UNIVERSIDADE ESTADUAL DE CAMPINAS

Faculdade de Engenharia Elétrica e de Computação

Pedro Tabacof

Exploring Adversarial Images in Deep Neural Networks

Explorando Imagens Adversárias em Redes Neurais Profundas

Campinas

2017

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Pedro Tabacof

Exploring Adversarial Images in Deep Neural Networks

Explorando Imagens Adversárias em Redes Neurais Profundas

Master's dissertation presented to the Graduate Program of the School of Electrical and Computer Engineering of the University of Campinas to obtain a Master's degree in Electrical Engineering, in the area of concentration of Computer Engineering.

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na área de concentração de Engenharia de Computação.

Supervisor: Prof. Dr. Eduardo Alves do Valle Junior

Este exemplar corresponde à versão final da tese defendida pelo aluno Pedro Tabacof, e orientada pelo Prof. Dr. Eduardo Alves do Valle Junior

Campinas

2017

Agência(s) de fomento e nº(s) de processo(s): Não se aplica.

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Luciana Pietrosanto Milla - CRB 8/8129

T111e

Tabacof, Pedro, 1990-
Exploring adversarial images in deep neural networks / Pedro Tabacof. – Campinas, SP : [s.n.], 2017.

Orientador: Eduardo Alves do Valle Junior.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Redes neurais (Computação). 2. Modelos lineares (Estatística). 3. Processamento de imagens - Métodos estatísticos. 4. Variáveis latentes. I. Valle Junior, Eduardo Alves do. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Explorando imagens adversárias em redes neurais profundas
Palavras-chave em inglês:
Neural networks (Computing)
Linear models (Statistics)
Image processing - Statistical methods
Latent variables
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Eduardo Alves do Valle Junior [Orientador]
Roseli Aparecida Francelin Romero
Fernando José Von Zuben
Data de defesa: 23-06-2017
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidato: Pedro Tabacof RA: 082493
Data da Defesa: 23 de junho de 2017

Título da Tese: Exploring Adversarial Images in Deep Neural Networks (Explorando Imagens Adversárias em Redes Neurais Profundas).

Prof. Dr. Eduardo Alves do Valle Junior (Presidente, FEEC/UNICAMP)
Profa. Dra. Roseli Aparecida Francelin Romero (ICMC/USP)
Prof. Dr. Fernando José Von Zuben (FEEC/UNICAMP)

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no processo de vida acadêmica do aluno.

Acknowledgements

First and foremost, I thank my supervisor, Eduardo Valle, for his mentorship and guidance of my work as a graduate student, and for all his help with the experimental design, presentation and editing of this dissertation.

My co-authors Julia Tavares and Ramon Oliveira contributed with me to two papers that were presented at different NIPS 2016 workshops, an achievement we all should be proud of. I especially thank Julia with respect to the research, development and writing of the article that lead to Chapter 4, that is now an integral part of this dissertation. I thank my former colleague Micael Carvalho for his helpful hints and revision of my first published article, which is part of this dissertation as Chapter 3. I also thank the professors Matthieu Cord and Fernando Von Zuben, who were part of my qualifying committee and provided many suggestions and corrections, now incorporated to this dissertation.

I thank all my colleagues from the Titans-A research group and the Bayesian study group for their patience to listen to my ideas, and their feedback, criticism, and suggestions, all of which were essential to my journey here and to the writing of this work. Among them, I especially thank George Gondim Ribeiro, who kindly helped review this dissertation. My laboratory, RECOD, provided me not only the computational resources necessary for the research and experimental design of this dissertation, but also a great research environment, where many interesting collaborations, opportunities, and discussions have taken place.

I thank my parents, Maria Eunice Santos and Germano Tabacof, for their emotional and financial support. They provided me with most of what I have today, for which I am deeply grateful. They shaped my life in many positive ways, and if they had not stilled in me intellectual curiosity and desire for academic achievement, I would not have pursued graduate studies. I also thank my partner Liara Aranha for her love and understanding. Living with me the past couple of years must not have been easy, and only her patience and care made this dissertation possible.

I worked for a startup company, I.Systems, during the majority of my Master studies, and I thank them for their generosity to allow me to pursue my graduate studies, granting me time, financial support, research experience, and learning opportunities. I especially thank the co-founders, Ronaldo Silva, Danilo Halla, and Igor Santiago. Now I have co-founded Datart with my partners Ramon Oliveira and Afonso Menegola, and I thank them for inviting me to be part of this enterprise of which I am really proud. I am continuing my graduate studies under the sponsorship of Motorola Mobility, that was generous enough to grant me a PhD scholarship, of which I am grateful.

I thank the Brazilian agencies CAPES, CNPq and FAPESP for overall financial support of my research group and laboratory. I thank IEEE-CIS and FAEPEX-Unicamp for their support of my trip to the IJCNN 2016 conference, where I presented my first published work.

"We are robust when errors in the representation of the unknown and understanding of random effects do not lead to adverse outcomes — fragile otherwise."

(Nassim Nicholas Taleb)

Resumo

Exemplos adversários levantaram questões da robustez e segurança de redes neurais profundas. Neste trabalho nós formalizamos o problema de imagens adversárias dado um classificador pré-treinado, mostrando que mesmo em modelos lineares a otimização resultante é não-convexa. Nós geramos imagens adversárias utilizando classificadores rasos e profundos nas bases de dados de imagens MNIST e ImageNet. Nós sondamos o espaço dos pixels das imagens adversárias com ruído de intensidade e distribuição variável. Nós trazemos visualizações novas que mostram o fenômeno e sua alta variabilidade. Nós mostramos que imagens adversárias aparecem em regiões grandes no espaço de pixels, entretanto, para a mesma tarefa, um classificador raso parece mais robusto a imagens adversárias que uma rede convolucional profunda.

Nós também propomos um novo ataque adversário a autoencoders variacionais. Nosso procedimento distorce uma imagem de entrada com o objetivo de confundir um autoencoder a reconstruir uma imagem alvo completamente diferente. Nós atacamos a representação interna e latente, com o objetivo de que a entrada adversária produza uma representação interna o mais similar possível da representação de uma imagem alvo. Nós verificamos que autoencoders são mais robustos ao ataque que classificadores: Apesar de alguns exemplos possuírem pequena distorção na entrada e similaridade razoável com a imagem alvo, há um compromisso quase linear entre esses objetivos. Nós demonstramos os resultados nas bases de dados MNIST e SVHN, e também testamos autoencoders determinísticos, chegando a conclusões similares em todos os casos. Finalmente, nós mostramos que o ataque adversário típico em classificadores, apesar de ser mais fácil, também apresenta uma relação proporcional entre a distorção da entrada e o erro da saída. No entanto, essa proporcionalidade é escondida pela normalização da saída, que mapeia uma camada linear em uma distribuição de probabilidades.

Palavras-chaves: Redes Neurais; Aprendizado Profundo; Imagens Adversárias; Autoencoders Variacionais.

Abstract

Adversarial examples have raised questions regarding the robustness and security of deep neural networks. In this work we formalize the problem of adversarial images given a pre-trained classifier, showing that even in the linear case the resulting optimization problem is nonconvex. We generate adversarial images using shallow and deep classifiers on the MNIST and ImageNet datasets. We probe the pixel space of adversarial images using noise of varying intensity and distribution. We bring novel visualizations that showcase the phenomenon and its high variability. We show that adversarial images appear in large regions in the pixel space, but that, for the same task, a shallow classifier seems more robust to adversarial images than a deep convolutional network.

We also propose a novel adversarial attack for variational autoencoders. Our procedure distorts the input image to mislead the autoencoder in reconstructing a completely different target image. We attack the internal latent representations, attempting to make the adversarial input produce an internal representation as similar as possible as the target's. We find that autoencoders are much more robust to the attack than classifiers: while some examples have tolerably small input distortion, and reasonable similarity to the target image, there is a quasi-linear trade-off between those aims. We report results on MNIST and SVHN datasets, and also test regular deterministic autoencoders, reaching similar conclusions in all cases. Finally, we show that the usual adversarial attack for classifiers, while being much easier, also presents a direct proportion between distortion on the input, and misdirection on the output. That proportionality however is hidden by the normalization of the output, which maps a linear layer into a probability distribution.

Keywords: Neural Networks; Deep Learning; Adversarial Images; Variational Autoencoders.

List of Figures

List of Tables

List of Acronyms and Abbreviations

AI Artificial Intelligence. 21

CG Conjugate Gradient. 31

ELU Exponential Linear Unit. 58

GAN Generative Adversarial Networks. 65

GPU Graphical Processing Unit. 16

IJCNN International Joint Conference on Neural Networks. 19

KL Kullback-Leibler. 36

LSTM Long Short-Term Memory Network. 31

MAP Maximum a Posteriori. 25

MLP Multilayer Perceptron. 27

MSE Mean Squared Error. 23

NIPS Neural Information Processing Systems. 19

PCA Principal Component Analysis. 22

ReLU Rectified Linear Unit. 30

SGD Stochastic Gradient Descent. 31

SVHN Street-view House Numbers dataset. 56

Contents

1 Introduction

After a long dominance of ensemble and kernel methods, neural networks regained — thanks to many empirical successes — primary importance in machine learning. In computer vision, convolutional networks have won the ILSVRC ImageNet competition in classification and localization since 2012 (RUSSAKOVSKY *et al.*, 2015); in natural language processing, recurrent neural networks can translate from English to French (SUTSKEVER *et al.*, 2014); in speech recognition, a recurrent neural network is state-of-the-art on the TIMIT phoneme recognition benchmark (GRAVES *et al.*, 2013); in reinforcement learning, two convolutional networks beat one of the best Go players of all time (SILVER *et al.*, 2016); in signal processing, an autoencoder is able to compress images better than JPEG or JPEG2000 (GREGOR *et al.*, 2016).

The recent successes of neural networks are tied to the development of Deep Learning (LECUN *et al.*, 2015). A deep neural network has many layers, but — otherwise — there is no formal difference between shallow and deep models. The successes are due to better architectures, optimization, initialization and regularization techniques, which exploit Graphical Processing Unit (GPU) parallelism and big data. More than a major theoretical breakthrough, a myriad of incremental improvements explain the performance of deep neural networks.

Since no unifying theory can explain the success of deep networks, for now we are forced to probe the learned representations to understand why and how the networks work and fail. Even if there were such theory, as Science is an empirical endeavor, it would have to eventually face the scrutiny of data. Empirical exploration of the networks — in the form of experiments or visualizations — not only improves our intuitive understanding of the models, but also helps to elicit, validate, and falsify theories. In this work, we advance that empirical avenue.

We explore different faces of an intriguing property of deep neural networks: adversarial images (SZEGEDY *et al.*, 2014). Adversarial images are one of the most glaring flaws of deep networks, and, as such, have attracted a lot of attention. Adversarial images may be thought as "optical illusions for artificial vision" — and in the same way that illusions provide a way to understand the cognitive processes of human brains (KOCH, 2006), adversarial images give us insights about the decision processes of deep neural networks.

Adversarial images present us fascinating questions and possibilities:

- It is possible to improve resistance to attacks, usually with other benefits, such as regularization (GOODFELLOW *et al.*, 2015; KENDALL; GAL, 2017).

- They need not know the inner workings of the network for the attack to be successful (PAPERNOT *et al.*, 2017; KURAKIN *et al.*, 2017).
- The attack can be transferred across different networks (SZEGEDY *et al.*, 2014; TRAMÈR *et al.*, 2017).
- Many strategies have been tried to completely stop adversarial attacks, but none has yet succeeded (GU; RIGAZIO, 2015; BRENDEL; BETHGE, 2017; CARLINI; WAGNER, 2017).

We also extend adversarial images to variational autoencoders. Variational autoencoders are unsupervised networks that learn probabilistic representations of data (KINGMA; WELLING, 2014). An autoencoder reduces input dimensionality by transforming it to a latent variable space, where the manifold of the input distribution is supposed to lie. Thus, we can experiment on variational autoencoder latent variables to understand the structure of deep representations. Here, however, we attack those latent variables via input distortion in order to fool the representation. We propose a novel method for generating adversarial images on variational autoencoders based on this latent variable attack.

1.1 Motivation

The debate over the dangers of overconfident machine learning has reached the headlines of mass media (VAUGHAN; WALLACH, 2016; CRAWFORD, 2016). Indeed, if our models are to drive cars, diagnose medical conditions, and even analyze the risk of criminal recidivism, unreliable predictions may have dire consequences. Adversarial examples provide one way to lead astray the predictions of most classifiers. Thus, we must understand how the attack works and how to prevent it. We focus on the former, specifically on the question of how large the space of adversarial images is, in order to better understand how dangerous the adversarial problem is for deep neural networks.

Our motivation to extend the concept of adversarial images to variational autoencoders is threefold: first, we want to understand how the latent representation of the autoencoders capture semantic knowledge, and attacking the representation is one possible way to understand their dynamic; second, we want to evidence the dangers of a possible adversarial attack, in order to inspire other researchers to look for protections and safeguards, similar to the line of research spun by the original adversarial images work; third, we want to warn users of autoencoders of the dangers of possible malicious attacks.

1.2 Objectives

The primary objective of this work is the empirical investigation of the space of adversarial images. The secondary objective is the extension of adversarial attacks to variational autoencoders. To achieve those goals, we aim at answering the following questions:

- How are adversarial images distributed in the input space?
- What can be said about the manifold of adversarial images?
- How does the nature of the probing change our conclusions?
- Can adversarial images be found in autoencoders?
- What is the relation between adversarial images for variational autoencoders and for classifiers?

1.3 Contributions

In this work we present the following contributions:

1. We probe the pixel space of adversarial images using noise of varying intensity and distribution. We bring novel visualizations that showcase the phenomenon and its variability.
2. We show that adversarial images appear in large regions in the pixel space, but that, for the same task, a shallow classifier seems more robust to adversarial images than a deep convolutional network.
3. We formalize the problem of adversarial image generation in order to show that it is in general a nonconvex optimization procedure, even for linear models.
4. We propose an adversarial attack on variational — and, for comparison, deterministic — autoencoders. Our attack aims not only at disturbing the reconstruction, but at fooling the autoencoder into reconstructing a completely different target image.
5. We show that while this attack may be successful given an appropriate regularization parameter, there is a quasi-linear trade-off between input noise and target reconstruction error.

6. We make a comparison between attacks for autoencoders and for classifiers, showing that while the former is much harder, in both cases the amount of distortion on the input is proportional to the amount of misdirection on the output. For classifiers, however, such proportionality is hidden by the normalization of the output, which maps a linear layer into non-linear probabilities.

1.4 Achievements

During these Master studies, the author has participated in four scholarly publications:

- First author of Tabacof & Valle (2016), published at the 2016 International Joint Conference on Neural Networks (IJCNN). The author received a Travel Grant from IEEE-CIS to attend the conference. The article was the subject of two media pieces Winiger (2015), James (2015).
- First author of Tabacof *et al.* (2016), presented at the Workshop on Adversarial Training at the 2016 Neural Information Processing Systems (NIPS) conference. The paper was selected as one of the spotlight presentations.
- Second author (with equal contribution to the first) of Oliveira *et al.* (2016), presented at the Workshop on Bayesian Deep Learning at NIPS 2016.
- Second author of Godoy *et al.* (2017), published at the journal PLOS ONE.

The author was the graduate teaching assistant of the Data Structures course, where he developed and taught all the practical programming laboratories. The author attended the Machine Learning Summer School, University of Texas at Austin, in 2015, and the selective Deep Learning Summer School, Université de Montréal, in 2016.

The author applied for no government scholarships during his Master studies, having opted instead to work as an engineer at a startup company (I.Systems, Campinas) during most of his studies. Later, he founded his own startup company (Datart, Campinas), as the Chief Research Officer. In 2017, he has started the Electrical Engineering Ph.D. program at FEEC, UNICAMP, funded by a scholarship from Motorola Mobility Brazil.

1.5 Outline

The remainder of the text is organized as follows:

Literature review: We review the basics of machine learning, where we present the basic building blocks necessary to understand deep learning. Then, we briefly overview the history of neural networks. Finally, we do a more thorough review of deep learning, focusing on variational autoencoders and adversarial images.

Exploring the space of adversarial images: We show how different probings of the space of adversarial images enables our understanding of the prevalence of adversarial images. Based on the work that has lead to the publication of Tabacof & Valle (2016).

Adversarial images for variational autoencoder: We propose an adversarial attack for variational autoencoders, where we slightly distort the input image in order to make the autoencoder reconstruct a completely different image. Based on the work that has lead to the publication of Tabacof *et al.* (2016).

Conclusion: We conclude this dissertation and point to possible future work.

2 Literature Review

The focus of this dissertation, adversarial images and variational autoencoders, arised in the context of *deep learning*. Deep learning, however, is a continuation of decades-old artificial neural networks. While there is a rich literature of neural networks from neurobiology, computational neuroscience, and cognitive science, we work from the point of view of statistical learning, as deep learning itself is mostly studied within this context. We motivate that point of view on section 2.1, where we show the connections between statistical learning and neural networks.

We begin this chapter with the basics of machine learning. We overview supervised and unsupervised learning on section 2.1, then we move to artificial neural networks on section 2.2, ending with the recent developments of deep learning on 2.3. Our brief reminder of machine learning fundamentals is not aimed at bringing readers with no background in the area up to speed; for that purpose we suggest Bishop (2007).

The literature upon which we build our contributions on chapters 3 and 4 is, respectively, on section 2.4 (adversarial images), and on section 2.5 (variational autoencoders). The reader already familiar with deep learning might want to skip directly to those sections.

2.1 Machine Learning

Machine learning is a subfield of Artificial Intelligence (AI). Defining AI is not straightforward because defining intelligence objectively is an open problem (LEGG, 2008). Is Intelligence the predictive capabilities of an animal (HAWKINS; BLAKESLEE, 2007), or is it the communication abilities of a human (TURING, 1950)? Here we will assume a broad view that includes goal-oriented behavior involving planning, reasoning, and learning in an uncertain world.

We are interested in the learning aspect of AI, which is known as machine learning. Machine learning uses data to train models to perform specific tasks. The models can be parametric, where the number of parameters is defined by an architect (which could be human or an AI), and non-parametric, where the number of parameters is undefined, and may increase with data. In this work we only deal with parametric models. Machine learning tasks are usually classified as three major types:

Supervised learning: A model learns a mapping between inputs and associated outputs that will generalize to unseen input-output pairs. The inputs are usually called features. Supervised learning can be further divided into two problems:

- *Regression*: The output is continuous (a “number”).
- *Classification*: The output is discrete (a “label”).

Unsupervised learning: A model learns the distribution, geometry, or interesting properties of unlabeled data.

Reinforcement learning: A model learns actions that maximize expected future rewards.

Here, we will focus on the first two types, as they are directly related to our subject: adversarial images were created to target supervised classifiers; variational autoencoders are unsupervised learning models.

As mentioned, throughout this brief review of machine learning we will adopt the point of view of statistical learning, since it has a deep interplay with neural networks. As we shall see throughout this chapter, the connections between statistical learning, neural networks (and deep learning) are often direct:

1. The logistic model can be interpreted as a single neuron with sigmoidal activation.
2. The logistic / softmax model is usually the last layer of a neural network for classification.
3. Learning a neural network for regression usually implies Gaussian likelihood.
4. Learning a neural network for classification usually implies Bernoulli or Categorical likelihood.
5. Weight decay implies Gaussian prior on the neural network weights.
6. Training a neural network usually consists in doing maximum likelihood or maximum a posteriori, with extra regularization techniques.
7. Many of the regularization techniques used to train neural networks have a Bayesian interpretation.
8. Principal Component Analysis (PCA) can be interpreted as a linear autoencoder.
9. Variational autoencoders learn by maximizing a lower bound to the marginal maximum likelihood.
10. Learning in variational autoencoders can be interpreted as minimizing the KL-divergence between the approximation and the true model.

Due to our exploration of variational autoencoders (chapter 4), we are particularly interested in the Bayesian viewpoint. Adding a prior on the unobserved variables (as we will do shortly to find the maximum a posteriori solution of a linear-Gaussian model), is not enough to characterize a model as Bayesian. True Bayesian models perform not only a point estimate (or prediction), but provide the entire probability distribution over possible estimates (predictions). Such extra information is important to measure uncertainty, and to make principled decisions that weight cost and risk (the subject of decision theory).

