Towards private and robust machine learning for information security

Jamie Hayes

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

November 30, 2021

I, Jamie Hayes, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Dedicated to my friend, Amar.

Abstract

Many problems in information security are pattern recognition problems. For example, determining if a digital communication can be trusted amounts to certifying that the communication does not carry malicious or secret content, which can be distilled into the problem of recognising the difference between benign and malicious content. At a high level, machine learning is the study of how patterns are formed within data, and how learning these patterns generalises beyond the potentially limited data pool at a practitioner's disposal, and so has become a powerful tool in information security.

In this work, we study the benefits machine learning can bring to two problems in information security. Firstly, we show that machine learning can be used to detect which websites are visited by an internet user over an encrypted connection. By analysing timing and packet size information of encrypted network traffic, we train a machine learning model that predicts the target website given a stream of encrypted network traffic, even if browsing is performed over an anonymous communication network. Secondly, in addition to studying how machine learning can be used to design attacks, we study how it can be used to solve the problem of hiding information within a cover medium, such as an image or an audio recording, which is commonly referred to as steganography. How well an algorithm can hide information within a cover medium amounts to how well the algorithm models and exploits areas of redundancy. This can again be reduced to a pattern recognition problem, and so we apply machine learning to design a steganographic algorithm that efficiently hides a secret message with an image.

Following this, we proceed with discussions surrounding why machine learning is not a panacea for information security, and can be an attack vector in and of itself. We show that machine learning can leak private and sensitive information about the data it used to learn, and how malicious actors can exploit vulnerabilities in these learning algorithms to compel them to exhibit adversarial behaviours. Finally, we examine the problem of the disconnect between image recognition systems learned by humans and by machine learning models. While human classification of an image is relatively robust to noise, machine learning models do not possess this property. We show how an attacker can cause targeted misclassifications against an entire data distribution by exploiting this property, and go onto introduce a mitigation that ameliorates this undesirable trait of machine learning.

Impact Statement

The work contained in part I will likely inform decisions about best practices with respect to applications of machine learning to information security problems, and contributed to the award of a Google PhD Fellowship. As part of an industry placement in the Government Digital Services (GDS), research ideas contained within this dissertation have been applied to the design of a UK government wide privacy-preserving anomaly detection pipeline. This is currently live and assists UK civil service analysts in the GDS with detecting fraudulent activity on the `.gov.uk` verify scheme [2].

The research detailed in part II works as guide and provides cautionary evidence for machine learning practitioners who work in privacy-sensitive environments, while the work in part III gives both theoretical and experimental evidence of the limitations of machine learning. We expect these research findings to act as warning of the limitations and dangers of applying machine learning blindly in privacy and security sensitive environments, and work as a guide as to when it is appropriate to trust decisions given by machine learning algorithms.

Where possible, data and code for work contained within this dissertation has been made freely available. Research on website fingerprinting in chapter 3, is freely available at `https://github.com/jhayes14/k-FP` and has become a benchmark within this field. Likewise steganographic work in chapter 4 is available at `https://github.com/jhayes14/advsteg`, and work on membership inference attacks in chapter 5 is available at `https://github.com/jhayes14/gen_mem_inf`.

Acknowledgements

Firstly, I would like to thank my supervisor, George Danezis. He has been a constant source of support, both professionally and personally. Upon matriculation, I was elated to find I had a supervisor who is not only a world-leading researcher, but has a constant appetite to learn. His willingness to learn and research new subjects and ideas was positively infectious, and gave me the confidence to broaden the scope of my research, which ultimately led to the work contained within this thesis. I am truly honoured to be his first doctoral student.

Alongside George Danezis, I am indebted to all the fantastic researchers I have had the opportunity to work with: Vasilios Mavroudis, Carmela Troncoso, Ania Piotrowska, Tariq Elahi, Sebastian Meiser, Giovanni Cherubin, Marc Juarez, Emiliano De Cristofaro, Nethanel Gelernter, Amir Herzberg, Olya Ohrimenko, Bogdan Kulynych, Nikita Samarin, Krishnamurthy (Dj) Dvijotham, Borja Balle, Zico Kolter, Chongli Qin, Andras Gyorgy, Kai Xiao, Sven Gowal, Pushmeet Kohli, Yutian Chen, Sander Dieleman, and Norman Casagrande.

I have been fortunate to experience research in a variety of industry environments. My thanks to Howard Staples, who was my host at the Government Digital Services, and gave me the opportunity to implement my research on production environments. To Aaron Johnson who hosted my time at the US Naval Research Laboratory, thank you for all the illuminating conversations and advice throughout my time there, and for all the extra work it took to get a non-US citizen into the building. Thank you to Olya Ohrimenko, Ian Fischer, and Pushmeet Kohli, who were my hosts at Microsoft Research, Google Research, and Google DeepMind, respectively, and gave me access to unique and exceptional research environments.

I am grateful to my funding sources that freed my mind from financial worries and allowed me to concentrate fully on research throughout my doctoral studies. From 2014 to 2017, I was funded by the UK Government Communications Headquarters (GCHQ), as part of University College London's status as a recognised Academic Centre of Excellence in Cyber Security Research. From 2017 to 2020, I was funded through a Google PhD Fellowship in Machine Learning.

Thanks to my parents, Sally and Paul, my partner, Nyla, and my siblings, Mairi and Tom. You all provided love, support, and guidance, even in times where I may have been less than deserving.

Finally, to my closest friend, Amar. Our ten years of friendship will instruct the way I live the rest of my life. You are greatly missed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine Learning models are increasingly relied upon for safety and business critical tasks such as in medicine [182, 215, 259], robotics and automotive [209, 221, 250], security [29, 139, 233] and financial [127, 150, 232] applications. However in environments and problems that necessitate *secure* solutions, questions have arisen around the pertinence of applying machine learning [188]. The success of a machine learning algorithm lies in its ability to generalise to unseen data. The standard assumption made in machine learning is that the data the algorithm learns on, is from the same distribution as any data it may observe in the future. The occurrence of an inconsistency in the distribution of data is called concept drift (also referred to as covariate shift or data nonstationarity), and is a real concern in production systems that cannot tolerate mistakes [51, 77]. Errors can also be the consequence of other underlying issues. If the machine learning algorithm does not receive a sufficient amount of data during training, it may be unable to generalise to new data; this often referred to as *overfitting*, where the algorithm is only accurate on the data it has observed when learning. Underparameterised, or excessively simple algorithms, can also lead to *underfitting*, where the algorithm cannot adequately capture the underlying structure of the data. For example, a linear model cannot adequately fit a non-linear function, and so using such a model will lead to poor performance.

If not addressed, all of these concerns can have devastating consequences on the accuracy of an algorithm, rendering it unsuitable for use in production. Threat modelling and a clear evaluation of the requirements of machine learning in production can alleviate these problems, however, there are additionally security vulnerabilities that appear to be inherent to the set of machine learning algorithms that are commonly used today. Firstly, errors in prediction can be intentionally induced through data poisoning, where an attacker can manipulate the data on which the algorithm learns. By replacing benign data with malicious data, an attacker can shift the underlying learned distribution. For example, clustering is often used in malware detection, where the goal is to distinguish between clusters of benign and malicious software. If an attacker can mislabel malware as benign and give this to the algorithm to learn on, this will reduce the efficacy of the algorithm when it is launched in production [21, 22, 41]. Secondly, even if an algorithm learns on entirely uncorrupted data, errors can be induced through corrupted data at prediction time. Adversarial perturbations can be added to benign inputs, that appear to be non-malicious to an oracle classifier, such as a human in vision tasks, but cause catastrophic failures in the algorithm [227].

In this dissertation, we discuss all of these failure cases. Our broad goal is to answer the following research questions:

1. Can machine learning be applied to well established problems in information security, and reach parity or improve upon standard techniques?
2. What are the inherent risks and dangers of using machine learning in information security? Are these risks fundamental in nature, or due to the threat model, problem assumptions, and implementation of the algorithms?

1.1 Thesis Contributions

This work presented in this dissertation make the following contributions to the fields of information security and machine learning:

1. In part I, we firstly present a website fingerprinting attack that leverages well established machine learning algorithms to design robust fingerprints of network traffic that are used to distinguish encrypted website traffic [97]. Sec-

ondly, we design a robust steganographic scheme for images, based on adversarial training of neural networks [83, 98]. The scheme approaches parity with established techniques in this space.

2. In part II, we propose attacks on generative machine learning models (i.e. machine learning models that attempt to learn the joint distribution of data and label $p(x,y)$ as opposed to discriminative models that learn the conditional $p(y|x)$.) that are designed to uncover information about the data used to train the model [102]. We also introduce a simple attack in the multi-party machine learning setting, that is designed to reduce to utility of a model. We show that a malicious party can introduce corrupted data into the pool of data that the model trains on, and consequently leads to unwanted behaviour when the model is used. Furthermore, we show this behaviour is difficult to observe using standard metrics such as validation accuracy on a hold-out dataset. We subsequently investigate defences to such an attack [100].

3. In part III, we first introduce a new method to design a universal adversarial perturbation. This perturbation can be applied to *any* data input to cause a misclassification by a machine learning model trained on similar data, with high likelihood [99]. Secondly, we introduce a defence to the general problem of adversarial examples in machine learning. Given a classifier, we show how to find a ball around an input with radius ε with respect to an ℓ_p norm, such that the classifier is guaranteed to return the same decision for any input within this ball with high likelihood.

1.2 Thesis Structure

The remainder of this dissertation is organised as follows. Chapter 2 gives an overview of the main research topics and related work that are used and referenced throughout this dissertation. Specifically, we survey research on website fingerprinting, steganography, and private and robust machine learning. Part I covers work on how machine learning can be applied to information security problems, designing a website fingerprinting attack, and designing a data hiding scheme. Part II details work on two problems of secure and private machine learning, membership inference attacks on generative machine learning models and contamination attacks in multi-party machine learning. Part III presents work on robust machine learning; we study the machine learning phenomenon commonly referred to as *adversarial examples*, in terms of designing effective attacks, and designing defences that provide robustness guarantees. Finally, Chapter 9 discusses and summarises the contributions presented in this dissertation.

1.3 Publications & contributions in joint work

The work presented in this dissertation has been published in several venues. Part I contains research from the following published works:

- ‘k-fingerprinting: A robust scalable website fingerprinting technique’ was published in USENIX Security Symposium 2016 [97].
- ‘Generating steganographic images via adversarial training’ was publish in Advances in Neural Information Processing Systems 2017 [98].

These works were both co-authored with my supervisor, George Danezis.

Part II contains findings from:

- ‘LOGAN: evaluating privacy leakage of generative models using generative adversarial networks’ was published in Proceedings on Privacy Enhancing Technologies 2019 [102]. This work was jointly lead by myself and Luca Melis, with co-authors Emiliano De Cristofaro and George Danezis. Both Luca Melis and I, contributed equally to the attack and experimental design, while I lead experimental evaluation on the CIFAR-10 and Diabetic Retinopathy datasets.
- ‘Contamination attacks and mitigation in multi-party machine learning’ was published in Advances in Neural Information Processing Systems 2018 [100]. This work was co-authored by my internship supervisor at Microsoft Research, Olya Ohrimenko.

Part II contains the following publish works:

- 'Learning universal adversarial perturbations with generative models' was published in IEEE Security and Privacy Workshop on Deep Learning and Security 2018 [99]. I was the principal investigator and my co-author was George Danezis.
- 'Extensions and limitations of randomised smoothing for robustness guarantees' was published in CVPR 2020 Workshop on Adversarial Machine Learning in Computer Vision [95] and I was the sole investigator.

1.4 Further contributions

In addition to the work presented in this dissertation, we have made further contributions in several areas in information security and machine learning. On the information security side we have made the following contributions:

1. Research on the Tor anonymity network routing protocol was presented in Proceedings on Privacy Enhancing Technologies 2015 in collaboration with George Danezis [96].
2. End-to-end traffic analysis attacks were presented in the NDSS Workshop track 2016 [93].
3. A new anonymous notification service was presented in Proceedings of the Workshop on Privacy in the Electronic Society 2017 in collaboration with Ania Piotrowska, Nethanel Gelernter, George Danezis, and Amir Herzberg [199].
4. Work on intersection attacks in anonymous communication systems was presented in Proceedings of the Workshop on Privacy in the Electronic Society 2016 in collaboration with Carmela Troncoso and George Danezis [101].
5. A new anonymous communication network was published in USENIX Security Symposium 2017 in collaboration with Ania Piotrowska, Tariq Elahi, Sebastian Meiser, and George Danezis [198].

6. Application layer defences to website fingerprinting attacks were published in Proceedings on Privacy Enhancing Technologies 2017 in collaboration with Giovanni Cherubin and Marc Juarez [46].

On the machine learning side, we published work on:

1. Designing defences to visible adversarial perturbations in the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops 2018 [94].

2. Provable defences to adversarial examples in International Conference on Learning Representation 2020 in collaboration with Krishnamurthy (Dj) Dvijotham, Borja Balle, Zico Kolter, Chongli Qin, Andras Gyorgy, Kai Xiao, Sven Gowal, and Pushmeet Kohli [62].

3. Designing watermarking schemes, done in collaboration with Krishnamurthy (Dj) Dvijotham, Yutian Chen, Sander Dieleman, Pushmeet Kohli, and Norman Casagrande [103].

4. Evading classifiers in discrete domains with provable optimality guarantees, done in collaboration with Bogdan Kulynych, Nikita Samarin, and Carmela Troncoso [134].

1.5 Experience

During my doctoral studies I was fortunate enough to be awarded a Google PhD Fellowship, and I spent time at the following institutions:

- From March to June, 2017, I interned with UK Government Digital Services, developing and implementing a privacy-preserving machine learning pipeline to aid threat analysis and improve Transaction Monitoring (TxM) on the `.gov.uk` verify system [2].

