Perturbation, Optimization and Statistics

Editors:

Tamir Hazan `tamir.hazan@technion.ac.il`
Technion - Israel Institute of Technology
Technion City, Haifa 32000, Israel

George Papandreou `gpapan@google.com`
Google Inc.
340 Main St., Los Angeles, CA 90291 USA

Daniel Tarlow `dtarlow@microsoft.com`
Microsoft Research
Cambridge, CB1 2FB, United Kingdom

This is a draft version of the author chapter.

The MIT Press
Cambridge, Massachusetts
London, England

1 Adversarial Perturbations of Deep Neural Networks

David Warde-Farley `wardefar@iro.umontreal.ca`
Montreal Institute for Learning Algorithms, Université de Montréal
Montreal, QC, Canada

Ian Goodfellow `goodfellow@google.com`
Google, Inc.
Mountain View, CA, USA

This chapter provides a review of a body of recent work on the topic of adversarial examples *and* generative adversarial networks*. Adversarial examples are examples created via worst-case perturbation of the input to a machine learning model. Adversarial examples have become a useful tool for the analysis and regularization of deep neural networks for classification. In the generative adversarial networks framework, the task of probabilistic modeling is reduced to the task of predicting worst-case perturbations of the input to a deep neural network. A discriminator network learns to recognize real data and reject fake samples, while a generator network learns to emit samples that deceive the discriminator. The GAN framework provides an alternative to maximum likelihood. The new framework has many advantageous computational properties, and is better suited than maximum likelihood to the task of generating realistic samples. More generally, games may be designed to have equilibria that direct learning algorithms to accomplish other goals, such as domain adaptation or preservation of privacy.*

1.1 Introduction

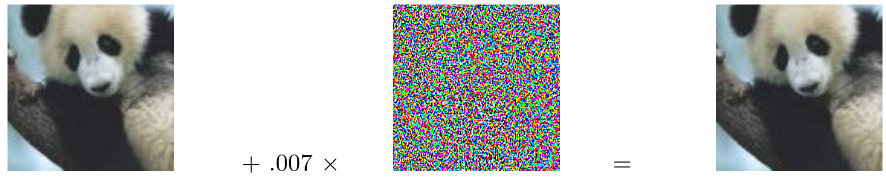
The past several years have given rise to two related lines of inquiry in deep learning research that view the training of neural networks through the lens of an adversarial game. The first body of work centers on the surprising result that discriminative classifiers are often highly sensitive to very small perturbations in the input space. This finding has led to algorithms designed to increase classifier robustness, to these perturbations and more generally, by exploiting these "adversarial examples". The second body of work frames generative model training as an adversarial game, pitting a sample generation process against a classifier trained to discriminate synthesized examples from training data.

This chapter describes how to construct adversarial perturbations in Section 1.2, then describes how to use the resulting adversarial examples to improve the robustness of a classifier in Section 1.3. Finally, Section 1.4 describes more sophisticated games in which one network is trained to generate inputs that deceive another network. These games between two machine learning models can be used for generative modeling, privatization of data, domain adaptation, and other applications.

1.2 Adversarial Examples

Neural networks have enjoyed much recent success in various application domains, owing to their ability to learn rich, non-linear parametric mappings from large amounts of data. While the general principles of training such networks via gradient descent are now well understood, a fully principled account of the internal representations they learn to compute remains elusive. As the commercial and industrial adoption of neural network technology hastens, the search for these insights becomes ever more important. Efforts to better understand how neural networks parameterize the input-output mappings they learn have yielded surprising results.

Szegedy et al. (2014b) discovered that small changes to the input of a neural network can have large, surprising effects on its output. For example, a well-chosen perturbation of pixels in the input to an image classifier can completely alter the class predicted by the network; in extreme cases, such as the one illustrated in Figure 1.1, the difference between the original and perturbed examples is imperceptible to a human observer. This surprising sensitivity to small perturbations has been found to exist not only in neural networks but also in more traditional machine learning methods, such as



$\boldsymbol{x}$ + .007 × $\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$ = $\boldsymbol{x} + \epsilon \cdot \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

$y =$"panda" w/ 57.7% confidence; "nematode" w/ 8.2% confidence; "gibbon" w/ 99.3 % confidence

Figure 1.1: An example of an adversarial perturbation of an ImageNet example, where the perturbation is so small that it is imperceptible to a human observer despite changing the model's classification of the input. The model assigns higher confidence to the incorrect classification of the adversarial example that it assigned to the correct classification of the original image. The model in this example is GoogLeNet (Szegedy et al., 2014a). Figure reproduced with permission from Goodfellow et al. (2014b).

linear and nearest neighbor classifiers. In-domain examples that have been altered in this fashion are known as *adversarial examples*.

Adversarial examples are interesting from many different perspectives. First, they demonstrate that machine learning methods do not yet truly understand the tasks they are asked to perform, even though these methods often achieve human level performance (or better) on a test set consisting of naturally occurring inputs. Improving performance on adversarial examples therefore naturally implies achieving a deeper understanding of the underlying task. To this end, improvements in classification of adversarial examples can indeed lead to improvements on the original, non-adversarial classification task, as described in Section 1.3. Second, adversarial examples also have important implications for computer security, discussed in Section 1.2.1. Adversarial examples suggest that contemporary machine learning algorithms deployed against artificial perception tasks are performing fundamentally different computations than the human perceptual system, as discussed in Section 1.2.2. Finally, adversarial examples are interesting because they present a major difficulty for certain forms of model-based optimization. In scenarios where automated classification is useful but the major task of interest is a search for examples with desirable properties (e.g. drug design), one might be tempted to employ a well-performing differentiable classifier and perform gradient ascent with respect to the input. However, the existence and relative abundance of adversarial examples suggests this approach will most often be fruitless.

1.2.1 Cross-Model, Cross-Dataset Generalization and Security

A shocking property of adversarial examples, discovered by Szegedy et al. (2014b), is that a specific input point $\tilde{\boldsymbol{x}}$ that was designed to deceive one model (model A) will often also deceive another model, model B. When model B has a different architecture than model A, this is called *cross-model generalization* of adversarial examples. When model B was trained on a different training set than model A, this is called *cross-dataset generalization*. It is not fully understood why this happens, but Section 1.2.3 offers some intuitive justification.

Both Szegedy et al. (2014b) and Goodfellow et al. (2014b) present several experiments demonstrating the transfer rate between various model families and subsets of the training set. Additional experimental results unique to this chapter are presented in Table 1.1, using the same adversarial example generation procedure as Goodfellow et al. (2014b). The crafting model for the majority of these experiments was a maxout neural network of the same architecture employed for the permutation-invariant MNIST task in Goodfellow et al. (2013a). Additionally, transfer between a smoothed, differentiable version of nearest neighbor classification and conventional nearest neighbor is examined, where the prediction of the smoothed nearest neighbor classifier predicts a probability for class i via the formula

$$y_i(\boldsymbol{x}) = \frac{1}{N} \sum_{n=1}^{N} w_n y_i^{(n)}$$

where $y_i^{(n)}$ is equal to 1 if training example n has class i and 0 otherwise, and w_n is the softmax-normalized squared Euclidean distance from the test example $\boldsymbol{x}$ to training example $\boldsymbol{x}^{(n)}$,

$$w_n = \frac{\exp\left(-\|\boldsymbol{x} - \boldsymbol{x}^{(n)}\|^2\right)}{\sum_{m=1} \exp\left(-\|\boldsymbol{x} - \boldsymbol{x}^{(m)}\|^2\right)}$$

These results show that there is a non-trivial error rate even when the adversarial examples are crafted to fool a neural network, then deployed against an extremely different machine learning model such as nearest neighbor classification. Because these models are so different from each other, nearest neighbor has a lower error rate on the transferred adversarial examples than has usually been reported previously, but the error rate remains significant. These results also show that models that are not differentiable (such as nearest neighbor) can easily be attacked using cross model transfer from a differentiable model (maxout networks or smoothed nearest neighbor).

Crafting model	Target model	Error rate
Maxout network	Nearest neighbor	25.3%
Smoothed nearest neighbor	Nearest neighbor	47.2%
Maxout network	ReLU network	47.2%
Maxout network	Tanh network	99.3%
Maxout network	Softmax regression	88.9%

Table 1.1: Results of additional cross-model adversarial transfer experiments. The maxout crafting model is identical in architecture to that employed for the permutation-invariant MNIST task by Goodfellow et al. (2013a). The ReLU and Tanh neural networks each contained two layers of 1,200 hidden units each. All nerual networks were trained with dropout (Srivastava et al., 2014).

Cross-model, cross-dataset generalization of adversarial examples implies that adversarial examples pose a security risk even under a threat model where the attacker does not have access to the target's model definition, model parameters, or training set. The attacker can prepare a training set (for the same task), train a model on their own training set, craft adversarial examples that deceive their own model, and then deploy these adversarial examples against the target system.