Thus, learning in the Bayesian sense consists in finding the posterior of all unobserved variables, given the prior, the likelihood, and the observed data. A Bayesian estimate would use the expected posterior. A Bayesian prediction involves marginalizing out the posteriors, which will lead to a probabilistic output that hopefully captures the prediction uncertainty better than point estimates.

2.1.1 Supervised Learning

The problem of supervised learning is to find a mapping from a set of input variables to an output variable[1]. Input and output variables may have time dependencies, but we will focus on the static domain. The main challenge is to generalize the mapping to unseen input-output pairs (otherwise a simple look-up table would suffice for a perfect mapping).

Fitting well observed data points must be balanced against generalizing well to unseen points. We can see such balance through different perspectives, but here we will use the bias-variance trade-off. Bias comes from the models being unable to reflect the true nature of the data; it arises when the model is too rigid to adapt to the data (or when assumptions about the model are plain wrong). Variance comes from adapting to the noise rather than the signal; that will happen even if there is no explicit noise term in the data, because we can only observe a sample of the data, and the sampling itself induces such noise.

When analyzing model fitness as the Mean Squared Error (MSE) between the true (unknown) and the learned model, we can interpret it as the squared bias plus the variance: $MSE \propto Bias^2 + Var$. That decomposition implies that unbiased models are not necessarily preferred over biased ones (with smaller variance), because their expected error can actually be greater.

We are dealing with the estimation of function approximators. That is different than estimating parameters of a model. For example, ordinary least-squares gives an

[1] That could be generalized to multiple output variables, where the learning can employ multi-objective optimization.

unbiased estimate of the coefficients of a linear model[2], but, as a function approximation, it will be biased unless the true data generating process is also linear. That point of view comes from machine learning, where the end goal is the prediction provided by the model, and not the parameters themselves.

To give further examples: a 1-nearest neighbors classifier has small bias but high variance (FRIEDMAN *et al.*, 2001); a regularized linear model will usually have high bias but lower variance. Neural networks are in-between, and as we shall see later on, many strategies are used to either reduce bias (e.g. increase network depth) or reduce variance (e.g. early stopping).

The prototypical supervised learning model is the linear regression, where the mapping between input and output is by an affine transformation:

$$y = \boldsymbol{w} \cdot \boldsymbol{x} + b \tag{2.1}$$

The parameters ($\boldsymbol{w}$ and b) can be learned different ways, the standard algorithm being ordinary least squares, which depends on the observed inputs ($\hat{\boldsymbol{X}}$) and outputs ($\hat{\boldsymbol{y}}$):

$$\begin{aligned} \underset{\boldsymbol{w},b}{\text{minimize}} \quad & \left\| \boldsymbol{y} - \hat{\boldsymbol{y}} \right\|_2^2 \\ \text{subject to} \quad & \boldsymbol{y} = \hat{\boldsymbol{X}}\boldsymbol{w} + b \end{aligned} \tag{2.2}$$

That is a strict convex procedure, i.e., there is only one local minimum, which is the global minimum. That particular optimization can be solved by calculating a pseudo-inverse, but, in general, convex problems can be solved in polynomial time by iterative procedures (BOYD; VANDENBERGHE, 2004).

Linear least squares is actually equivalent to finding the maximum likelihood parameters of the following linear-Gaussian model:

$$\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{w} \cdot \boldsymbol{x} + b, \sigma^2) \tag{2.3}$$

Maximum likelihood finds the parameters that best explain the observed data and, for exactly that reason, are susceptible to overfitting, when the noise is learned instead of the signal (i.e. low bias and high variance). That can be exemplified through a polynomial regression[3] of varying order (Figure 1).

A more principled learning procedure puts a prior on the unknown parameters, which reduces the degrees of freedom of the problem and prevents overfitting (that is,

[2] As shown by the Gauss-Markov theorem, assuming the errors are uncorrelated, homoscedastic, and with expected value of zero.

[3] A polynomial regression is linear in the parameters: $y = a + bx + cx^2 + dx^3 \ldots$

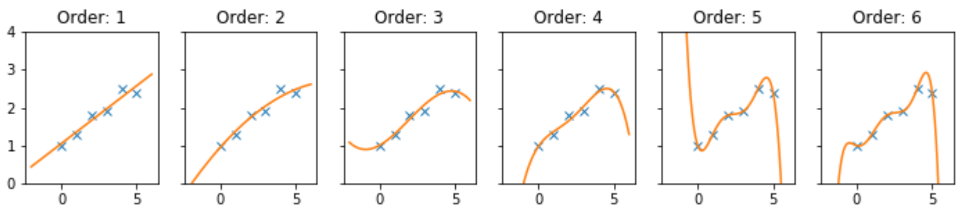


Figure 1 – Polynomial regression of varying order. The data was generated by hand.

increases the bias but greatly decreases the variance):

$$\begin{aligned} \boldsymbol{y} &\sim \mathcal{N}(\boldsymbol{w} \cdot \boldsymbol{x} + b, \sigma^2) \\ \boldsymbol{w} &\sim \mathcal{N}(0, \sigma_w^2) \end{aligned} \tag{2.4}$$

That is known as ridge regression or Tikhonov regularization. From a statistical perspective, this is called Maximum a Posteriori (MAP) estimation, which simply is maximum likelihood with priors on the estimated parameters. The intuition is that by shrinking the parameters value towards zero, we are biasing our solution towards a more likely configuration. That also stabilizes the optimization. Regularizing the linear model increases the bias but often greatly decreases the variance, thus reducing total error. The same polynomial regression with Tikhonov regularization shows more sensible results (Figure 2).

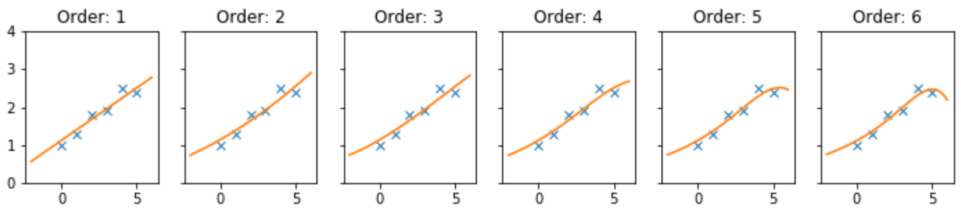


Figure 2 – Polynomial ridge regression of varying order. The data was generated by hand.

One open question is how to determine the regularization strength, which in our statistical model is represented by the parameter σ_w^2. A Bayesian perspective would suggest putting an hierarchical prior over σ_w^2, shared by all coefficients. A frequentist perspective would favor cross-validation to find the σ_w^2 that minimizes validation error. Such problem is known as hyperparameter optimization and is very important for neural networks, where the frequentist perspective usually prevails.

If we use a Laplace (instead of Gaussian) prior on the parameters of the linear model, the maximum a posteriori procedure would be equivalent to the Lasso algorithm (PARK; CASELLA, 2008), which favors sparse solutions (i.e., individual parameters values not only shrink towards zero, but can become exactly zero)(TIBSHIRANI, 1996). That is specially important if there are more features than data points, or when performing feature selection.

The extension of linear models to binary classification involves a change in the likelihood function, from Gaussian to Bernoulli[4]:

$$y \sim \mathit{Bernoulli}(\sigma(\boldsymbol{w} \cdot \boldsymbol{x} + b)) \tag{2.5}$$

The linear model is transformed via the inverse logit link function[5] from the real space to a valid probability range $(0 - 1)$. Other link functions that map the real line into the probability space interval can be used, such as the probit. This model is known as logistic regression, and it can also be regularized by a Gaussian or Laplace prior. The multiclass version uses the softmax link function and thus is known as softmax regression (JORDAN *et al.*, 1995).

2.1.2 Unsupervised Learning

We can learn the distribution or geometry of data in many ways, such as clustering, density estimation, or dimensionality reduction. Here we focus on the latter, as it is closely related to the variational autoencoders we will explore later.

Assume we have unobserved factors that cause the observed data. We will call those unobserved factors latent variables. The simplest transformation from the latent variables to the observations is a linear-Gaussian model (represented as a graphical model in Figure 3):

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu} + \epsilon \\ \boldsymbol{z} &\sim \mathcal{N}(0, I) \\ \epsilon &\sim \mathcal{N}(0, \sigma^2 I) \end{aligned} \tag{2.6}$$

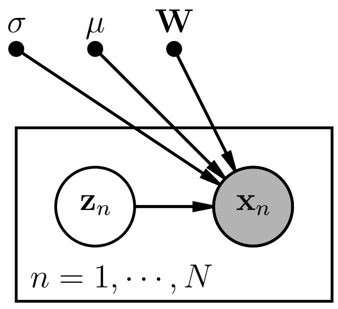


Figure 3 – Probabilistic PCA.

To find the linear transformation matrix $\boldsymbol{W}$ we can marginalize the latent variables and optimize the resulting likelihood, a procedure called maximum marginal likelihood.

[4] Multiclass or multinomial classification involves the Categorical distribution instead of the Bernoulli.

[5] $\sigma(x) = \dfrac{1}{1 + \exp(-x)}$

That procedure is known as the probabilistic Principal Component Analysis (PCA) (TIPPING; BISHOP, 1999), which we can view as an autoencoder with linear-Gaussian decoder and encoder. Autoencoders will be explained in further detail in section 2.5.

The decoder is the explicit generative model above $p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu}, \sigma^2 I)$, while the encoder is the posterior of the latent variables $p(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}((\boldsymbol{W}^T + \boldsymbol{W} + \sigma^2 I)^{-1}\boldsymbol{W}^T(\boldsymbol{x} - \boldsymbol{\mu}), \sigma^2(\boldsymbol{W}^T + \boldsymbol{W} + \sigma^2 I)^{-1})$ (TIPPING; BISHOP, 1999). With such probabilistic model, we can use the observed data to return the latent variables, which we can use to reduce the dimensionality of the data, or to reconstruct the data from the latent variables — tasks related to compression.

2.2 Artificial Neural Networks

In 1943, McCulloch and Pitts proposed a simple thresholding neuron, and proved that a network of such neurons could implement any Boolean function (MCCULLOCH; PITTS, 1943). In the late 1950s, Rosenblatt proposed the perceptron, which included adaptive weights (ROSENBLATT, 1958). Those neurons apply a nonlinear function to the output of what is basically a linear model — a weighed combination of input values plus a bias term:

$$f(\boldsymbol{x}) = \phi(\boldsymbol{w} \cdot \boldsymbol{x} + b) \tag{2.7}$$

The nonlinear function $\phi(x)$ is called, in this context, the activation function, $\boldsymbol{w}$ is the adaptive weight vector, b is the bias term, $\boldsymbol{x}$ is the input vector, and $f(\boldsymbol{x})$ is the scalar output. A layer of neurons is the parallel application of those functions:

$$f(\boldsymbol{x}) = \phi(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) \tag{2.8}$$

Now the output $f(\boldsymbol{x})$ is also a vector, $\boldsymbol{W}$ is the weight matrix and $\boldsymbol{b}$ is the bias vector. The activation function ϕ is applied element-wise. A feedforward neural network is a nonlinear mapping between input and output consisting of successive applications of layers of neurons. The prototypical neural network is the Multilayer Perceptron (MLP) with one hidden layer:

$$\boldsymbol{y} = f_1(f_2(\boldsymbol{x})) \tag{2.9}$$

Where f_2 is the hidden layer and f_1 is the output layer. Typically, the hidden layer uses a hyperbolic tangent (tanh) nonlinearity and the output layer is linear. Neural networks are usually trained with maximum likelihood or maximum a posteriori optimization.

As in the linear models described in section 2.1, the typical choice of likelihood is Gaussian for regression problems and Bernoulli for categorical problems. Maximum likelihood with Gaussian likelihood leads to the mean squared error loss function minimization:

$$\mathcal{L}_\theta(\boldsymbol{x}, \hat{y}) = \frac{1}{N}\sum_{i=0}^{N}(f_\theta(\boldsymbol{x}_i) - \hat{y}_i)^2 \tag{2.10}$$

Where $\boldsymbol{x}_i$ and $\hat{y}_i$ are respectively the input and output observations. θ is the concatenation of all free parameters of all layers (e.g. the bias vector $\boldsymbol{b}_1$ of the first layer, the weight matrix $\boldsymbol{W}_2$ of the second layer, etc). Maximum likelihood with Bernoulli likelihood leads to the cross-entropy loss function minimization:

$$\mathcal{L}_\theta(\boldsymbol{x}, \hat{y}) = \frac{1}{N}\sum_{i=0}^{N} -(\hat{y}_i \cdot \log(\sigma(f_\theta(\boldsymbol{x}_i))) + (1 - \hat{y}_i) \cdot \log(1 - \sigma(f_\theta(\boldsymbol{x}_i)))) \tag{2.11}$$

The two classical ways to regularize neural networks are via weight decay or early stopping. Weight decay adds a term to the cost proportional to the Euclidean norm of the weights, and corresponds, from a Bayesian viewpoint to adding a Gaussian prior to the weights:

$$\mathcal{L}_\theta(\boldsymbol{x}, \hat{y}) = \frac{1}{N}\sum_{i=0}^{N}(f_\theta(\boldsymbol{x}_i) - \hat{y}_i)^2 + \lambda\left\|\theta\right\|_2^2 \tag{2.12}$$

Early stopping, as the name suggests, stops the optimization procedure (be it maximum likelihood or MAP) before a local minima has been reached. A metric (e.g. accuracy) computed over a validation set is used to decide when to stop, as further training will lead to overfitting.

Finding the optimal set of weights for a nonlinear model is generally a hard optimization problem — a nonconvex procedure with no guarantees of finding the global optimum. However, a local solution can be found using the assumption of local convexity, applying first- and second-order optimizations. Those techniques require the gradient of the loss function with respect to the weights. The backpropagation algorithm (rediscovered many times (SCHMIDHUBER, 2015)) achieves that by exploiting the chain rule, and the associativity of matrix multiplication. We can also seen backpropagation as a special case of automatic differentiation (the reverse accumulation case) (BASTIEN *et al.*, 2012).

The basic idea behind backpropagation is differentiating the loss function with respect to the weights, a direct application of the chain rule on the loss function:

$$\nabla\mathcal{L}_\theta(\boldsymbol{x}, \hat{y}) = \frac{\partial\mathcal{L}(f_\theta(\boldsymbol{x}), \hat{y})}{\partial f_\theta(\boldsymbol{x})}\frac{df_\theta(\boldsymbol{x})}{d\theta} \tag{2.13}$$

The first term, the derivative of the loss function with respect to its inputs, is usually simple, unless unusual loss functions are used. The second term, the gradient of the neural network output with respect to its weights and biases (encapsulated by θ), is the source of most practical optimization and numerical issues. On the next section, we show the general formula for the second term, as it applied to both shallow and deep networks, but the problems are exarcerbated in the latter.

2.3 Deep Learning

Deep learning is a rebranding of neural network models with more than one hidden layer. The difference between a deep and shallow network is subjective — it is hard to pinpoint where shallow ends and deep begins — but any model with more than 10 hidden layers is definitely considered deep learning (SCHMIDHUBER, 2015).

Deep models leverage massive datasets through stochastic training of mini-batches of data within GPUs. The rise of massive datasets, such ILSVRC2016 containing over one million images (DENG *et al.*, 2009), allowed the training of ever increasing neural networks, reaching hundreds of millions of neurons within a single deep network (SIMONYAN; ZISSERMAN, 2015).

While research on specific hardware for neural networks has existed since the 1990s, they have never enjoyed the popularity of graphical-processing units (GPUs). Originally intended for video rendering and gaming, GPUs soon brought massive parallelism to the masses, with what became known as general-purpose computing on GPU (GPGPU). GPGPU sped up the training of neural networks by a factor of 50 or more, allowing the use of massive datasets to train deep neural networks (SCHMIDHUBER, 2015).

Deep neural networks with more than one hidden layer can be seen as a complex nonlinear function composed by the successive application of individual layers:

$$\boldsymbol{y} = (f_N \circ f_{N-1} \circ \ldots \circ f_1)(\boldsymbol{x}) \tag{2.14}$$

The strong non-convexity of such models makes the training of deep neural networks difficult. For a long time, escaping bad local minima was considered the biggest challenge in training neural networks (LECUN *et al.*, 2012), but recent research suggests that all local minima are likely to have similar values, close to the global minimum (KAWAGUCHI, 2016). Instead, deep models are plagued by saddle points (CHOROMANSKA *et al.*, 2015), which slow the optimization down, and mislead it towards poor solutions (DAUPHIN *et al.*, 2014).

Many practical problems of training deep models are due to vanishing or exploding gradients. As explained in the previous section, the gradient is defined using backpropagation, which is an application of the chain rule in an efficient manner. The gradient of a deep neural network output with respect to its weights is the following expression:

$$\begin{aligned} \frac{d}{d\theta} f_N \circ \ldots \circ f_1(\boldsymbol{x}) &= \frac{\partial f_N}{\partial \theta_N}\frac{d\theta_N}{d\theta} + \frac{\partial f_N}{\partial f_{N-1}}\frac{df_{N-1}}{d\theta} \\ &= \frac{\partial f_N}{\partial \theta_N}\frac{d\theta_N}{d\theta} + \frac{\partial f_N}{\partial f_{N-1}}\frac{\partial f_{N-1}}{\partial \theta_{N-1}}\frac{d\theta_{N-1}}{d\theta} + \frac{\partial f_N}{\partial f_{N-1}}\frac{\partial f_{N-1}}{\partial f_{N-2}}\frac{df_{N-2}}{d\theta} \\ &= \ldots \end{aligned} \tag{2.15}$$

Where θ is the concatenation of all the free parameters of the network (i.e., weight matrices and bias vectors), and θ_j are the parameters of some particular layer j. Therefore, the gradient with respect to the weights of the first layer will be the multiplication of the partial derivative of each layer with respect to its input times the partial derivative of the first layer with respect to its weights:

$$\frac{\partial}{\partial \boldsymbol{W}_1} f_N \circ \ldots \circ f_1(\boldsymbol{x}) = \frac{\partial f_N}{\partial f_{N-1}}\frac{\partial f_{N-1}}{\partial f_{N-2}} \ldots \frac{\partial f_2}{\partial f_1}\frac{\partial f_1}{\partial \boldsymbol{W}_1} \tag{2.16}$$

If each partial derivative is smaller than 1, their multiplication will lead to a vanishing value. If they are bigger than 1, their multiplication will lead to an exploding value. Thus, training can only be possible if the partial derivative of each layer is very close to 1. That can be achieved through proper weight initialization, or a thoughtful choice of activation functions. For example, the Xavier initialization rule assumes that each activation function is linear and tries to set the expected value of the activation to 0 and the variance to 1 (GLOROT; BENGIO, 2010). The Rectified Linear Unit (ReLU) nonlinearity is $max(0, x)$, so when the input is positive, its gradient will be exactly 1 everywhere (NAIR; HINTON, 2010). Batch normalization ensures the feedforward activations have zero expectation and unitary variance within each mini-batch of training samples (IOFFE; SZEGEDY, 2015).

2.3.1 Regularization and Optimization

Weight decay is not enough to regularize deep models. The complex interaction of layers allows overfitting even with small weight values. Dropout randomly drops neurons from one particular layer in order to simulate an ensemble of an exponential number of networks and prevent the co-adaption of units (SRIVASTAVA *et al.*, 2014). Dropout can also be interpreted as a Bayesian inference procedure, which makes the network more robust to overfitting if compared to point estimation procedures such as maximum likelihood or maximum a posteriori (GAL; GHAHRAMANI, 2016) (we will explore that interpretation in more detail in section 2.3.2).

Weight tying reduces the number of weights and exploits the structure of the input. The most popular strategy for that is the convolutional layer (LECUN; BENGIO, 1995). A convolutional unit uses the same weights to process different parts of the input image, so one convolutional unit may learn to detect one kind of regularity in the image, such as edges. Deep convolutional networks are able to learn invariant and high-level features of the input image (ZEILER; FERGUS, 2014).

