1619.

[5] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. “Intriguing Properties of Neural Networks”. In: International Conference on Learning Representations (ICLR). Ed. by Yoshua Bengio and Yann LeCun. 2014.

[6] Han Xu, Yao Ma, Haochen Liu, Debayan Deb, Hui Liu, Jiliang Tang, and Anil K. Jain. “Adversarial Attacks and Defenses in Images, Graphs and Text: A Review”. In: (Oct. 9, 2019). arXiv: 1909.08072 [cs, stat].

[7] Nicholas Carlini and David Wagner. “Defensive Distillation Is Not Robust to Adversarial Examples”. In: (July 14, 2016). arXiv: 1607.04311 [cs].

[8] Anish Athalye, Nicholas Carlini, and David A Wagner. “Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples”. In: International Conference on Machine Learning (ICML). Ed. by Jennifer G Dy and Andreas Krause. Vol. 80. 2018, pp. 274–283.

[9] Nicholas Carlini and David A. Wagner. “Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods”. In: Workshop on Artificial Intelligence and Security at the ACM Conference on Computer and Communications Security (AISec@CCS). ACM, 2017, pp. 3–14.

[10] Changliu Liu, Tomer Arnon, Christopher Lazarus, Clark Barrett, and Mykel J. Kochenderfer. “Algorithms for Verifying Deep Neural Networks”. In: CoRR abs/1903.06758 (2019). arXiv: 1903.06758.

[11] Ian J Goodfellow. “NIPS 2016 Tutorial: Generative Adversarial Networks”. In: CoRR abs/1701.0 (2017). arXiv: 1701.00160.

[12] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. “GAN Dissection: Visualizing and Understanding Generative Adversarial Networks”. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, 2019.

[13] Antonia Creswell and Anil Anthony Bharath. “Inverting the Generator of a Generative Adversarial Network”. In: IEEE Trans. Neural Netw. Learning Syst. 30.7 (2019), pp. 1967–1974.

[14] Eric Wong and J. Zico Kolter. “Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope”. In: Proceedings of the 35th International Conference on Machine Learning (ICML). 2018, pp. 5283–5292.

[15] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. “Aggregated Residual Transformations for Deep Neural Networks”. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017. IEEE Computer Society, 2017, pp. 5987–5995.

[16] Andrew Brock, Jeff Donahue, and Karen Simonyan. “Large Scale GAN Training for High Fidelity Natural Image Synthesis”. In: International Conference on Learning Representations (ICLR). 2019.

[17] Andrew Brock and Alex Andonian. “BigGAN-PyTorch”. In: (2019). Online; accessed 2019-05-15.

[18] Jonas Rauber, Wieland Brendel, and Matthias Bethge. “Foolbox: A Python Toolbox to Benchmark the Robustness of Machine Learning Models”. In: (Mar. 20, 2018). arXiv: 1707.04131 [cs, stat].

[19] Nicholas Carlini and David A. Wagner. “Towards Evaluating the Robustness of Neural Networks”. In: 2017 IEEE Symposium on Security and Privacy, SP. 2017, pp. 39–57.

[20] Justin Gilmer and Dan Hendrycks. “A Discussion of ‘Adversarial Examples Are Not Bugs, They Are Features’: Adversarial Example Researchers Need to Expand What Is Meant by ‘Robustness’”. In: Distill 4.8 (Aug. 6, 2019), e00019.1. ISSN: 2476-0757.

[21] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and Harnessing Adversarial Ex-
amples”. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[22] Lakshya Jain, Wilson Wu, Steven Chen, Uyeong Jang, Varun Chandrasekaran, Sanjit Seshia, and Somesh Jha. “Generating Semantic Adversarial Examples with Differentiable Rendering”. In: (Oct. 1, 2019). arXiv: 1910.00727 [cs, stat].