Attacks that leverage cross-model and cross-dataset generalization of adversarial examples have been acknowledged as a theoretical possibility since the work of Szegedy et al. (2014b) introduced these effects. Papernot et al. (2016a) provided the first practical demonstration of attacks based on adversarial examples in a realistic scenario: they trained a classifier for the MNIST dataset using the MetaMind API, wherein the model parameters reside on MetaMind's servers and its definition is not disclosed to the user. By training another model locally and crafting adversarial examples that fooled it, the authors were able to successfully fool the model they had trained via the MetaMind API. This suggests that modern machine learning methods require new defenses before they can be safely used in situations where they might face an actual adversary.

1.2.2 Adversarial Examples and the Human Brain

It is natural to wonder whether the human brain is vulnerable to adversarial examples. At first glance, it seems difficult to test, because there is no known method for obtaining a description of the brain as a differentiable model in the form used by adversarial example construction algorithms. However, the cross-model, cross-dataset generalization property of adversarial examples suggests that if the brain were even remotely similar to modern machine learning algorithms, it should be fooled by the same images that fool machine learning models. So far this seems not to be the case.

However, the brain can be easily fooled by many illusions; see Robinson (2013) for a review. For example, optical illusions in which one line appears to be longer than another despite both lines being the same length can be interpreted as adversarial examples for the line length regression task.

Audible and visible stimuli can also cause a range of beneficial or detrimental involuntary side effects in human observers, ranging from pain relief to seizures. Many of these effects rely on synchronizing the temporal frequency of a visual stimulus to the temporal frequency of changes in brain activity measured by EEG. This might be analogous to adversarial example construction techniques that match a spatial pattern of inputs to the spatial distribution of neural network weights. See Frederick et al. (2005) for a useful review of the effects of audible and visible stimuli constructed using information from EEG.

1.2.3 The Linearity Hypothesis

When Szegedy et al. (2014b) discovered the existence of adversarial examples, their cause was unknown. Initially, they were suspected to be caused by neural networks being highly complex, non-linear models that can assign very random classifications to test set inputs.

Goodfellow et al. (2014b) argued that these explanations failed to explain two important experimental observations. First, adversarial examples affect some very simple models, such as shallow linear classifiers, just as much as they affect deep models. Model complexity and overfitting would therefore not seem to be the primary problem. Second, adversarial examples can consistently fool models other than the one from which they are initially derived, as described in Section 1.2.1. If adversarial examples were just a manifestation of overfitting, then different models should respond to each adversarial example differently. Goodfellow et al. (2014b) demonstrated, to the contrary, that distinct models not only mislabel the same adversarial examples, but also mislabel them with the same class.

Goodfellow et al. (2014b) introduced *the linearity hypothesis*, which predicts that most adversarial examples affecting current machine learning models arise due to the model behaving extremely linearly as a function of its inputs. To confirm this hypothesis, Goodfellow et al. (2014b) demonstrated that adversarial attacks against linear approximations of deep models are highly successful, and introduced visualizations showing that the *logits* (i.e. the inputs to a final softmax output layer) of a deep neural network classifier are piecewise linear with large pieces as a function of the input to the model. This hypothesis is based on the observation that modern deep networks are based on components that have been designed to be extremely linear, such as rectified linear units (Jarrett et al., 2009; Glorot et al., 2011). Though deep neural networks are very nonlinear as a function of their parameters, they can nonetheless be very linear as a function of their inputs. Deep rectifier networks divide input space into several regions, with the output of the rectified linear layers being linear within each region. These regions are often extremely large, especially compared to the size of perturbations used to construct adversarial examples.

To understand why linear functions are highly vulnerable to adversarial examples, consider the output of a regression model $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$. If the input is perturbed by $\epsilon \cdot \mathrm{sign}(\boldsymbol{w})$, then the output increases by $\epsilon||\boldsymbol{w}||_1$. When $\boldsymbol{w}$ is high dimensional, the increase in the output can be extremely large. In other words, linear functions can add up very many tiny pieces of evidence to reach an extreme conclusion. If $\boldsymbol{x}$ has large feature values that are not closely aligned with $\boldsymbol{w}$, it will have less of an effect on the output than a perturbation consisting of many small values that are all closely aligned to $\boldsymbol{w}$.

Even in low dimensional spaces, linear functions behave in ways that seem disadvantageous for machine learning. A logistic regression model applied to a one-dimensional input space that classifies an input of $x = -1$ as belonging to the negative class and an input of $x = 1$ as belonging to the positive class must classify an input of $x = 2$ as belonging to the positive class with extremely high confidence, even if no value as large as 2 occurred in the training set. Larger values of x result in more confidence, even if they are even farther from examples that were seen at training time.

Because neural networks are parameterized in terms of linear components, they are biased toward learning functions that make wild predictions when extrapolating far from previously seen inputs. In high-dimensional spaces, even small perturbations of each input can take the input vector very far in Euclidean distance from the starting point. This explains the majority of adversarial examples affecting modern neural networks.

It is natural to wonder how adversarial examples are distributed throughout space. For example, one could imagine that they are rare and occur in small, fine pockets that must be found with careful search procedures. The linearity hypothesis predicts instead that adversarial examples occupy large volumes of space. If the cost function $J(\boldsymbol{x}, y)$ increases in a roughly linear fashion in direction $\boldsymbol{d}$, then an adversarial example $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\eta}$ will be misclassified so long as $\boldsymbol{\eta}^\top \boldsymbol{d}$ is large. The linearity hypothesis thus predicts that a hyperplane where $\boldsymbol{\eta}^\top \boldsymbol{d} = C$ for some constant C divides the space $\mathbb{R}^n$ into two half-spaces. The original input $\boldsymbol{x}$ is correctly classified, and a large region of points on the same side of the hyperplane as $\boldsymbol{x}$ are also classified the same as $\boldsymbol{x}$. On the opposite side of the hyperplane, nearly all points have a different classification.

Goodfellow et al. (2014b) provided a variety of sources of indirect evidence for the linearity hypothesis. This chapter introduces some visualizations that show the resulting half-spaces of adversarial examples more directly. These visualizations are called *church window plots* due to their resemblance to stained glass windows. These plots show two-dimensional cross sections of the classification function, exploring input space near test set examples. Figure 1.2 shows cross sections exploring the adversarial direction defined by the fast gradient sign method and a random direction. Figure 1.3 shows cross sections exploring two random orthogonal directions. Figure 1.4 shows cross sections exploring two adversarial directions, with the first defined by the fast gradient sign method and the second defined by the component of the gradient that is orthogonal to the first direction.

1.2.4 Crafting Adversarial Examples

Several different methods of crafting adversarial examples are available. When adversarial examples were first discovered, they were generated with general purpose methods that make no assumption about the underlying cause of adversarial examples, but that are expensive and require multiple iterations. Later, inexpensive methods based on linearity assumptions were developed. Most adversarial example crafting techniques require training set labels, but *virtual adversarial examples* remove this requirement. Specialized methods provide fast methods of attacking classifiers specifically or crafting perturbations that change as few input dimensions as possible.

Let $\boldsymbol{x} \in \mathbb{R}^n$ be a vector of input features (usually the pixels of an image) and y be an integer specifying the desired output class of the model. Let f be the classification function learned by the model, so that $f(\boldsymbol{x})$ is an integer giving the model's prediction. Let $J(\boldsymbol{x}, y)$ be the cost used to train the model.

The goal of adversarial example crafting is to find an input point $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\eta}$ that causes the model to perform poorly. Different methods of crafting adversarial examples use different criteria to determine how poorly the model behaves, different approaches to limit the size of η, and different approximations to optimize the chosen criterion. In all cases, the goal is to find a perturbation $\boldsymbol{\eta}$ to be small enough that an ideal classifier (usually approximated by human judgment) would still assign class y to $\tilde{\boldsymbol{x}}$. Guaranteeing that $\tilde{\boldsymbol{x}}$ truly belongs to the same class as $\boldsymbol{x}$ is a subtle point, discussed further in Section 1.2.5.