The flow of gradients is made explicit in architectures such as the Long Short-Term Memory Network (LSTM) recurrent network (HOCHREITER; SCHMIDHUBER, 1997) and the residual convolutional network (HE *et al.*, 2016). They both use connections between layers that are not hampered by nonlinearities, so the gradient cannot vanish through these connections. Gradient clipping can be used to prevent exploding values (PASCANU *et al.*, 2013). That allows training networks using more layers (up to a 1000 in the residual network case) or more timesteps in the recurrent case.

Most deep learning models are trained using some variant of Stochastic Gradient Descent (SGD) applied to mini-batches of data. Second-order models, such as L-BFGS, usually do not scale well with the number of parameters and data points used in modern neural networks. However, there are efficient second-order methods, such as the Nonlinear Conjugate Gradient (CG) algorithm, that do not require the direct calculation of the Hessian and scale as well as first-order methods with the number of parameters (SANTOS; VON ZUBEN, 1999). Unfortunately, they do not appear to provide advantage in practice over the stochastic gradient descent. For example, a variant of the CG algorithm was proposed recently to train LSTMs (MARTENS; SUTSKEVER, 2011), but it was soon superseded by SGD and momentum with better initialization (SUTSKEVER *et al.*, 2013).

The basic stochastic gradient descent algorithm is:

$$\theta_{t+1} = \theta_t - \alpha \nabla \mathcal{L}_\theta(\boldsymbol{x}, \hat{y}) \tag{2.17}$$

where α is the learning rate and $\nabla \mathcal{L}_\theta(\boldsymbol{x}, \hat{y})$ is the gradient of the loss function with respect to one or few input points. A popular variant includes momentum:

$$\begin{aligned} \boldsymbol{v} &= \gamma \boldsymbol{v} + \alpha \nabla \mathcal{L}_\theta(\boldsymbol{x}, \hat{y}) \\ \theta_{t+1} &= \theta_t - \boldsymbol{v} \end{aligned} \tag{2.18}$$

Other variants include Adagrad (DUCHI *et al.*, 2011), Adadelta (ZEILER, 2012) and ADAM (KINGMA; BA, 2015), which offer some kind of adaptation of the learning rate to the local curvature of the loss function. This is done by assigning an individual rate to each weight of the network and using their recent history to adapt their values. Those methods can be seen as a simplification of second-order methods that scale better with the dimensionality of the network and the number of training samples.

Recently, ADAM has been widely used in practice (KARPATHY, 2017). It uses an unbiased estimation of the first two moments of the gradient ($\boldsymbol{m}_t$ and $\boldsymbol{v}_t$, respectively) to adapt the optimization steps (KINGMA; BA, 2015):

$$\theta_{t+1} = \theta_t - \alpha \frac{\boldsymbol{m}_t}{\sqrt{\boldsymbol{v}_t} + \epsilon} \tag{2.19}$$

2.3.2 Bayesian Deep Learning

A Bayesian neural network treats its weights as unobserved variables, whose distribution must be inferred using the data. The prior distribution is usually simple and uninformative, as there is no direct *a priori* physical interpretation for the weights. The standard Gaussian distribution is the most commonly used prior: $\theta \sim \mathcal{N}(0, 1)$.

The process of learning in a Bayesian neural network is the process of Bayesian inference: Given the prior on the weights, the likelihood function, and the data, we infer the posterior distribution of each weight. In practice, we can perform the inference in three ways:

- Markov chain Monte-Carlo (MCMC): We construct a chain of posterior samples, which will approximate the true posterior. Hamiltonian Monte Carlo (HMC) is the most used method, as it leverages the gradient to scale the inference to the high dimensions of neural networks (NEAL *et al.*, 2011).
- Variational approximation: We approximate the posterior to a function or functional, whose parameters or forms we find by maximizing a lower bound on the marginal log-likelihood (BLUNDELL *et al.*, 2015).
- Dropout: We use multiple dropout forward passes in test time, as they are equivalent to a Bayesian prediction (marginalized over the parameters' posteriors) given a particular variational approximation (GAL; GHAHRAMANI, 2016).

Bayesian neural networks are robust to overfitting, and give better estimates for the uncertainty of the prediction, as they consider two sources of uncertainty: *aleatoric*, which comes from the likelihood function, and *epistemic*, which comes from the weights (KENDALL; GAL, 2017).

Due to the costs of Bayesian inference, Bayesian neural networks are not widely used in practice. However, we may profitably reinterpret under a Bayesian perspective some of the *ad hoc* regularizations used in ordinary deep learning, including dropout (GAL; GHAHRAMANI, 2016; BLUM *et al.*, 2015), early stopping (MACLAURIN *et al.*, 2015), and weight decay (BISHOP, 2007; BLUNDELL *et al.*, 2015). In addition, Bayesian neural networks may provide protection against adversarial attacks (LI; GAL, 2017; LOUIZOS; WELLING, 2017).

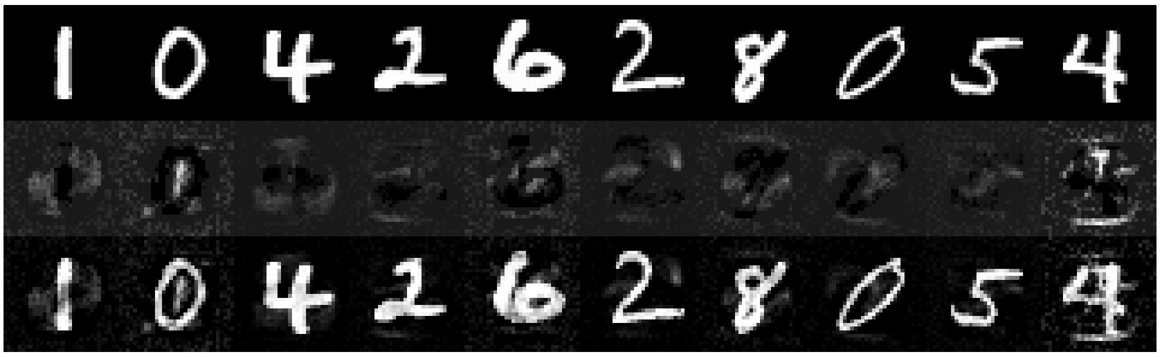
2.4 Adversarial Images

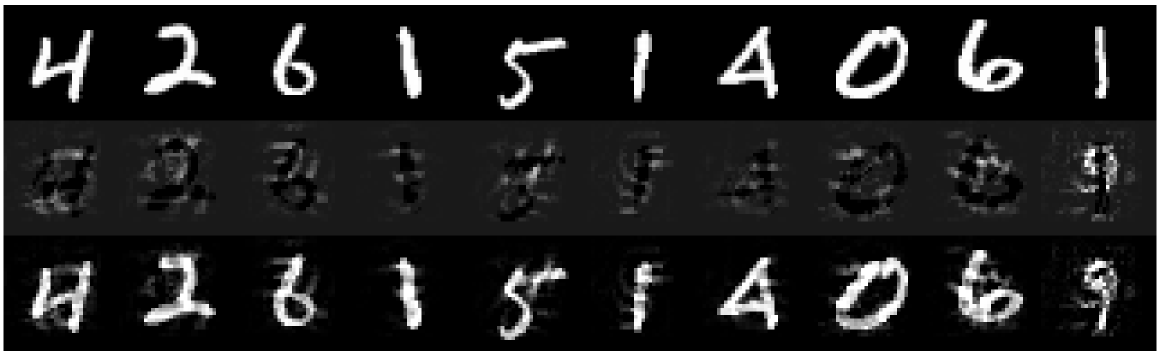
After the huge empirical success of deep neural networks, Szegedy *et al.* surprised the community, showing that small but purposeful pixel distortions can easily fool the best convolutional networks for image classification (SZEGEDY *et al.*, 2014). Szegedy *et al.* used the gradient of the network output with respect to its input to find the minimal pixel distortion that leads to misclassification — small distortions which are hardly visible to humans. We present some examples in Figure 4 for two different datasets (MNIST and ImageNet) and three different network architectures. The optimization problem, the algorithm to solve it, and the practical details are explained in Chapter 3.

Adversarial images even generalize across different network architectures (TRAMÈR *et al.*, 2017). Nguyen *et al.* showed how adversarial images can be generated using evolutionary approaches (NGUYEN *et al.*, 2015). Goodfellow *et al.* demonstrated that only one gradient evaluation is necessary to arrive at an adversarial image (GOODFELLOW *et al.*, 2015). Papernot *et al.* showed that even a black-boxes can be adversarially attacked, given an oracle to provide labels for input images by training a local substitute model (PAPERNOT *et al.*, 2017). That black-box attack can be used in a more general manner to steal machine learning models that are available as prediction APIs (TRAMÈR *et al.*, 2016). Sara Sabour *et al.* (SABOUR *et al.*, 2016) show that adversarial attacks can not only lead to mislabeling, but also manipulate the internal representations of the network. Adversarial examples can also attack neural network policies in the context of reinforcement learning (HUANG *et al.*, 2017).

The problem of adversarial images has divided the Machine Learning community, with some hailing it as a "deep flaw" of deep neural networks (BI, 2014); and others promoting a more cautious interpretation, and showing, for example, that most classifiers are susceptible to adversarial examples (GOODFELLOW *et al.*, 2015; FAWZI *et al.*, 2016).

Despite the controversy, adversarial images surely suggest a lack of robustness, since they are (for humans) essentially equal to correctly classified images. Immunizing a network against those perturbations increases its ability to generalize, a form of regularization (GOODFELLOW *et al.*, 2015) whose statistical nature deserves further investigation. Even the traditional backpropagation training procedure can be improved with adversarial gradient (NØKLAND, 2015). The idea behind adversarial regularization is to make the input space smooth, therefore small changes in the input will not lead to large changes on the output, which is one explanation of the existence of adversarial images. This idea is


(a) MNIST with logistic regression. The correct labels are self-evident.


(b) MNIST with convolutional network. The correct labels are self-evident.

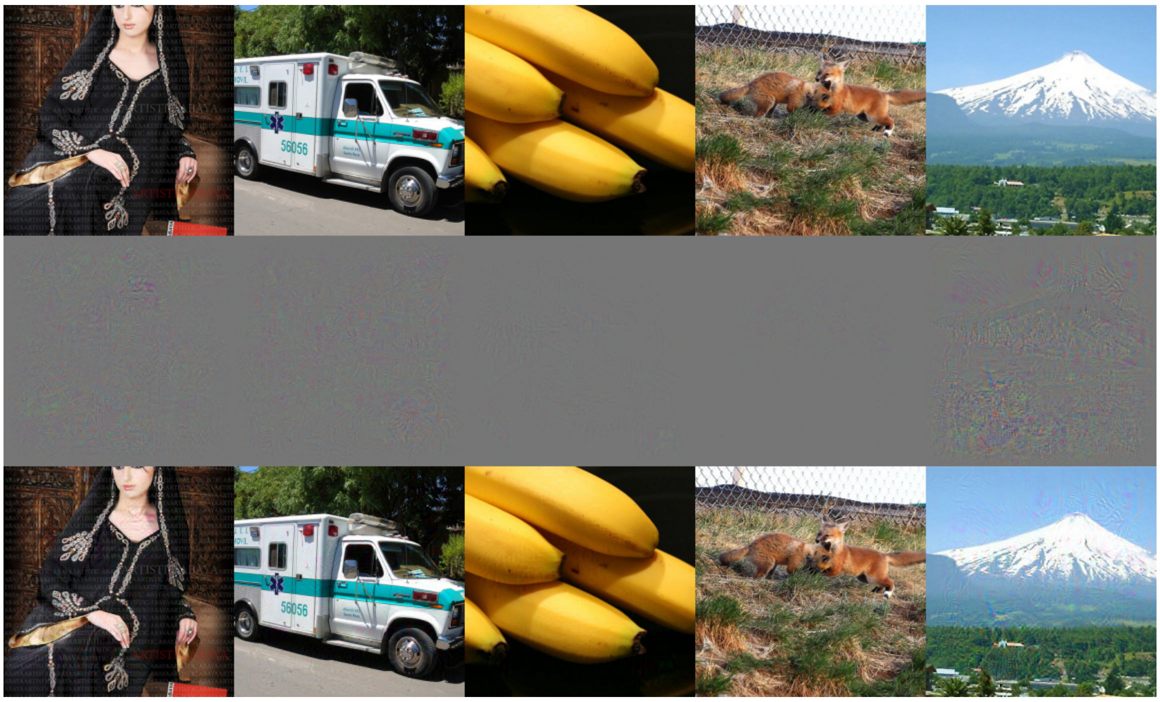

(c) OverFeat on ImageNet. From left to right, correct labels: 'Abaya', 'Ambulance', 'Banana', 'Kit Fox', 'Volcano'. Adversarial labels for all: 'Bolete' (a type of mushroom).

Figure 4 – Adversarial examples for each network. For all experiments: original images on the top row, adversarial images on the bottom row, distortions (difference between original and adversarial images) on the middle row.

formalized in the Virtual Adversarial Training, where a regularization term of input space smoothness is added to the training procedure, allowing even unsupervised networks to benefit from this concept (MIYATO *et al.*, 2016).

Recent work shows that it is possible for the training procedure to make classifiers robust to adversarial examples by using a strong adversary (HUANG *et al.*, 2015), defensive distillation (PAPERNOT *et al.*, 2016), bio-inspired units that saturate (NAYEBI; GANGULI, 2017), and Bayesian neural networks (LI; GAL, 2017). Those attempts of protection against adversarial images make the attack harder, but not impossible, in the sense that either new techniques are required for the attack or that more input distortion is necessary. For example, after the publication of (NAYEBI; GANGULI, 2017), it was shown that a different optimization procedure enables an attack (BRENDEL; BETHGE, 2017). We have found via preliminary unpublished experiments that Bayesian neural networks can be attacked, given an appropriate optimization objective, therefore, the results of (LI; GAL, 2017) shows how to increase the attack difficulty (i.e. require more distortion in order to arrive at the same misdirection), instead of preventing it completely.

2.5 Variational Autoencoders

Autoencoders are unsupervised learning models that learn a mapping between the input data to a latent space of smaller dimensionality, which can be used to reconstruct the original input. Autoencoders are trained to minimize the reconstruction error plus some regularization cost. Autoencoders are composed of two parts: an encoder, which transforms the input into the latent variable representation, and a decoder, which transforms the latent representation back into the input:

$$\begin{tikzcd}
\boxed{\begin{matrix}\text{Input}\\ \boldsymbol{x}\end{matrix}} \arrow[r] & \boxed{\begin{matrix}\text{Encoder}\\ p(\boldsymbol{z}|\boldsymbol{x})\end{matrix}} \arrow[r] & \boxed{\begin{matrix}\text{Latent space}\\ \boldsymbol{z}\end{matrix}} \arrow[r] & \boxed{\begin{matrix}\text{Decoder}\\ p(\boldsymbol{x}|\boldsymbol{z})\end{matrix}} \arrow[r] & \boxed{\begin{matrix}\text{Output}\\ \boldsymbol{x}\end{matrix}}
\end{tikzcd}$$

A nonlinear autoencoder uses nonlinear mappings in the encoder and decoder. Some famous variants include the sparse autoencoder, which uses a sparsity (L1) regularization term (NG, 2011), and the denoising autoencoder, which regularizes implicitly by feeding the input signal with noise, while the reconstruction term uses the original input signal (VINCENT *et al.*, 2010).

A modern variant that is growing in popularity is the variational autoencoder (KINGMA; WELLING, 2014). It gives a Bayesian interpretation to the latent variables to determine the reconstruction and regularization objectives in a principled manner. The variational autoencoder is a probabilistic generative model, so we find the probability distribution of the data by marginalizing the latent variables:

$$p_\theta(\boldsymbol{x}) = \int p_\theta(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z} = \int p_\theta(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{z}) d\boldsymbol{z} \tag{2.20}$$

The likelihood $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is the probabilistic explanation of the observed data, but in practice it is usually just the decoder network under some noise consideration. All the decoder parameters are comprised in θ (i.e. weights and biases). The $p(\boldsymbol{z})$ prior is usually considered as a standard Gaussian ($\mathcal{N}(0, I)$) (KINGMA; WELLING, 2014), but it can be discrete (e.g. Bernoulli) (KINGMA *et al.*, 2014), a mixture model (DILOKTHANAKUL *et al.*, 2016), or even have some geometric interpretation ("what" and "where" latent variables) (ESLAMI *et al.*, 2016).

The expression above is generally intractable, so we maximize the variational lower bound instead:

$$\begin{aligned} \log p(\boldsymbol{x}) &\geq \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln p_\theta(\boldsymbol{x}|\boldsymbol{z})] - KL(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})) \\ &= -KL(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z}|\boldsymbol{x})) \end{aligned} \tag{2.21}$$

Note that the variational lower bound is equivalent to the Kullback-Leibler (KL) divergence[6] between the approximated posterior and the real posterior (which is unknown). So another interpretation of the lower bound maximization is finding the best posterior approximation in the KL sense. The KL divergence measures how similar two probability distributions are, though it is not a metric as it is not symmetric. The approximated posterior is often parametrized as an independent multivariate Gaussian distribution with means and variances determined by the encoder network:

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}_\phi(\boldsymbol{x}), \exp(\boldsymbol{\sigma}_\phi^2(\boldsymbol{x}))) \tag{2.22}$$

If the prior and posterior are both Gaussian, their KL divergence has analytical form:

$$KL(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})) = \frac{1}{2}\sum_{j=1}^{J} 1 + \log \sigma_{\phi_j}(\boldsymbol{x})^2 - \mu_{\phi_j}(\boldsymbol{x})^2 - \sigma_{\phi_j}(\boldsymbol{x})^2 \tag{2.23}$$

The likelihood expectation under the posterior approximation can be approximated using Monte Carlo samples (i.e. by averaging the likelihood of the reconstruction of multiple samples). The reparameterization trick makes the variance of the gradient of the lower bound scale with the inverse of the mini-batch size (BLUM *et al.*, 2015), so in practice it is common to see just one Monte Carlo sample. The reparameterization trick is just the sampling of simple standard distributions and their transformation into the actual posterior (latent variable) sample. For example, for a Gaussian posterior, we can use the affine transformation property of the Gaussian distribution:

$$\begin{aligned} \epsilon &\sim \mathcal{N}(0, 1) \\ \boldsymbol{z} &= \mu(\boldsymbol{x}) + \sigma(\boldsymbol{x})\epsilon \end{aligned} \tag{2.24}$$

[6] $KL(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log(\frac{p(x)}{q(x)}) dx$

where $\mu(\boldsymbol{x})$ and $\sigma(\boldsymbol{x})$ are the outputs of the encoder network.

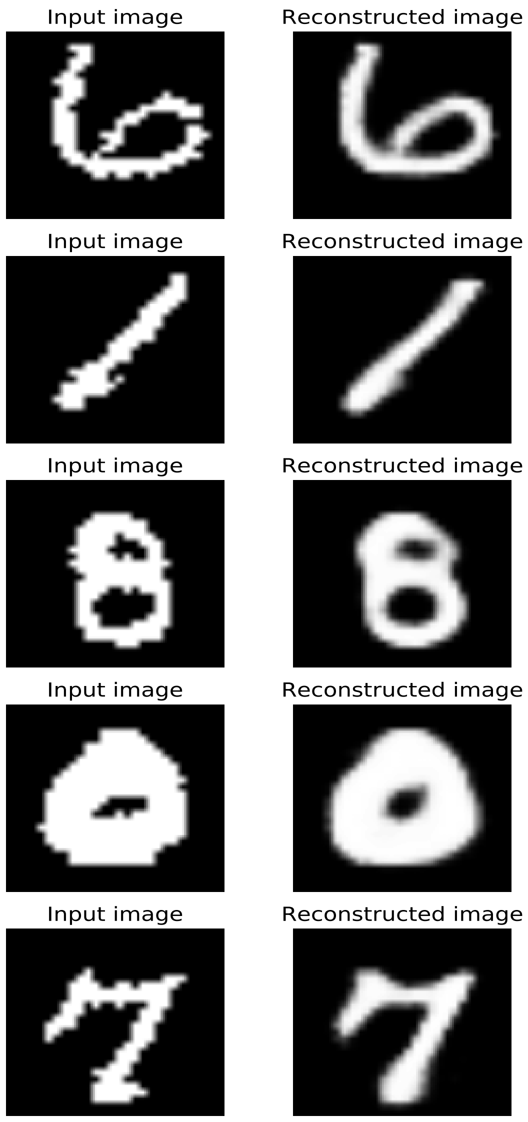


Figure 5 – MNIST reconstructions using a fully-connected variational autoencoder and 100 latent variables.