- From August to October, 2017, I interned at the US Naval Research Laboratory under Paul Syverson and Aaron Johnson, developing a project on using machine learning techniques to perform privacy-preserving experiments on live traffic analysis attacks on Tor.

- From February to May, 2018, I interned with Olya Ohrimenko at Microsoft Research, where we researched attacks in multi-party machine learning.
- From July to September, 2018, I interned with Ian Fischer at Google Research. We developed new techniques for unsupervised style transfer, and probabilistic conditioning methods for generative models.
- From May to September, 2019, I interned with Pushmeet Kohli at Google DeepMind. We developed zero-bit watermarking techniques that resist various distortion attacks.

Chapter 2

Background

In this chapter, we cover background material necessary to understand and contextualise research contributions made within this thesis. In section 2.1, we survey machine learning techniques used throughout subsequent chapters. Following this, in section 2.2, section 2.3, and section 2.4, we give an overview of sub-fields intersecting machine learning, privacy, and security, that are the topics studied in the remaining chapters.

2.1 Machine learning

2.1.1 Random forests

Random forests are a classification technique consisting of an ensemble of decision trees, taking a consensus vote of how to classify a new object. They have been shown to perform well in classification, regression [28, 148] and anomaly detection [153]. Each tree in the forest is trained using labelled objects represented as feature vectors of a fixed size. Training includes some randomness to prevent overfitting; this is referred to as bootstrap aggregation, or bagging, where the training set for each tree is sampled from the entire available training set with replacement. We use random forests in chapter 3 to develop a robust website fingerprinting attack.

2.1.2 Neural networks

Neural networks and stochastic gradient descent (SGD) are the core work horses behind the "deep learning revolution" [82]. Given access to data, X, with label set

Y, the goal of supervised machine learning is to learn the conditional distribution $p(Y|X)$. Neural networks define a parameterised function $f_w : X \rightarrow Y$ that when trained with SGD attempt to approximate the conditional $p(Y|X)$. A neural network is a composition of one or layers of neurons: an operation that takes it's input, multiplies it by a weight vector and then passes the sum through an activation function to the other neurons. Given an input x, to layer i, the input to layer $i+1$ is given by

$$\sigma(w_i^T \cdot x + b_i)$$

where w_i, a model weights that govern the strength of a connection between two neurons, b_i is a bias vector, and σ is a non-linear activation function. Popular choices of non-linear activation functions are ReLU(x) = max(0, x) and the sigmoid function. One can interpret the final layer as a probability vector by applying the softmax function to outputs (sometimes referred to as logits). Given an input x, logits $f_w(x)$, where $f_w(x)_k$ is the logit value of the k^{th} class, and true label y, a loss function is defined that outputs a scalar based on how strongly an input would be assigned the true class label. The most common loss function to use in classification-based supervised neural network training is the log-loss defined by $-y\log(f_w(x)_y)$. Model weights are then updated based on $\frac{\partial - y\log(f_w(x)_y)}{\partial w}$; averaging this loss over batches of inputs, computing the derivative with respect to model weights, and updating these weights in the opposite direction to this derivative is what is known as SGD and has produced state-of-the-art results on classification problems in a large number of fields (cf. Goodfellow et al. [82]).

2.1.3 Generative models

Given a supervised learning task, and given the features x of a data-point and the corresponding label y, discriminative models attempt to predict y on future x by learning a discriminative function f from (x, y); the function takes in input x and outputs the most likely label y. By contrast, generative models describe how data is generated by learning the joint probability distribution of $p(X,Y)$. Generative models based on deep neural networks, such as Generative Adversarial Networks (GAN) [81] (introduced below) and Variational Auto-encoders (VAE) [130] are considered the state-of-the-art for producing samples of realistic images [124].

Generative Adversarial Networks (GANs) [81] are neural networks trained in an adversarial manner to generate data mimicking some distribution. One network takes noise as input and generates data samples – and so is called the *generator*. The other model, the *discriminator*, receives samples from both the generator and the training data, and has to be able to distinguish between the two sources. The two networks play a continuous game where the generator is learning to produce more and more realistic samples, and the discriminator is learning to get better and better at distinguishing generated data from real data.

More formally, to learn the generator's output distribution over data-points x, we define a prior on input noise variables $p_z(z)$, then represent a mapping to data space as $G(z;\theta_g)$, where G is a generative deep neural network with parameters θ_g. We also define a discriminator $D(x;\theta_d)$ that outputs $D(x) \in [0,1]$, representing the probability that x was taken from the training set rather than from the generator G. D is trained to maximise the probability of assigning the correct label to both real training examples and fake samples from G. We simultaneously train G to minimise $\log(1-D(G(z)))$. The final optimisation problem solved by the two networks D and G follows a two-player minimax game as:

$$\min_G \max_D \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$

First, gradients of D are computed to discriminate fake samples from training data, then G is updated to generate samples that are more likely to be classified as data. GANs represent a zero-sum game between two networks, while using first-order optimisation techniques often fail to converge to a stable solution, if G and D have enough capacity and a Nash equilibrium is achieved, they will reach a point at which both cannot improve [81].

Work by Lucic et al. [158] showed that, despite a large number of proposed changes to the original GAN model [12, 19, 87] it is still difficult to assess if one performs better than another. They also show that the original GAN performs equally well against other state-of-the-art GANs, concluding that any improvements are due to computational budgets and hyper-parameter tuning, rather than scientific breakthroughs.

Variational Auto-encoders (VAE) [130] VAEs [130] consist of two neural networks (an *encoder* and a *decoder*) and a loss function. The encoder compresses data into a latent space z while the decoder reconstructs the data given the hidden representation. Rather than attempting to maximise the likelihood, one could maximise a lower bound of the likelihood, thus, if the lower bound increases to a given level, the likelihood must be at least as high. More formally, let x be a random vector of observed variables, which are either discrete or continuous. Let z be a random vector of latent continuous variables.

The probability distribution between x and z assumes the form $p_\theta(x,z) = p_\theta(z)p_\theta(x \mid z)$, where θ indicates that p is parameterised by θ. Also, let $q_\phi(z \mid x)$ be a recognition model whose goal is to approximate the true and intractable posterior distribution $p_\theta(z \mid x)$. We can then define a lower-bound on the log-likelihood of x as follows: $\mathscr{L}(x) = -D_{KL}(q_\phi(z \mid x) \mid\mid p_\theta(z)) + \mathrm{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)]$. The first term pushes $q_\phi(z \mid x)$ to be similar to $p_\theta(z)$ ensuring that, while training, the VAE learns a decoder that, at generation time, will be able to process samples from the prior distribution such they resemble the training data. The second term can be seen as a form of reconstruction cost, and needs to be approximated by sampling from $q_\phi(z \mid x)$. In VAEs, the gradient signal is propagated through the sampling process and through $q_\phi(z \mid x)$, using the so-called re-parameterisation trick. This is done by making z a deterministic function of ϕ and some noise ε, i.e., $z = f(\phi, \varepsilon)$. For instance, sampling from a normal distribution can be done as $z = \mu + \sigma\varepsilon$, where $\varepsilon \sim \mathscr{N}(0, I)$. VAEs are trained using stochastic gradient descent to optimise the loss w.r.t. the parameters of the encoder and decoder θ and ϕ. For an in-depth discussion of VAEs, we refer the reader to Doersch [58].

Larsen et al. [140] combine VAEs and GANs into an unsupervised generative model that simultaneously learns to encode and generate new samples, which contain more details, sampled from the training data-points.

2.2 Security of machine learning

2.2.1 Membership inference attacks

In chapter 5, we study *membership inference* attacks against generative models. That is, given access to a generative model and an individual data record, can an attacker tell if a specific record was used to train the model? Membership inference on generative models is likely to be more challenging than on discriminative ones (see, e.g., [57]). The latter attempt to predict a label given a data input, and an attacker can use the confidence the model places on an input belonging to a label to perform the attack. In generative models, there is no such signal, thus, it is difficult to both detect overfitting and infer membership.

Specific to membership inference are attacks against supervised models by Shokri et al. [218]. Their approach exploits differences in the model's response to inputs that were or were not seen during training. For each class of the targeted black-box model, they train a *shadow model*, with the same machine learning technique. Whereas, our approach targets generative models and relies on GANs to provide a general framework for measuring the information leakage. Performing a membership inference attack amounts to detecting overfitting in generative models, which is regarded as one of the most important research problems in machine learning [255].

Additional membership inference attacks focus on genomic research studies [16, 113], whereby an attacker aims to infer the presence of a particular individual's data within an aggregate genomic dataset, or aggregate locations [202].

Tangentially related are *model inversion* attacks [74], where an adversary extracts training data from outputted model predictions. Fredrikson et al. [73] show how an attacker can rely on outputs from a model to infer sensitive features used as inputs to the model itself: given the model and some demographic information about a patient whose records are used for training, an attacker predicts sensitive attributes of the patient. However, the attack does not generalise on inputs not seen at training time, thus, the attacker relies on statistical inference about the total population [166]. The record extracted by the attacker is not an actual training record, but an average representation of the inputs that are classified in a particular class. Long et al. [155] and Yeom et al. [261] investigate connections between membership inference and model inversion attacks against machine learning classifiers. In particular, Yeom et al. [261] assumes that the adversary knows the distribution from which the training set was drawn and its size, and that the adversary colludes with the training algorithm. Their attacks are close in performance to Shokri et al. [218], and show that, besides overfitting, the influence of target attributes on model's outputs also correlates with successful attacks. Then, Tramèr et al. [233] present a model extraction attack to infer the parameters from a trained classifier, however, it only applies to scenarios where the attacker has access to the probabilities returned for each class.

Song et al. [222] develop attacks that force a machine learning model to memorise the training data in such a way that an adversary can later extract training inputs with only black-box access to the model. Carlini et al. [38] show that deep learning-based language models trained on text data can unintentionally memorise specific training inputs, which can then be extracted with black-box access, however, demonstrating it only for simple sequences of digits artificially introduced into the text. Ateniese et al. [13] present a few attacks against SVM and HMM classifiers aimed to reconstruct properties about training sets, by exploiting knowledge of model parameters.

Recent work [11, 109, 167] present inference attacks against distributed deep learning [164]. In particular, Aono et al. [11] target the collaborative privacy-preserving deep learning protocol of Shokri and Shmatikov [217], and show that an honest-but-curious server can partially recover participants' data points from the shared gradient updates. However, they operate on a simplified setting where the batch consists of a single data point. Also, Hitaj et al. [109] introduce a white-box attack against Shokri and Shmatikov [217], which relies on GAN models to generate valid samples of a particular class from a targeted private training set, however, it cannot be extended to black-box scenarios. Furthermore, evaluation of the attack is limited to the MNIST dataset of handwritten digits where all samples in a class look very similar, and the AT&T dataset of Faces, which consists of only 400 grayscale images of faces. By contrast, our evaluation in chapter 5 is performed on 13,233, 60,000, and 88,702 images for the LFW, CIFAR-10, and Diabetic Retinopathy datasets, respectively.

Finally, Truex et al. [236] show how membership inference attacks are data-driven and largely transferable, while Melis et al. [167] demonstrate how an adversarial participant can successfully perform membership inference in distributed learning [164, 217], as well as inferring sensitive properties that hold only for a subset of the participants' training data.

2.2.2 Membership inference defences

Privacy-enhancing tools based on secure multiparty computation and homomorphic encryption have been proposed to securely train supervised machine learning models, such as decision trees [151], linear regressors [60], and neural networks [26, 59]. However, these mechanisms do not prevent an attacker from running inference attacks on the privately trained models as the final parameters are left unchanged.

Differential Privacy [63] can be used to mitigate inference attacks, and it has been widely applied to various machine learning models [8, 137, 190, 217, 242, 253]. Shokri and Shmatikov [217] support distributed training of deep learning networks in a privacy-preserving way, where independent entities collaboratively build a model without sharing their training data, but selectively share subsets of noisy model parameters during training. Abadi et al. [8] show how to train deep neural networks (DNNs) with non-convex objectives with an acceptable privacy budget, while Rahman et al. [205] show that the Abadi et al. [8] proposal partially mitigates the effects of the Shokri et al. [218] membership inference attack.

Papernot et al. [190, 192] combine multiple models trained with disjoint datasets without exposing the models, while, Papernot et al. [189] present "defensive distillation" to reduce the effectiveness of adversarial samples on DNNs.

Then, Beaulieu-Jones et al. [17] apply the noisy gradient descent from Abadi et al. [8] to train the discriminator of a Generative Adversarial Network with differential privacy. The resulting model is then used to generate synthetic subjects based on the population of clinical trial data. Finally, Jia and Gong [120] use adversarial machine learning to defend against attribute inference attacks, in the setting where an attacker trains a classifier to infer a target user's sensitive attributes from their public data, while Nasr et al. [178] leverage adversarial regularisers to design a privacy-preserving training mechanism with provable protections against membership inference attacks against discriminative models.

2.2.3 Multi-party machine learning

Multi-party machine learning allows several parties (e.g., hospitals, banks, government agencies) to combine their datasets and run algorithms on their joint data in order to get insights that may not be present in their individual datasets. As there could be competitive and regulatory restrictions as well as privacy concerns about sharing datasets, there has been extensive research on developing techniques to perform secure multi-party machine learning. The main guarantee of secure multi-party computation (MPC) is to allow each party to obtain only the output of their mutually agreed-upon computation without seeing each others data nor trusting a third-party to combine their data for them.

Secure MPC can be enabled with cryptographic techniques [26, 79, 107, 171, 180], and systems based on trusted processors such as Intel SGX [10, 24, 183]. In the latter, a (untrusted) cloud service collects encrypted data from multiple parties who decide on an algorithm and access control policies of the model, and runs the code inside of a Trusted Execution Environment (TEE) protected by hardware guarantees of the trusted processor. The data is decrypted with the TEE and stays encrypted in memory. This ensures that nothing except the output is revealed to the parties, while no one else (including the cloud provider) learns neither the data nor the output, and any tampering with the data during the computation is detected. Additionally, it allows parties to outsource potentially expensive computation and guarantees that they do not see model parameters during training that have to be shared, for example, in distributed settings [91, 163, 194, 206, 217].