Different methods of crafting adversarial examples quantify poor model

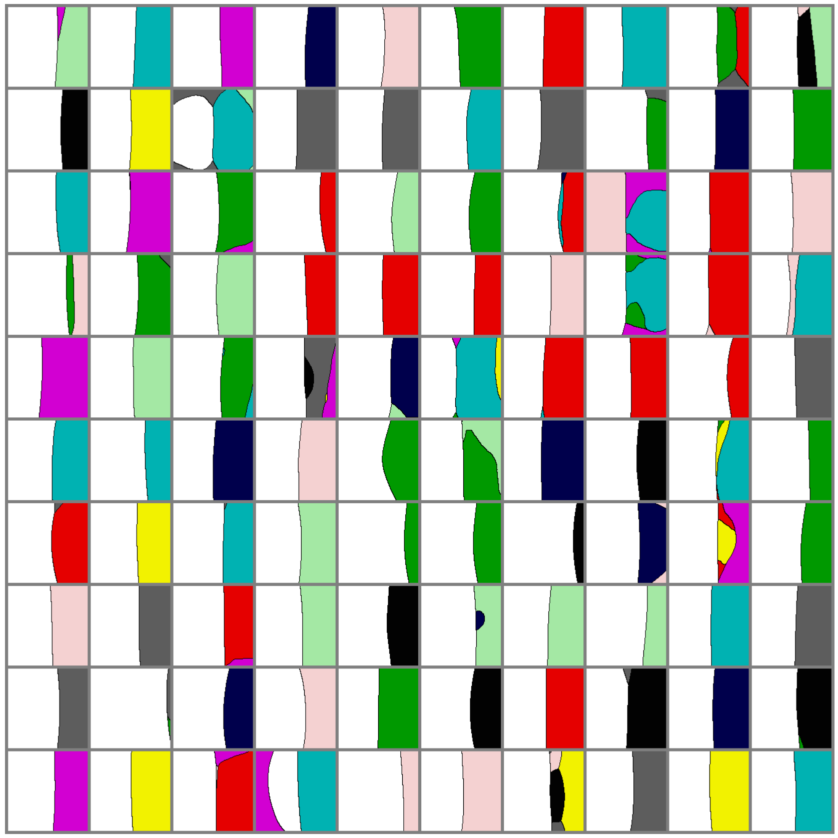


Figure 1.2: *Church window plots* applied to a convolutional network trained on CIFAR-10. The convolutional network is that of Goodfellow et al. (2013b). Each cell in the 10 × 10 grid in the figure is a different church window plot corresponding to a different CIFAR-10 test example. Here the model is viewed as a function $f : \mathbb{R}^n \to \{1, \ldots, 10\}$. At coordinate (h, v) within the plot, the pixel is drawn with a unique shade (in the book, the pixel is printed with a unique grayscale shade indicating the class, while on a computer monitor, each pixel may be displayed with a unique color indicating the class) for each class, indicating the class output by $f(\boldsymbol{x} + h\boldsymbol{u}^{(1)} + v\boldsymbol{u}^{(2)})$, where $\boldsymbol{u}^{(1)}$ and $\boldsymbol{u}^{(2)}$ are orthogonal unit vectors that span a 2-D subspace of $\mathbb{R}^n$. The correct class for each example, given by the test set label, is always plotted as white. To aid visibility, a black contour is drawn around the boundary of each class region. The horizontal coordinate h within the plot begins at $-\epsilon$ on the left side of the plot and increases to ϵ on the right side of the plot. The vertical coordinate v spans the same range, beginning at $-\epsilon$ at the top of the plot. The center of the plot thus corresponds to the classification of the unperturbed input $\boldsymbol{x}$. In all cases these visualizations use .25 for ϵ, which corresponds to large perturbations on our preprocessing of CIFAR-10. Such large perturbations seriously degrade the quality of the image but do not prevent a human observer from recognizing the class. In this figure, $\boldsymbol{u}^{(1)}$ is the direction defined by a fast technique for finding adversarial examples discussed later in this section, while $\boldsymbol{u}^{(2)}$ is a direction chosen uniformly at random among those orthogonal to $\boldsymbol{u}^{(1)}$. From this figure, one can see that the adversarial direction usually roughly divides space into a half-space of correct classification and incorrect classification, with the test example usually lying on the correct side but somewhat near the decision boundary. One can also see that in these cross-sections, the decision boundaries have simple, roughly linear, shapes.

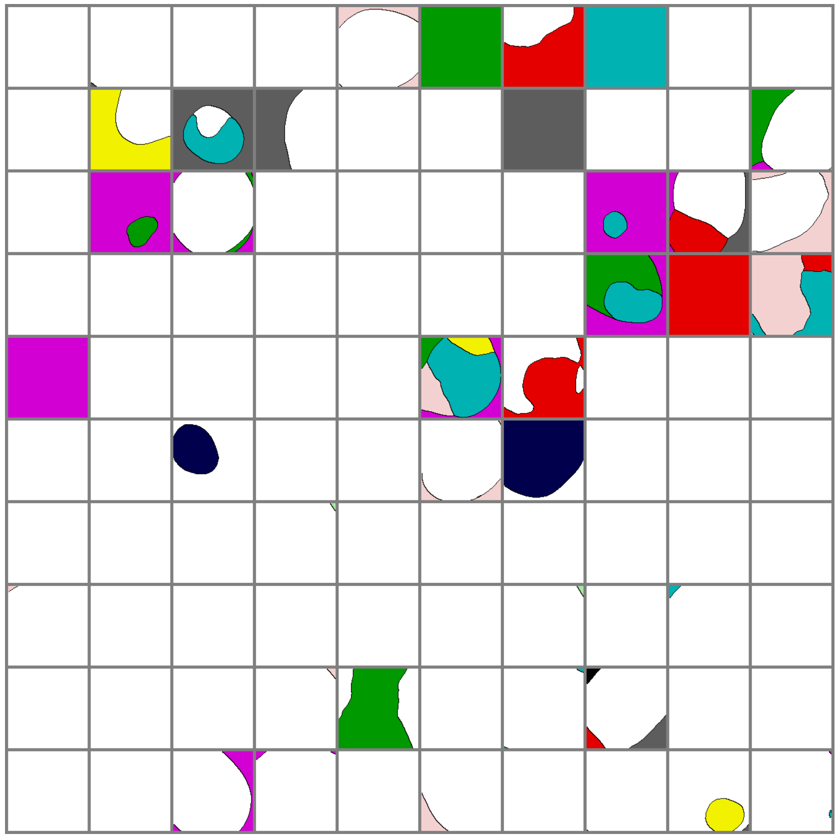


Figure 1.3: Church window plots with both basis directions chosen randomly. See Figure 1.2 for a description of church window plots. In this plot, one can see that random directions rarely cause the class to change. Many authors mistakenly speak of "adversarial noise." This figure illustrates that noise actually does not change the classification very often compared to adversarial directions of perturbation. The empirical observation that noise is less harmful than adversarial directions dates back to Szegedy et al. (2014b), but the church window plots make the mechanism clear. The classification decision is sensitive mostly to a small subspace of adversarial directions that are unlikely to be chosen randomly.

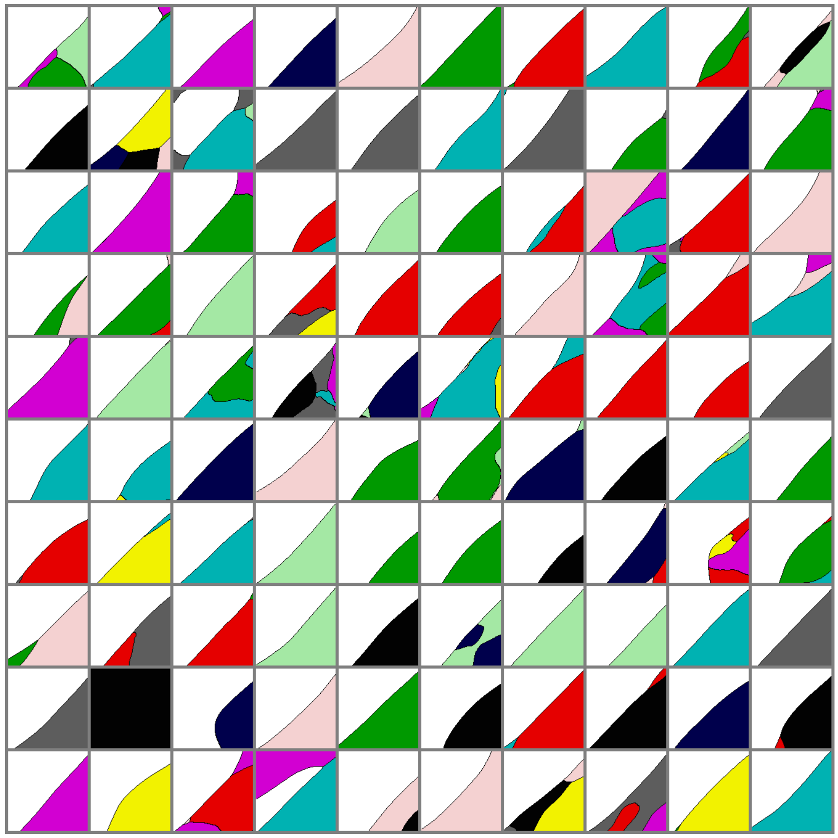


Figure 1.4: Church window plots with both basis directions chosen adversarially. See Figure 1.2 for a description of church window plots. In this plot, the first direction is the one given by the fast gradient sign method (Equation 1.3) and the second direction is the component of the gradient that is orthogonal to the first direction. One can still see linear decision boundaries within this subspace. From this one can see that adversarial examples do not lie in small pockets whose exact coordinates are difficult to find. Instead, adversarial examples may be found by moving in any direction that has large dot product with the gradient.

performance in different ways. Some methods are explicitly designed to cause the model to label $\tilde{\boldsymbol{x}}$ as belonging to class $\tilde{y}$, where $\tilde{y} \neq y$. Other methods make use of the cost function $J(\boldsymbol{x}, y)$ used to train the model, and seek a perturbation that results in a large (ideally, maximal) value of $J(\tilde{\boldsymbol{x}}, y)$.