The encoder and decoder networks can be any neural network, such as a multi-layer perceptron (KINGMA; WELLING, 2014) or a convolutional network (RADFORD *et al.*, 2016). We show some examples of input and reconstruction of a fully-connected (i.e. both encoder and decoder networks are MLPs) variational autoencoder trained on the MNIST dataset in Figure 5. In Chapter 4 we give more details about the procedures used to train the variational autoencoders and generate those images.

A recent development are recurrent variational autoencoders, which use soft attention to encode and decode patches of the image (GREGOR *et al.*, 2015). The encoder and decoder are LSTM recurrent networks (HOCHREITER; SCHMIDHUBER, 1997). A related approach using convolutional LSTMs achieved better lossy image compression than the JPEG algorithm (GREGOR *et al.*, 2016).

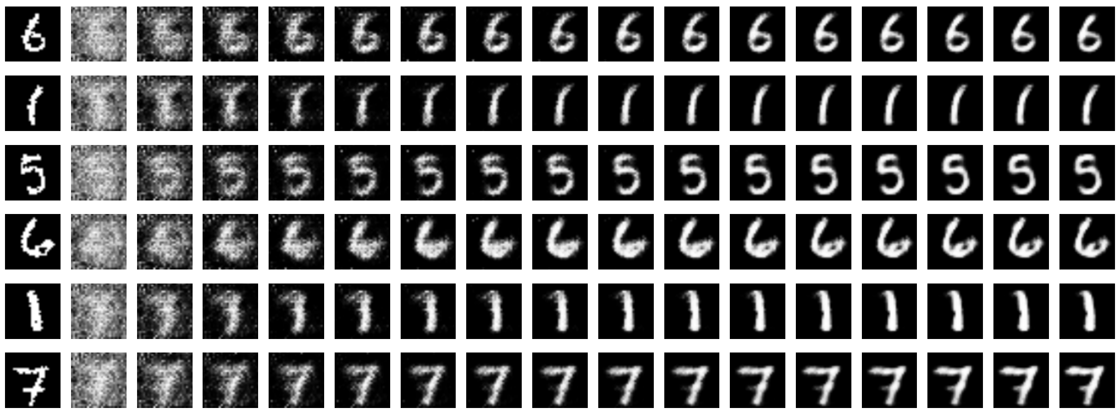


Figure 6 – Recurrent variational autoencoder. The encoder and decoder are LSTM recurrent networks. The encoder receives the reconstruction error (the difference of the previous reconstruction and the input image). We use 10 latent variables for 16 time-steps. The first column shows the input image, and the others show from left to right the evolution of the reconstruction at each time-step.

Recurrent variational autoencoders build the reconstruction step by step, using the feedback between the real image and the current reconstruction to guide the process. With an attention mechanism, that reconstruction by feedback process allows such autoencoders to achieve the state of the art of lossy compression for images (GREGOR *et al.*, 2016). See figure 6 for some examples of the step by step reconstruction process (without attention) on the MNIST dataset.

There are many possible applications to variational autoencoders. By simulating a chain of samples of the latent variables and likelihood function it is possible to do image denoising and missing data imputation (image inpainting) (REZENDE *et al.*, 2014). The latent variables of a variational autoencoder can be used for visual analogy and interpolation (RADFORD *et al.*, 2016). The latent variables can also represent states in dynamical systems, allowing the control of complex systems from high-dimensional inputs, such as images (WATTER *et al.*, 2015).

2.6 Conclusion

We reviewed the basics of machine learning from a statistical learning perspective, neural networks, and deep learning, with a focus on adversarial images and variational autoencoders, which will be the subject of the following chapters. We showed that many procedures used in neural networks and deep learning — including variational autoencoders — may be interpreted under the lens of statistics, and in particular of Bayesian statistics.

We also showed that even though adversarial examples for deep learning are a rather new research topic, there is already plenty of work presenting many different adversarial attacks. There are, as well, many attempts to defend the models against adversarial attacks, but none has so far convincingly succeeded.

3 Exploring the Space of Adversarial Images

In this chapter, we extend previous works on adversarial images for deep neural networks (SZEGEDY *et al.*, 2014), by exploring the pixel space of such images using random perturbations. This chapter in based on the paper Tabacof & Valle (2016), published at the IEEE International Joint Conference on Neural Networks (IJCNN) 2016.

Initial skepticism about the relevance of adversarial images suggested they existed as isolated points in the pixel space, reachable only by a guided procedure with complete access to the model. More recent works (GOODFELLOW *et al.*, 2015; GU; RIGAZIO, 2015) claim that they inhabit large and contiguous regions in the space. The correct answer has practical implications: if adversarial images are isolated or inhabit very thin pockets, they deserve much less worry than if they form large, compact regions. In this work we intend to shed light to the issue with an in-depth analysis of adversarial image space. We propose a framework (Figure 7) that allows us to ask interesting questions about adversarial images.

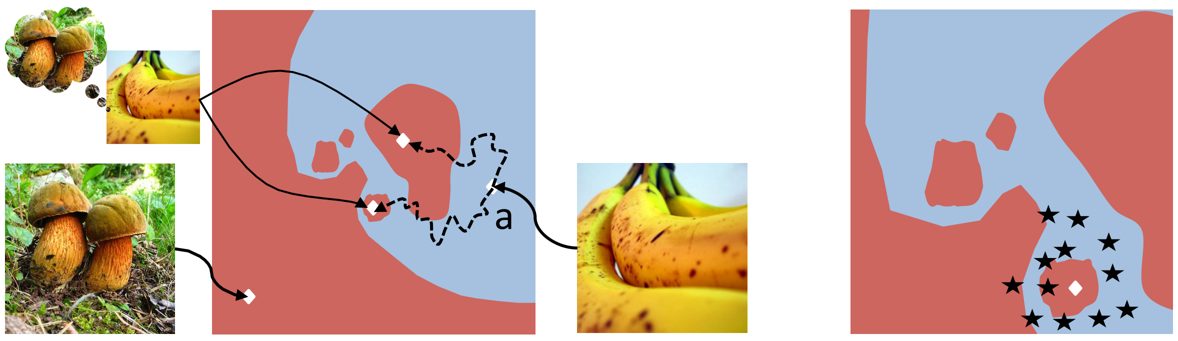


Figure 7 – Fixed-sized images occupy a high-dimensional space spanned by their pixels (one pixel = one dimension), here depicted as a 2D colormap. **Left:** classifiers associate points of the input pixel space to output class labels, here 'banana' (blue) and 'mushroom' (red). From a correctly classified original image (a), an optimization procedure (dashed arrows) can find adversarial examples that are, for humans, essentially equal to the original, but that will fool the classifier. **Right:** we probe the pixel space by taking a departing image (white diamond), adding random noise to it (black stars), and asking the classifier for the label. In compact, stable regions, the classifier will be consistent (even if wrong). In isolated, unstable regions, as depicted, the classifier will be erratic.

We can embed fixed-sized images into a vector space, each pixel corresponding to a dimension. The pixel space, however, contains a lot of structure that the vector space discards. Most researchers currently believe that images of a given appearance (or a given meaning) exist in low-dimensional, but extremely intricate manifolds within the whole vector space (BENGIO, 2009). However, given that determining and exploring the manifolds of deep neural networks is still a matter of ongoing research, we restrict our analyses to the pixel space.

3.1 Creating Adversarial Images

Assume we have a pre-trained classifier $\boldsymbol{p} = f(\boldsymbol{x})$ that, for each input $\boldsymbol{x} \in \mathcal{I}$, corresponding to the pixels of a fixed-sized image, outputs a vector of probabilities $\boldsymbol{p} = [p_1 \cdots p_i \cdots p_n]$ of the image belonging to the class label i. We will be rather lax in what we accept as output: most uncertainties behaving like probabilities will do (i.e., ranging from 0 to 1, additive, normalized, etc.). We can assign h to the label corresponding to the highest probability p_h. Assume further that $\mathcal{I} = [L - U]$, for grayscale images, or $\mathcal{I} = [L - U]^3$ for RGB images, where L and U are the lower and upper limits of the pixel scale. In most cases L is 0, and U is either 1 or 255.

Assume that c is the correct label and that we start with $h = c$, otherwise there is no point in fooling the classifier. We want to add the smallest distortion $\boldsymbol{d}$ to $\boldsymbol{x}$, such that the highest probability will no longer be assigned to h. The distortions must keep the input inside its space, i.e., we must ensure that $\boldsymbol{x} + \boldsymbol{d} \in \mathcal{I}$. In other words, the input is box-constrained. Thus, we have the following optimization:

$$\begin{aligned} \underset{\boldsymbol{d}}{\text{minimize}} \quad & \left\|\boldsymbol{d}\right\| \\ \text{subject to} \quad & L \leq \boldsymbol{x} + \boldsymbol{d} \leq U \\ & \boldsymbol{p} = f(\boldsymbol{x} + \boldsymbol{d}) \\ & \max(p_1 - p_c, ..., p_n - p_c) > 0 \end{aligned} \tag{3.1}$$

That formulation is more general than the one presented by (SZEGEDY *et al.*, 2014), for it ignores non-essential details, such as the choice of the adversarial label. It also showcases the non-convexity: since $\max(x) < 0$ is convex, the inequality is clearly concave (BOYD; VANDENBERGHE, 2004), making the problem non-trivial even if the model $\boldsymbol{p} = f(\boldsymbol{x})$ were linear in $\boldsymbol{x}$. Deep networks, of course, exacerbate the non-convexity due to their highly non-linear model. For example, a simple multi-layer perceptron for binary classification could have $f(\boldsymbol{x}) = \text{logit}^{-1}(W_2 \cdot \tanh(W_1 \cdot \boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2)$, which is neither convex nor concave due to the hyperbolic tangent.

3.1.1 Procedure

Training a classifier usually means minimizing the classification error by changing the model weights. To generate adversarial images, however, we hold the weights fixed, and find the minimal distortion that still fools the network. Because any two images can be directly switched with a large-enough distortion, the problem is only interesting for small distortions, preferably those imperceptible to humans.

We can simplify the optimization problem of equation 3.1 by exchanging the max inequality for a term in the loss function that measures how adversarial the probability output is:

$$\begin{aligned} \underset{\boldsymbol{d}}{\text{minimize}} \quad & \left\|\boldsymbol{d}\right\|_2^2 + C \cdot \mathrm{H}(\boldsymbol{p}, \boldsymbol{p}^A) \\ \text{subject to} \quad & L \leq \boldsymbol{x} + \boldsymbol{d} \leq U \\ & \boldsymbol{p} = f(\boldsymbol{x} + \boldsymbol{d}) \end{aligned} \tag{3.2}$$

where we introduce the adversarial probability target $\boldsymbol{p}^A = [\mathbb{1}_{i=a}]$, which assigns zero probability to all but a chosen adversarial label a. We use the square of the ℓ_2-distance as the penalty term to the adversarial distortion. We experimented with the ℓ_1-distance penalty, but we did not find any improvements in the adversarial attack. The formulation in equation 3.2 is essentially the same of Szegedy *et al.* (2014), picking an explicit (but arbitrary) adversary label. We stipulate the loss function: the cross-entropy (H) between the probability assignments; while Szegedy *et al.* (2014) keep that choice open.

The constant C balances the importance of the two objectives. The lower the constant, the more we will minimize the distortion norm. Values too low, however, may turn the optimization unfeasible. We want the lowest, but still feasible, value for C.

We can solve the new formulation with any local search compatible with box-constraints. Since the optimization variables are the pixel distortions, the problem size is exactly the size of the network input, in our case $28 \times 28 = 784$ for MNIST (LECUN *et al.*, 1998) and $221 \times 221 \times 3 = 146\,523$ for OverFeat (SERMANET *et al.*, 2013)[1]. Such input sizes make numeric differentiation (e.g. finite differences) impractical to compute the huge number of gradients required to find an adversarial image: a single gradient of the output with respect to the input of a $256{\times}256$ pixels$\times$3 channels image requires almost 200 thousand feedforward evaluations. We must, thus, resort to backpropagation, just as if we were training the network.

In contrast to current neural network training, that reaches hundreds of millions of weights, those sizes are small enough to allow second-order procedures, which converge faster and with better guarantees (NOCEDAL; WRIGHT, 2006). We chose L-BFGS-B, a box-constrained version of the popular L-BFGS second-order optimizer[2] (ZHU *et al.*, 1997). We set the number of corrections in the limited-memory matrix to 15, and the maximum number of iterations to 150. We used Torch7 to model the networks and

[1] In neural networks we often constraint images to some fixed size, as required by the dense layers.

[2] L-BFGS is the limited memory version of the BFGS optimization algorithm.

extract their gradients with respect to the inputs[3] (COLLOBERT *et al.*, 2011).

Finally, we implemented a bisection search to determine the optimal value for C (BURDEN; FAIRES, 1985). The algorithm is explained in detail in the next section.

3.1.2 Algorithm

Algorithm 3.1 implements the optimization procedure used to find the adversarial images. The algorithm is essentially a bisection search for the constant C, where in each step we solve a problem equivalent to equation 3.2. Bisection requires initial lower and upper bounds for C, such that the upper bound succeeds in finding an adversarial image, and the lower bound fails. It will then search the transition point from failure to success (the "zero" in a root-finding sense): that will be the best C. We can use $C = 0$ as lower bound, as it always leads to failure (the distortion will go to zero). To find an upper bound leading to success, we start from a very low value, and exponentially increase it until we succeed. During the search for the optimal C we use warm-starting in L-BFGS-B to speed up convergence: the previous optimal value found for $\boldsymbol{d}$ is used as initialization for the next attempt.

To achieve the general formalism of equation 3.1 we would have to find the adversarial label leading to minimal distortion. However, in datasets like ImageNet (DENG *et al.*, 2009), with hundreds of classes, that search would be too costly. Instead, in our experiments, we opt to consider the adversarial label as one of the sources of random variability. As we will show, that does not upset the analyses.

3 In regular neural network training, the gradient required would be with respect to the weights. Backpropagation also requires the gradient with respect to the input for internal computations, so it is easy to extract such information from Torch7 or an equivalent library.

Algorithm 3.1 Adversarial image generation algorithm

```
Require: A small positive value $\epsilon$
Ensure: $L\text{-}BFGS\text{-}B(\boldsymbol{x}, \boldsymbol{p}^A, C)$ solves optimization 3.2
 1: {Finding initial $C$}
 2: $C \leftarrow \epsilon$
 3: repeat
 4:     $C \leftarrow 2 \times C$
 5:     $\boldsymbol{d}, \boldsymbol{p} \leftarrow L\text{-}BFGS\text{-}B(\boldsymbol{x}, \boldsymbol{p}^A, C)$
 6: until $\max(p_i)$ in $\boldsymbol{p}$ is $p_a$
 7: {Bisection search}
 8: $C_{low} \leftarrow 0$, $C_{high} \leftarrow C$
 9: repeat
10:     $C_{half} \leftarrow (C_{high} + C_{low})/2$
11:     $\boldsymbol{d}', \boldsymbol{p} \leftarrow L\text{-}BFGS\text{-}B(\boldsymbol{x}, \boldsymbol{p}^A, C_{half})$
12:     if $\max(p_i)$ in $\boldsymbol{p}$ is $p_a$ then
13:         $\boldsymbol{d} \leftarrow \boldsymbol{d}'$
14:         $C_{high} \leftarrow C_{half}$
15:     else
16:         $C_{low} \leftarrow C_{half}$
17:     end if
18: until $(C_{high} - C_{low}) < \epsilon$
19: return $\boldsymbol{d}$
```

3.2 Adversarial Space Exploration

In this section we explore the vector space spanned by the pixels of the images to investigate the “geometry” of adversarial images: are they isolated, or do they exist in dense, compact regions? Most researchers currently believe that images of a certain appearance (and even meaning) are contained into relatively low-dimensional manifolds inside the whole space (BENGIO, 2009). However, those manifolds are exceedingly convoluted, discouraging direct geometric approaches to investigate the pixel space.

Thus, our approach is indirect, probing the space around the images with small random perturbations. In regions where the manifold is nice — round, compact, occupying most of the space — the classifier will be consistent (even if wrong). In the regions where the manifold is problematic — sparse, discontinuous, occupying small fluctuating subspaces — the classifier will be erratic.

3.2.1 Datasets and Models

To allow comparison with the results of Szegedy *et al.* (2014), we employ the MNIST handwritten digits database (10 classes, 60k training and 10k testing images), and the 2012 ImageNet Large Visual Recognition Challenge Dataset (1000 classes, 1.2M+

Table 1 – Convolutional network for MNIST classification.

Layer	Size	Filter	Nonlinearity
Input	28x28		
Convolution	32	5x5	ReLU
Max pooling		3x3	
Convolution	64	5x5	ReLU
Max pooling		2x2	
Linear	200		ReLU
Linear	10		
Softmax			

training and 150k testing images).

For MNIST, Szegedy *et al.* (2014) tested convolutional networks and autoencoders. We employ both convolutional networks and a logistic linear classifier. While logistic classifiers have limited accuracy (∼7.5% error), their training procedure is convex (BOYD; VANDENBERGHE, 2004). They also allowed us to complement the original results (SZEGEDY *et al.*, 2014) by investigating adversarial images in a shallow classifier.

The convolutional network we employed for MNIST/ConvNet consisted of two convolutional layers, two max-pooling layers, one fully-connected layer, and a softmax layer as output. We used ReLU for the nonlinearity. More details of the architecture can be found in table 1. The network was trained with SGD and momentum. Without data augmentation, that model achieves 0.8% error on the test set.

For ImageNet, we used the pre-trained OverFeat network (SERMANET *et al.*, 2013), which achieved 4th place at the ImageNet competition in 2013, with 14.2% top-5 error in the classification task, and won the localization competition the same year. (SZEGEDY *et al.*, 2014) employed AlexNet (KRIZHEVSKY *et al.*, 2012), which achieved 1st place at the ImageNet competition in 2012, with 15.3% top-5 error. The OverFeat network consists of five convolutional and two fully-connected layers. More details can be found in Sermanet *et al.* (2013).

On each dataset, we preprocess the inputs by standardizing each pixel with the global mean and standard deviation of all pixels in the training set images. We used Torch7 (COLLOBERT *et al.*, 2011) for the implementation[4].

Figure 4, displayed in Chapter 2 for didactic purposes, illustrates all three cases generated by the procedure above. Original and adversarial images are virtually indistinguishable. The pixel differences (middle row) do not show any obvious form — although a faint "erasing-and-rewriting" effect can be observed for MNIST. Figures 4(a) and 4(b) also suggest that the MNIST classifiers are more robust to adversarial images, since the dis-

[4] The source code for adversarial image generation and pixel space analysis can be found at <https://github.com/tabacof/adversarial>.

tortions are larger and more visible. We will see, throughout this chapter, that classifiers for MNIST and for ImageNet have important differences in how they react to adversarial images.

3.2.2 Methods

Each case (MNIST/Logistic, MNIST/ConvNet, ImageNet/OverFeat) was investigated independently, by applying the optimization procedure explained in Section 3.1.1. For ImageNet we sampled 5 classes (Abaya, Ambulance, Banana, Kit Fox, and Volcano), 5 correctly classified examples from each class, and sampled 5 adversarial labels (Schooner, Bolete, Hook, Lemur, Safe), totaling 125 adversarial images. For MNIST, we just sampled 125 correctly classified examples from the 10K examples in the test set, and sampled an adversarial label (from 9 possibilities) for each one. All random sampling was made with uniform probability. To sample only correctly classified examples, we rejected the misclassified ones until we accumulated the needed amount. We call, in the following sections, those correctly classified images *originals*, since the adversarial images are created from them.

The probing procedure consisted in picking an image pair (an adversarial image and its original), adding varying levels of noise to their pixels, resubmitting both to the classifier, and observing if the newly assigned labels corresponded to the original class, to the adversarial class, or to some other class.