Multi-party machine learning raises concerns regarding what parties can learn about each others data through model outputs as well as how much a malicious party can influence training. The number of parties and how much data each one of them contributes influences the extent of their malicious behaviour. For example, the influence of each party is limited in the case where a model is trained from hundreds or thousands of parties (e.g., users of an email service) where each party owns a small portion of training data. As a result, differential privacy guarantees at a per-party level have shown to be successful [8, 165]. Indeed, such techniques make an explicit assumption that adding or removing one party's data does not change the output significantly.

2.2.4 Adversarial examples

Szegedy et al. [227] casts the construction of adversarial examples as an optimisation problem. Given a *target model*, f, and a *source* input x, which is classified correctly by f as c, the attacker aims to find a perturbation, δ, such that $x+\delta$ is perceptually identical to x but $f(x+\delta) \neq c$. The attacker tries to minimise the distance between the source input and adversarial input under an appropriate measure. The problem space can be framed to find a specific misclassification in a *targeted* attack, or *any* misclassification, referred to as a *non-targeted* attack.

In the absence of a distance measure that accurately captures the perceptual differences between a source and adversarial input, the ℓ_p metric is commonly used [227]. Related work commonly uses the ℓ_2 and ℓ_∞ metrics [34, 35, 54, 105, 132, 136, 173, 172, 264]. The ℓ_2 metric measures the Euclidean distance between two inputs, while the ℓ_∞ metric measures the largest pixel-wise difference between two inputs (Chebyshev distance).

2.3 Website fingerprinting

Traditional encryption obscures only the content of communications and does not hide metadata such as the size and direction of traffic over time. Anonymous communication systems obscure both content and metadata, preventing a passive attacker from observing the source or destination of communication. Anonymous communications tools, such as Tor [57], route traffic through relays to hide its ultimate destination. Tor is designed to be a low-latency system to support interactive activities such as instant messaging and web browsing, and does not significantly alter the shape of network traffic. This allows an attacker to exploit information leaked via the order, timing and volume of resources requested from a website. As a result, many works have shown that website fingerprinting attacks are possible even when a client is browsing with encryption or using an anonymity tool such as Tor [31, 86, 138, 185, 216, 247]. Website fingerprinting is commonly formulated as a classification problem. An attacker wishes to know whether a client browses one of *n* web pages. The attacker first collects many examples of traffic traces from each of the *n* web pages by performing web-requests through the protection mechanism under attack; features are extracted and a machine learning algorithm is trained to classify the website using those features. When a client browses a web page, the attacker passively collects the traffic, passes it in to their classifier and checks if the client visited one of the *n* web pages. In the literature this is referred to as the closed-world setting – a client is restricted to browse a limited number of web pages, monitored by the attacker. However, the closed-world model has been criticised for being unrealistic [121, 196] since a client is unlikely to adhere to such a constraint. The open-world setting attempts to model a more realistic setting where the attacker monitors a small number of web pages, but allows a client to additionally browse to a large world size of unmonitored web pages [186, 244].

2.4 Steganography

Steganography research can be split into two subfields: the construction of steganographic algorithms and the study of steganalysis. Research into steganographic algorithms concentrates on finding methods to embed undetected information within a medium while minimising the perturbations of the added information within that medium. Steganalysis research seeks to discover methods to detect such perturbations. Steganalysis is a binary classification task: discovering whether or not secret information is present with a message, and so machine learning classifiers are commonly used in this field.

Least significant bit (LSB) [169] is a simple steganographic algorithm used to embed a secret message within a cover image. Each pixel in an image is made up of three RGB colour channels (or one for grayscale images), and each colour channel is represented by a number of bits. For example, it is common to represent a pixel in a grayscale image with an 8-bit binary sequence. The LSB technique then replaces the least significant bits of the cover image by the bits of the secret message. By only manipulating the least significant bits of the cover image, the variation in colour of the original image is minimised. However, information from the original image is always lost when using the LSB technique, and is known to be vulnerable to steganalysis [75].

Most steganographic schemes for images use a distortion function that forces the embedding process to be localised to parts of the image that are considered noisy. Steganographic algorithms attempt to minimise the distortion function between a cover image, C, and a steganographic image, C',

$$d(C,C') = f(C,C') \cdot |C - C'|$$

It is the choice of the function f, the cost of distorting a pixel, which changes for different steganographic algorithms. State-of-the-art embedding algorithm, HUGO [197], is considered to be one of the most secure steganographic techniques [39]. It defines a distortion function domain by assigning costs to pixels based on the effect of embedding some information within a pixel, the space of pixels is condensed into a feature space using a weighted norm function. WOW (Wavelet Obtained Weights) [111] is another advanced steganographic method that embeds information into a cover image according to regions of complexity. If a region of an image is more textually complex than another, the more pixel values within that region will be modified. Finally, S-UNIWARD [112] proposes a universal distortion function that is agnostic to the embedding domain. However, the end goal is much the same: to minimise this distortion function, and embed information in noisy regions or complex textures, avoiding smooth regions of the cover images. In terms of steganalysis, we measure our work in chapter 4 against the ATS [145] steganalysis tool. ATS uses labelled data to build artificial training sets of cover and steganographic images, and is trained using an SVM with a Gaussian kernel. The authors show that this technique outperforms other popular steganalysis tools.

Part I

Machine learning applications to information security

In chapter 2, we introduced two problems in information security; website fingerprinting and the problem of designing a secure steganographic algorithm for digital media. We saw that machine learning has been extensively applied to the problem of website fingerprinting. Indeed, the challenge of website fingerprinting is to design a function that accepts a network traffic flow as input and a website label as output, and therefore standard classification techniques in machine learning are suited to such a problem. While machine learning techniques have been applied to the problem of steganalysis, there is a dearth of research into how machine learning techniques can be used for steganography. This is perhaps surprising, since the success of both steganography and steganalysis depend on the ability of the technique to model the cover medium, a task that is again suited to machine learning techniques. In chapter 3 and chapter 4, we show how machine learning can be used to tackle both of these problems.

Chapter 3

Robust & scalable website fingerprinting

Website fingerprinting has been studied extensively in previous works. Early work by Wagner and Schneier [241], and Cheng and Avnur [45] exposed the possibility that encrypted HTTP GET requests may leak information about the URL, conducting preliminary experiments on a small number of websites. They asked clients in a lab setting to browse a website for 5-10 minutes, pausing two seconds between page loading. With caching disabled they were able to correctly identify 88 pages out of 92 using simple packet features. Early website fingerprinting defences were usually designed to safeguard against highly specific attacks. In 2009, Wright et al. [252] designed ‘traffic morphing’ that allowed a client to shape their traffic to look as if it was generated from a different website. They were able to show that this defence does well at defeating early website fingerprinting attacks that heavily relied on exploiting unique packet length features [149, 226].

A simple defence instantiated by Tor is to pad all packets to a fixed-size cells of 512 bytes. Tor also implemented randomised ordering of HTTP pipelines [195] in response to the attack by Panchenko et al. [185], who used packet ordering features to train an SVM classifier. This attack on Tor achieved an accuracy of 55%, compared to a previous attack that did not use such fine grained features achieving 3% accuracy on the same dataset using a Naive-Bayes classifier [106]. Other defences such as the decoy defence [185] loads a camouflage website in parallel to a legitimate website, adding a layer of background noise. They were able to show using this defence attack accuracy of the SVM again dropped down to 3%.

Luo et al. [159] designed the HTTPOS fingerprinting defence at the application layer. HTTPOS acts as a proxy accepting HTTP requests and obfuscating them before allowing them to be sent. It modifies network features on the TCP and HTTP layer such as packet size, packet time and payload size, along with using HTTP pipelining to obfuscate the number of outgoing packets. They showed that HTTPOS was successful in defending against a number of classifiers [23, 42, 149, 226].

More recently Dyer et al. [67] created a defence, BuFLO, that combines many previous countermeasures, such as fixed packet sizes and constant rate traffic, and showed this defence improved upon others at the expense of a high bandwidth overhead. Cai et al. [32] made modifications to the BuFLO defence based on rate adaptation again at the expense of a high bandwidth overhead. Following this Nithyanand et al. [181] proposed Glove, that groups website traffic into clusters that cannot be distinguished from any other website in the set. This provides information theoretic privacy guarantees and reduces the bandwidth overhead by intelligently grouping web traffic in to similar sets.

Cai et al. [31] modified the kernel in Panchenko et al. [185] SVM to improve an attack on Tor, and was further improved in an open-world setting by Wang and Goldberg [245], achieving a true positive rate (TPR) of over 0.95 and a false positive rate (FPR) of 0.002 when monitoring one web page. Wang et al. [248] conducted attacks on Tor using large open-world sets. Using a *k*-nearest neighbour classifier they achieved a TPR of 0.85 and FPR of 0.006 when monitoring 100 web pages out of 5100 web pages. More recently Wang and Goldberg [243] suggested a defence using a browser in half-duplex mode – meaning a client cannot send multiple requests to servers in parallel. In addition to this simple modification they add random padding and show they can even foil an attacker with perfect classification accuracy with a comparatively (to other defences) small bandwidth overhead. Wang and Goldberg [247] then investigated the practical deployment of website fingerprinting attacks on Tor. By maintaining an up-to-date training set and splitting a full packet sequence in to components comprising of different web page load traces they show that practical website fingerprinting attacks are possible. By considering a time gap of 1.5 seconds between web page loads, their splitting algorithm can successfully parse a single packet sequence in to multiple packet sequences with no loss in website fingerprinting accuracy. Gu et al. [86] studied website fingerprinting in multi-tab browsing setting. Using a Naive Bayes classifier on the 50 top-ranked websites in Alexa, they show when tabs are opened with a delay of 2 seconds, they can classify the first tab with 75.9% accuracy, and the background tab with 40.5%. More recently, Kwon et al. [138] showed that website fingerprinting attacks can be applied to Tor hidden services, and due to the long lived structure of hidden services, attacks can be even more accurate than when compared to non-hidden pages. They correctly de-anonymise 50 monitored hidden service servers with TPR of 88% and FPR of 7.8% in an open world setting. In concurrent work to ours, Panchenko et al. [186] have experimented with large datasets. Using a mix of different sources they produced two datasets, one of 34,580 unique websites (118,884 unique web pages) and another of 65,409 unique websites (211,148 unique web pages). Using a variation of a sequence of cumulative summations of packet sizes in a traffic trace they show their attack, CUMUL, was of similar accuracy to *k*-NN [248] under normal browsing conditions.

3.1 *k*-fingerprints from random forests

We consider an attacker that can passively collect a client's encrypted or anonymised web traffic, and aims to infer which web resource is being requested. Dealing with an open-world (as defined in section 2.3), makes approaches based purely on classifying previously seen websites inapplicable. Our technique, *k*-fingerprinting, aims to define a distance-based classifier. It automatically manages unbalanced sized classes and assigns meaningful distances between packet sequences, where close-by 'fingerprints' denote requests likely to be for the same resources.

Our technique, *k*-fingerprinting, is based on random forests, which are used to extract a fingerprint for each traffic instance[1], instead of directly using the classification decision of the random forest. We define a distance metric between two traces based on the output of the forest like so: given a feature vector each tree in the forest associates a leaf identifier with it, forming a vector of leaf identifiers for the input, which we refer to as the *fingerprint*.

Once fingerprint vectors are extracted for two traces, we use the Hamming distance[2] to calculate the distance between these fingerprints[3]. We classify a test instance as the label of the closest k training instances via the Hamming distance of fingerprints – assuming all labels agree. We evaluate the effect of varying k, the number of fingerprints used for comparison, in section 3.6, section 3.7, and section 3.8.

The vector output by the random forest represents a robust fingerprint: we expect similar traffic sequences are more likely to fall on the same leaves than dissimilar traffic sequences. Since the forest has been trained on a classification task, traces from the same websites are preferentially aggregated in the same leaf nodes, and those from different websites kept apart. We can vary the number of training instances, k, a fingerprint should match, to allow an attacker to trade the true positive rate (TPR) for false positive rate (FPR). This is not possible using the direct classification of the random forest. By using a k-closest fingerprint technique for classification, the attacker can choose how they wish to decide upon final classification[4]. For the closed-world setting we do not need the additional fingerprint layer for classification, we can simply use the classification output of the random forest since all classes are balanced and our attack does not have to differentiate between false positives and false negatives. For the closed-world setting we measure the mean accuracy of the random forest.

[1]We define a traffic instance as the network traffic generated via a web page load.

[2]We experimented with using the Hamming, Euclidean, Mahalanobis and Manhattan distance functions and found Hamming to provide the best results.

[3]For example, given the Hamming distance function $d : V \times V \to \mathbb{R}$, where V is the space of leaf symbols, we expect given two packet sequences generated from loading `https://google.com`, with fingerprints vectors f_1, f_2 and a packet sequence generated from loading `https://facebook.com` with fingerprint f_3, that $d(f_1, f_2) < d(f_1, f_3)$ and $d(f_1, f_2) < d(f_2, f_3)$.

[4]We chose to classify a traffic instance as a monitored page if all k fingerprints agree on the label, but an attacker could choose some other metric such as majority label out of the k fingerprints.

3.2 The *k*-fingerprinting attack

Our *k*-fingerprinting attack proceeds in two phases: The attacker chooses which web pages they wish to monitor and captures network traffic generated via loading the monitored web pages and a large number of unmonitored web pages. These target traces for monitored websites, along with some traces for unmonitored websites, are used to train a random forest for classification. Given a packet sequence representing each training instance of a monitored web page, it is converted to a fixed length fingerprint as described in section 3.1 and stored.