Szegedy et al. (2014b) introduced the first method for crafting adversarial examples. This method was based on solving the optimization problem

$$\boldsymbol{\eta} = \underset{\boldsymbol{\eta}}{\operatorname{argmin}}\, \lambda||\boldsymbol{\eta}||_2^2 + J(\boldsymbol{x} + \boldsymbol{\eta}, \tilde{y}) \text{ subject to } (\boldsymbol{x} + \boldsymbol{\eta}) \in [0, 1]^n,$$

where $\tilde{y}$ is an incorrect class of the attacker's choice. The initial experiments on adversarial examples used box-constrained L-BFGS to accomplish the minimization, but in principle any gradient-based optimization algorithm would suffice. The minimization was repeated multiple times with different values of λ in order to find the smallest $\boldsymbol{\eta}$ that resulted in successfully causing $f(\boldsymbol{x} + \eta) = \tilde{y}$. This method is extremely effective, finds very small perturbations, and can cause the model to output specific, desired classes, and makes no assumptions about the structure of the model, but is also highly expensive, requiring multiple calls to an iterative optimization procedure for each example.

Szegedy et al. (2014b) included a constraint that $(\boldsymbol{x} + \boldsymbol{\eta}) \in [0, 1]^n$. This constraint ensures that the adversarial example has the same range of pixel values as the original data, and that it lies within the domain of the original function. Later authors frequently omitted this constraint for simplicity, because the perturbations $\boldsymbol{\eta}$ are typically small and thus do not move the input significantly far outside the original domain.

For the cost functions that are used to train neural network classifiers, such as $J(\boldsymbol{x}, y) = -\log P(y \mid \boldsymbol{x})$, a model that is linear over wide regions of its input domain also yields a cost that is approximately linear over wide regions of the input domain. This motivated the development of a fast adversarial example generation scheme based on a linear approximation of the cost function. The method of Szegedy et al. (2014b) fixes a desired target class and minimizes the size of $\boldsymbol{\eta}$. The method of Goodfellow et al. (2014b) simplifies the problem by fixing the allowed size of $\boldsymbol{\eta}$ and maximizing the cost incurred by the perturbation:

$$\boldsymbol{\eta} = \underset{\boldsymbol{\eta}}{\operatorname{argmax}}\, J(\boldsymbol{x} + \boldsymbol{\eta}, y) \text{ subject to } ||\boldsymbol{\eta}||_\infty \leq \epsilon, \tag{1.1}$$

where ϵ is a hyperparameter chosen by the attacker, specifying the maximum desired pertubation size. The use of the max norm $||\boldsymbol{\eta}||_\infty$ is motivated in Section 1.2.5, but this method could also work with other norms, including the L^2 norm. Solving Equation 1.1 requires iterative optimization in general.

To obtain a fast, closed-form solution, Goodfellow et al. (2014b) replaced J with a first-order Taylor series approximation:

$$\boldsymbol{\eta} = \operatorname*{argmax}_{\boldsymbol{\eta}} J(\boldsymbol{x}, y) + \boldsymbol{\eta}^\top \boldsymbol{g} \text{ subject to } ||\boldsymbol{\eta}||_\infty \leq \epsilon. \tag{1.2}$$

where $\boldsymbol{g} = \nabla_{\boldsymbol{x}} J(\boldsymbol{x}, y)$. The solution to Equation 1.2 is given by

$$\boldsymbol{\eta} = \epsilon \cdot \mathrm{sign}(\boldsymbol{g}) \tag{1.3}$$

This is called the *fast gradient sign method* of generating adversarial examples. The method has the advantage of being extremely fast compared to the L-BFGS method (computing the gradient once instead of hundreds of times), making adversarial example generation feasible for use within the inner loop of a learning algorithm, as described in Section 1.3. The method has some disadvantages, namely that its justification rests on the linearity hypothesis. In some cases, when the linear approximation poorly represents the function, this method requires larger perturbations than other methods. In extreme cases, such as when a model has been explicitly trained to resist the fast gradient sign method, the fast gradient sign method might cease to find adversarial examples while the L-BFGS method continues to do so. The L-BFGS method was also designed to cause the model to predict a specific class $\tilde{y}$ chosen by the attacker. While the fast gradient sign method as outlined above does not allow for the specification of a target class, it can be trivially extended to this setting by following the gradient of $\log P(\tilde{y} \mid \boldsymbol{x})$ rather than $J(\boldsymbol{x}, y)$. Finally, the fast gradient sign method is highly general because it is based on maximizing J. This allows it to be applied to models other than classifiers. For example, it can be used to find inputs to an autoencoder that incur high reconstruction error.

Both the L-BFGS method and the fast gradient sign method rely on access to the true class label y. Miyato et al. (2015) devised a way to remove this requirement. After a model has been at least partially trained, it is usually able to provide mostly accurate labels. Therefore, rather than making a perturbation intended to reduce the probability of the label provided in the training set, the attacker can make a perturbation intended to make the model change its prediction. *Virtual adversarial examples* are thus designed to approximately maximize

$$D_{\mathrm{KL}}\left(p(y \mid \boldsymbol{x}) \| p(y \mid \boldsymbol{x} + \boldsymbol{\eta})\right)$$

with respect to $\boldsymbol{\eta}$, under appropriate constraints on $\boldsymbol{\eta}$. The ability to construct adversarial examples without access to ground truth labels enables the use of adversarial examples for semi-supervised learning, described in

Section 1.3.1.

Other specialized methods of crafting adversarial examples provide different benefits. Huang et al. (2015) introduced an attack specialized for classifiers. While the fast gradient sign method linearizes the **cost function**, the attack of Huang et al. (2015) linearizes the **model**. Under the linear approximation of the model, it is possible to solve for the smallest perturbation that yields a change in the output class in closed form. By more tightly modeling the problem of changing the output class, this method is able to achieve class changes with smaller perturbation sizes than the fast gradient sign method.

Most methods of crafting adversarial examples change many input dimensions, each by a small amount. Papernot et al. (2016b) introduced a different approach, that changes few input dimensions, but may change each one by a large amount.

Finally, Sabour et al. (2015) showed that it is possible to construct adversarial examples that cause the model to assign a hidden representation to $\tilde{\boldsymbol{x}}$ that closely resembles the hidden representation of a different example $\boldsymbol{x}'$. For example, an image of farm equipment may be perturbed so that it has approximately the same hidden representation as an image of a bird. This is a stronger condition than perturbing the image to take on a specific class. For example, when the image of farm equipment is perturbed to have the same hidden representation as the image of a bird, the hidden representation may be decoded to obtain the same color of bird standing in the same location with the same pose—it is not just the concept of the output class "bird" that is imposed on the adversarial example.

1.2.5 Ensuring That Class Changes are Mistakes

One subtle point when constructing adversarial examples is that the perturbation $\boldsymbol{\eta}$ must not change the *true* class of the input – that is, the adversarial example should be such that for the task at hand, it would still be desirable that a classifier assign it the same class as it would the original. If $\boldsymbol{\eta}$ is "too large", an adversarial perturbation could subtract the true identifying characteristics of the original class identity and replace them with the true identifying characteristics of another class, yielding an adversarial example $\tilde{\boldsymbol{x}}$ that truly does belong to a different class $\tilde{y}$. In other words, it is sometimes correct for the classifier to change its class output when the input changes. Adversarial examples must be crafted in such a way that it remains a mistake for the class output to change.

So far there is no general principle determining how to tell whether the class should change for an arbitrary new input, and it seems that if such a principle were known there would no longer be a need for machine learning classifiers. Instead, Goodfellow et al. (2014b) advocate devising a set of sufficient conditions that guarantee that a perturbation $\boldsymbol{\eta}$ will not change the class for a particular application area. For the specific application of object recognition in images, Goodfellow et al. (2014b) suggests that a perturbation $\boldsymbol{\eta}$ that does not change any specific pixel by more than some amount ϵ cannot change the output class. The value of ϵ should be chosen based on knowledge of the task. For example, on the MNIST dataset, the input values are typically normalized to lie in the range $[0, 1]$. The images are of written digits, typically displayed as white digits on a black background. The information content of each pixel is thus roughly binary. Consequently, ϵ may be chosen to be quite large for this task. An ϵ of .25 turns a white pixel with value 1.0 into a bright gray pixel with value .75, which may still easily be recognized as carrying the same semantics as a white pixel. Because some pixels in the original data are gray, perturbations larger than .25 become difficult for human observers to classify. For other object recognition datasets, one might choose ϵ to be small enough that the change to a pixel is imperceptible to a human observer, or to be small enough that a change to the 32 bit floating point encoding of the input does not change the 8 bit representation used to store the images on disk. This principle of ensuring that no pixel changes by more than some negligible amount motivates the use of the max norm to constrain the size of $\boldsymbol{\eta}$ in Equation 1.1. Figure 1.5 provides some illustrations showing how the max norm can be superior to the L^2 norm for ensuring that perturbations do not alter the true class.