We measured the *levels of noise* (λ) relative to the difference between each image pair. We initially tested an independent and identically distributed (i.i.d.) Gaussian model for the noise. For each image $\boldsymbol{x} = \{x_i\}$, our procedure creates an image $\boldsymbol{x}' = \{\text{clamp}(x_i + \epsilon)\}$ where $\epsilon \sim \mathcal{N}(\mu, \lambda\sigma^2)$, and μ and σ^2 are the sample mean and variance of the distortion pixels. In the experiments we ranged λ from 2^{-5} to 2^5. To keep the pixel values of $\boldsymbol{x}'$ within the original range $[L - U]$ we employ $\text{clamp}(x) = \min(\max(x, L), U)$. In practice, we observed that clamping has little effect on the noise statistics.

An i.i.d. Gaussian model discards two important attributes of the distortions: spatial correlations, and higher-order momenta. We wanted to evaluate the relative importance of those, and thus performed an extra round of experiments that, while still discarding all spatial correlations by keeping the noise i.i.d., adds higher momenta information by modeling non-parametrically the distribution of the distortion pixels. Indeed, a study of those higher momenta (Table 2) suggests that the adversarial distortions have a much heavier tail than the Gaussians (shown by the positive excess kurtosis), and we wanted to investigate how that affects the probing. The procedure is exactly the same as before, but with $\epsilon \sim \mathcal{M}$, where $\mathcal{M}$ is an empirical distribution induced by a non-parametric observation of the distortion pixels. That is, we resample the distortion pixels

Table 2 – Descriptive statistics of the adversarial distortions for the two datasets averaged over the 125 adversarial examples. Pixels values in $[0 - 255]$. Logistic and ConvNet refer to MNIST dataset, OverFeat refers to ImageNet dataset.

	Mean	S.D.	Skewness	Ex. Kurtosis
Logistic	30.7 ± 4.3	18.3 ± 11.3	0.1 ± 1.0	7.8 ± 3.2
ConvNet	27.5 ± 2.1	23.0 ± 9.4	-0.5 ± 1.6	17.6 ± 7.3
OverFeat	118.4 ± 0.1	1.9 ± 2.0	0.0 ± 0.1	6.5 ± 4.1

with replacement, like the bootstrap procedure (EFRON; TIBSHIRANI, 1994). In those experiments we cannot control the level: the variance of the noise is essentially the same as the variance of the distortion pixels.

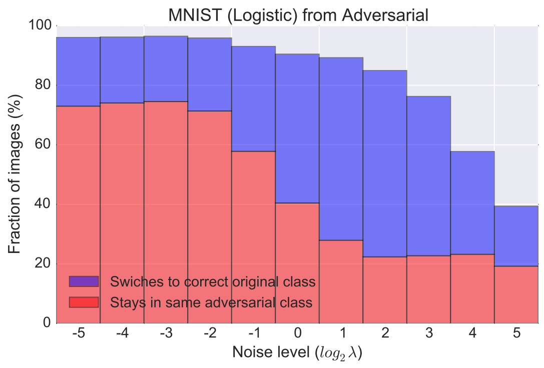
Our main metric is the fraction of images (in %) that keep or switch labels when noise is added to a departing image, which we use as a measure of the stability of the classifier at the departing image in the pixel space. The fraction is computed over a sample of 100 probes, each probe being a repetition of the experiment with all factors held fixed but the sampling of the random noise.

3.2.3 Results

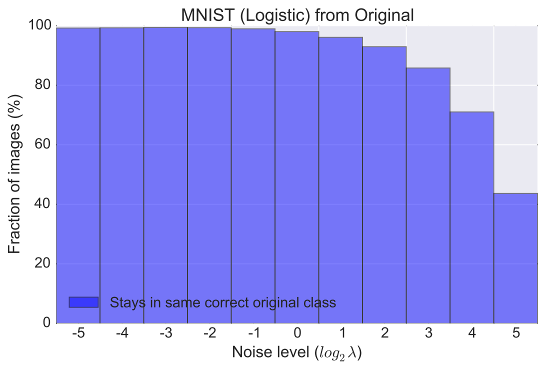
Figure 8 shows that adversarial images do not appear isolated. On the contrary, to completely escape the adversarial pocket we need to add a noise with much higher variance — notice that the horizontal axis is logarithmic — than the distortion used to reach the adversarial image in the first place.

In both networks, the original images display a remarkable robustness against Gaussian noise (Figures 8(b) and 8(f)), confirming that robustness to random noise does not imply robustness to adversarial examples (FAWZI *et al.*, 2016).

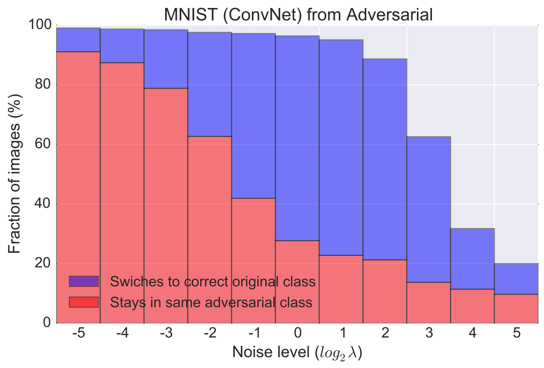
The results in Figure 8 are strongly averaged, each data point summarizing, for a given level of noise, the result of 125 experiments: the fraction of images that fall in each label for *all* five original class labels, *all* five original samples from each label, and *all* five adversarial class labels. In reality there is a lot of variability that can be better appreciated in Figure 9. Here each curve alongside the axis *experiments* represents a *single* choice of original class label, original sample, and adversarial class label, thus there are 125 curves. (The order of the curves along that axis is arbitrary and chosen to minimize occlusions and make the visualization easier). The graphs show that depending on a specific configuration, the label may be very stable and hard to switch (curves that fall later or do not fall at all), or very unstable (curves that fall early). Those 3D graphs also



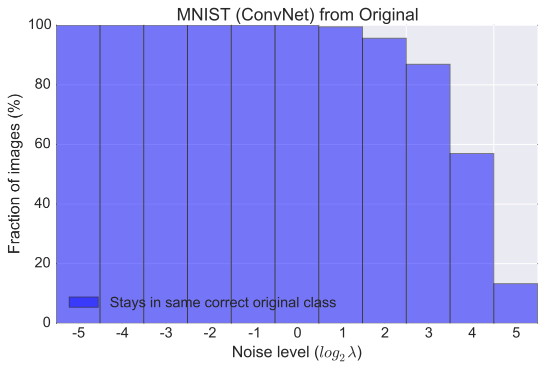
(a)



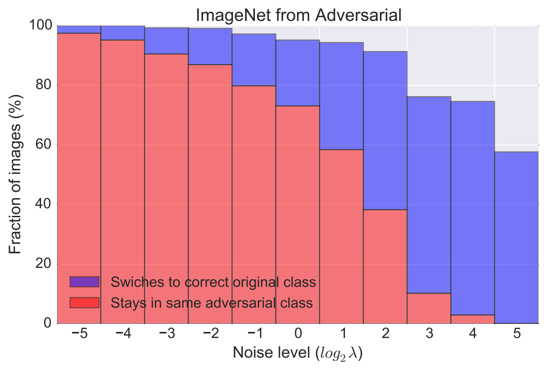
(b)



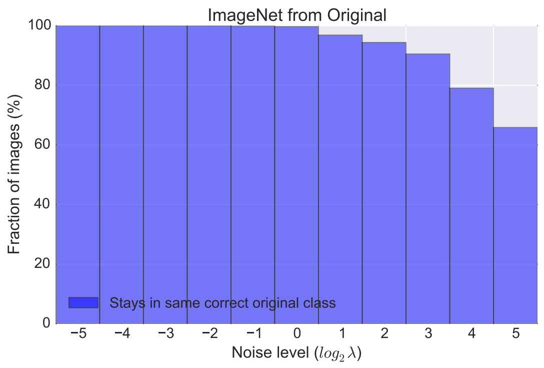
(c)



(d)



(e)

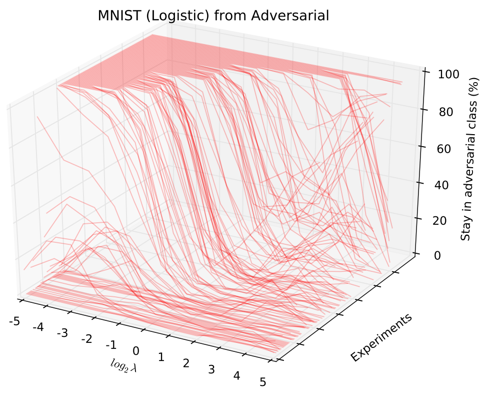


(f)

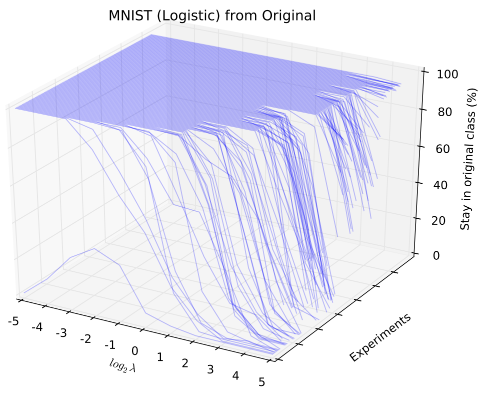
Figure 8 – Adding Gaussian noise to the images. We perform the probing procedure explained in Section 3.2.2 to measure the stability of the classifier boundaries at different points of the pixel space. To escape the adversarial pockets completely we have to add a noise considerably stronger than the original distortion used to reach them in the first place: adversarial regions are not isolated. That is especially true for ImageNet/OverFeat. Still, the region around the correctly classified original image is much more stable. This graph is heavily averaged: each stacked column along the horizontal axis summarizes 125 experiments $\times$ 100 random probes.

reinforce the point about the stability of the correctly classified original images.

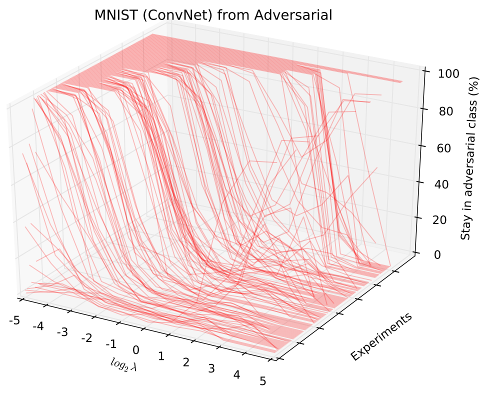
The results suggest that the classifiers for MNIST are more resilient against adversarial images than ImageNet/OverFeat. Moreover, the shallow MNIST/logistic behaves



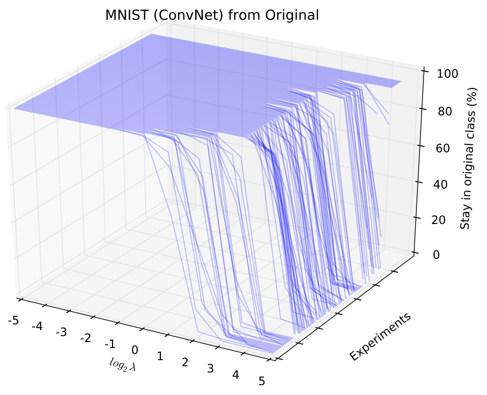
(a)



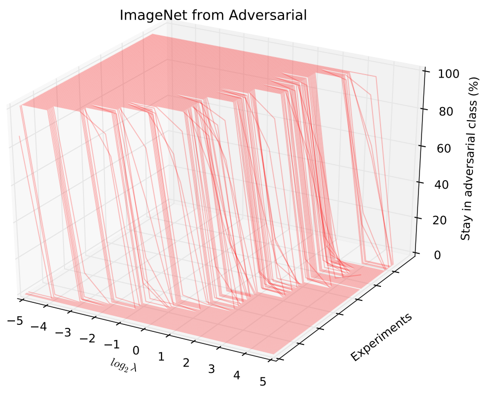
(b)



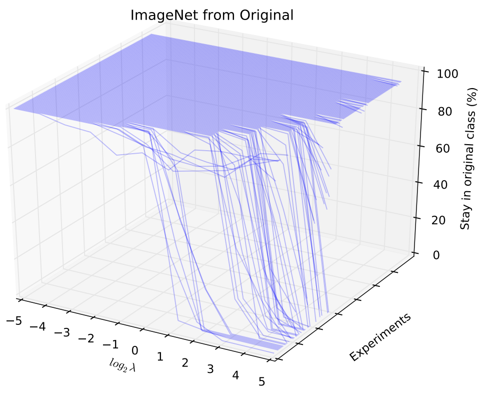
(c)



(d)



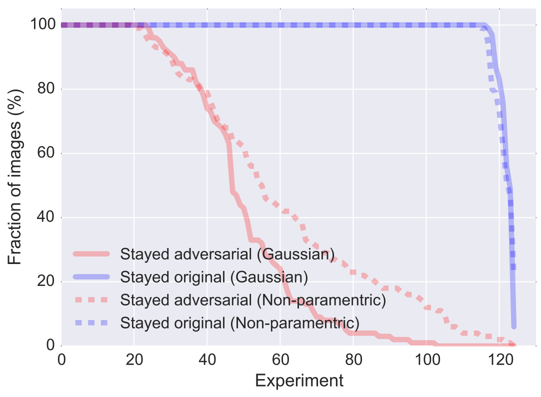
(e)



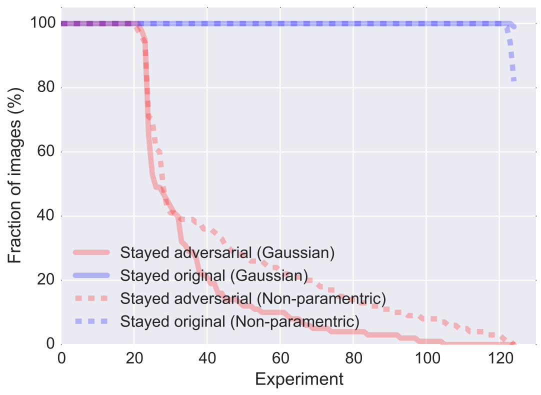
(f)

Figure 9 – Adding Gaussian noise to the images. Another view of the probing procedure explained in Section 3.2.2. Contrarily to the averaged view of Figure 8, here each one of the 125 experiments appears as an independent curve along the *Experiments* axis (their order is arbitrary, chosen to reduce occlusions). Each point of the curve is the fraction of probes (out of a hundred performed) that keeps their class label.

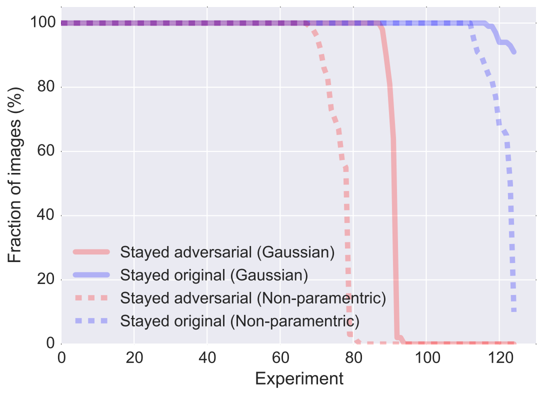
differently than the deep MNIST/ConvNet, as shown by the "falling columns" in Figure 8: initially, a small push in MNIST/logistic throws a larger fraction of the adversarial ex-



(a) MNIST / logistic regression



(b) MNIST / convolutional network



(c) ImageNet / OverFeat

Figure 10 – For each of the 125 experiments we measure the fraction of the probe images (i.e., departing image + random noise) that stayed in the same class label. Those fractions are then sorted from biggest to lowest along the *Experiments* axis. The area under the curves indicates the entire fraction of probes among all experiments that stayed in the same class.

amples back to the correct space. However, at large noise levels, MNIST/logistic saturates with a larger fraction of images still adversarial than MNIST/ConvNet.

Finally, we wanted to investigate how the nature of the noise added affected the experiments. Recall that our i.i.d. Gaussian noise differs from the original optimized distortion in two important aspects: no spatial correlations, and no important higher-order momenta. To explore the influence of those two aspects, we introduced a noise modeled after the empirical distribution of the distortion pixels. That still ignores spatial correlations, but captures higher-order momenta. The statistics of the distortion pixels are summarized in Table 2, and reveal a distribution that is considerably heavier-tailed than the Gaussians we have employed so far.

Figure 10 contrasts the effect of that noise modeled non-parametrically after the distortions with the effect of the comparable Gaussian noise ($\lambda = 1$). Each point in the curves is one of the 125 experiments, and represents the fraction of the 100 probe images that stays in the same class as the departing — adversarial or original — image. The experiments where ordered by that value in each curve (thus the order of the experiments in the curves is not necessarily the same). Here the individual experiments are not important, but the shape of the curves: how early and how quickly they fall.

For ImageNet, the curves for the non-parametric noise (dotted lines) fall before the curves for the Gaussian noise (continuous line), showing that, indeed, the heavier tailed noise affects the images more, even without the spatial correlation. In addition, all curves fall rather sharply, showing that in almost all experiments, either all probes stay in the same label as the original, or all of them switch. Few experiments present intermediate results. Such rather bimodal behavior was already present in the curves of Figure 9. For MNIST, again, the effect is different: Gaussian and heavy-tailed noises behave much more similarly and the curves fall much more smoothly.

3.3 Conclusion

Our in-depth analysis reinforces previous claims found in the literature (GOODFELLOW *et al.*, 2015; GU; RIGAZIO, 2015): adversarial images are not necessarily isolated, spurious points: many of them inhabit relatively dense regions of the pixel space. That helps to explain why adversarial images tend to stay adversarial across classifiers of different architectures, or trained on different sets (SZEGEDY *et al.*, 2014): slightly different classification boundaries stay confounded by the dense adversarial regions.

The nature of the noise affects the resilience of both adversarial and original images. The effect is clear in ImageNet/OverFeat, where a Gaussian noise affects the images less than a heavy-tailed noise modeled after the empirical distribution of the distortions used to reach the adversarial images in the first place. An important next step in the exploration, in our view, is to understand the spatial nature of the adversarial distortions, i.e., the role spatial correlations play.

Recent works have attributed susceptibility to adversarial attacks to the linearity in the network (GOODFELLOW *et al.*, 2015), but our experiments suggest the phenomenon may be more complex. A weak, shallow, and relatively more linear classifier (logistic regression), seems no more susceptible to adversarial images than a strong, deep classifier (deep convolutional network), for the same task (MNIST). A strong deep model on a complex task seems to be more susceptible. Are adversarial images an inevitable Achilles' heel of powerful complex classifiers? Speculative analogies with the illusions of the Human Visual System are tempting, but the most honest answer is that we still know too little. Our hope is that our work will keep the conversation about adversarial images ongoing and help further explore those intriguing properties.

4 Adversarial Images for Variational Autoencoders

In this chapter we extend adversarial images to variational autoencoders. We propose an optimization — and its algorithm — to generate adversarial images for variational autoencoders. We test our proposal on two different datasets, autoencoder architectures, and compare to a version for deterministic autoencoders. This chapter in based on the paper Tabacof *et al.* (2016), presented at the Workshop on Adversarial Training at the conference on Neural Information Processing Systems (NIPS) 2016.

Adversarial attacks expressly optimize the input to "fool" models, e.g., in image classification, the adversarial input — while visually tantamount to an ordinary original image — leads to mislabeling with high confidence. In this chapter, we explore adversarial images for autoencoders — models optimized to reconstruct their inputs from compact internal representations. In an autoencoder, the attack targets not a single label, but a whole reconstruction.

We propose an adversarial attack on variational — and, for comparison, deterministic — autoencoders. Our attack aims not only at disturbing the reconstruction, but at fooling the autoencoder into reconstructing a completely different target image;

We make a comparison between attacks for autoencoders and for classifiers, showing that while the former is much harder, in both cases the amount of distortion on the input is proportional to the amount of misdirection on the output. For classifiers, however, such proportionality is hidden by the normalization of the output, which maps a linear layer into non-linear probabilities.

Evaluating generative models is hard, there are no clear-cut success criteria for autoencoder reconstruction (THEIS *et al.*, 2016), and therefore, neither for the attack. We attempt to bypass that difficulty by analyzing how inputs and outputs differ across varying regularization constants.