The attacker then passively collects instances of web page loads from a client's browsing session. A fingerprint is extracted from the newly collected packet sequence. The attacker then computes the Hamming distance of this new fingerprint against the corpus of fingerprints collected during training and is classified as a monitored page if and only if all *k* fingerprints agree on classification, as described in section 3.1, otherwise it is classified as an unmonitored page.

We define the following performance measures for the attack:

- **True Positive Rate (TPR).** The probability that a monitored page is classified as the correct monitored page.
- **False Positive Rate (FPR).** The probability that an unmonitored page is incorrectly classified as a monitored page.
- **Bayesian Detection Rate (BDR).** The probability that a page corresponds to the correct monitored page given that the classifier recognised it as that monitored page. Assuming a uniform distribution of pages BDR can be found from TPR and FPR using the formula

  $$\frac{TPR.\Pr(M)}{(TPR.\Pr(M)+FPR.\Pr(U))}$$

  where

  $$\Pr(M)=\frac{|\text{Monitored Pages}|}{|\text{Total Pages}|},\quad \Pr(U)=1-P(M).$$

Ultimately BDR indicates the practical feasibility of the attack as it measures the main concern of the attacker, the probability that the classifier made a correct prediction.

3.3 Experimental set-up

We collect two datasets: one via Tor[5] DS_{Tor}, and another via a standard web browser, DS_{Norm}. DS_{Norm} consists of 30 instances from each of 55 monitored web pages, along with 7,000 unmonitored web pages chosen from Alexa's top 20,000 web sites [1]. We collected DS_{Norm} using a number of `Amazon EC2 instances`[6], `Selenium`[7] and the headless browser `PhantomJS`[8]. We used `tcpdump`[9] to collect network traces for 20 seconds with 2 seconds between each web page load. Monitored pages were collected in batches of 30 and unmonitored web pages were collected successively. Page loading was performed with no caches and time gaps between multiple loads of the same web page, as recommended by Wang and Goldberg [245]. We chose to monitor web pages from Alexa's top 100 web sites [1] to provide a comparison with the real world censored web pages used in the Wang et al. [248] data set[10]. DS_{Tor} was collected in a similar manner to DS_{Norm} but was collected via the Tor browser. DS_{Tor} consists of two subsets of monitored web pages: (i) 100 instances from each of the 55 top Alexa monitored web pages and (ii) 80 instances from each of 30 popular Tor hidden services. A Tor hidden service is a website that is hosted behind a Tor client's *Onion Proxy*, which serves as the interface between application and network. Tor hidden services allow both a client accessing the website and the server hosting the website to remain anonymous to one another and any external observers. We chose hidden services to fingerprint based on popularity as listed by the `.onion` search engine Ahmia[11]. The unmonitored set is comprised of the top 100,000 Alexa web pages, excluding

[5]The most recent version at the time of collection was used, Tor Browser 5.0.6.
[6]`https://aws.amazon.com/ec2/`
[7]`http://www.seleniumhq.org/`
[8]`http://phantomjs.org/`
[9]`http://www.tcpdump.org/`
[10]We used TCP/IP packets for final classification over abstracting to the Tor cell layer [245], preliminary experiments showed no consistent improvements from using one data layer for classification over the other.
[11]`http://www.ahmia.fi/`

the top 55. We chose to fingerprint web pages as listed by Alexa as these constitute the most popular web pages in the world over extended periods of time, and hence provide a more realistic dataset than choosing pages at random and/or using transiently popular website links as included in recent work from Panchenko et al. [186]. By including website visits to trending topics we argue that this diminishes the ability to properly measure how effective a website fingerprinting attack will perform in general.

For comparison to previous work, we evaluated our attack on one of the previous largest website fingerprinting datasets [248], which collected 90 instances from each of 100 monitored sites, along with 5000 unmonitored web pages. The Wang et al. [248] monitored web pages are various real-world censored websites from UK, Saudi Arabia and China providing a realistic set of web pages an attacker may wish to monitor. The unmonitored web pages are chosen at random from Alexa's top 10,000 websites – with no intersection between monitored and unmonitored web pages.

This allows us to validate k-fingerprinting on two different datasets while allowing for direct comparison against the state-of-the-art k-nearest neighbour attack [248]. We can also infer how well the attack works on censored web pages which may not have small landing pages or be set up for caching like websites in the top Alexa list. Testing k-fingerprinting on both real-world censored websites and top Alexa websites indicates how the attack performs across a wide range of websites.

For the sake of comparison, according to a study by research firm Nielsen [3] the number of unique websites visited per month by an average client in 2010 was 89. Another study [121, 184] collected web site statistics from 80 volunteers in a virtual office environment. Traffic was collected from each volunteer for a total of 40 hours. The mean unique number of websites visited per volunteer was 484, this is substantially smaller than the world sizes we consider in our experiments.

3.4 Information leakage from network traffic features

Our first contribution is a systematic analysis of feature importance. Despite some preliminary work by Panchenko et al. [185], there is a notable absence of feature analysis in the website fingerprinting literature. Instead features are picked based on heuristic arguments. All feature importance experiments were performed with the Wang et al. [248] dataset so as to allow direct comparison with their attack results.

We train a random forest classifier in the closed-world setting using a feature vector comprised of features commonly used in related work, and labels corresponding to the monitored sites. We use the gini coefficient as the purity criterion for splitting branches and estimate feature importance using the standard methodology described by Breiman [28], Friedman [76]. Each time a decision tree branches on a feature the weighted sum of the gini impurity index for the two descendant nodes is higher than the purity of the parent node. We add up the gini decrease for each individual feature over the entire forest to get a consistent measure of feature importance.

Figure 3.1 illustrates the effect of using a subset of features for random forest classification. We first train a random forest classifier to establish feature importance; and then train new random forests with only subsets of the most informative features. As we increase the number of features we observe a monotonic increase in accuracy; however there are diminishing returns as we can achieve nearly the same accuracy after using the 30 most important features. We chose to use 150 features in all following experiments since the difference in training time when using fewer features was negligible.

Figure 3.2 identifies the top-20 ranked features and illustrates their variability across 100 repeated experiments. As seen in fig. 3.1 there is a reduction in gradient when combining the top 15 features compared to using the top 10 features. Figure 3.2 shows that the top 13 features are comparatively more important than the rest of the top 20 features, hence there is only a small increase in accuracy when

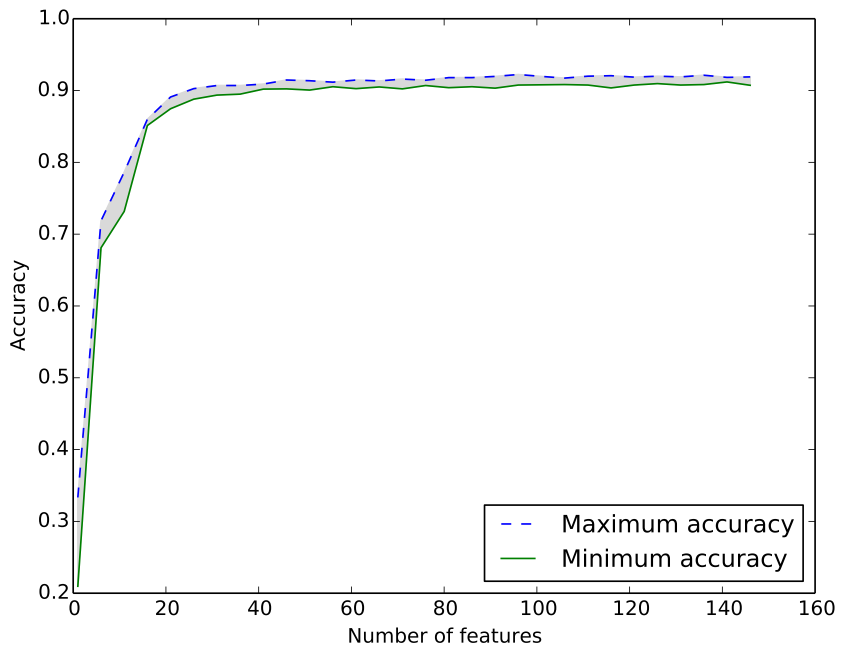


Figure 3.1: Accuracy of *k*-fingerprinting in a closed-world setting as the number of features is varied.

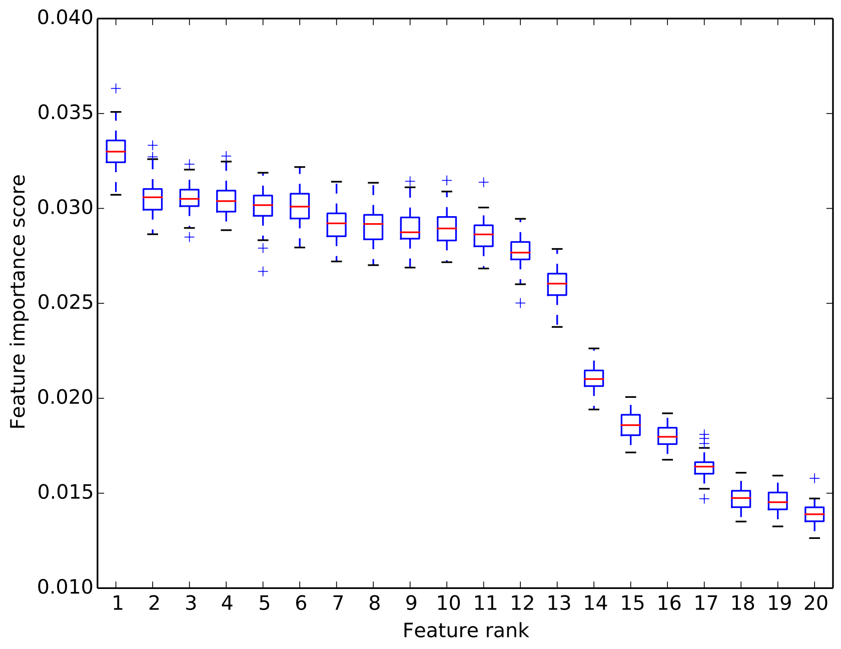
using the top 15 features compared to using the top 10. After the drop between the rank 13 and rank 14 features, feature importance falls steadily until feature rank 40, after which the reduction in feature importance is less prominent[12]. Note that there is some interchangeability in rank between features, we assign ranks based on the average rank of a feature over the 100 experiments.

From each packet sequence we extract the following features:

- **Number of packets statistics.** The total number of packets, along with the number of incoming and outgoing packets [67, 185, 248]. The number of incoming packets during transmission is the most important feature, and together with the number of outgoing packets during transmission are always two of the five most important features. The total number of packets in transmission has rank 10.

- **Incoming & outgoing packets as fraction of total packets.** The number of incoming and outgoing packets as a fraction of the total number of packets [185]. Always two of the five most important features.

[12]The total feature importance table is shown in appendix A.1.



№	Feature Description
1.	Number of incoming packets.
2.	Number of outgoing packets as a fraction of the total number of packets.
3.	Number of incoming packets as a fraction of the total number of packets.
4.	Standard deviation of the outgoing packet ordering list.
5.	Number of outgoing packets.
6.	Sum of all items in the alternative concentration feature list.
7.	Average of the outgoing packet ordering list.
8.	Sum of incoming, outgoing and total number of packets.
9.	Sum of alternative number packets per second.
10.	Total number of packets.
11-18.	Packet concentration and ordering features list.
19.	The total number of incoming packets stats in first 30 packets.
20.	The total number of outgoing packets stats in first 30 packets.

Figure 3.2: The 20 most important features.

- **Packet ordering statistics.** For each successive incoming and outgoing packet, the total number of packets seen before it in the sequence [31, 185, 248]. The standard deviation of the outgoing packet ordering list has rank 4, the average of the outgoing packet ordering list has rank 7. The standard deviation of the incoming packet ordering list has rank 12 and the average of the incoming packet ordering list has rank 13.

- **Concentration of outgoing packets.** The packet sequence split into non-overlapping chunks of 20 packets. Count the number of outgoing packets in each of the chunks. Along with the entire chunk sequence, we extract the standard deviation (rank 16), mean (rank 11), median (rank 64) and max (rank 65) of the sequence of chunks. This provides a snapshot of where outgoing packets are concentrated [248]. The features that make up the concentration list are between the 15th and 30th most important features, but also make up the bulk of the 75 least important features.

- **Concentration of incoming & outgoing packets in first & last 30 packets.** The number of incoming and outgoing packets in the first and last 30 packets [248]. The number of incoming and outgoing packets in the first thirty packets has rank 19 and 20, respectively. The number of incoming and outgoing packets in the last thirty packets has rank 50 and 55, respectively.

- **Number of packets per second.** The number of packets per second, along with the mean (rank 44), standard deviation (rank 38), min (rank 117), max (42), median (rank 50).

- **Alternative concentration features.** This subset of features is based on the concentration of outgoing packets feature list. The outgoing packets feature list split into 20 evenly sized subsets and sum each subset. This creates a new list of features. Similarly to the concentration feature list, the alternative concentration feature list are regularly in the top 20 most important features and bottom 50 features. Note though concentration features are never seen in the top 15 most important features whereas alternative concentration features are, – at rank 14 and 15, – so information is gained by summing the concentration subsets.

- **Packet inter-arrival time statistics.** For the total, incoming and outgoing packet streams extract the lists of inter-arrival times between packets. For each list extract the max, mean, standard deviation, and third quartile [23]. These features have rank between 40 and 70.

- **Transmission time statistics.** For the total, incoming and outgoing packet sequences we extract the first, second, third quartile and total transmission time [248]. These features have rank between 30 and 50. The total transmission time for incoming and outgoing packet streams are the most important out of this subset of features.

- **Alternative number of packets per second features.** For the number of packets per second feature list we create 20 even sized subsets and sum each subset. The sum of all subsets is the 9th most important feature. The features produced by each subset are in the bottom 50 features - with rank 101 and below. The important features in this subset are the first few features with rank between 66 and 78, that are calculated from the first few seconds of a packet sequence.