The use of the max norm to constrain $\boldsymbol{\eta}$ is of course a sufficient condition for preventing a class change when the task is object recognition. One could imagine other tasks where no norm of $\boldsymbol{\eta}$ provides a useful restriction on the perturbation. For example, consider a regression task where the true output should be $\boldsymbol{d}^\top \boldsymbol{x}$. Then any perturbation that has non-zero dot product with $\boldsymbol{d}$ will change the true output that the regression model should return. The norm of the perturbation is not relevant for this hypothetical task, but rather the direction.

1.2.6 Rubbish Class Examples

Adversarial examples are closely related to the idea of rubbish class examples (LeCun et al., 1998). Rubbish class examples are pathological inputs that do not belong to any class encountered during training. For example, an image where the pixels are drawn from a uniform distribution usually does not belong to any class of images of objects. Ideally, one would like a classifier that assigns normalized probabilities to various output classes to report a

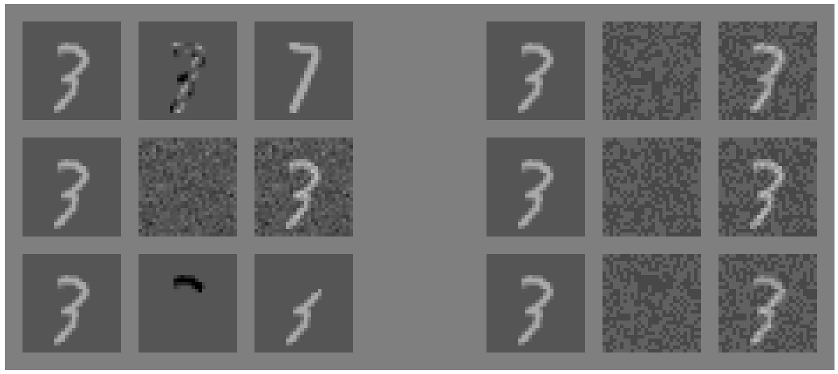


Figure 1.5: Examples of perturbations, illustrating that an L^2 perturbation can behave unpredictable, while a perturbation subject to a max norm constraint can be guaranteed to preserve the object class. The grid on the left shows the result of L^2-constrained perturbation while the grid on the right shows the result of max norm-constrained perturbation. Within each grid, each row shows a the results of a single perturbation. Each row of three images consists of (left to right) an image of input $\boldsymbol{x}$, an image of a perturbation $\boldsymbol{\eta}$, and an image of a resulting perturbed input, $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\eta}$.
In the grid on the left, three different perturbations are shown. From top to bottom, the first perturbation causes the true class to change from 3 to 7 (the pertubation is just the difference between an example 7 and an example 3 from the dataset), the second perturbation causes no change, and the final perturbation causes the class to change from 3 to the rubbish class. All three of these perturbations have the same L^2 norm.
In the grid on the right, see three new perturbations that still have the same L^2 norm as the first three, but that have been modified to obey a max norm constraint. These perturbations were constructed by taking the sign of the corresponding perturbation on the left, assigning zero entries to be -1 or 1 randomly, and multiplying by a scaling factor. Randomly replacing zero entries with -1 or 1 is necessary to increase the perturbation size enough to maintain the same L^2 norm as the perturbation on the left. None of the max norm constrained perturbations change the class.
All six perturbations shown have the same L^2 norm but yield different outcomes. This suggests that the L^2 norm is not a useful way of constraining $\boldsymbol{\eta}$ while constructing adversarial examples for object recognition. The max norm provides a sufficient (but not necessary) condition that guarantees an adversarial example will not change the true underlying class.
The perturbations used in this visualization are relatively small, with an L^2 norm of roughly 4 for all six perturbations and a max norm of roughly 0.14 for the perturbations on the right. When using the max norm constraint, it is possible to construct adversarial examples with max norm .25. Such perturbations have a L^2 norm of 7.

uniform distribution over output classes when presented with such an input. Similarly, a model that reports an independent probability estimate for the detection of each class should preferably indicate that no classes are present. However, both formulations are easily fooled into reporting that a specific class is present with high probability simply by using Gaussian noise as input to the model (Goodfellow et al., 2014b). Nguyen et al. (2014) demonstrated that large, state of the art convolutional networks can also be fooled using rich, structured images generated by genetic algorithms. Because rubbish class examples do not correspond to small perturbations of a realistic input example, they are beyond the scope of this chapter.

1.2.7 Defenses

To date, the most effective strategy for defending against adversarial examples is to explicitly train the model on them, as described in Section 1.3.

Many traditional regularization strategies such as weight decay, ensemble methods, and so on, are not viable defenses. Regularization strategies can fail in two different ways. Some of them reduce the error rate of the model on the test set, but do not reduce the error rate of the model on adversarial examples. Others reduce the sensitivity of the model to adversarial perturbation, but only have a significant effect if they are applied so powerfully (e.g., with such a large weight decay coefficient) that they cause the performance of the model to seriously degrade on the validation set. The failure of some traditional regularization strategies to provide a defense against adversarial examples is discussed by Szegedy et al. (2014b) and the failure of many more traditional regularization strategies is discussed by Goodfellow et al. (2014b). In summary, most traditional neural network regularization techniques have been tested and do not provide a viable defense.

In addition to these traditional methods, some new methods have been devised to defend against adversarial examples. However, none of these methods are yet very effective. For even the best methods, the error rate on adversarial examples remains noticeably higher than on unperturbed examples.

Gu and Rigazio (2014) trained a denoising autoencoder, where the noise corruption process was the adversarial example generation process. In other words, the autoencoder is trained with adversarial example to predict the corresponding unperturbed example as output. The goal was to use the autoencoder as a preprocessing step before applying a classifier, in order to make the classifier resistant to adversarial examples. Unfortunately, the combination of the autoencoder and the classifier then becomes vulnerable to a different class of adversarial examples that the autoencoder has not been trained to resist. Gu and Rigazio (2014) reported that the combined system was vulnerable to adversarial examples with smaller perturbation size than the original classifier. The authors of this chapter speculate that this can be explained by the linearity hypothesis; the autoencoder is still built mostly out of linear components. If one views the classifier as being roughly a product of matrices, then the autoencoder simply introduces two more matrix factors into this product. If these matrices have any singular values that are larger than one, then they amplify adversarial perturbations in the corresponding directions.

Papernot et al. (2015) introduced an approach called *defensive distillation*. First, a teacher model is trained to maximize the likelihood of the training set labels:

$$\boldsymbol{\theta}^{(t)*} = \operatorname{argmax} \sum_{i=1}^{m} \log p^{(t)} \left(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta}^{(t)} \right).$$

The teacher model is then used to provide soft targets for a second network, called the student network. The student network is trained not just to predict the same class as the teacher network, but to predict the same probability distribution over classes:

$$\boldsymbol{\theta}^{(s)*} = \underset{\boldsymbol{\theta}^{(s)}}{\operatorname{argmin}} \sum_{i=1}^{m} D_{\mathrm{KL}} \left(p^{(t)} \left(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta}^{(t)} \right) \| p^{(s)} \left(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta}^{(s)} \right) \right).$$

This technique noticeably reduces the vulnerability of a model to adversarial examples but does not completely resolve the problem.

As an original contribution of this chapter, an experimental observation shows that a simpler method than defensive distillation also has a beneficial effect. Rather than training a teacher network to provide soft targets, it is possible to simply modify the targets from the training set to be soft, e.g., for a k class problem, replace a target value of 1 for the correct class with a target value of .9, and for the incorrect classes replace the target of 0 with a target of $\frac{1}{10k}$. This technique is called *label smoothing* and is a component of some state of the art object recognition systems (Szegedy et al., 2015). The label smoothing experiment was based on a near-replication of the MNIST classifier of Goodfellow et al. (2013a). This classifier is a feedforward network with two hidden layers consisting of maxout units, trained with dropout (Srivastava et al., 2014). The model was trained on only on the first 50,000 examples and was not re-trained on the validation set, so the test error rate was higher than in the original investigation of Goodfellow et al. (2013a). The model obtained an error rate of 1.28% on the MNIST test set. The error rate of the model on adversarial examples on the MNIST test set using the fast gradient sign method (Equation 1.3) with $\epsilon = .25$ was 99.97%. A second instantiation of exactly the same model was trained using label smoothing. The error on the test set dropped to 1.17%, and the error rate on the adversarially perturbed test set dropped to 33.0%. This error rate indicates a significant remaining vulnerability but it is a vast improvement over the pre-smoothing adversarial example error rate.

The linearity hypothesis can explain the effectiveness of label smoothing. Without label smoothing, a softmax classifier is trained to make infinitely confident predictions on the training set. This encourages the model to learn large weights and strong responses. When values are pushed outside the areas where training data concentrates, the model makes even more extreme predictions when extrapolating linearly. Label smoothing penalizes the model for making overly confident predictions on the training set, forcing it to learn either a more non-linear function or a linear function with smaller slope. Extrapolations by the label-smoothed model are consequently less extreme.