Gu *et al.* used autoencoders to preprocess the input and try to reinforce the network against adversarial attacks, finding that although in some cases resistance improved, attacks with small distortions remained possible (GU; RIGAZIO, 2015). A more recent trend is training adversarial *models*, in which one attempts to generate "artificial" samples (from a generative model) and the other attempts to recognize those samples (GOODFELLOW *et al.*, 2014). Makhzani *et al.* employ such scheme to train an autoencoder (MAKHZANI *et al.*, 2016).

Although autoencoders appear in the literature of adversarial images as an attempt to obtain robustness to the attacks (GU; RIGAZIO, 2015), and in the literature of adversarial training as models that can be trained with the technique (MAKHZANI *et al.*, 2016), we are unaware of any attempts to create attacks targeted to them. In the closest related literature, Sara Sabour *et al.* show that adversarial attacks can not only lead to mislabeling, but also manipulate the internal representations of the network. In this chapter, we show that an analogous manipulation allows us to attack autoencoders, but that those remain much more resistant than classifiers to such attacks (SABOUR *et al.*, 2016).

4.1 Adversarial Attack on Autoencoders

Adversarial procedures minimize an adversarial loss to mislead the model (e.g., misclassification), while distorting the input as little as possible. If the attack is successful, humans should hardly be able to distinguish between the adversarial and the regular inputs (SZEGEDY *et al.*, 2014; TABACOF; VALLE, 2016). We can be even more strict, and only allow a distortion below the input quantization noise (GOODFELLOW *et al.*, 2015; SABOUR *et al.*, 2016).

To build adversarial images for classification, one can maximize the misdirection towards a certain wrong label (SZEGEDY *et al.*, 2014; TABACOF; VALLE, 2016) or away from the correct one (GOODFELLOW *et al.*, 2015). The distortion can be minimized (SZEGEDY *et al.*, 2014; TABACOF; VALLE, 2016) or constrained to be small (GOODFELLOW *et al.*, 2015; SABOUR *et al.*, 2016). Finally, one often requires that images stay within their valid space (i.e., no pixels "below black or above white").

In autoencoders, there is not a single class output to misclassify, but instead a whole image output to scramble. The attack attempts to mislead the reconstruction: if a slightly altered image enters the autoencoder, but the reconstruction is wrecked, then the attack worked. A more dramatic attack — the one we attempt in this chapter — would be to change slightly the input image and make the autoencoder reconstruct a completely different valid image (Figure 11).

Our attack consists in selecting an original image and a target image, and then feeding the network the original image added to a small distortion, optimized to get an output as close to the target image as possible (Figure 11). Our attempts to attack the output directly failed: minimizing its distance to the target only succeeded in blurring the reconstruction. As autoencoders reconstruct from the latent representation, we can attack it instead. The latent layer is the information bottleneck of the autoencoder, and thus particularly convenient to attack. At a high level, our adversarial attack is the solution to the following optimization problem:

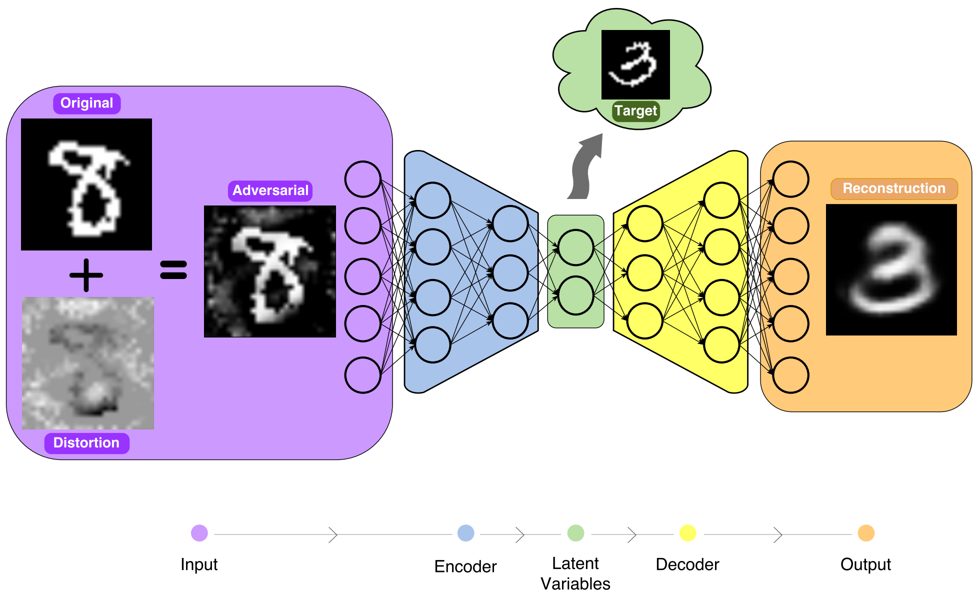


Figure 11 – Adversarial attacks for autoencoders add (ideally small) distortions to the input, aiming at making the autoencoder reconstruct a different target. We attack the latent representation, attempting to match it to the target image's.

$$\begin{aligned} \min_{\boldsymbol{d}} \quad & \Delta(\boldsymbol{z_a}, \boldsymbol{z_t}) + C\left\|\boldsymbol{d}\right\|_2^2 \\ \text{s.t.} \quad & L \leq \boldsymbol{x} + \boldsymbol{d} \leq U \\ & \boldsymbol{z_a} = \text{encoder}(\boldsymbol{x} + \boldsymbol{d}) \end{aligned} \tag{4.1}$$

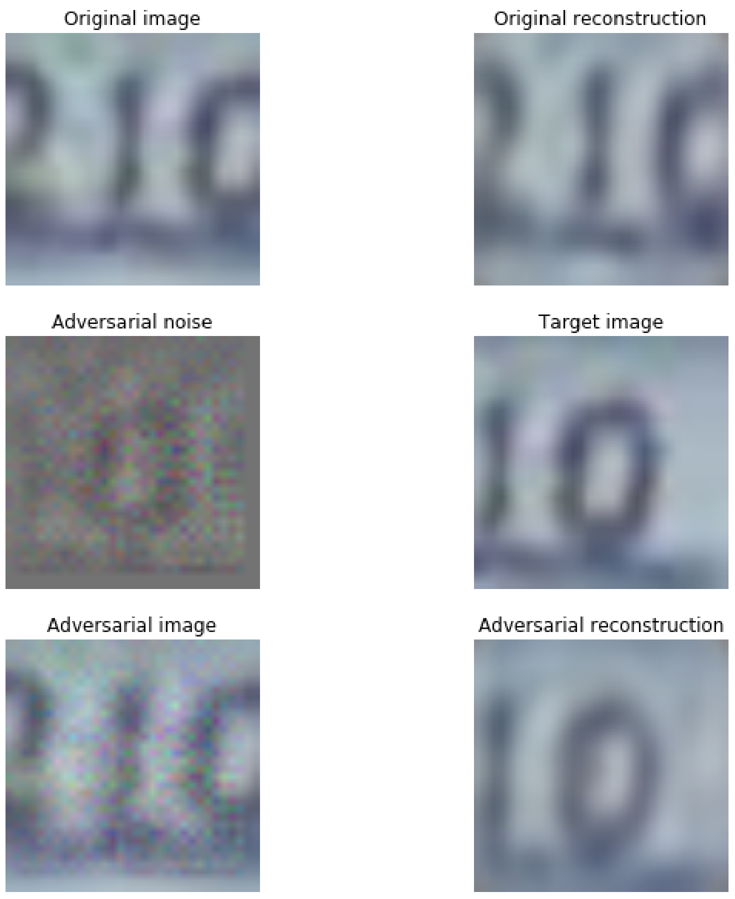
where $\boldsymbol{d}$ is the adversarial distortion; $\boldsymbol{z_a}$ and $\boldsymbol{z_t}$ are the latent representations, respectively, for the adversarial and the target images; $\boldsymbol{x}$ is the original image; $\boldsymbol{x} + \boldsymbol{d}$ is the adversarial image; L and U are the bounds on the input space; and C is the regularizing constant the balances reaching the target and limiting the distortion.

We must choose a function Δ to compare representations. For regular autoencoders a simple ℓ_2-distance sufficed; however, for variational autoencoders, the KL-divergence between the distributions induced by the latent variables not only worked better, but also offered a sounder justification. This leads to the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{d}}{\text{minimize}} \quad & KL(\boldsymbol{z}_{target} \parallel \boldsymbol{z}_{adversarial}) + C\left\|\boldsymbol{d}\right\|_2^2 \\ \text{subject to} \quad & L \leq \boldsymbol{x}_{original} + \boldsymbol{d} \leq U \\ & \boldsymbol{z}_{target} \sim \mathcal{N}(\boldsymbol{\mu}_\phi(\boldsymbol{x}_{target}), \exp(\boldsymbol{\sigma}_\phi^2(\boldsymbol{x}_{target}))) \\ & \boldsymbol{z}_{adversarial} \sim \mathcal{N}(\boldsymbol{\mu}_\phi(\boldsymbol{x}_{original} + \boldsymbol{d}), \exp(\boldsymbol{\sigma}_\phi^2(\boldsymbol{x}_{original} + \boldsymbol{d}))) \end{aligned} \tag{4.2}$$

where the $\boldsymbol{z}_*$ are uncorrelated multivariate Gaussian distributions with parameters given by the encoder. The output of the encoder is composed by two parts: one which determines the representation mean vector ($\boldsymbol{\mu}_\phi(\boldsymbol{x})$); another which determines the (diagonal) covariance matrix ($\boldsymbol{\sigma}^2_\phi(\boldsymbol{x})$). In practice, those two outputs come from the same neural network, and they differ only in the last layer, where each part is transformed by independent linear layers into the means and variances of the latent representation. ϕ are the autoencoder parameters — learned previously by training it for its ordinary task of reconstruction. During the entire adversarial procedure, ϕ remains fixed.

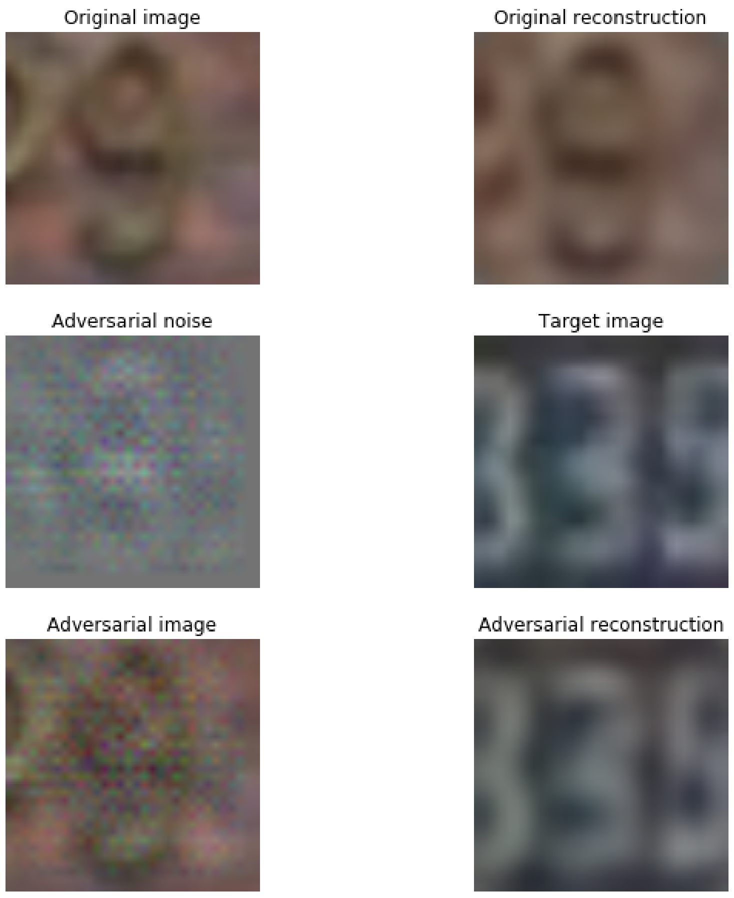
We show examples of adversarial images for variational autoencoders in Figure 12. We used a convolutional autoencoder (i.e. the encoder is a convolutional and the decoder is a deconvolutional network) trained on the Street-view House Numbers dataset (SVHN) to generate those images. The details of the procedure used to train the autoencoder and to generate those images are explained in the next section.



(a) Original image label: 1. Target image label: 0.

4.1.1 Data and Methods

We worked on the binarized MNIST (LECUN *et al.*, 1998) and SVHN datasets (NETZER *et al.*, 2011). The former allows for very fast experiments and very controlled conditions; the latter, while still allowing to manage a large number of experiments, provides much more noise and variability. Following literature (KINGMA; WELLING, 2014), we



(b) Original image label: 9. Target image label: 3.

Figure 12 – Adversarial images for a convolutional variational autoencoder trained on SVHN with 100 latent variables.

Table 3 – Fully-connected variational autoencoder for MNIST.

Layer	Size	Nonlinearity
Input	28x28	
Linear	512	ReLU
Linear	512	ReLU
Latent	20	Identity
Linear	512	ReLU
Linear	512	ReLU
Linear	784	Sigmoid
Output	28x28	

modeled pixel likelihoods as independent Bernoullis (for binary images), or as independent Gaussians (for RGB images). We used Parmesan and Lasagne (DIELEMAN *et al.*, 2015) for the implementation[1].

The loss function to train the variational autoencoder (equation 2.21) is the expectation of the likelihood under the approximated posterior plus the KL divergence

[1] The code for the experiments can be found at <https://github.com/tabacof/adv_vae>

Table 4 – Convolutional variational autoencoder for SVHN.

Layer	Size	Filter	Stride	Nonlinearity
Input	3x32x32			
Convolution	32	4x4	2	ELU
Convolution	64	4x4	2	ELU
Convolution	128	4x4	2	ELU
Linear	256			ELU
Latent	100			Identity
Linear	256			ELU
Deconvolution	128	5x5	2	ELU
Deconvolution	64	5x5	2	ELU
Deconvolution	32	5x5	2	ELU
Deconvolution	3	4x4		Identity
Output	3x32x32			

between the approximated posterior and the prior. We approximate the expectation of the likelihood with one sample of the posterior. We extract the gradients of the lower bound using automatic differentiation and maximize it using stochastic gradient ascent via the ADAM algorithm (KINGMA; BA, 2015). We used 20 and 100 latent variables for MNIST and SVHN, respectively. The prior of the latent variables are independent Gaussians, with zero mean and unitary variance. We parameterized the approximated posterior as an independent multivariate Gaussian distribution with means and variances determined by the encoder network (see equation 2.21), so the KL divergence has an analytical form (KINGMA; WELLING, 2014).

We set the encoder and decoder as fully-connected networks in the MNIST case, and as convolutional and deconvolutional (ZEILER *et al.*, 2010) networks in the SVHN case. The architectures are described in detail in tables 3 and 4[2]. After the training is done, we can use the autoencoder to reconstruct some image samples through the latent variables, which are the learned representation of the images. An example of a pair of input image/reconstructed output appears in Figure 5.

For classification tasks, the regularization term C (equation 4.1) may be chosen by bisection as the smallest constant that still leads to success (TABACOF; VALLE, 2016). Autoencoders complicate such choice, for there is no longer a binary criterion for success. Goodfellow *et al.* (GOODFELLOW *et al.*, 2015) and Sabour *et al.*(SABOUR *et al.*, 2016) optimize differently, choosing for Δ an ℓ_∞-norm constrained to make the distortion imperceptible, while maximizing the misdirection. We found such solution too restrictive, leading to reconstructions visually too distinct from the target images. Our solution was instead to forgo a single choice for C, and analyze the behavior of the system throughout a series of values. We formalize our exploration of adversarial images for

[2] Exponential Linear Unit (ELU) is an alternative to the ReLU activation function (CLEVERT *et al.*, 2015).

Algorithm 4.1 Adversarial images for variational autoencoders experimental methodology

```
Require: An original image $\boldsymbol{x}_{original}$, a target image $\boldsymbol{x}_{target}$, a set of regularization constants $\mathfrak{C}$, number of samples $N$
Ensure: $\textit{L-BFGS-B}(\boldsymbol{x}, \boldsymbol{\mu}_{target}, \boldsymbol{\sigma}_{target}, C)$ solves optimization 4.2
 1: $\boldsymbol{\mu}_{target}, \boldsymbol{\sigma}_{target} \leftarrow encoder(\boldsymbol{x}_{target})$
 2: for $C \in \mathfrak{C}$ do
 3:     $\boldsymbol{d} \leftarrow \textit{L-BFGS-B}(\boldsymbol{x}_{original}, \boldsymbol{\mu}_{target}, \boldsymbol{\sigma}_{target}, C)$
 4:     for $i = 1$ to $N$ do
 5:         $\boldsymbol{\mu}, \boldsymbol{\sigma} \leftarrow encoder(\boldsymbol{x}_{original} + \boldsymbol{d})$
 6:         $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$
 7:         $\boldsymbol{x}_{recon} \leftarrow decoder(\boldsymbol{z})$
 8:         $e[i] \leftarrow \left\| \boldsymbol{x}_{recon} - \boldsymbol{x}_{target} \right\|_2$
 9:     end for
10:     $Errors[C] \leftarrow Mean(\boldsymbol{e})$
11:     $Distortions[C] \leftarrow \boldsymbol{d}$
12: end for
13: return $Errors, Distortions$
```

variational autoencoders in algorithm 4.1.

In our experiments, we pick at random 25 pairs of original/target images (axis "experiment" in the graphs). For each pair, we span 100 different values for the regularization constant C in a logarithmic scale (from 2^{-20} to 2^{20}), measuring the ℓ_2-distance between the adversarial input and the original image (axis "distortion"), and the ℓ_2-distance between the reconstructed output and the target image (axis "adversarial−target"). The "distortion" axis is normalized between 0.0 (no attack) and the ℓ_2-distance between the original and target images in the pair (a large distortion that could reach the target directly). The "adversarial−target" is normalized between the ℓ_2-distance of the reconstruction of the target and the target (the best expected attack) and the ℓ_2-distance of the reconstruction of the original and the target (the worst expected attack). The geometry of such normalization is illustrated by the colored lines in the graphs of Figure 13. For variational autoencoders, the reconstruction is stochastic: therefore, each data point is sampled 100 times, and the average is reported.

For comparison purposes, we use the same protocol above to generate a range of adversarial images for the usual classification tasks on the same datasets. The aim is to contrast the behavior of adversarial attacks across the two tasks (autoencoding / classification). In those experiments we pick pairs of original image / adversarial class (axis "experiment"), and varying C (from 2^{-10} to 2^{20}), we measure the distortion as above, and the probability (with corresponding logit) attributed to the adversarial (red lines) and to the original classes (blue lines). The axes here are no longer normalized, but we center at 0 in the "distortion" axis the transition point between attack failure and success — the point where red and blue lines cross.