We conclude that the total number of incoming packets is the most informative feature. This is expected as different web pages have different resource sizes, that are poorly hidden by encryption or anonymisation, and are unaffected by differing network conditions. The number of incoming and outgoing packets as a fraction of the total number of packets are also informative for the same reason. The least important features are from the padded concentration of outgoing packets list, since the original concentration of outgoing packets lists were of non-uniform size and so have been padded with zeros to give uniform length. Clearly, if most packet sequences have been padded with the same value this will provide a poor criterion for splitting, hence being a feature of low importance. Packet concentration statistics, while making up the bulk of "useless features" also regularly make up a few of the top 30 most important features, they are the first few items that are unlikely to be zero. In other words, the first few values in the packet concentration list do split the data well.

Packet ordering features have rank 4, 7, 12 and 13, indicating these features are a good criterion for classification. Packet ordering features exploit the information leaked via the way in which browsers request resources and the end server orders the resources to be sent. This supports conclusions made by Cai et al. [31], Wang et al. [248] on the importance of packet ordering features.

We also found that the number of incoming and outgoing packets in the first thirty packets, with rank 19 and 20, were more important than the number of incoming and outgoing packets in the last thirty packets, with rank 50 and 55. In the alternative number packets per second feature list the earlier features were a better criterion for splitting than the later features in the list. This supports claims by Wang et al. [248] that the beginning of a packet sequence leaks more information than the end of a packet sequence. In contrast to Bissias et al. [23], we found packet inter-arrival time statistics, with rank between 40 and 70, only slightly increase the attack accuracy, despite being a key feature in their work.

3.5 Attack on hardened defences

For direct comparison we tested our random forest classifier in a closed-world setting on various defences against the *k*-NN attack [248] and the more recent CUMUL[13] [186] attack using the Wang et al. [248] dataset. Note that most of these defences require large bandwidth overheads that may render them unusable for the average client. We test against the following defences:

- **BuFLO [67].** This defence sends packets at a constant size during fixed time intervals. This potentially extends the length of transmission and requires dummy packets to fill in gaps.

- **Decoy pages [185].** This defence loads a decoy page whenever another page is loaded. This provides background noise that degrades the accuracy of an attack. This is essentially a defence that employs multi-tab browsing.

- **Traffic morphing [252].** Traffic morphing shapes a client's traffic to look like another set of web pages. A client chooses the source web pages that

[13]Note we did not have access to original authors implementation code at the time of writing and so recreated the attack using `sklearn.svm` with RBF kernel and `sklearn.gridsearch`.

they would like to defend, as well as a set of target web pages that they would like to make the source processes look like.

- **Tamaraw [246].** Tamaraw operates similarly to BuFLO but fixes packet sizes depending on their direction. Outgoing traffic is fixed at a higher packet interval, this reduces overhead as outgoing traffic is less frequent.

- **Adaptive Padding (AP) [122, 216].** AP protects anonymity by introducing traffic in to statistically unlikely delays between packets in a flow. This limits the amount of extra bandwidth required and does not incur any latency costs. AP uses previously computed histograms of inter-arrival packet times from website loads to determine when a dummy packet should be injected[14]. This is currently the favoured option if padding were to be implemented in Tor [4].

Table 3.1 shows the performance of k-fingerprinting against k-NN and CUMUL under various website fingerprinting defences in a closed-world setting. Under every defence k-fingerprinting is comparable or achieves better results than the k-NN attack and performs significantly better than CUMUL. Note that k-fingerprinting does equally well when traffic morphing is applied compared to no defence. As Lu et al. [157] note, traffic morphing is only effective when the attacker restricts attention to the same features targeted by the morphing process. Our results confirm that attacks can succeed even when traffic morphing is employed. k-fingerprinting also performs nearly 10% better than k-NN when decoy pages are used, which is in effect a marker for how well the attack performs on multi-tab browsing. Due to the dependency of packet length and sequence length features, CUMUL performs substantially worse than the other two attacks under website fingerprinting defences. Though CUMUL uses a similar number of features and is of similar computational efficiency to k-fingerprinting, simple defences remove the feature vector patterns between similar web pages used in CUMUL, rendering the attack ineffectual. More generally, any attack that uses a restricted set of features

[14]As Juarez et al. [122] note, the distribution of histogram bins is dependent on the individual client bandwidth capacity. Optimising histograms for a large number of clients is an open problem. Here we implement a naive version of AP with one master histogram for all clients.

Table 3.1: Attack comparison under various website fingerprinting defences.

defences	**This work**	***k*-NN [248]**	**CUMUL [186]**	**Bandwidth overhead (%)**
No defence	0.91 ±0.01	0.91 ±0.03	0.91 ±0.04	0
Morphing [252]	**0.90** ±0.03	0.82 ±0.06	0.75 ±0.07	50 ±10
Decoy pages [185]	**0.37** ±0.01	0.30 ±0.06	0.21 ±0.02	130 ±20
Adaptive Padding [216]	**0.30** ±0.04	0.19 ±0.03	0.16 ±0.03	54
BuFLO [67]	**0.21** ±0.02	0.10 ±0.03	0.08 ±0.03	190 ±20
Tamaraw [246]	**0.10** ±0.01	0.09 ±0.02	0.08 ±0.03	96 ±9

will suffer greatly from a defence that targets those features. *k*-fingerprinting performs well under defences due to its feature set that captures traffic information not used in CUMUL such as packet timings and burst patterns. The *k*-NN attack performs slightly better than CUMUL but requires an order of magnitude more features than both CUMUL and *k*-fingerprinting. Our attack is both more efficient and more accurate than CUMUL and *k*-NN under defences.

3.6 Attack on the Wang et al. [248] dataset

We first evaluate *k*-fingerprinting on the Wang et al. [248] dataset. This dataset was collected over Tor, and thus implements padding of packets to fixed-size cells (512-bytes) and randomization of request orders [195]. Thus the only available information to *k*-fingerprinting are timing and volume features. We train on 60 out of the 90 instances for each of the 100 monitored web pages; we vary the number of pages on which we train from the 5000 unmonitored web pages. For the attack evaluation we use fingerprints of length 200 and 150 features. Final classification is as described in section 3.2, if all *k* fingerprints agree on classification a test instance is classified as a monitored web page, otherwise it is classified as an unmonitored web page.

The scenario for the attack is as follows: an attacker monitors 100 web pages; they wish to know whether a client is visiting one of those pages, and establish which one. The client can browse to any of these web pages or to 5000 unmonitored web pages, which the attacker classifies in bulk as an unmonitored page.

Using the *k*-fingerprinting method for classifying a web page we measure a TPR of 0.88 ± 0.01 and a FPR of 0.005 ± 0.001 when training on 3500 unmonitored

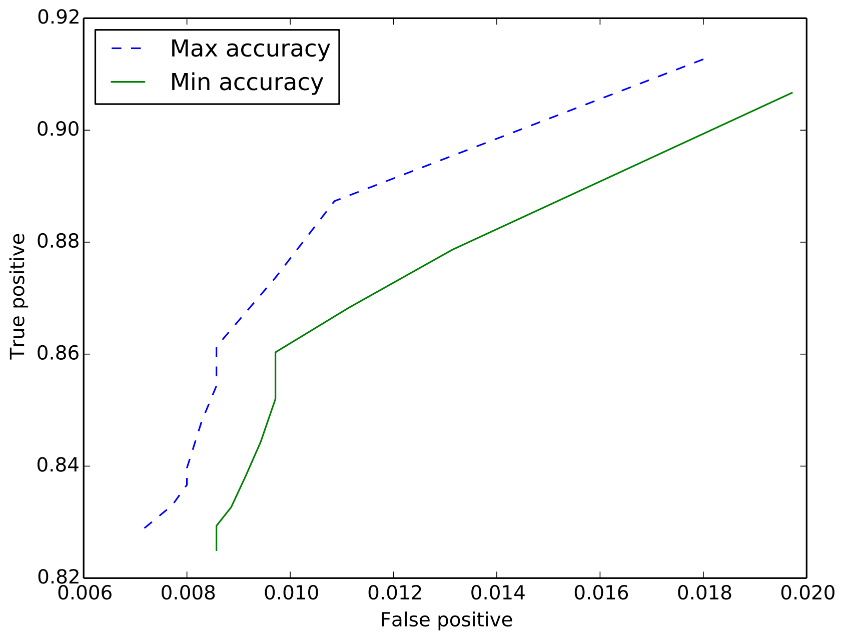


Figure 3.3: Attack results for 1500 unmonitored training pages while varying the number of fingerprints used for comparison, *k*, over 10 experiments.

Table 3.2: *k*-fingerprinting results for *k*=3 while varying the number of unmonitored training pages.

Training pages	TPR	FPR	BDR
0	0.90±0.02	0.750±0.01	0.419
1500	0.88±0.02	0.013±0.007	0.983
2500	0.88±0.01	0.007±0.001	0.993
3500	0.88±0.01	0.005±0.001	0.997
4500	0.87±0.02	0.009±0.001	0.998

web pages and *k*, the number of training instances used for classification, set at *k*=3. *k*-fingerprinting achieves better accuracy than the state-of-the-art *k*-NN attack that has a TPR of 0.85 ± 0.04 and a FPR of 0.006 ± 0.004.

Best results are achieved when training on 3500 unmonitored web pages. Table 3.2 reports TPR and FPR when using different numbers of unmonitored web pages for training with *k*=3. As we train more unmonitored web pages we decrease our FPR with almost no reduction in TPR. After training 3500 unmonitored pages there is no decrease in FPR and so no benefit in training on more unmonitored web pages. This is confirmed by the marginal increase in BDR after training on at least some of the unmonitored set. Furthermore without training on any of the unmonitored web pages, despite the high FPR, the classifier has more than 40% probability of being correct when classifying a web page as monitored.

Figure 3.3 illustrates how classification accuracy changes as, k, the number of fingerprints used for classification changes. For a low k the attack achieves a FPR of around 1%, as we increase the value of k we reduce the number of misclassifications since it is less likely that all k fingerprints will belong to the same label, but we also reduce the TPR. Altering the number of fingerprints used for classification allows an attacker to tune the classifier to either a low FPR or high TPR depending on the desired application of the attack.

We find that altering the number of fingerprints used for classification, k, affects the TPR and FPR more than the number of unmonitored training pages. This suggests that while it is advantageous to have a large world size of unmonitored pages, increasing the number of unmonitored training pages does not increase accuracy of the classifier dramatically. This supports similar claims from Wang et al. [248]. In practice an attacker will need to train on at least some unmonitored pages to increase the BDR, though this does not need to be a substantial amount; training 1500 unmonitored web pages leads to a 98.3% chance the classifier is correct when claiming to have recognised a monitored web page.

Fingerprint length. Changing the length of the fingerprint vector will affect k-fingerprinting accuracy. For a small fingerprint length there may not be enough diversity to provide an accurate measure of distance over all packet sequences. Figure 3.4 shows the resulting TPR and FPR as we change the length of fingerprints in the Wang et al. [248] data set. We set k=1 and train on 4000 unmonitored web pages. Using only a fingerprint of length one results in a TPR of 0.51 and FPR of 0.904. Clearly using a fingerprint of length one results in a high FPR since there is a small universe of leaf symbols from which to create the fingerprint. A fingerprint of length 20 results in a TPR of 0.87 and FPR of 0.013. After this there are diminishing returns for increasing the length of the fingerprint vector.

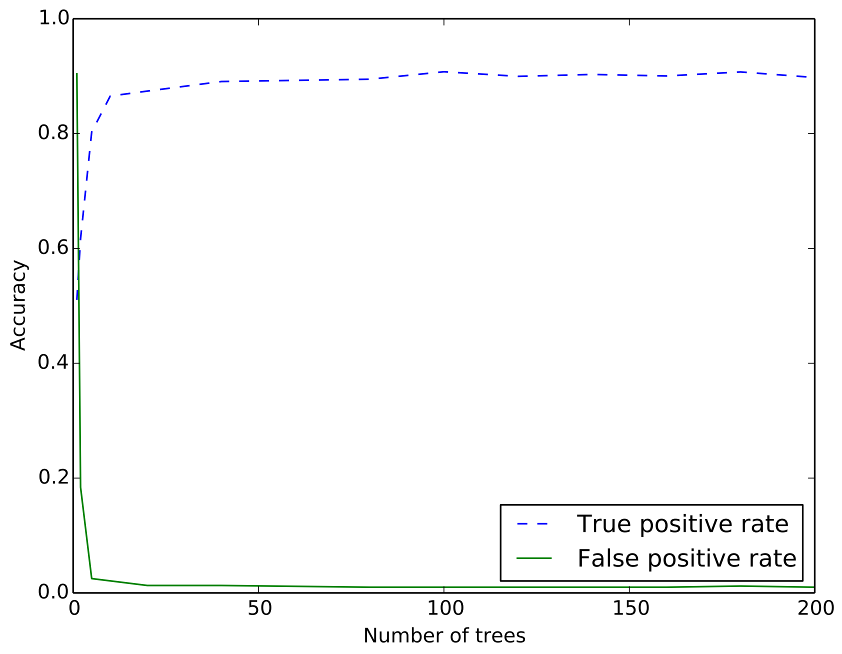


Figure 3.4: Accuracy of k-fingerprinting as we vary the number of trees in the forest.

3.7 Attack evaluation on DS_{Tor}

We now evaluate k-fingerprinting on DS_{Tor}. First we evaluate the attack given a monitored set of the top 55 Alexa web pages, with 100 instances for each web page. Then we evaluate the attack given a monitored set of 30 Tor hidden services, with 80 instances for each hidden service. The unmonitored set remains the same for both evaluations, the top 100,000 Alexa web pages with one instance for each web page.