[23] Hsueh-Ti Derek Liu, Michael Tao, Chun-Liang Li, Derek Nowrouzezahrai, and Alec Jacobson. “Beyond Pixel Norm-Balls: Parametric Adversaries Using an Analytically Differentiable Renderer”. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, 2019.

[24] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. “Constructing Unrestricted Adversarial Examples with Generative Models”. In: Advances in Neural Information Processing Systems (NeurIPS). Ed. by Samy Bengio, Hanna M Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. 2018, pp. 8322–8333.

[25] Isaac Dunn, Hadrien Pouget, Tom Melham, and Daniel Kroening. “Adaptive Generation of Unrestricted Adversarial Inputs”. In: (Oct. 1, 2019). arXiv: 1905.02463 [cs, stat].

[26] Logan Engstrom, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. “A Rotation and a Translation Suffice: Fooling CNNs with Simple Transformations”. In: CoRR abs/1712.02779 (2017). arXiv: 1712.02779.

[27] Dan Hendrycks and Thomas Dietterich. “Benchmarking Neural Network Robustness to Common Corruptions and Perturbations”. In: (Mar. 28, 2019). arXiv: 1903.12261 [cs, stat].

[28] Haonan Qiu, Chaowei Xiao, Lei Yang, Xinchen Yan, Honglak Lee, and Bo Li. “SemanticAdv: Generating Adversarial Examples via Attribute-Conditional Image Editing”. In: (June 19, 2019). arXiv: 1906.07927 [cs, eess].

[29] Yunjey Choi, Min-Je Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. “StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation”. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018. IEEE Computer Society, 2018, pp. 8789–8797.

[30] Anand Bhattacharjee, Min Jin Chong, Kaizhao Liang, Bo Li, and David A. Forsyth. “Big but Imperceptible Adversarial Perturbations via Semantic Manipulation”. In: (Apr. 12, 2019). arXiv: 1904.06347 [cs].

[31] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. “Image Style Transfer Using Convolutional Neural Networks”. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA: IEEE, June 2016, pp. 2414–2423. ISBN: 978-1-4673-8851-1.

[32] Richard Zhang, Phillip Isola, and Alexei A. Efros. “Colorful Image Colorization”. In: Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III. Ed. by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling. Vol. 9907. Lecture Notes in Computer Science. Springer, 2016, pp. 649–666.

[33] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. In: 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings. Ed. by Yoshua Bengio and Yann LeCun. 2016.

A. Details of Experimental Setup

A.1. MNIST: Convolutional GAN

For MNIST, we tried a range of generators and found that they all worked roughly as well as one another. For the experiments, we use a simple convolutional generator, inspired by the Deep Convolutional GAN [33]. Details are shown in Table 3. Inputs to the generator are drawn from a 128-dimensional standard Gaussian. The sigmoid output transformation ensures that pixels are in the range $[0,1]$, as expected by the classifier.

We perform adversarial perturbations before ReLU layers, rather than after, to prevent ReLU output values from being perturbed to become negative, which would not have been encountered during training and so may not result in plausible images being generated.

Note that perturbing before and after the sigmoid transformation has different effects because perturbations to values not close to 0 are diminished in magnitude if passed through the sigmoid function.

A.2. ImageNet: BigGAN

We use the BigGAN [16] generator at $128 \times 128$ resolution; Table 4 details the stages at which adversarial perturbations are performed. At this resolution, the randomly-sampled component of the input is 120 dimensions wide. This is decomposed into six chunks of 20 dimensions each, which are fed in (along with the desired output label) to the different blocks
Table 3: MNIST generator architecture. Horizontal lines mark the stages at which adversarial perturbations are performed.