4.1.2 Results and Discussion

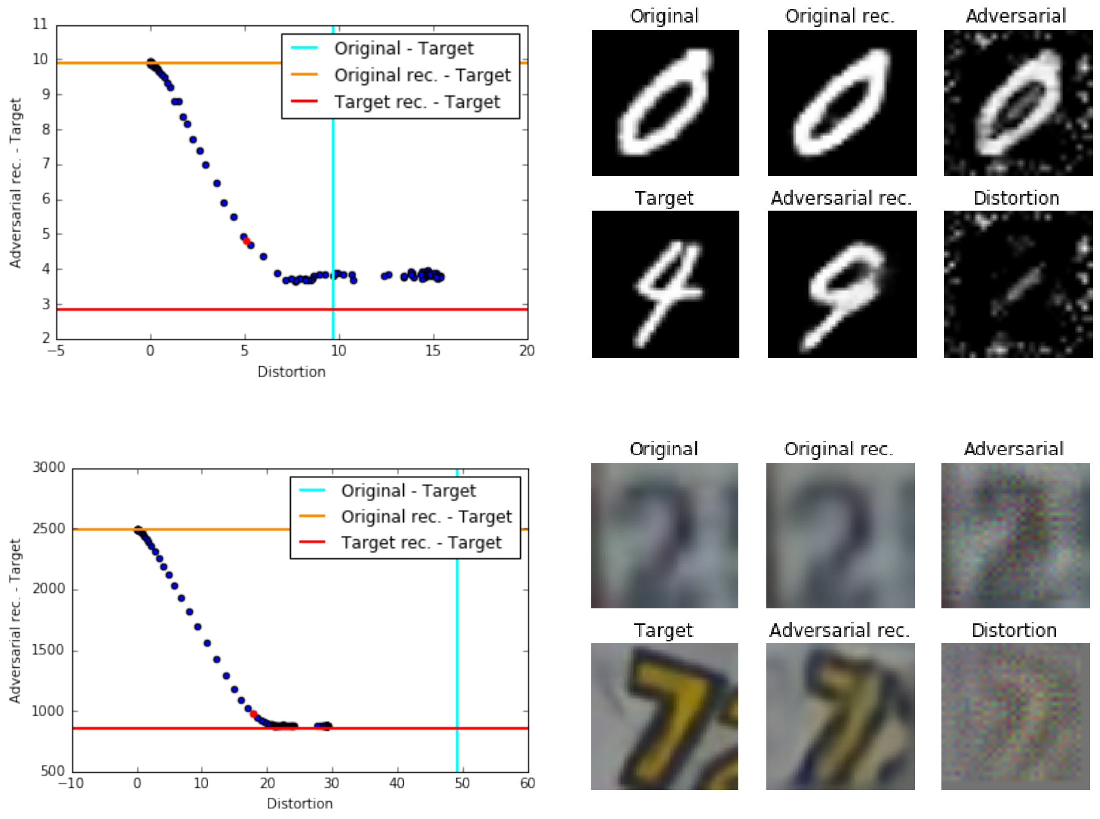


Figure 13 – Top row: MNIST. Bottom row: SVHN. The figures on the left show the trade-off between the quality of adversarial attack and the adversarial distortion magnitude, with changing regularization parameter (implicit in the graphs, chosen from a logarithmic scale). The figures on the right correspond to the points shown in red in the graphs, illustrating adversarial images and reconstructions using fully-connected, and convolutional variational autoencoders (for MNIST and SVHN, respectively).

We found that generating adversarial images for autoencoders is a much harder task than for classifiers. If we apply little distortion (comparable to those used for misleading classifiers), the reconstructions stay essentially untouched. To get reconstructions very close to the target's, we have to apply heavy distortions to the input. However, by hand-tuning the regularization parameter, it is possible to find trade-offs where the reconstruction approaches the target's and the adversarial image will still resemble the input (two examples in Figure 13).

The plots for the full set of 25 original/target image pairs appear in Figure 14. All series saturate when the latent representation of the adversarial image essentially equals the target's. That saturation appears well before the upper distortion limit of 1.0, and provides a measure of how resistant the model is to the attack: Variational Autoencoders appear slightly more resistant than Deterministic Autoencoders, and MNIST much more resistant than SVHN. The latter is not surprising, since large complex models seem, in general, more susceptible to adversarial attacks. Before the "hinge" where the attack saturates, there is a quasi-linear trade-off between input distortion and output similarity to target, for all combinations of dataset and autoencoder choice. We were initially hoping for a more non-linear behavior, with a sudden drop at some point in the scale, but data suggests that there is a give-and-take for attacking autoencoders: each gain in the attack requires a *proportional* increase in distortion.

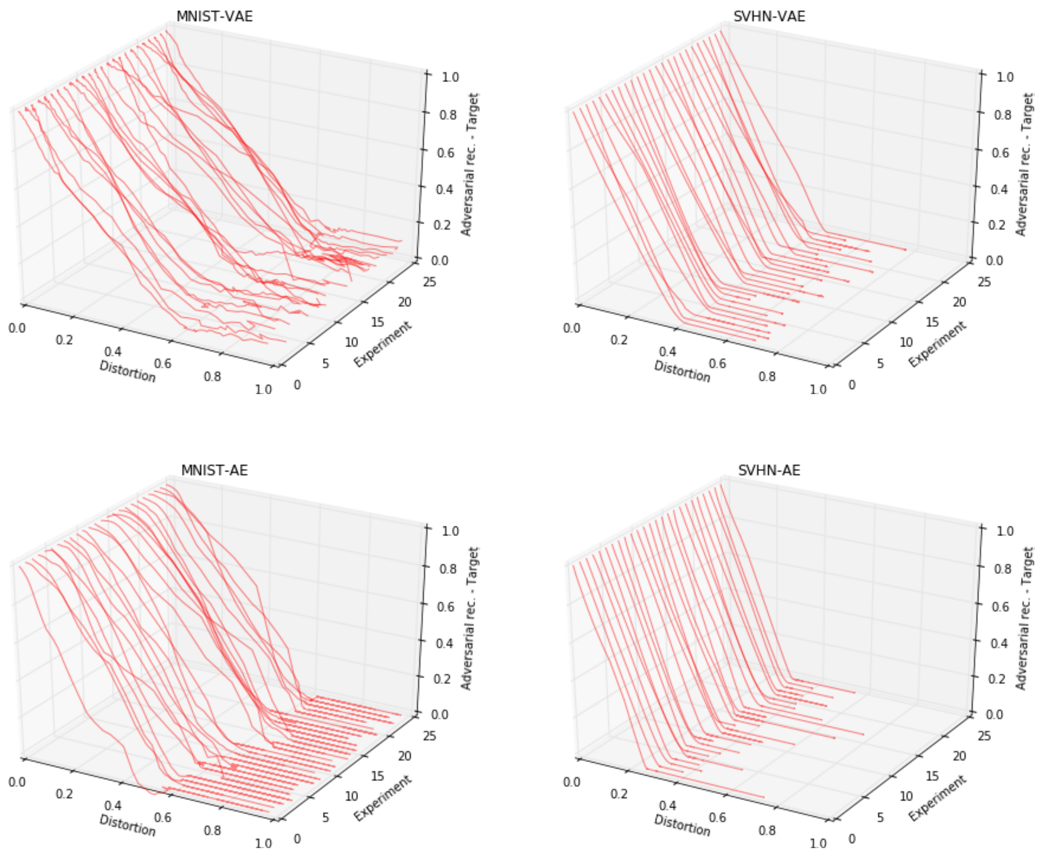


Figure 14 – Plots for the whole set of experiments in MNIST and SVHN. Top: variational autoencoders (VAE). Bottom: deterministic autoencoders (AE). Each line in a graph corresponds to one experiment with adversarial images from a single pair of original/target images, varying the regularization parameter C (like shown in Figure 13). The "distortion" and "adversarial−target" axes show the trade-off between cost and success. The "hinge" where the lines saturate show the point where the reconstruction is essentially equal to the target's: the distortion at the hinge measures the resistance to the attack.

The comparison with the (much better-studied) attacks for classifiers, showed, at the beginning, a much different behavior: when we contrasted the probability attributed to the adversarial class *vs.* the distortion imposed on the input, we observed the non-linear, sudden change we were expecting (left column of Figure 16). The question remained, however whether such non-linearity was intrinsic, or whether it was due to the highly non-linear nature of the probability scale. The answer appears in the right column of Figure 16, where, with a logit transformation of the probabilities, the linear behavior appears again. It seems that the attack on classifiers show, internally, the same linear give-and-take present in autoencoders, but that the normalization of the outputs of the last layer into valid probabilities aids the attack: changes in input lead to proportional changes in logit, but to much larger changes in probability. That makes feasible for the attack on classifiers to find much better sweet spots than the attack on autoencoders (Figure 15). Goodfellow *et al.* (GOODFELLOW *et al.*, 2015) suggested that the linearity of deep models make them susceptible to adversarial attacks. Our results seems to reinforce that such linearity plays indeed a critical role, with "internal" success of the attack being proportional to the distortion on inputs. On classification networks, however, which are essentially piecewise linear until the last layer, the non-linearity of the latter seems to compound the problem.

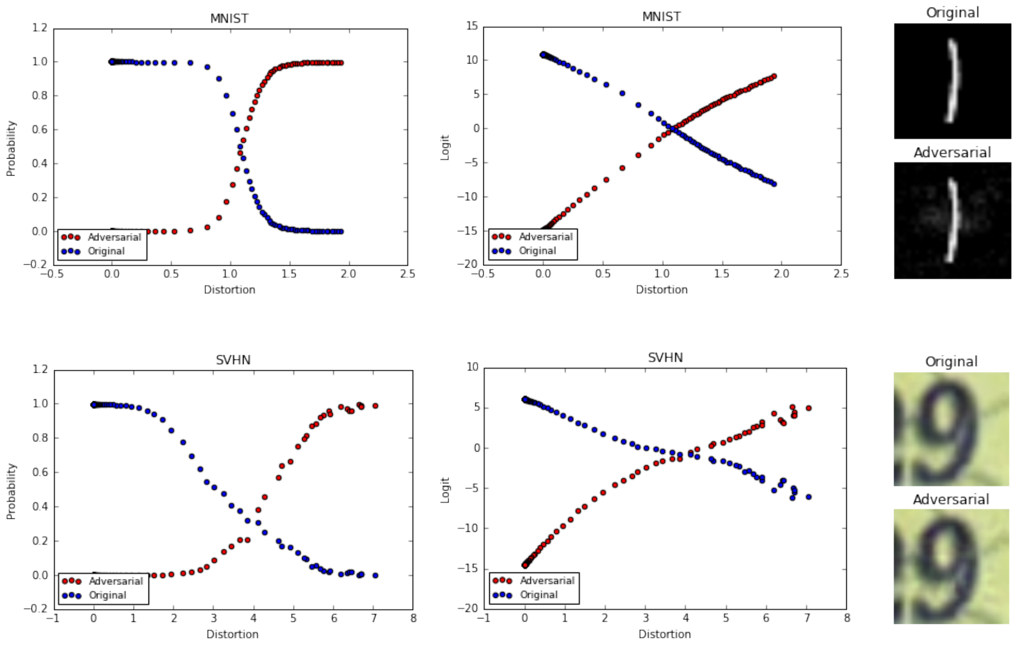


Figure 15 – Examples for the classification attacks. Top: MNIST. Bottom: SVHN. Left: probabilities. Middle: logit transform of probabilities. Right: images illustrating the intersection point of the curves. The adversarial class is '4' for MNIST, and '0' for SVHN. The red curve shows the probability/logit for the adversarial class, and the blue curve shows the same for the original class: the point where the curves cross is the transition point between failure and success of the attack.

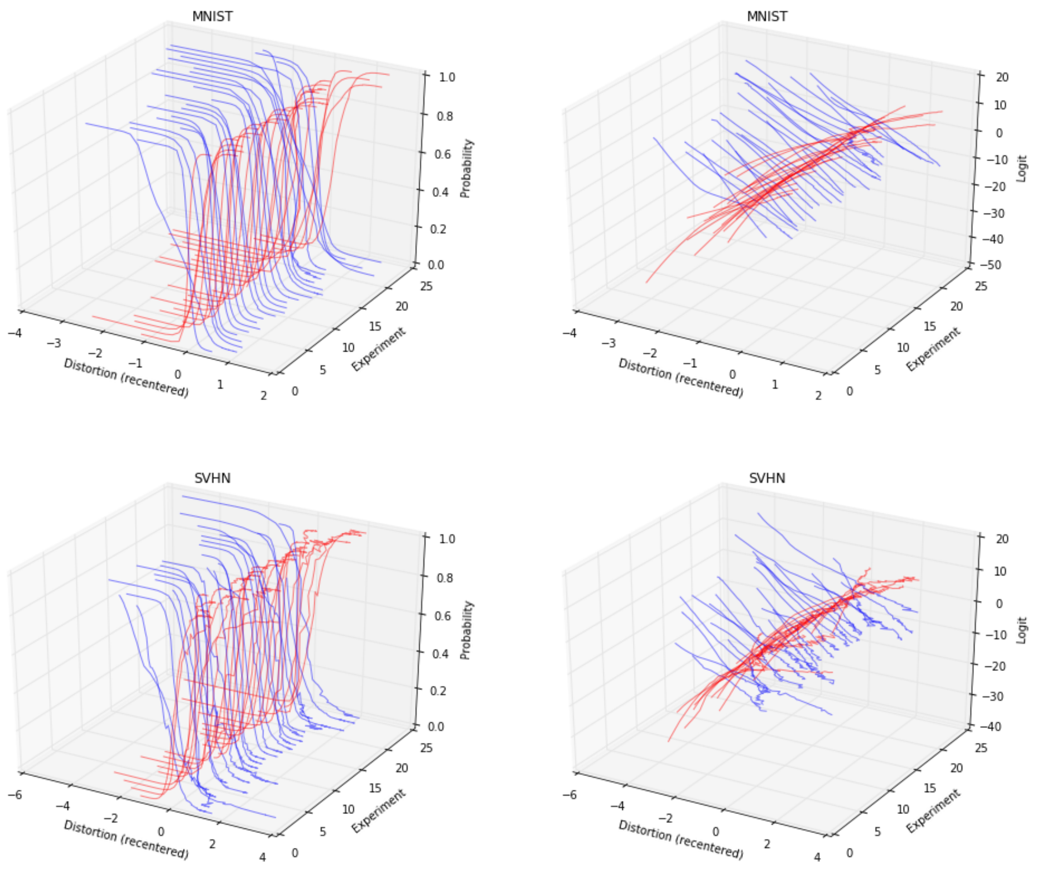


Figure 16 – Plot of whose set of experiments for classifiers. Top: MNIST. Bottom: SVHN. Left: probabilities. Right: logit transform of probabilities. Each experiment corresponds to one of the graphs shown in Figure 15, centered to make the crossing point between the red and blue lines stay at 0 in the "distortion" axis.

4.2 Conclusion

We proposed an adversarial method to attack autoencoders, and evaluated their robustness to such attacks. We showed that there is a linear trade-off between how much the adversarial input is similar to the original input, and how much the adversarial reconstruction is similar to the target reconstruction — frustrating the hope that a small change in the input could lead to drastic changes in the reconstruction. Surprisingly, such linear trade-off also appears for adversarial attacks on classification networks, if we "undo" the non-linearity of the last layer. In the future, we intend to extend our empirical results to datasets with larger inputs and more complex networks (e.g. ImageNet) — as well as to different autoencoder architectures. For example, the DRAW variational autoencoder (GREGOR *et al.*, 2015) uses feedback from the reconstruction error to improve the reconstruction — and thus could be more robust to attacks. To arrive at the state of the art of variational autoencoders, we need to add a differentiable attention mechanism (GREGOR *et al.*, 2015) or convolutional LSTMs (GREGOR *et al.*, 2016) to the recurrent autoencoder. One possible research direction is adding attention to the latent variables themselves, instead of applying it to the input and output, as it is usually done. We are also interested in advancing theoretical explanations to illuminate our results.

Besides adversarial attacks, we are interested in understanding the latent representation created by the autoencoders. One possibility is to generate a synthetic dataset where you control some aspects of the image (e.g. position, size, shape, etc) and then correlate it with the learned representation. Another possibility is to use the synthetic dataset to create images by interpolation and analogy and verify whether latent variable arithmetic can accomplish the same task.

Conclusion

We have explored the intriguing subject of adversarial attacks in two different contexts — their classical setting of classification, and in a novel setting of autoencoders. Although autoencoders appear elsewhere in the literature of adversarial images, as far as we know, we were the first to attempt an adversarial attack on them. We have also reviewed neural networks in machine learning, the recent developments of deep learning, and covered in greater depth the recent literature on adversarial images and variational autoencoders.

Adversarial images have taken the community aback, and cast doubts on the successes of deep learning. Literature abounds with strategies attempting to immunize deep neural networks from them, but it seems that the more powerful and trainable a model is, the more susceptible it will be to adversarial attacks. Our own brains are plagued by optical illusions and cognitive biases, and even knowing that they exist is not enough to prevent the brain from being fooled. That suggests such flaws may be inherent to complex decision systems. Nevertheless, neural networks are products of our design — unlike the brain, which was shaped by millions of years by evolution — so we must stride to mitigate flaws like adversarial images. Some strategies, such as adversarial training, had partial success, with additional benefits such as improved generalization. That indicates we should keep probing neural networks, and find novel ways to prevent or alleviate the issue.

Variational autoencoders are part of a recent wave of Bayesian deep learning and deep generative models that have achieved state-of-the-art results in semi-supervised learning, image compression, and reconstruction. A competing family of generative models, Generative Adversarial Networks (GAN), has attracted much attention the past couple of years and has improved the results of VAEs in some problems, such as realistic image generation (GOODFELLOW *et al.*, 2014; RADFORD *et al.*, 2016). The nomenclature is confusing: GANs do not use adversarial images in training, but rather two networks *against each other*, one generative to create artificial samples, and one discriminator that classifies images into "real" (coming from the dataset) or "fake" (generated by the first network). GANs are otherwise unrelated to classical adversarial images. There are attempts to merge both families of generative models, VAEs and GANs, into a more complete probabilistic model (MAKHZANI *et al.*, 2016; MESCHEDER *et al.*, 2017). Perhaps such models will be more resistant to our adversarial attack, but the experience in classical adversarial images indicates that to be unlikely.

We showed that it is possible to fool autoencoders in similar ways a classifier is fooled. It is, however, much harder to do so, as there is no binary decision on whether an autoencoder was fooled. One important result shows that there is quasi-linear trade-off between input noise and reconstruction error (and that the nonlinear part is simply the saturation that happens when the fooling is as good as possible). That is fortunate as it implies there is no free lunch: the more you want to fool an autoencoder, the higher the price in distortion you have to pay. This is unlike optical illusions, where a small change of shape or color may unleash the illusion. With that result, we went back to the classical adversarial images for classifiers, and showed that their nonlinear behavior is in large part due those neural networks being probabilistic classifiers. A linear change in the input of a sigmoidal / softmax function will bring a nonlinear change to the probabilistic output. That reinforces the idea that adversarial susceptibility comes from the linearity, rather than the nonlinearity, of deep neural networks, as presented by Goodfellow *et al.* (GOODFELLOW *et al.*, 2015).

An interesting question for future explorations is on the slope of the noise-error trade-off. Are there different slopes for different models and attackers? What makes the trade-off slope increase or decrease? Ideally, we would want slopes to be as flat as possible, forcing the attacker to exert great effort in all situations.

Another line of inquiry comes from the Bayesian perspective. Why are Bayesian neural networks and variational autoencoders susceptible to adversarial images? Intuitively, the noise that is injected by Bayesian procedures should make it harder to fool such models, which has been found to be the case (LI; GAL, 2017; LOUIZOS; WELLING, 2017). Despite that work, we have preliminary experiments showing that Bayesian neural networks can be attacked given an optimization objective that takes its uncertainty into account. We have published a workshop paper on evaluating the uncertainty quality of Bayesian neural networks, but in that paper we did not consider the adversarial problem. As suggested by Li & Gal (2017), Louizos & Welling (2017), we can use adversarial attacks as yet another measure of uncertainty quality, which we intend to explore further.

We intend to keep pursuing the question of how a deep neural network with internal noise, be it from the the weight uncertainty or from the latent variables, can be attacked by small distortions in the input, and what can de done to thwart such attacks.

Bibliography

BASTIEN, F.; LAMBLIN, P.; PASCANU, R.; BERGSTRA, J.; GOODFELLOW, I.; BERGERON, A.; BOUCHARD, N.; WARDE-FARLEY, D.; BENGIO, Y. Theano: new features and speed improvements. In: *NIPS Deep Learning Workshop*. 2012. Cited in page 28.

BENGIO, Y. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, Now Publishers Inc., v. 2, n. 1, p. 1–127, 2009. Cited 2 times in pages 40 and 44.

BI, R. *Does Deep Learning Have Deep Flaws?* 2014. Accessed: 2015-09-08. Available at: <http://www.kdnuggets.com/2014/06/deep-learning-deep-flaws.html>. Cited in page 33.

BISHOP, C. M. *Pattern recognition and machine learning.* : Springer, 2007. (Information science and statistics). ISBN 9780387310732. Cited 2 times in pages 21 and 32.

BLUM, A.; HAGHTALAB, N.; PROCACCIA, A. D. Variational dropout and the local reparameterization trick. In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada.* 2015. p. 2575–2583. Cited 2 times in pages 32 and 36.

BLUNDELL, C.; CORNEBISE, J.; KAVUKCUOGLU, K.; WIERSTRA, D. Weight uncertainty in neural network. In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015.* 2015. p. 1613–1622. Cited in page 32.