3.7.1 Alexa web pages monitored set

Table 3.3 shows the accuracy of k-fingerprinting as the number of unmonitored training pages is varied. For the monitored web pages, 70 instances per web page were trained upon and testing was done on the remaining 30 instances of each web page. As expected, the FPR decreases as the number of unmonitored training samples grows. Similar to section 3.6, there is only a marginal decrease in TPR while we see a large reduction in the FPR as the number of training samples grows. Meaning an attacker will not have to compromise on TPR to decrease the FPR; when scaling the number of unmonitored training samples from 2% to 16% of the entire set

Table 3.3: Attack results on top Alexa sites for *k*=2 while varying the number of unmonitored training pages.

Training pages	**TPR**	**FPR**	**BDR**
2000	0.93 ±0.03	0.032 ±0.01	0.33
4000	0.93 ±0.01	0.018 ±0.007	0.47
8000	0.92 ±0.01	0.008 ±0.002	0.67
16000	0.91 ±0.02	0.003 ±0.001	0.86

the TPR decreases from 93% to 91% while the FPR decreases from 3.2% to 0.3%. There is a more pronounced shift in BDR with the increase of unmonitored training pages, however an attacker needs to train on less than 10% of the entire dataset to have nearly 70% confidence that classifier was correct when it claims to have detected a monitored page.

Clearly the attack will improve as the number of training samples grows, but in reality an attacker may have limited resources and training on a significant fraction of 100,000 web pages may be unfeasible. Figure 3.5 shows the TPR and FPR of *k*-fingerprinting as the number of unmonitored web pages used for testing grows while the number of unmonitored web pages used for training is kept at 2000, for different values of *k*. We may think of this as the evaluation of success of *k*-fingerprinting as a client browses to more and more web pages over multiple browsing sessions. Clearly for a small *k*, both TPR and FPR will be comparatively high. Given that, with *k*=5 only 2.5% of unmonitored web pages are falsely identified as monitored web pages, out of 98,000 unmonitored web pages.

3.7.2 Hidden services monitored set

Table 3.4 shows the accuracy of *k*-fingerprinting as the number of unmonitored training pages is varied. For the monitored set, 60 instances per hidden service were trained upon and testing was done on the remaining 20 instances of each hidden service. Again we observe a marginal loss of TPR and a large reduction in FPR as the number of training samples grows. When scaling the number of unmonitored training samples from 2% to 16% of the entire set the TPR decreases from 82% to 81% while the FPR decreases by an order of magnitude from 0.2% to 0.02%. As a result, when training on 16% of the unmonitored set only 16 unmonitored web

Table 3.4: Attack results on Tor hidden services for k=2 while varying the number of unmonitored training pages.

Training pages	TPR	FPR	BDR
2000	0.82 ±0.03	0.0020 ±0.0015	0.72
4000	0.82 ±0.04	0.0007 ±0.0006	0.88
8000	0.82 ±0.02	0.0002 ±0.0001	0.96
16000	0.81 ±0.02	0.0002 ±0.0002	0.97

pages out of 84,000 were misclassified as a Tor hidden service. In comparison to the Alexa web pages monitored set the TPR is around 10% lower, while the FPR is also greatly reduced. This is evidence that Tor hidden services are easy to distinguish from standard web pages loaded over Tor. There is also a higher but more gradual increase in BDR compared to standard web pages. An attacker need only train on as little as 2% of unmonitored pages to have over 70% confidence that classification of a monitored page was correct, with this rising to 97% when training on 16% of the unmonitored dataset.

Similar to fig. 3.5, fig. 3.6 shows the TPR and FPR of k-fingerprinting as the number of unmonitored web pages used for testing grows while the number of unmonitored web pages used for training is kept at 2000, for different values of k. Both the TPR and FPR is lower than in fig. 3.5. For example using k=5, the FPR is 0.2% which equates to only 196 out of 98,000 unmonitored pages being falsely classified as monitored pages.

From fig. 3.7 we observe that the BDR of both standard web pages and hidden services monitored sets depends heavily on not only the world size but the number of fingerprints used for classification. With k=10, when a web page is classified as a monitored hidden service page, there is over an 80% chance that the classifier was correct, despite the unmonitored world size (98,000) being over 160 times larger than the monitored world size (600). The high BDR regardless of the disparity in world sizes makes it clear that our attack is highly effective under realistic large world size conditions.

It is clear that an attacker need only train on a small fraction of data to launch a powerful fingerprinting attack. It is also clear that Tor hidden services are easily distinguished from standard web pages, rendering them vulnerable to website fingerprinting attacks. We attribute the lower FPR of Tor hidden services when compared to a monitored training set of standard web page traffic to this distinguishability. A standard web page over Tor is more likely to be confused with another standard web page than a Tor hidden service.

Comparison with Kwon et al. [138] hidden services results. For comparison we ran *k*-fingerprinting on the dataset used in the Kwon et al. [138] study on fingerprinting hidden services. This dataset simulated a client connecting to a hidden service. The dataset consists of 50 instances for each of 50 monitored hidden services and an unmonitored set of 950 hidden services. When training on 100 of the unmonitored pages they report attack accuracy of 0.9 TPR and 0.4 FPR. *k*-fingerprinting achieved a similar true positive rate but the FPR is reduced to 0.22. This FPR reduction in comparison with Kwon et al. [138] continued regardless of the amount of data used for training.

3.8 Attack evaluation on DS_{Norm}

Besides testing on DS_{Tor}, Wang et al. [248] dataset and the Kwon et al. [138] dataset we tested the efficacy of *k*-fingerprinting on DS_{Norm}. This allows us to establish how accurate *k*-fingerprinting is over a standard encrypted web browsing session or through a VPN.

3.8.1 Attack on encrypted browsing sessions

An encrypted browsing session does not pad packets to a fixed size so the attacker may extract the following features in addition to time features:

- **Size transmitted.** For each packet sequence we extract the total size of packets transmitted, in addition, we extract the total size of incoming packets and the total size of outgoing packets.

- **Size transmitted statistics.** For each packet sequence we extract the average, variance, standard deviation and maximum packet size of the total sequence, the incoming sequence and the outgoing sequence.

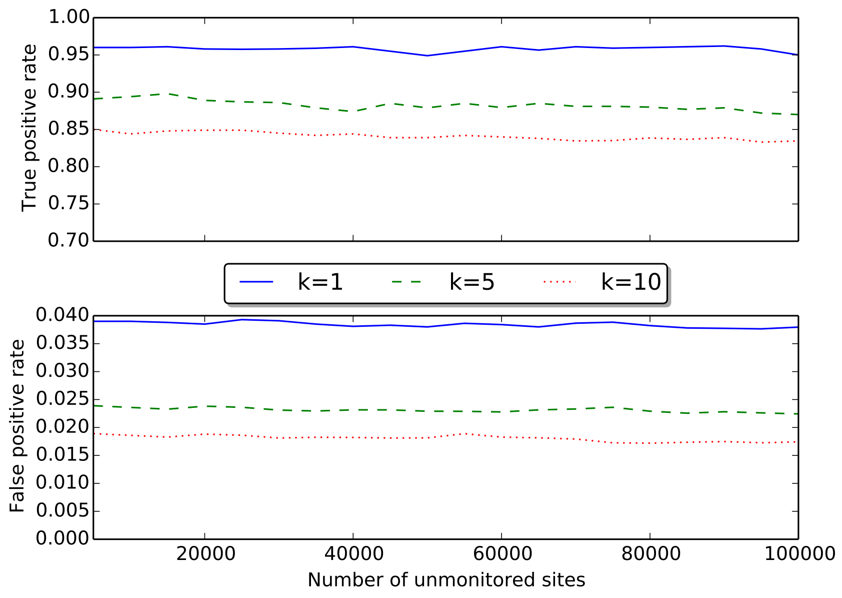


Figure 3.5: Attack accuracy on DS_{Tor} with Alexa monitored set.

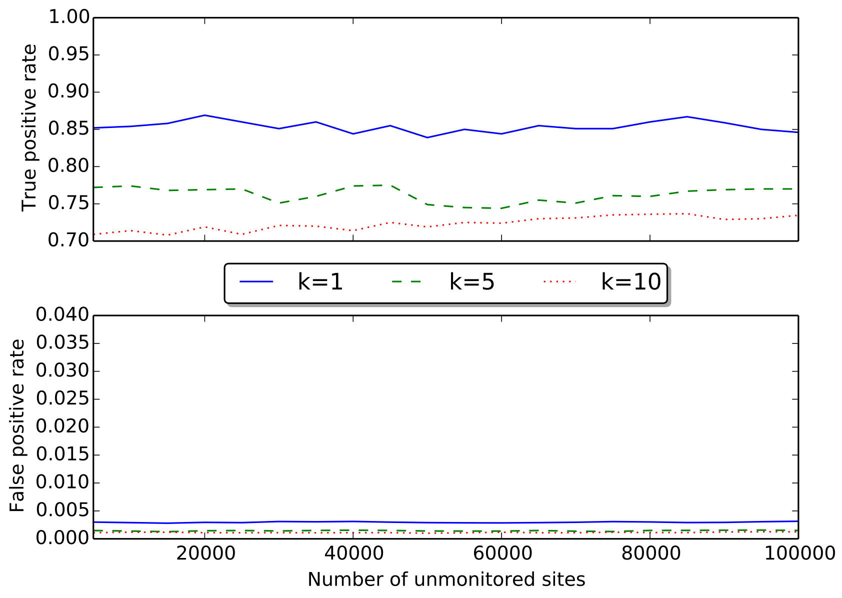


Figure 3.6: Attack accuracy on DS_{Tor} with Tor hidden services monitored set.

Apart from this modification in available features, the attack setting is similar: An attacker monitors a client browsing online and attempts to infer which web pages they are visiting. The only difference is that browsing with the Transport Layer Security (TLS) protocol, or Secure Sockets Layer (SSL) protocol, versions 2.0 and 3.0, exposes the destination IP address and port. The attack is now trying to

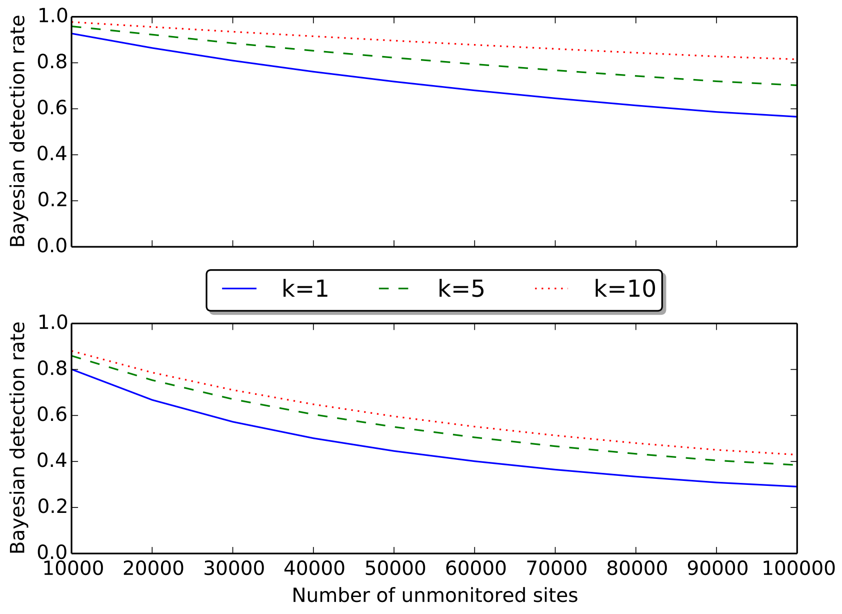


Figure 3.7: BDR for hidden services monitored set (above) and Alexa monitored set (below).

Table 3.5: Attack results with packet size features for a varying number of unmonitored training pages.

Training pages	**TPR**	**FPR**	**BDR**
0	0.95 ±0.01	0.850 ±0.010	0.081
2000	0.90 ±0.01	0.01 ±0.004	0.908
4000	0.87 ±0.02	0.004 ±0.001	0.976
6000	0.86 ±0.01	0.005 ±0.002	0.990

infer which web page the client is visiting from the known website[15].

The attacker monitors 55 web pages; they wish to know if the client has visited one of these pages. The client can browse to any of these web pages or to 7000 other web pages, which the attacker does not care to classify other than as unmonitored. We train on 20 out of the 30 instances for each monitored page and vary the number of unmonitored pages on which we train.

Despite more packet sequence information to exploit, the larger cardinality of world size gives rise to more opportunities for incorrect classifications. The attack achieves a TPR of 0.87 and a FPR of 0.004. We achieved best results when training on 4000 unmonitored web pages. Table 3.5 reports results for training on different

[15]Note that the datasets are composed of traffic instances from some websites without SSL and TLS, as well as websites using the protocols.

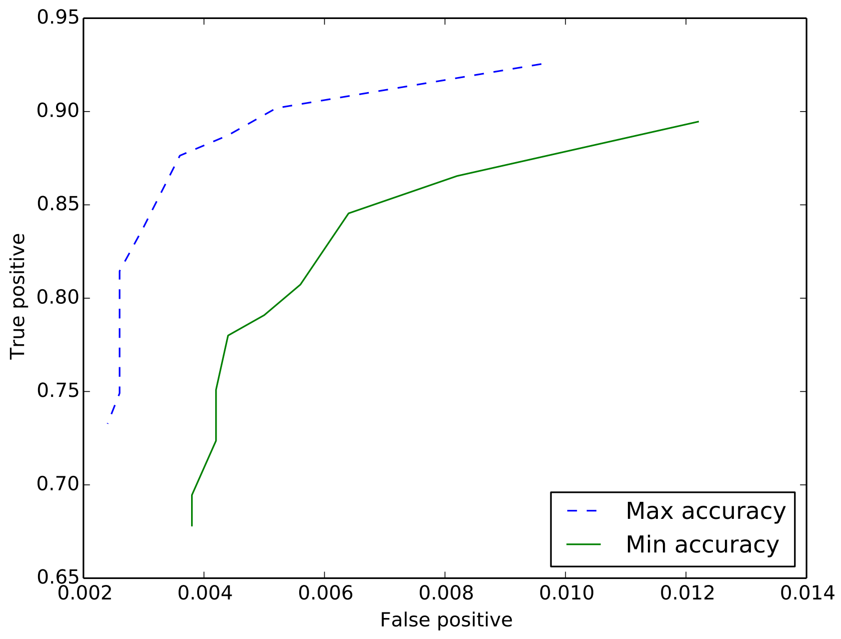


Figure 3.8: Attack results for 2000 unmonitored training pages while varying the number of fingerprints used for comparison, k, over 10 experiments.