| Depth  | Component                      | Description                                      |
|--------|--------------------------------|--------------------------------------------------|
| 0      | Fully-Connected                | (64 units)                                       |
| 1      | ReLU Transposed Convolution    | (5 × 5 kernel, 2×2 stride, 32 feature maps)     |
| 2      | Batch Normalisation Leaky ReLU| (Slope −0.2)                                     |
|        | Dropout Transposed Convolution | (5 × 5 kernel, 2×2 stride, 8 feature maps)      |
| 3      | Batch Normalisation Leaky ReLU| (Slope −0.2)                                     |
|        | Dropout Transposed Convolution | (5 × 5 kernel, 2×2 stride, 4 feature maps)      |
| 4      | Batch Normalisation Leaky ReLU| (Slope −0.2)                                     |
|        | Dropout Fully-Connected        | (784 units)                                      |
| 5      | Sigmoid                        | (tanh used during training)                      |
| 6      |                                |                                                  |

Table 4: BigGAN architecture. Horizontal lines mark the stages at which adversarial perturbations are performed.

| Depth  | Component                      | Description                                      |
|--------|--------------------------------|--------------------------------------------------|
| 0      | Dense                          | (4×4×16×96 output units)                         |
| 1      | ResBlock                       | (8×8×16×96 output units)                         |
| 2      | ResBlock                       | (16×16×8×96 output units)                        |
| 3      | ResBlock                       | (32×32×4×96 output units)                        |
| 4      | ResBlock                       | (64×64×2×96 output units)                        |
| 5      | ResBlock                       | (128×128×1×96 output units)                      |
| 6      | BatchNorm ReLU Convolution     | (3×3 kernel, 1×1 stride, 1×1 zero-padding, 1 output channel) |
| 7      | Sigmoid                        | (tanh used during training)                      |

B. Examples of Semantic Perturbations

This appendix contains examples of our semantic adversarial perturbations. In particular, we include examples for MNIST and for ImageNet, targeted and untargeted cases, single-depth and multiple-depth attacks, and for different perturbation algorithms.

in the architecture. Please refer to Appendix B of the BigGAN for detailed descriptions, in particular of the ResBlocks which comprise the majority of the network. Note that we simply perform perturbations after each ResBlock; if desired, perturbations could also be performed within each block.
Figure 10: Untargeted single-depth semantic adversarial perturbations for MNIST. Green pixels in the perturbation represent an increase in value; red represent a decrease. Observe that the closer the perturbation is performed to pixel space (depth 6), the more scattered, smaller magnitude, and less macro-level the changes are.

Figure 11: Untargeted single-depth semantic adversarial perturbations for ImageNet. Note how the granularity of the features altered by the perturbation varies with depth.
Figure 12: Targeted single-depth semantic adversarial perturbations for MNIST. The target class is 0 for the top row and 1 for the bottom row. Note that the success rate is much lower for the targeted attack: there are both more examples which have changed true label (e.g. some in this figure may have changed to 0) and also more examples which have become nonsense. In this figure, there is also one instance (depth 1, top row, bottom left instance) of Foolbox being unable to find a suitable perturbation at all.

Figure 13: Targeted single-depth semantic adversarial perturbations for ImageNet (target class 951: lemon). Note that perturbations near the beginning of the generator still tend to have macro-level effects such as colour changes or introduction of more bush (bottom left). Note also that the targeted case has a lower success rate for ImageNet too: besides some obvious distortions, there are other examples that may have changed class (for instance, the disappearing boat).
Figure 14: Targeted multiple-depth adversarial examples for ImageNet. The target class is ‘lemon’. Note the much higher success rate than for the targeted single-depth attacks. Note also the variety of semantic changes made: pose, colour, camera position, zoom, object location, object size, texture, and shape.
Figure 15: Targeted multiple-depth adversarial perturbations for MNIST. Note that some target classes appear to be more difficult than others: the perturbations when targeting a 0, 1 or 6 are larger than when targeting a 4 or 7. We can see that each perturbation is, as expected, affecting features at all levels of granularity, including digit shape and orientation, stroke thickness, the presence or absence of certain small strokes, and pixel-level noise.
Figure 16: Tables showing examples of and success rates for single- and multiple-depth untargeted attacks on MNIST for a variety of perturbation algorithms.