1.3 Adversarial Training

Adversarial training corresponds to the process of explicitly training a model to correctly label adversarial examples. In other words, given a training example $\boldsymbol{x}$ with label y, the training set may be augmented with an adversarial example $\tilde{\boldsymbol{x}}$ that is still associated with training label y. Szegedy et al. (2014b) proposed this method, but were unable to generate large amounts of adversarial examples due to reliance on the expensive L-BFGS method of crafting adversarial examples. Goodfellow et al. (2014b) introduced the fast gradient sign method and showed that it enabled practical adversarial training. In their approach, the model is trained on a minibatch consisting of both unmodified examples from the training set and adversarially perturbed versions of the same examples. Crucially, the adversarial perturbation is recomputed using the latest version of the model parameters every time a minibatch is presented. Adversarial training can be interpreted as a minimax game,

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \mathbb{E}_{\boldsymbol{x},y} \max_{\boldsymbol{\eta}} \left[J(\boldsymbol{x}, y, \boldsymbol{\theta}) + J(\boldsymbol{x} + \boldsymbol{\eta}, y) \right],$$

with the learning algorithm as the minimizing player and a fixed procedure (such as L-BFGS or the fast gradient sign method) as the maximizing player.

Goodfellow et al. (2014b) found that adversarial training on MNIST reduced both the test set error rate and the adversarially perturbed test set error rate of a maxout network. The reduction in error rate on the unperturbed test set is presumably due to adversarial training forcing the model to learn a more parsimonious function that can explain a wide variety of adversarial examples with a small number of parameters.

Training with the fast gradient sign method means that the model is selectively resistant to adversarial examples that were constructed with this method. However, some resistance to other forms of adversarial examples is achieved. Goodfellow et al. (2014b) reported that their maxout network had an error rate of 18% on the MNIST test set when perturbed by the fast gradient sign method with $\epsilon = .25$. This chapter introduces the observation that using gradient descent on the true model to find the best perturbation with max norm less than .25 increases the error rate to 97%. However, this does not mean that adversarial training with the fast gradient sign method was ineffective. If the max norm constraint is tightened to $\epsilon = .1$, then the error rate of the adversarially trained maxout network falls to 22%. Without adversarial training, the error rate at this perturbation magnitude is 79%. Adversarial training with the fast gradient sign method thus confers robustness to other types of perturbation, but with a smaller perturbation size than was used for training.

1.3.1 Virtual Adversarial Training

Miyato et al. (2015) extended adversarial training to the semi-supervised setting by introducing the virtual adversarial example construction technique, which allows the construction of adversarial examples when no class label is available. This approach allows the model to be trained to have a highly robust classification function in the neighborhood of unlabeled examples. This technique improved the state of the art on semi-supervised learning on the MNIST dataset, outperforming much more complicated methods based on training generative models of unlabeled examples.

1.4 Generative Adversarial Networks

The generative adversarial network (GAN) framework introduced in Goodfellow et al. (2014a) phrases the problem of estimating a generative model in terms of a sample generation process $G : \mathbb{R}^d \to \mathbb{R}^n$, which takes as its argument a random variate $\boldsymbol{z} \sim p(\boldsymbol{z})$; $p(\boldsymbol{z})$ is often chosen from some simple family such as an isotropic Gaussian distribution, or a uniform distribution on $[-1, 1]^d$. $G(\cdot)$ is a machine parameterized by Θ_G which learns to map a sample from the base distribution $p(\boldsymbol{z})$ to a corresponding sample from an implicitly defined distribution $p_g(\boldsymbol{x})$. The combined procedure of drawing a sample $\boldsymbol{z}$ from $p(\boldsymbol{z})$ and applying G to $\boldsymbol{z}$ is referred to as the *generator*.

In contrast with many existing generative modeling frameworks, GANs may be trained without an explicit algebraic representation of $p_{\text{model}}(x)$, tractable or otherwise. The GAN framework is compatible with some models that explicitly define a probability distribution—any directed graphical model whose sampling process is compatible with stochastic backpropagation (Williams, 1992; Kingma and Welling, 2014; Rezende et al., 2014) may be used as a GAN generator—but the framework does not require explicit specification of any conditional or marginal distributions, only the sample generation process. In frameworks based on explicit specification of probabilities it is typical to maximize the empirical expectation of $\log p_{\text{model}}(\boldsymbol{x})$, applying Monte Carlo or variational approximations if faced with intractable terms (often in the form of a normalizing constant). Instead, GANs are trained to match the data distribution indirectly with the help of a *discriminator*, i.e. a binary classifier $D : \mathbb{R}^n \to [0, 1]$, parameterized by Θ_D, whose output represents a calibrated probability estimate that a given example was sampled from $p_{\text{data}}(\boldsymbol{x})$. The conditional log likelihood of the discriminator, on a balanced dataset of real and synthetic examples, is (in the usual fashion) *maximized* with respect to the parameters of D, but simultaneously *minimized* with respect to the parameters of G.

1.4.1 Adversarial Networks In Theory and Practice

The joint training procedure for the generator G and the discriminator D can be viewed as a two-player, continuous minimax game with a certain value function. In their introduction of the GAN framework, Goodfellow et al. (2014a) proved that the GAN training criterion has a unique global optimum in the space of distributions represented by G and D, wherein the distribution sampled by the generator exactly matches that of the data generating process, and the discriminator D is completely unable to distinguish real data from synthetic. It can also be proved, under certain assumptions, that the game converges to this optimum if G is improved at every round and D is chosen to be the ideal discriminator between $p_g(\boldsymbol{x})$ and $p_{\text{data}}(\boldsymbol{x})$, i.e. $D^\star(\boldsymbol{x}) = p_{\text{data}}(\boldsymbol{x})/(p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x}))$.

Goodfellow (2014) advanced the theoretical understanding of the GAN training criterion and its relationship to other distinguishability-based learning criteria. In particular, *noise-contrastive estimation (NCE)* (Gutmann and Hyvarinen, 2010) can be viewed as a variant of the GAN criterion wherein the generator is fixed, and the discriminator is a generatively parameterized classifier that learns an explicit model of $p(\boldsymbol{x})$ as a side effect of discriminative training, while a variant of noise contrastive estimation employing (a copy of) the learned generative model is shown to be equivalent, in expectation, to maximum likelihood. Perhaps most importantly, Goodfellow (2014) noted a subtlety of theoretical results outlined above, pointing out that they are significantly weakened by the setting in which GANs are typically optimized in practice.

Optimization of the generator and discriminator necessarily takes place in the space of parameterized families of functions, and the cost surface in the space of these parameters may have symmetries and other pathologies that imply non-uniqueness of the optima as well as practical difficulties locating them. One does not typically have analytical access to $p_g(\boldsymbol{x})$ and certainly not to $p_{\text{data}}(\boldsymbol{x})$, and must attempt to infer the optimal discriminator from data and samples. It is often prohibitively expensive to fully optimize the parameters of D after every change in the parameters of G – therefore, in practice, one settles for a parameter update aimed at improving D, such as one or more stochastic gradient steps. This means that the generator's role in the minimax game of minimizing with respect to $p_g(\boldsymbol{x})$ given a *maximum* of the value function with respect to D, is instead *minimizing a lower bound* on the correct objective. It is not at all clear whether the minimization of this lower bound improves the quantity of interest or simply loosens the bound.

Note that Goodfellow et al. (2014a) optimize a slightly different but equivalent criterion than described above. Let $D(x) = p\,(\boldsymbol{x} \text{ is data} \mid \boldsymbol{x})$, the discriminator's estimate that a given sample $\boldsymbol{x}$ comes from the data. Rather than minimize

$$\mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})} \log\left(1 - D(G(\boldsymbol{z}))\right)$$

(a term that already appears in the training criterion for the discriminator) with respect to the parameters of G, one can instead maximize $\mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})} \log\left(D(G(\boldsymbol{z}))\right)$; this criterion was found to work better in practice. The motivation for this lies in the fact that early in training, when G is producing samples that look nothing at all like data, the discriminator D can quickly learn to distinguish the two and $\log\left(1 - D(G(\boldsymbol{z}))\right)$ can quickly saturate to zero. The derivative of the per-sample objective contains a factor of $(1 - D(G(\boldsymbol{z})))^{-1}$, thus scaling the gradients which G receives via backpropagation to have very small magnitude. Pushing upward on $\log(D(G(\boldsymbol{z})))$ yields a multiplicative factor of $D(G(\boldsymbol{z}))^{-1}$ instead, resulting in gradients with a more favourably scaled magnitude if $D(G(\boldsymbol{z}))$ is small.

As G and D are both parameterized learners, the balance between the respective modeling capacities (and effective capacities during learning) can have a profound effect on the learning dynamics and the success of generative learning. In particular, the discriminator must be sufficiently flexible to reliably model the difference between the data distribution and the generated distribution, as the latter gradually tends towards reproducing the statistical structure of the former. At the same time, the discriminator must not become too effective too quickly, or else the gradients it provides the generator will be uninformative: no small change in the generated sample will move it significantly closer to the discriminator's decision boundary.