Table 3.6: Attack results without packet size features for a varying number of unmonitored training pages.

Training pages	TPR	FPR	BDR
0	0.90 ±0.01	0.790 ±0.020	0.082
2000	0.83 ±0.01	0.009 ±0.001	0.910
4000	0.81 ±0.02	0.006 ±0.001	0.961
6000	0.80 ±0.02	0.005 ±0.001	0.989

numbers of unmonitored web pages, with $k = 2$. Figure 3.8 shows our results when modifying the number of fingerprints used (k) and training on 2000 unmonitored pages. We find that altering the number of unmonitored training pages decreases the FPR while only slightly decreasing the TPR. This mirrors our experimental findings from DS_{Tor} and the Wang et al. [248] dataset.

3.8.2 Attack without packet size features

DS_{Norm} was not collected via Tor and so also contains packet size information. We remove this to allow for comparison with DS_{Tor} and the Wang et al. [248] dataset which was collected over Tor. This also gives us a baseline for how much more powerful k-fingerprinting is when we have additional packet size features available.

We achieved a TPR of 0.81 and FPR of 0.005 when training on 5000 unmonitored web pages. Table 3.6 shows our results at other sizes of training samples, with k=2. Removing packet size features reduces the TPR by over 0.05 and increases the FPR by 0.001. Clearly packet size features improve our classifier in terms of correct identifications but do not decrease the number of unmonitored test instances that were incorrectly classified as a monitored page. Despite the difference in FPR when including packet size information, the BDR is similar, suggesting that BDR is dominated by the amount of information that can be trained upon.

Closed-World. In the closed-world setting in which the client can only browse within the 55 monitored web pages k-fingerprinting is 0.91, compared to 0.96 when packet size features are available. In the closed-world setting attack accuracy improves by 5% when we include packet size features.

3.9 Fine grained open-world false positives on Alexa monitored set of DS_{Tor}

The classification error is not uniform across all web pages[16] Some pages are misclassified many times, and confused with many others, while others are never misclassified. An attacker can leverage this information to estimate the misclassification rate of each of the web page classes instead of using the global average misclassification rate. A naive approach to this problem would be to first find which fingerprints contribute to the many misclassifications and remove them. Our analysis shows that the naive approach of removing "bad" fingerprints that contribute to many misclassifications will ultimately lead to a higher misclassification rate. Figure 3.9 shows the 60 fingerprints that cause the most misclassifications, and also shows for those same fingerprints the number of correct classifications they contribute towards. Nearly all "bad" fingerprints actually contribute to many correct classifications. One may think it may still be beneficial to remove these fingerprints as the cumulative sum of misclassifications outweigh the number of correct classifications. This removal will then promote fingerprints that are further away in terms

[16]See additional evidence in appendix A.2.

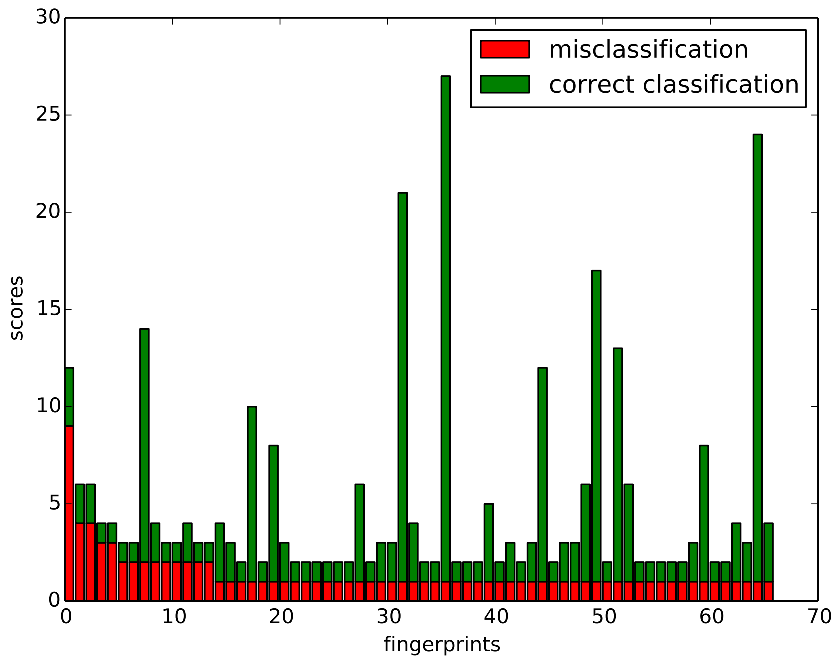


Figure 3.9: The fingerprints that lead to the most misclassifications and the correct classifications they contribute towards. Training on 2% of unmonitored pages with k=3.

of Hamming distance from the fingerprinting that is being tested, which will lead to a greater number of misclassifications.

Instead an attacker can use their training set of web pages to estimate the TPR and FPR of each web page class, by splitting the training set in to a smaller training set and validation set. Since both sets are from the original training set the attacker has access to the true labels. The attacker then computes the TPR and FPR rates of each monitored class, this is used as an estimation for TPR and FPR when training on the entire training set and testing on new traffic instances. More specifically we split, for the monitored training set of 70 instance for each of the Alexa top 55 web pages, into smaller training sets of 40 instances and validation sets of 30 instances. This is used as a misclassification estimator for the full monitored training set against the monitored test set of 30 instances per class. Similarly we split the unmonitored training in half, one set as a smaller training set and the other as a validation set.

Figure 3.10 shows the TPR and FPR estimation accuracy for 2000 unmonitored training pages. Monitored classes are first ordered from best to worst in terms of their classification accuracy. Even with a small unmonitored training set of 2000 web pages, which is then split in to a training set of 1000 web pages and a validation set of 1000 web pages, an attacker can accurately estimate the FPR of the attack if some of the monitored classes were removed. For example, using only the best 20 monitored classes (in terms of TPR), an attacker would estimate that using those 20 classes as a monitored set, the FPR would be 0.012. Using the entire dataset we see that the true FPR of these 20 classes is 0.010; the attacker has nearly precisely estimated the utility of removing a large fraction of the original monitored set.

There is a small difference between estimated and the actual FPR in both fig. 3.10 and fig. 3.11. There is little benefit in training more unmonitored data if the attacker wants to accurately estimate the FPR; fig. 3.10 has a similar gap between the estimated FPR and true FPR when compared to fig. 3.11. It is evident even with a small training set, an attacker can identify web pages that are likely to be misclassified and then accurately calculate the utility of removing these web pages from their monitored set. Due to the overwhelmingly large world size of unmonitored web pages the BDR of fig. 3.10 does not grow dramatically with the removal of web pages that are likely to be misclassified; using the entire monitored set the BDR is 0.33, removing half of the monitored web pages the BDR is 0.35.

3.10 Attack summary & discussion

Attack Summary. Best attack results on datasets were achieved when training on approximately two thirds of the unmonitored web pages. Despite this, results from DS_{Tor} show that an attacker can achieve a small false positive rate while only training on 2% of the unmonitored data. Training on 2% of 100,000 unmonitored web pages greatly reduces the attack set up costs while only marginally reducing the accuracy compared to training on more data, providing a realistic setting under which an attack could be launched. Results on all datasets also suggest that altering k, the number of fingerprints used for classification, has a greater influence on accuracy

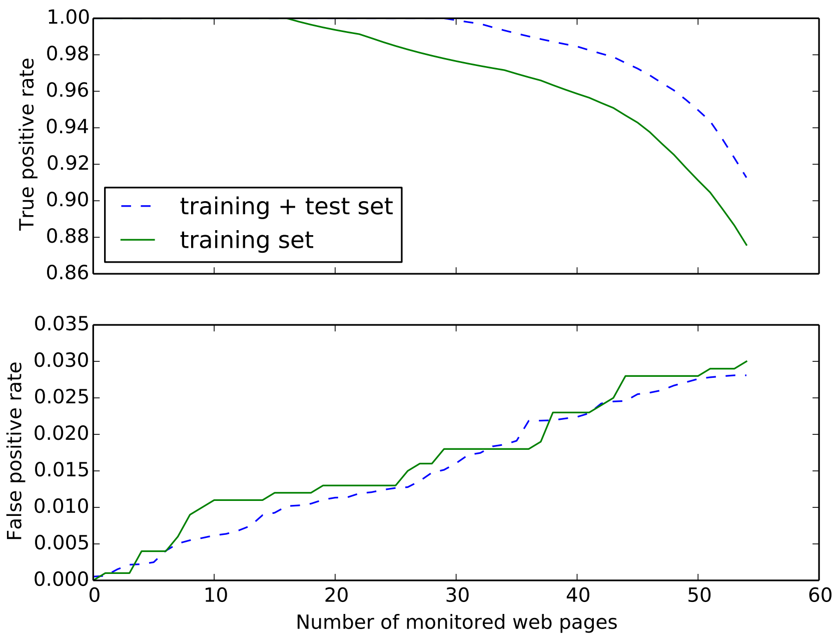


Figure 3.10: Rates for training on 1000 unmonitored pages, testing on 1000, and comparison when training on the full 2000 unmonitored pages and testing on the remaining 98000 unmonitored pages in DS_{Tor}, k=3.

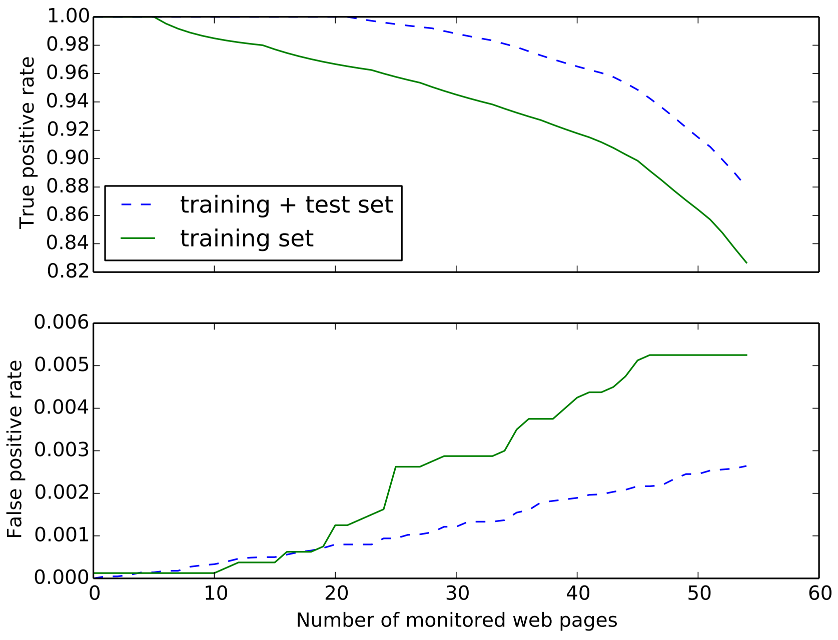


Figure 3.11: Rates for training on 8000 unmonitored pages, testing on 8000, and comparison when training on the full 16000 unmonitored pages and testing on the remaining 84000 unmonitored pages in DS_{Tor}, k=3.

than the number of training samples[17].

[17]Figure A.6 illustrates that compared to training on a small number of monitored instances increasing the size of the monitored training set only incrementally increases accuracy.

k-fingerprinting is robust; our technique achieves the same accuracy regardless of the type of monitored set or the manner in which it was collected (through Tor or standard web browsers). The monitored set in the Wang et al. [248] dataset consists of real world censored websites, the Kwon et al. [138] monitored set consist of Tor hidden services and the $DS_{Tor/Norm}$ monitored sets were taken from Tor hidden services and top Alexa websites. We do see a reduction in FPR when the target monitored set are Tor hidden services due to the distinguishability between the hidden services and unmonitored standard web pages.

We also highlight the non-uniformity of classification performance: when a monitored web page is misclassified, it is usually misclassified on multiple tests. We show that an attacker can use their training set to estimate the error rate of *k*-fingerprinting per web page, and select targets with low misclassification rates.

Computational Efficiency. *k*-fingerprinting is more accurate and uses fewer features than state-of-the-art attacks. Furthermore *k*-fingerprinting is faster than current state-of-the-art website fingerprinting attacks. On the Wang et al. [248] dataset training time for 6,000 monitored and 2,500 unmonitored training pages is 30.738 CPU seconds on an 1.4 GHz Intel Core i5z. The *k*-NN attack [248] has training time per round of 0.064 CPU seconds for 2500 unmonitored training pages. For 6,000 rounds training time is 384.0 CPU seconds on an AMD Opteron 2.2 GHz cores. This can be compared to around 500 CPU hours using the attack described by Cai et al. [31]. Testing time per instance for *k*-fingerprinting is around 0.1 CPU seconds, compared to 0.1 CPU seconds to classify one instance for *k*-NN and 450 CPU seconds for the attack described by Cai et al. [31].