(a) Results for $l_\infty$ attacks.

| Depth | Attack | Examples | Success |
|-------|--------|----------|---------|
| (None) | | ![Example](image1) | 75% |
| 0     | FGSM   | ![Example](image2) | 59% |
| 1     | FGSM   | ![Example](image3) | 79% |
| 2     | FGSM   | ![Example](image4) | 72% |
| 3     | FGSM   | ![Example](image5) | 78% |
| 4     | FGSM   | ![Example](image6) | 49% |
| 5     | FGSM   | ![Example](image7) | 96% |
| 6     | FGSM   | ![Example](image8) | 81% |
| Multiple | FGSM | ![Example](image9) | 84% |
| 0     | BIM    | ![Example](image10) | 87% |
| 1     | BIM    | ![Example](image11) | 86% |
| 2     | BIM    | ![Example](image12) | 93% |
| 3     | BIM    | ![Example](image13) | 92% |
| 4     | BIM    | ![Example](image14) | 93% |
| 5     | BIM    | ![Example](image15) | 94% |
| 6     | BIM    | ![Example](image16) | 97% |
| Multiple | BIM | ![Example](image17) | 92% |
| 0     | DeepFool | ![Example](image18) | 84% |
| 1     | DeepFool | ![Example](image19) | 80% |
| 2     | DeepFool | ![Example](image20) | 84% |
| 3     | DeepFool | ![Example](image21) | 87% |
| 4     | DeepFool | ![Example](image22) | 86% |
| 5     | DeepFool | ![Example](image23) | 87% |
| 6     | DeepFool | ![Example](image24) | 56% |
| Multiple | DeepFool | ![Example](image25) | 84% |

(b) Results for $l_2$ attacks.

| Depth | Attack | Examples | Success |
|-------|--------|----------|---------|
| (None) | | ![Example](image26) | 86% |
| 0     | BIM    | ![Example](image27) | 85% |
| 1     | BIM    | ![Example](image28) | 91% |
| 2     | BIM    | ![Example](image29) | 94% |
| 3     | BIM    | ![Example](image30) | 93% |
| 4     | BIM    | ![Example](image31) | 92% |
| 5     | BIM    | ![Example](image32) | 94% |
| 6     | BIM    | ![Example](image33) | 90% |
| Multiple | BIM | ![Example](image34) | 84% |
| 0     | C&W    | ![Example](image35) | 85% |
| 1     | C&W    | ![Example](image36) | 82% |
| 2     | C&W    | ![Example](image37) | 91% |
| 3     | C&W    | ![Example](image38) | 90% |
| 4     | C&W    | ![Example](image39) | 91% |
| 5     | C&W    | ![Example](image40) | 98% |
| 6     | C&W    | ![Example](image41) | 95% |
| Multiple | C&W | ![Example](image42) | 90% |
| 0     | NewtonFool | ![Example](image43) | 67% |
| 1     | NewtonFool | ![Example](image44) | 76% |
| 2     | NewtonFool | ![Example](image45) | 89% |
| 3     | NewtonFool | ![Example](image46) | 75% |
| 4     | NewtonFool | ![Example](image47) | 89% |
| 5     | NewtonFool | ![Example](image48) | 85% |
| 6     | NewtonFool | ![Example](image49) | 69% |
| Multiple | NewtonFool | ![Example](image50) | 82% |
Figure 17: Targeted multiple-depth adversarial examples for ImageNet, with target class ‘lemon’. The perturbations in the left column have been found using $l_\infty$ Projected Gradient Descent; those in the right have been found using the $l_2$ Carlini & Wagner attack. Note that the $l_2$ attack results in larger perturbations to fewer features; this is likely because the $l_2$ metric encourages sparsity, in contrast to $l_\infty$. 