BOYD, S.; VANDENBERGHE, L. *Convex optimization.* : Cambridge university press, 2004. Cited 3 times in pages 24, 41, and 45.

BRENDEL, W.; BETHGE, M. Comment on "Biologically inspired protection of deep networks from adversarial attacks". *arXiv preprint arXiv:1704.01547*, 2017. Available at: <http://arxiv.org/abs/1704.01547>. Cited 2 times in pages 17 and 35.

BURDEN, R. L.; FAIRES, J. D. The bisection algorithm. *Numerical Analysis. Prindle, Weber & Schmidt, Boston, MA., pp. x*, v. 676, 1985. Cited in page 43.

CARLINI, N.; WAGNER, D. Adversarial examples are not easily detected: Bypassing ten detection methods. *arXiv preprint arXiv:1705.07263*, 2017. Available at: <http://arxiv.org/abs/1705.07263>. Cited in page 17.

CHOROMANSKA, A.; HENAFF, M.; MATHIEU, M.; AROUS, G. B.; LECUN, Y. The loss surfaces of multilayer networks. In: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015.* 2015. Cited in page 29.

CLEVERT, D.-A.; UNTERTHINER, T.; HOCHREITER, S. Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv preprint arXiv:1511.07289*, 2015. Available at: <https://arxiv.org/abs/1511.07289>. Cited in page 58.

COLLOBERT, R.; KAVUKCUOGLU, K.; FARABET, C. Torch7: A matlab-like environment for machine learning. In: *BigLearn, NIPS Workshop*. 2011. Cited 2 times in pages 43 and 45.

CRAWFORD, K. Artificial intelligence's white guy problem. *The New York Times*, 2016. Cited in page 17.

DAUPHIN, Y. N.; PASCANU, R.; GULCEHRE, C.; CHO, K.; GANGULI, S.; BENGIO, Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In: *Advances in neural information processing systems*. 2014. p. 2933–2941. Cited in page 29.

DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. Imagenet: A large-scale hierarchical image database. In: IEEE. *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. 2009. p. 248–255. Cited 2 times in pages 29 and 43.

DIELEMAN, S.; SCHLÜTER, J.; RAFFEL, C.; OLSON, E.; SØNDERBY, S. K.; NOURI, D.; MATURANA, D.; THOMA, M.; BATTENBERG, E.; KELLY, J. *et al. Lasagne: First release.* 2015. Cited in page 57.

DILOKTHANAKUL, N.; MEDIANO, P. A.; GARNELO, M.; LEE, M. C.; SALIMBENI, H.; ARULKUMARAN, K.; SHANAHAN, M. Deep unsupervised clustering with Gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*, 2016. Available at: <http://arxiv.org/abs/1611.02648>. Cited in page 36.

DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, JMLR. org, v. 12, p. 2121–2159, 2011. Cited in page 31.

EFRON, B.; TIBSHIRANI, R. J. *An introduction to the bootstrap.* : CRC press, 1994. Cited in page 47.

ESLAMI, S. M. A.; HEESS, N.; WEBER, T.; TASSA, Y.; SZEPESVARI, D.; KAVUKCUOGLU, K.; HINTON, G. E. Attend, infer, repeat: Fast scene understanding with generative models. In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain.* 2016. p. 3225–3233. Cited in page 36.

FAWZI, A.; MOOSAVI-DEZFOOLI, S.; FROSSARD, P. Robustness of classifiers: from adversarial to random noise. In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain.* 2016. p. 1624–1632. Cited 2 times in pages 33 and 47.

FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. *The elements of statistical learning.* : Springer series in statistics Springer, Berlin, 2001. v. 1. Cited in page 24.

GAL, Y.; GHAHRAMANI, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016.* 2016. p. 1050–1059. Cited 2 times in pages 30 and 32.

GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In: *International conference on artificial intelligence and statistics*. 2010. p. 249–256. Cited in page 30.

GODOY, A.; TABACOF, P.; VON ZUBEN, F. J. The role of the interaction network in the emergence of diversity of behavior. *PloS one*, Public Library of Science, v. 12, n. 2, p. e0172073, 2017. Cited in page 19.

GOODFELLOW, I.; POUGET-ABADIE, J.; MIRZA, M.; XU, B.; WARDE-FARLEY, D.; OZAIR, S.; COURVILLE, A.; BENGIO, Y. Generative adversarial nets. In: *Advances in Neural Information Processing Systems*. 2014. p. 2672–2680. Cited 2 times in pages 53 and 65.

GOODFELLOW, I. J.; SHLENS, J.; SZEGEDY, C. Explaining and harnessing adversarial examples. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2015. arXiv:1412.6572. Cited 9 times in pages 17, 33, 40, 51, 52, 54, 58, 62, and 66.

GRAVES, A.; MOHAMED, A.; HINTON, G. E. Speech recognition with deep recurrent neural networks. In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*. 2013. p. 6645–6649. Cited in page 16.

GREGOR, K.; BESSE, F.; REZENDE, D. J.; DANIHELKA, I.; WIERSTRA, D. Towards conceptual compression. In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. 2016. p. 3549–3557. Cited 3 times in pages 16, 38, and 64.

GREGOR, K.; DANIHELKA, I.; GRAVES, A.; REZENDE, D. J.; WIERSTRA, D. DRAW: A recurrent neural network for image generation. In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. 2015. p. 1462–1471. Cited 3 times in pages 37, 63, and 64.

GU, S.; RIGAZIO, L. Towards deep neural network architectures robust to adversarial examples. In: *International Conference on Learning Representations (ICLR) Workshop*. 2015. arXiv:1412.5068. Available at: <http://arxiv.org/abs/1412.5068>. Cited 5 times in pages 17, 40, 51, 53, and 54.

HAWKINS, J.; BLAKESLEE, S. *On intelligence*. : Macmillan, 2007. Cited in page 21.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016. p. 770–778. Cited in page 31.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997. Cited 2 times in pages 31 and 37.

HUANG, R.; XU, B.; SCHUURMANS, D.; SZEPESVÁRI, C. Learning with a strong adversary. *arXiv preprint arXiv:1511.03034*, 2015. Available at: <http://arxiv.org/abs/1511.03034>. Cited in page 35.

HUANG, S.; PAPERNOT, N.; GOODFELLOW, I.; DUAN, Y.; ABBEEL, P. Adversarial attacks on neural network policies. In: *International Conference on Learning Representations (ICLR) Workshop*. 2017. arXiv:1702.02284. Available at: <http://arxiv.org/abs/1702.02284>. Cited in page 33.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. 2015. p. 448–456. Cited in page 30.

JAMES, M. *The Flaw In Every Neural Network Just Got A Little Worse*. 2015. Accessed: 2016-05-31. Available at: <http://www.i-programmer.info/news/105-artificial-intelligence/9090-the-flaw-in-every-neural-network-just-got-a-little-worse.html>. Cited in page 19.

JORDAN, M. I. *et al. Why the logistic function? A tutorial discussion on probabilities and neural networks*. : Computational Cognitive Science Technical Report, 1995. Cited in page 26.

KARPATHY, A. *A Peek at Trends in Machine Learning*. 2017. <https://medium.com/@karpathy/a-peek-at-trends-in-machine-learning-ab8a1085a106>. Cited in page 32.

KAWAGUCHI, K. Deep learning without poor local minima. In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. 2016. p. 586–594. Cited in page 29.

KENDALL, A.; GAL, Y. What uncertainties do we need in Bayesian deep learning for computer vision? *arXiv preprint arXiv:1703.04977*, 2017. Available at: <http://arxiv.org/abs/1703.04977>. Cited 2 times in pages 17 and 32.

KINGMA, D.; BA, J. Adam: A method for stochastic optimization. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2015. arXiv:1412.6980. Available at: <http://arxiv.org/abs/1412.6980>. Cited 3 times in pages 31, 32, and 58.

KINGMA, D. P.; MOHAMED, S.; REZENDE, D. J.; WELLING, M. Semi-supervised learning with deep generative models. In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. 2014. p. 3581–3589. Cited in page 36.

KINGMA, D. P.; WELLING, M. Auto-encoding variational Bayes. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2014. arXiv:1312.6114. Available at: <http://arxiv.org/abs/1312.6114>. Cited 6 times in pages 17, 35, 36, 37, 56, and 58.

KOCH, C. *The Quest for Consciousness: A Neurobiological Approach*. : {Roberts & Company Publishers}, 2006. Cited in page 16.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. 2012. p. 1097–1105. Cited in page 45.

KURAKIN, A.; GOODFELLOW, I.; BENGIO, S. Adversarial examples in the physical world. In: *International Conference on Learning Representations (ICLR) Workshop*. 2017. arXiv:1607.02533. Available at: <http://arxiv.org/abs/1607.02533>. Cited in page 17.

LECUN, Y.; BENGIO, Y. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, v. 3361, n. 10, p. 1995, 1995. Cited in page 31.

LECUN, Y.; BENGIO, Y.; HINTON, G. E. Deep learning. *Nature*, v. 521, n. 7553, p. 436–444, 2015. Cited in page 16.

LECUN, Y.; CORTES, C.; BURGES, C. J. *The MNIST database of handwritten digits*. 1998. Cited 2 times in pages 42 and 56.

LECUN, Y. A.; BOTTOU, L.; ORR, G. B.; MÜLLER, K.-R. Efficient backprop. In: *Neural networks: Tricks of the trade*. : Springer, 2012. p. 9–48. Cited in page 29.

LEGG, S. *Machine super intelligence*. Tese (Doutorado) — University of Lugano, 2008. Cited in page 21.

LI, Y.; GAL, Y. Dropout inference in bayesian neural networks with alpha-divergences. *arXiv preprint arXiv:1703.02914*, 2017. Available at: <http://arxiv.org/abs/1703.02914>. Cited 3 times in pages 33, 35, and 66.

LOUIZOS, C.; WELLING, M. Multiplicative normalizing flows for variational Bayesian neural networks. *arXiv preprint arXiv:1703.01961*, 2017. Available at: <http://arxiv.org/abs/1703.01961>. Cited 2 times in pages 33 and 66.

MACLAURIN, D.; DUVENAUD, D.; ADAMS, R. P. Early stopping is nonparametric variational inference. In: *NIPS 2015 Advances in Approximate Bayesian Inference*. 2015. Cited in page 32.

MAKHZANI, A.; SHLENS, J.; JAITLY, N.; GOODFELLOW, I.; FREY, B. Adversarial autoencoders. In: *International Conference on Learning Representations (ICLR) Workshop*. 2016. arXiv:1511.05644. Available at: <http://arxiv.org/abs/1511.05644>. Cited 3 times in pages 53, 54, and 65.

MARTENS, J.; SUTSKEVER, I. Learning recurrent neural networks with hessian-free optimization. In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 2011. p. 1033–1040. Cited in page 31.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943. Cited in page 27.

MESCHEDER, L.; NOWOZIN, S.; GEIGER, A. Adversarial variational Bayes: Unifying variational autoencoders and generative adversarial networks. *arXiv preprint arXiv:1701.04722*, 2017. Available at: <http://arxiv.org/abs/1701.04722>. Cited in page 65.

MIYATO, T.; MAEDA, S.-i.; KOYAMA, M.; NAKAE, K.; ISHII, S. Distributional smoothing with virtual adversarial training. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2016. arXiv:1507.00677. Available at: <http://arxiv.org/abs/1507.00677>. Cited in page 35.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted Boltzmann machines. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 2010. p. 807–814. Cited in page 30.

NAYEBI, A.; GANGULI, S. Biologically inspired protection of deep networks from adversarial attacks. *arXiv preprint arXiv:1703.09202*, 2017. Available at: <http://arxiv.org/abs/1703.09202>. Cited in page 35.

NEAL, R. M. *et al.* MCMC using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, v. 2, p. 113–162, 2011. Cited in page 32.

NETZER, Y.; WANG, T.; COATES, A.; BISSACCO, A.; WU, B.; NG, A. Y. Reading digits in natural images with unsupervised feature learning. In: *NIPS 2011 Workshop on Deep Learning and Unsupervised Feature Learning*. 2011. Cited in page 56.

NG, A. Sparse autoencoder. *CS294A Lecture notes*, v. 72, p. 1–19, 2011. Cited in page 35.

NGUYEN, A. M.; YOSINSKI, J.; CLUNE, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. 2015. p. 427–436. Cited in page 33.

NOCEDAL, J.; WRIGHT, S. *Numerical optimization.* : Springer Science & Business Media, 2006. Cited in page 42.

NØKLAND, A. Improving back-propagation by adding an adversarial gradient. *arXiv preprint arXiv:1510.04189*, 2015. Available at: <http://arxiv.org/abs/1510.04189>. Cited in page 33.

OLIVEIRA, R.; TABACOF, P.; VALLE, E. Known unknowns: Uncertainty quality in Bayesian neural networks. In: *NIPS 2016 Bayesian Deep Learning Workshop*. 2016. arXiv:1612.01251. Available at: <http://arxiv.org/abs/1612.01251>. Cited in page 19.

PAPERNOT, N.; MCDANIEL, P. D.; GOODFELLOW, I. J.; JHA, S.; CELIK, Z. B.; SWAMI, A. Practical black-box attacks against machine learning. In: *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*. 2017. p. 506–519. Cited 2 times in pages 17 and 33.

PAPERNOT, N.; MCDANIEL, P. D.; WU, X.; JHA, S.; SWAMI, A. Distillation as a defense to adversarial perturbations against deep neural networks. In: *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*. 2016. p. 582–597. Cited in page 35.

PARK, T.; CASELLA, G. The Bayesian Lasso. *Journal of the American Statistical Association*, Taylor & Francis, v. 103, n. 482, p. 681–686, 2008. Cited in page 25.

PASCANU, R.; MIKOLOV, T.; BENGIO, Y. On the difficulty of training recurrent neural networks. In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*. 2013. p. 1310–1318. Cited in page 31.

RADFORD, A.; METZ, L.; CHINTALA, S. Unsupervised representation learning with deep convolutional generative adversarial networks. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2016. arXiv:1511.06434. Available at: <http://arxiv.org/abs/1511.06434>. Cited 3 times in pages 37, 38, and 65.

REZENDE, D. J.; MOHAMED, S.; WIERSTRA, D. Stochastic backpropagation and approximate inference in deep generative models. In: *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*. 2014. p. 1278–1286. Cited in page 38.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958. Cited in page 27.

RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATHY, A.; KHOSLA, A.; BERNSTEIN, M. S.; BERG, A. C.; LI, F. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, v. 115, n. 3, p. 211–252, 2015. Cited in page 16.

SABOUR, S.; CAO, Y.; FAGHRI, F.; FLEET, D. J. Adversarial manipulation of deep representations. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2016. arXiv:1511.05122. Available at: <http://arxiv.org/abs/1511.05122>. Cited 3 times in pages 33, 54, and 58.

SANTOS, E. P. dos; VON ZUBEN, F. J. Efficient second-order learning algorithms for discrete-time recurrent neural networks. In: *Recurrent neural networks: design and applications*. : CRC Press, 1999. Cited in page 31.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural Networks*, Elsevier, v. 61, p. 85–117, 2015. Cited 2 times in pages 28 and 29.

SERMANET, P.; EIGEN, D.; ZHANG, X.; MATHIEU, M.; FERGUS, R.; LECUN, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. Available at: <http://arxiv.org/abs/1312.6229>. Cited 2 times in pages 42 and 45.

SILVER, D.; HUANG, A.; MADDISON, C. J.; GUEZ, A.; SIFRE, L.; DRIESSCHE, G. van den; SCHRITTWIESER, J.; ANTONOGLOU, I.; PANNEERSHELVAM, V.; LANCTOT, M.; DIELEMAN, S.; GREWE, D.; NHAM, J.; KALCHBRENNER, N.; SUTSKEVER, I.; LILLICRAP, T. P.; LEACH, M.; KAVUKCUOGLU, K.; GRAEPEL, T.; HASSABIS, D. Mastering the game of Go with deep neural networks and tree search. *Nature*, v. 529, n. 7587, p. 484–489, 2016. Cited in page 16.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2015. arXiv:1409.1556. Available at: <http://arxiv.org/abs/1409.1556>. Cited in page 29.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUT-DINOV, R. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014. Cited in page 30.

SUTSKEVER, I.; MARTENS, J.; DAHL, G.; HINTON, G. On the importance of initialization and momentum in deep learning. In: *International conference on machine learning*. 2013. p. 1139–1147. Cited in page 31.

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. Sequence to sequence learning with neural networks. In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. 2014. p. 3104–3112. Cited in page 16.

SZEGEDY, C.; ZAREMBA, W.; SUTSKEVER, I.; BRUNA, J.; ERHAN, D.; GOODFELLOW, I.; FERGUS, R. Intriguing properties of neural networks. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2014. arXiv:1312.6199. Available at: <http://arxiv.org/abs/1312.6199>. Cited 10 times in pages 16, 17, 33, 40, 41, 42, 44, 45, 51, and 54.

TABACOF, P.; TAVARES, J.; VALLE, E. Adversarial images for variational autoencoders. In: *NIPS 2016 Workshop on Adversarial Training*. 2016. arXiv:1612.00155. Available at: <http://arxiv.org/abs/1612.00155>. Cited 3 times in pages 19, 20, and 53.

TABACOF, P.; VALLE, E. Exploring the space of adversarial images. In: *2016 International Joint Conference on Neural Networks, IJCNN 2016, Vancouver, BC, Canada, July 24-29, 2016*. 2016. p. 426–433. Cited 5 times in pages 19, 20, 40, 54, and 58.

THEIS, L.; OORD, A. v. d.; BETHGE, M. A note on the evaluation of generative models. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2016. arXiv:1511.01844. Available at: <http://arxiv.org/abs/1511.01844>. Cited in page 53.

TIBSHIRANI, R. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, JSTOR, p. 267–288, 1996. Cited in page 25.

TIPPING, M. E.; BISHOP, C. M. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Wiley Online Library, v. 61, n. 3, p. 611–622, 1999. Cited in page 27.

TRAMÈR, F.; PAPERNOT, N.; GOODFELLOW, I.; BONEH, D.; MCDANIEL, P. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017. Available at: <http://arxiv.org/abs/1704.03453>. Cited 2 times in pages 17 and 33.

TRAMÈR, F.; ZHANG, F.; JUELS, A.; REITER, M. K.; RISTENPART, T. Stealing machine learning models via prediction APIs. In: *USENIX Security*. 2016. Cited in page 33.

TURING, A. M. Computing machinery and intelligence. *Mind*, JSTOR, v. 59, n. 236, p. 433–460, 1950. Cited in page 21.

VAUGHAN, J.; WALLACH, H. *The Inescapability of Uncertainty: AI, Uncertainty, and Why You Should Vote No Matter What Predictions Say.* 2016. <https://medium.com/@jennwv/uncertainty-edd5caf8981b>. Cited in page 17.

VINCENT, P.; LAROCHELLE, H.; LAJOIE, I.; BENGIO, Y.; MANZAGOL, P.-A. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, JMLR. org, v. 11, p. 3371–3408, 2010. Cited in page 35.

WATTER, M.; SPRINGENBERG, J.; BOEDECKER, J.; RIEDMILLER, M. Embed to control: A locally linear latent dynamics model for control from raw images. In: *Advances in Neural Information Processing Systems.* 2015. p. 2746–2754. Cited in page 38.

WINIGER, S. *Adversarial Machines: Fooling A.Is (and turn everyone into a Manga).* 2015. Accessed: 2016-05-31. Available at: <https://medium.com/@samim/adversarial-machines-998d8362e996>. Cited in page 19.

ZEILER, M. D. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012. Available at: <http://arxiv.org/abs/1212.5701>. Cited in page 31.

ZEILER, M. D.; FERGUS, R. Visualizing and understanding convolutional networks. In: *Computer vision–ECCV 2014.* : Springer, 2014. p. 818–833. Cited in page 31.

ZEILER, M. D.; KRISHNAN, D.; TAYLOR, G. W.; FERGUS, R. Deconvolutional networks. In: IEEE. *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on.* 2010. p. 2528–2535. Cited in page 58.

ZHU, C.; BYRD, R. H.; LU, P.; NOCEDAL, J. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, ACM, v. 23, n. 4, p. 550–560, 1997. Cited in page 42.