1.4.2 Generator collapses

Note that in theory, a perfectly optimal discriminator could exploit any subtle mismatch between $p_{\text{data}}(\boldsymbol{x})$ and $p_g(\boldsymbol{x})$ to give itself a better-than-chance ability to correctly distinguish real and synthetic examples; the generator could then use the gradients obtained from this optimal discriminator to correct its misallocations of probability mass. In practice, when using richly parameterized neural networks for generation and discrimination, the objective functions used to train the generator are non-convex and (due to the dependence between the learning tasks for the generator and the discriminator) highly nonstationary; it is impractical and even theoretically intractable to globally optimize the discriminator prior to each change in the generator. A failure mode for the training criterion therefore manifests when the generator learns to place too much probability mass on a subregion of the data distribution. In the most extreme cases, a generator could elect to place all of its mass on a single point, perfectly reproducing a single training example. A well-trained discriminator can quickly learn to exploit this and confidently classify every other point in the training set correctly. This presents a problem for generator learning, in that the gradients the generator receives are entirely with respect to a single synthetic example, most local perturbations of which will result in gradients that point back towards the singularity. To date, strategies to mitigate this type of failure are an active area of research. Radford et al. (2015) noted that the judicious use of batch normalization (Ioffe and Szegedy, 2015) appears, empirically, to prevent these kinds of collapses to a large degree.

1.4.3 Sample Fidelity and Learning the Objective Function

Machine learning problems are classically posed in terms of an objective function that is a fixed function of the parameters given a training set, often the log likelihood of training data under some parametric model. Viewed from the perspective of the generator G, the GAN training procedure does not involve a single, fixed objective function: G's objective is defined at any moment by the discriminator D, the parameters of which are being continually adapted to both the data and to the current state of G. This can be considered a *learned objective function*, whereby the objective function for G is *automatically* adapted to the data distribution being estimated. The inductive bias for G is characterized by the family of functions from which D is chosen: G is optimized so as to elude detection via any statistical difference between p_g and p_{data} that D can learn to detect.

It is this property that is arguably responsible for the perceived visual quality of generated samples of GANs trained on natural images. Models trained via objective functions involving reconstruction terms, such as the variational autoencoder (Kingma and Welling, 2014; Rezende et al., 2014), implicitly commit to a static definition of sample plausibility. In the case of conditionally Gaussian likelihood, this takes the form of mean squared error, which is a particularly poor perceptual metric for natural image pixel intensities: it considers all perturbations of a given magnitude equivalent, without regard for the fact that changes in luminance which blur out sharp edges decrease the plausibility of the sample as a natural image much more than minor shifts in chroma across the entire image. While one popular approach in the case of models of natural images, and in many other domains, is to design the static objective so as to mitigate the mismatch between training criterion and the statistical properties of the domain, the solution offered by GANs is in some sense more universal: train D to detect and exploit any difference it can between the distributions of samples and real data, train G to outwit this new discriminator, and repeat. This often results in generated samples that more closely match human conceptions of saliency, illustrated in Figure 1.6 in an application to parameterized image generation, where an adversarial loss allows the model to accurately extrapolate the presence of ears, a visually salient feature which a model trained with mean squared error sees fit to discard.

1.4.4 Extensions and Refinements

Since the initial introduction of generative adversarial networks, the framework has been extended in several notable directions. Many of these rely on a straightforward extension to the *conditional* setting, where the generator and discriminator receive additional contextual inputs, first explored by Mirza and Osindero (2014). For example, in the aforementioned work, the authors train a class-conditional generator on the MNIST handwritten digits by feeding the network an additional input consisting of a “one-hot” vector indicating the desired class. The discriminator is fed the generated or

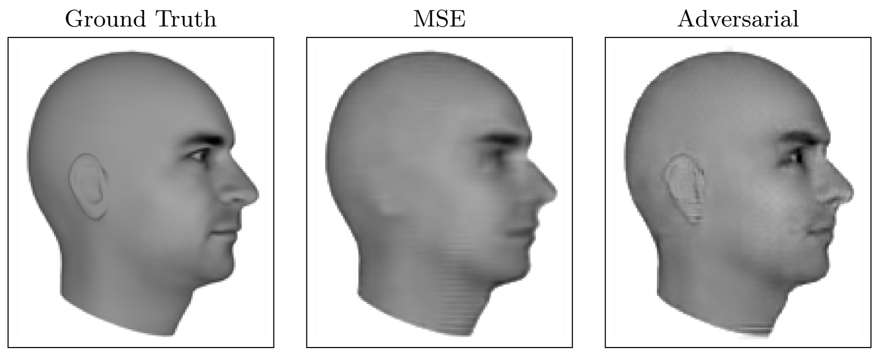


Figure 1.6: Predictive generative networks provide an example of how a learned cost function can correspond more closely to human intuition for which aspects of the data are salient and important to model than a fixed, hand-designed cost function such as mean squared error. These images show the results when predictive generative networks are trained to generate images of 3-D models of human heads at specified viewing angles. *(Left)* An example output frame from the test set. This is the target image that the model is expected to predict. *(Center)* When trained using mean squared error, the model fails to predict the presence of ears. Ears are not salient under the mean squared error loss because they do not cause a major change in brightness for a large enough number of pixels. *(Right)* When trained using a combination of mean squared error and adversarial loss, the model successfully predicts the presence of ears. Because ears have a repeated, predictable structure, they are highly salient to the discriminator network. Future research work may discover better ways of determining which aspects of the input should be considered salient. Figures reproduced with permission from Lotter et al. (2015).

real image as well as the class label (the assigned label if the image is real, the desired label if the image is generated). Through training, the discriminator learns that in the presence of a given class label, the image should resemble instances of that class from the training data. Likewise, in order to succeed at fooling the discriminator, the generator must learn to use the class label input to inform the characteristics of its generated sample.

In pursuit of more realistic models of natural images, Denton et al. (2015) introduced a hierarchical model, dubbed LAPGAN, which interleaved conditional GAN generators with spatial upsampling in a Laplacian pyramid (Burt et al., 1983). The first generator, either class-conditional or traditional, is trained to generate a small thumbnail image. A fixed upsampling and blurring is performed and a second conditional generator, conditioned on the newly upsampled image, is trained to reproduce the *difference* between the image at the current resolution and the upsampled thumbnail. This process is iterated, with subsequent conditional generators predicting residuals at ever higher resolutions.

Also in the space of natural image generation, Radford et al. (2015) leveraged recent advances in the design and training of discriminative convolutional networks to successfully train a single adversarial pair to generated realistic images of relatively high resolution. These generator networks employ "fractionally strided convolutions", otherwise recognizable as the transpose operation of "valid"-mode strided convolution commonly used when back-propagating gradients through a strided convolutional layer, to learn their own upsampling operations. The authors identify a set of architectural constraints on the generator and discriminator which allow for relatively stable training, including the elimination of downsampling in favour of strided convolution in the discriminator, the use of the bounded tanh() function at the generator output layer, careful application of batch normalization (Ioffe and Szegedy, 2015) and the use of rectified linear units (Jarrett et al., 2009; Glorot et al., 2011) and leaky rectified linear units (Maas et al., 2013) throughout the generator and discriminator, respectively. Inspired by recent work on word embeddings (e.g. Mikolov et al. (2013)), the authors also interrogate the latent representations, i.e. samples from $p(\boldsymbol{z})$, and find that they obey surprising arithmetic properties when trained on a dataset of faces as shown in Figure 1.7.

1.4.5 Hybrid Models

A recent body of work has examined the combination of the adversarial network training criterion with other formalisms, notably autoencoders. Larsen et al. (2015) combine a GAN with a variational autoencoder (VAE) (Kingma

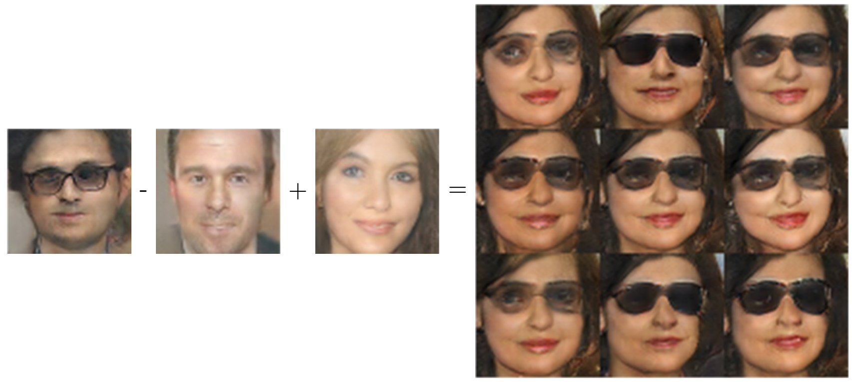


Figure 1.7: Deep Convolutional Generative Adversarial Networks (DCGANs) learn distributed representations that can separate semantically distinct concepts from each other. In this example, a DCGAN has learned one direction in representation space that corresponds to gender and another direction that corresponds to the presence or absence of glasses. Arithmetic can also be performed in this vector space. From left to right, let $\boldsymbol{a}$ be the representation of an image of a man with glasses, $\boldsymbol{b}$ the representation of a man without glasses, and $\boldsymbol{c}$ the representation of a woman without glasses. The vector $\boldsymbol{d} = \boldsymbol{a} - \boldsymbol{b} + \boldsymbol{c}$ now represents the concept of a woman with glasses. The generator maps $\boldsymbol{d}$ to rich images from this class. Images reproduced with permission from Radford et al. (2015).

and Welling, 2014; Rezende et al., 2014), dispensing with the VAE's reconstruction error term in favor of an squared error expressed in the space of the discriminator's hidden layers, combining the resulting modified VAE objective with the usual GAN objective. Makhzani et al. (2015) employs an adversarial cost as a regularizer on the hidden layer representation of a conventional autoencoder, forcing the aggregate posterior distribution of the hidden layer to match a particular synthetic distribution. This formulation closely resembles the VAE. The VAE maximizes a lower bound on the log likelihood that includes both a reconstruction term and terms regularizing the variational posterior to resemble the model's prior distribution over the latent variables. The adversarial autoencoder removes the regularization term and uses the adversarial game to enforce the desired conditions.

The adversarial network paradigm has also been extended in the direction of supervised and semi-supervised learning. Springenberg (2016) generalizes the convention adversarial network setting to employ a categorical (softmax) output layer in the discriminator. The discriminator and generator compete to shape the *entropy* of this distribution while respecting constraints on its marginal distribution, and an optional likelihood term can add semantics to this output layer if class labels are available. Sutskever et al. (2015) propose an unsupervised criterion designed expressly with the intent of improving performance on downstream supervised tasks in settings where the space of possible outputs is large, and it is easy to obtain independent examples from both the input and output domains. The proposed supervised mapping is adversarially trained to have an output distribution resembling the distribution of independent output domain examples.

1.4.6 Beyond Generative Modeling

Generative adversarial networks were originally introduced in order to provide a means of performing generative modeling. The idea has since proven to be more general. Adversarial pairs of networks may in fact be used for a broad range of tasks.

Two recent methods have shown that the adversarial framework can be used to impose desired properties on the features extracted by a neural network. The feature extractor can be thought of as analogous to the generator in the GAN framework. A second network, analogous to the discriminator, then tries to obtain some forbidden information from the extracted features. The feature extractor is then trained to learn features that are both useful for some original task, such as classification, and that yield little information to the second network. Ganin and Lempitsky (2015) use this approach for domain adaptation. The second network attempts to predict which domain the input was drawn from. When the feature extractor is trained to fool this network, it is forced to learn features that are invariant to the choice of input domain. Edwards and Storkey (2015) use a similar technique to learn representations that do not contain private information. In this case, the second network attempts to recover the private information from the representation. This approach could be used to remove prejudice from a decision making process. For example, if a machine learning model is used to make hiring decisions, it should not use protected information such as the race or gender of applicants. If the machine learning model is trained on the decisions made by human hiring managers, and if the previous hiring managers made biased decisions, the machine learning model could discover other features of the candidates that are correlated with their race or gender. By applying the method of Edwards and Storkey (2015), the machine learning model is encouraged to remove features that have a statistical relationship with the protected information, ideally leading to more fair decisions.

1.5 Discussion

The staggering gains in many application areas brought by the introduction of deep neural networks have inspired much excitement and widespread adoption. In addition to remarkable success tackling difficult supervised classification tasks, it is often the case that even misclassifications the errors made by state-of-the-art neural networks appear to be quite reasonable (as remarked, for example, by Krizhevsky et al. (2012)). The existence of adversarial examples as a problem plaguing a wide variety of model families suggests surprising deficits both in the degree to which these models understand their tasks, and to which human practitioners truly understand their models. Research into such phenomena can yield immediate gains in robustness and resistance to attack for neural networks deployed in commercial and industrial systems, as well as guide research into new model classes which naturally resist such perturbation through a deeper comprehension of the learning task.

Simultaneously, the adversarial perspective can be fruitfully leveraged for tasks other than simple supervised learning. While the focus of generative modeling in the past has often been on models that directly optimize likelihood, many application domains express a need for realistic synthesis, including the generation of speech waveforms, image and video inpainting and super-resolution, the procedural generation of video game assets, and forward prediction in model-based reinforcement learning. Recent work (Theis et al., 2015) suggests that these goals may be at odds with this likelihood-centric paradigm. Generative adversarial networks and their extensions provide one avenue attack on these difficult synthesis problems with an intuitively appealing approach: to learn to generate convincingly, aim to fool a motivated adversary. An important avenue for future research concerns the quantitative evaluation of generative models intended for synthesis; particular desiderata include generic, widely applicable evaluation procedures which nonetheless can be made to respect domain-specific notions of similarity and verisimilitude.

Acknowledgements

The authors of this chapter would like to thank Martin Wattenberg and Christian Szegedy for insightful suggestions that improved the church window plots, and to thank Martin in particular for the name “church window plots.” Ilya Sutskever provided the observation that visual stimuli can cause

seizures.

1.6 References

P. J. Burt, Edward, and E. H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31:532–540, 1983.

E. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. *NIPS*, 2015.

H. Edwards and A. J. Storkey. Censoring representations with an adversary. *CoRR*, abs/1511.05897, 2015. URL `http://arxiv.org/abs/1511.05897`.

J. A. Frederick, D. L. Timmermann, H. L. Russell, and J. F. Lubar. Eeg coherence effects of audio-visual stimulation (avs) at dominant and twice dominant alpha frequency. *Journal of neurotherapy*, 8(4):25–42, 2005.

Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML'2015*, 2015.

X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *JMLR W&CP: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2011)*, Apr. 2011.

I. J. Goodfellow. On distinguishability criteria for estimating generative models. In *International Conference on Learning Representations, Workshops Track*, 2014.

I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML'13)*, pages 1319–1327. ACM, 2013a. URL `http://icml.cc/2013/`.

I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *ICML'2013*, 2013b.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. In *NIPS'2014*, 2014a.

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2014b. URL `http://arxiv.org/abs/1412.6572`.

S. Gu and L. Rigazio. Towards deep neural network architectures robust to adversarial examples. In *NIPS Workshop on Deep Learning and Representation Learning*, 2014.

M. Gutmann and A. Hyvarinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS'2010*, 2010.

R. Huang, B. Xu, D. Schuurmans, and C. Szepesvári. Learning with a strong adversary. *CoRR*, abs/1511.03034, 2015. URL `http://arxiv.org/abs/1511.03034`.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015.

K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *Proc. International Conference on Computer Vision (ICCV'09)*, pages 2146–2153. IEEE, 2009.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *Proceedings of* *the International Conference on Learning Representations (ICLR)*, 2014.

A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25 (NIPS'2012)*. 2012.

A. B. L. Larsen, S. K. Sønderby, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. *CoRR*, abs/1512.09300, 2015. URL `http://arxiv.org/abs/1512.09300`.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov. 1998.

W. Lotter, G. Kreiman, and D. Cox. Unsupervised learning of visual structure using predictive generative networks. *arXiv preprint arXiv:1511.06380*, 2015.

A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech, and Language Processing*, 2013.

A. Makhzani, J. Shlens, N. Jaitly, and I. J. Goodfellow. Adversarial autoencoders. *CoRR*, abs/1511.05644, 2015. URL `http://arxiv.org/abs/1511.05644`.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations: Workshops Track*, 2013.

M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

T. Miyato, S. Maeda, M. Koyama, K. Nakae, and S. Ishii. Distributional smoothing with virtual adversarial training. In *ICLR*, 2015. Preprint: arXiv:1507.00677.

A. Nguyen, J. Yosinski, and J. Clune. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. *ArXiv e-prints*, Dec. 2014.

N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. *arXiv preprint arXiv:1511.04508*, 2015.

N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint arXiv:1602.02697*, 2016a.

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*. IEEE, 2016b.

A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML'2014*, 2014.

J. O. Robinson. *The psychology of visual illusion*. Courier Corporation, 2013.

S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet. Adversarial manipulation of deep representations. *CoRR*, abs/1511.05122, 2015. URL `http://arxiv.org/abs/1511.05122`.

J. T. Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. In *International Conference on Learning Representations*, 2016.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

I. Sutskever, R. Józefowicz, K. Gregor, D. J. Rezende, T. Lillicrap, and O. Vinyals. Towards principled unsupervised learning. *CoRR*, abs/1511.06440, 2015. URL `http://arxiv.org/abs/1511.06440`.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. Technical report, arXiv:1409.4842, 2014a.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *ICLR*, abs/1312.6199, 2014b. URL `http://arxiv.org/abs/1312.6199`.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision. *ArXiv e-prints*, Dec. 2015.

L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. arXiv:1511.01844, Nov 2015. URL `http://arxiv.org/abs/1511.01844`.

R. J. Williams. Simple statistical gradient-following algorithms connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.