Discussion. Website fingerprinting research has been criticised for not being applicable to real-world scenarios [121, 196]. We have shown that a website fingerprinting attack can scale to the number of traffic instance an attacker may sample over a long period of time with a high BDR and low FPR. However, we did not consider the cases where background traffic may be present, for example from multitab browsing, or the effect that short-lived websites will have on our attack. Gu et al. [86] show in their work that a simple Naive-Bayes attack achieves highly accurate results even when a client browses in multiple tabs. Wang and Goldberg [247] also show that website fingerprinting is effective in practical scenarios. With no prior attack set-up to tailor to a multi-tab browsing session our attack was able to classify nearly 40% of monitored pages correctly when the decoy defence was employed.

Website content rapidly changes which will negatively affect the accuracy of a website fingerprinting attack [121]. As the content of a website changes so will the generated packet sequences, if an attacker cannot train on this new data then an attack will suffer. However we note that an attack will suffer from the ephemeral nature of websites at different rates depending on the type of website being monitored. For example, an attack monitoring a news or social media site can expect a faster degradation in performance compared to an attack monitoring a landing page of a top 10 Alexa site [1]. Also note Tor does not cache by default, so if in the realistic scenario where an attacker wanted to monitor `https://socialmediawebsite.com` a client would be forced to navigate to the social media website landing page, which is likely to host content that is long lived and not subject to change. The problem of content change is weakened when fingerprinting Tor hidden services. As shown by Kwon et al. [138] hidden pages show minimal changes in comparison to non-hidden pages, resulting in devastatingly accurate attacks on hidden services that can persist.

3.11 Has *k*-fingerprinting withstood the test of time?

The *k*-fingerprinting attack was published in 2016, and has since been used a baseline to compare with new attacks and defences in this space. Here, we survey the performance of *k*-fingerprinting attack in comparison to these newer works.

It will come as no surprise to the interested reader to learn that website fingerprinting research has not been immune to the 'deep learning revolution'. Neural networks based approaches to website fingerprinting attacks have been proposed by Bhat et al. [20], Rimmer et al. [208], Sirinam et al. [220]. Each of these works reported improved attack results compared to non-deep learning based approaches such as *k*-fingerprinting, *k*-NN [248], and CUMUL [186]. For example, Sirinam et al. [220] develop the Deep Fingerprinting (DF) attack using a convolutional neural network, and report that in a closed-world setting on the top 100 Alexa web pages, DF achieves 98.3% accuracy while *k*-fingerprinting achieves 95.5% accuracy. However, recent works have brought into question how much deep learning based approaches improve website fingerprinting attacks. Similar to adaptive padding [122, 216], Gong and Wang [80] introduce a website fingerprinting defence, FRONT, that randomly insert dummy packets into the front portion of traces, and use a dataset consisting of the top 100 Alexa monitored web pages each visited 100 times as the monitored set, and 10,000 other unmonitored web pages, for evaluation. With no defence in place, *k*-NN has an F1 score of 0.86, CUMUL has an F1 score of 0.76, *k*-fingerprinting has an F1 score of 0.93, and DF has an F1 score of 0.94. However, under the FRONT defence, the *k*-NN F1 score shrinks to 0.048, the CUMUL F1 score shrinks to 0.18, the *k*-fingerprinting F1 score shrinks to 0.54, and the DF F1 score shrinks to 0.47. A similar observation is made by De la Cadena et al. [53], who showed that while in the undefended setting, DF outperforms *k*-fingerprinting (94.5% accuracy versus 92.09%), there exists settings under a traffic splitting defence where *k*-fingerprinting consistently outperforms DF.

Furthermore, Liang et al. [147] have recently demonstrated that a random forest approach can outperform deep learning based approaches to website classification providing that the number of classes to identify is small. Work from Di Martino et al. [56] studied how effectively attacks perform at fingerprinting HTTPS traffic in realistic scenarios where caching and dynamic content may be present, and show that under these settings, *k*-fingerprinting outperforms both DF and the deep learning based website fingerprinting attack presented by Bhat et al. [20].

Finally, we have recently shown that deep learning based embedding models can improve upon other techniques in the large open-world setting [162]. Our experiments show that adaptive adversaries can reliably uncover the webpage visited by a user among several thousand potential pages, even under considerable distributional shift (e.g., the webpage contents change significantly over time). Such adversaries could infer the products a user browses on shopping websites or log the browsing habits of state dissidents on online forums and encyclopedias. Our technique achieves 90% accuracy in a top-15 setting where the model distinguishes the article visited out of 6,000 Wikipedia webpages, while the same model achieves 80% accuracy in a dataset of 13,000 classes that were not included in the training set.

Chapter 4

Steganography guided by machine learning

We now move on to assessing the practicality of using machine learning to develop steganography schemes, inspired by the work of Abadi and Andersen [5] on learning cryptographic schemes.

Steganography and cryptography both provide methods for secret communication. *Authenticity* and *integrity* of communications are central aims of modern cryptography. However, traditional cryptographic schemes do not aim to hide the presence of secret communications. Steganography conceals the presence of a message by embedding it within a communication the adversary does not deem suspicious. Recent details of mass surveillance programs have shown that meta-data of communications can lead to devastating privacy leakages (cf. EFF's report on mass surveillance programs [52].). General Michael Hayden, former director of the NSA and the CIA, famously stated that they "kill people based on meta-data" [92]; the mere presence of a secret communication can have life or death consequences even if the content is not known. Concealing both the content *as well as* the presence of a message is necessary for privacy sensitive communication.

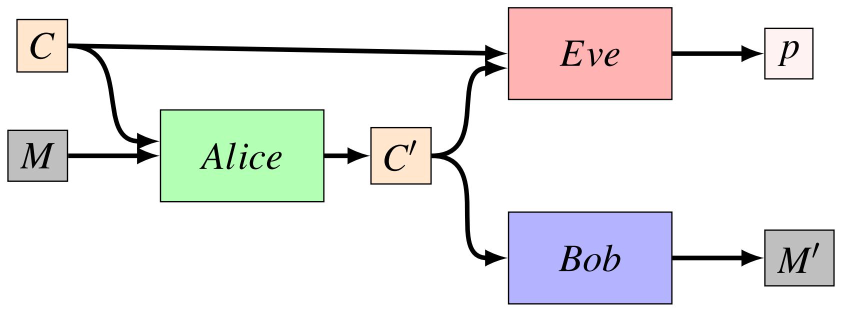
Steganographic algorithms are designed to hide information within a *cover* message such that the cover message appears unaltered to an external adversary. A great deal of effort is afforded to designing steganographic algorithms that minimise the perturbations within a cover message when a secret message is embedded within, while allowing for recovery of the secret message. In this chapter, we ask if a steganographic algorithm can be *learnt*, without human domain knowledge. Note that steganography only aims to hide the presence of a message. Thus, it is nearly always the case that the message is encrypted prior to embedding using a standard cryptographic scheme; the embedded message is therefore indistinguishable from a random string. The receiver of the steganographic image will then decode to reveal the ciphertext of the message and then decrypt using an established shared key.

For the design of steganographic techniques, we leverage ideas from the field of adversarial training [81]. Typically, adversarial training is used to train generative models on tasks such as image generation and speech synthesis. We design a scheme that aims to embed a secret message within an image. Our task is discriminative, the embedding algorithm takes in a *cover* image and produces a *steganographic* image, while the adversary tries to learn weaknesses in the embedding algorithm, resulting in the ability to distinguish cover images from steganographic images.

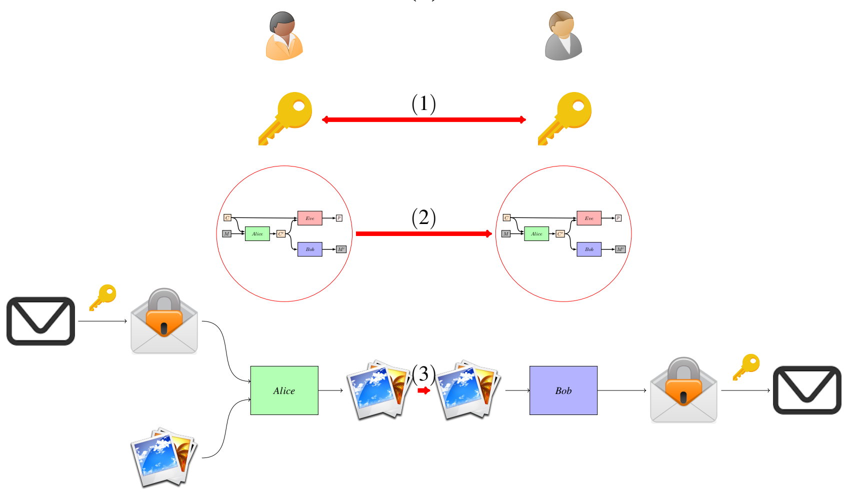
The success of a steganographic algorithm or a steganalysis technique over one another amounts to its ability to model the cover distribution correctly [72]. So far, steganographic schemes have used human-based rules to 'learn' this distribution and perturb it in a way that disrupts it least. However, steganalysis techniques commonly use machine learning models to learn the differences in distributions between the cover and steganographic images. Based on this insight we pursue the following hypothesis:

Hypothesis: Machine learning is as capable as human-based rules for the task of modelling the cover distribution, and so naturally lends itself to the task of designing steganographic algorithms, as well as performing steganalysis.

Next, we introduce the steganographic algorithm designed through a novel adversarial training scheme. We show that our scheme can be successfully implemented in practice between two communicating parties, and additionally that with supervised training, the steganalyser, Eve, can compete against state-of-the-art ste-



(a)



(b)

Figure 4.1: **(a)** Diagram of the training game. **(b)** How two parties, Carol and David, use the scheme in practice: (1) Two parties establish a shared key. (2) Carol trains the scheme on a set of images. Information about model weights, architecture and the set of images used for training is encrypted under the shared key and sent to David, who decrypts to create a local copy of the models. (3) Carol then uses the *Alice* model to embed a secret encrypted message, creating a steganographic image. This is sent to David, who uses the *Bob* model to decode the encrypted message and subsequently decrypt.

ganalysis methods. To the best of our knowledge, this is one of the first real-world applications of adversarial training, aside from traditional adversarial learning applications such as image generation tasks.

4.1 Learning objectives

Our training scheme involves three parties: Alice, Bob and Eve. Alice sends a message to Bob, Eve can eavesdrop on the link between Alice and Bob and would like to discover if there is a secret message embedded within their communication. In classical steganography, Eve (the *steganalyser*) is passed both unaltered images, called *cover* images, and images with secret messages embedded within, called *steganographic* images. Given an image, Eve places a confidence score of how likely this is a cover or steganographic image. Alice embeds a secret message within the cover image, producing a steganographic image, and passes this to Bob. Bob knows the embedding process and so can recover the message. In our scheme, Alice, Bob and Eve are neural networks. Alice is trained to learn to produce a steganographic image such that Bob can recover the secret message, and such that Eve can do no better than randomly guess if a sample is a cover or steganographic image.

The full scheme is depicted in fig. 4.1a: Alice receives a cover image, C, and a secret encrypted message, M, as inputs. Alice outputs a steganographic image, C', which is given to both Bob and Eve. Bob outputs M', the secret message he attempts to recover from C'. We say Bob performs perfectly if $M = M'$. In addition to the steganographic images, Eve also receives the cover images. Given an input X, Eve outputs the probability, p, that $X = C$. Alice tries to learn an embedding scheme such that Eve always outputs $p = \frac{1}{2}$. We do not train Eve to maximise her prediction error, since she can then simply flip her decision and perform with perfect classification accuracy. Figure 4.1b shows how the scheme should be used in practice if two people wish to communicate a steganographic message using our scheme. The cost of sending the encrypted model information from Carol to David is low, with an average of 70MB. Note that in fig. 4.1b, steps (1) and (2), the set-up of the shared key and sharing of model information, is performed offline. We assume, as is common in cryptographic research, that this initial set-up phase is not visible to an adversary.

At the beginning of training, a human can easily separate cover images from steganographic images, as Alice has not learned yet how to embed the secret message such that there is no visible difference in the cover image. However, we train Eve much like a discriminator in a GAN, where we tie her predictive power to the embedding capacity of Alice. When Alice produces a steganographic image that does not resemble the cover image, Eve does not have the ability to perfectly separate cover from steganographic images. As training continues, Eve becomes better at her task, but then so does Alice as her weights are updated, in part, based on the loss of Eve.

Similarly to Abadi and Andersen [5], we let $\theta_A, \theta_B, \theta_C$ denote the parameters of Alice, Bob and Eve, respectively. We write $A(\theta_A, C, M)$ for Alice's output on C and M, $B(\theta_b, C')$ for Bob's output on C', and $E(\theta_E, C, C')$ for Eve's output on C and C'. Let L_A, L_B, L_C denote the loss of Alice, Bob and Eve, respectively. Then, we have the following relations:

$$B(\theta_b, C') = B(\theta_b, A(\theta_A, C, M))$$
$$E(\theta_E, C, C') = E(\theta_E, C, A(\theta_A, C, M))$$

We set Bob's loss (the secret message reconstruction loss), to be the Euclidean distance between M and M'. In practical applications, Bob can use error correction to recover the signal is errors accumulate during transmission.

$$\begin{aligned} L_B(\theta_A, \theta_B, M, C) &= d(M, B(\theta_b, C')) \\ &= d(M, B(\theta_b, A(\theta_A, C, M)) \\ &= d(M, M') \end{aligned}$$

As is common with GAN discriminator implementations, we set the Eve's loss to be sigmoid cross entropy loss:

$$L_E(\theta_A, \theta_E, C, C') = -y \cdot \log(E(\theta_E, x)) - (1-y) \cdot \log(1 - E(\theta_E, x)),$$

where $y = 0$ if $x = C'$ and $y = 1$ if $x = C$. Alice's loss is given as a weighted sum of Bob's loss, Eve's loss on steganographic images, and an additional reconstructive loss term:

$$L_A(\theta_A, C, M) = \lambda_A \cdot d(C, C') + \lambda_B \cdot L_B + \lambda_E \cdot L_E(\theta_E, C'),$$

where $d(C, C')$ is the Euclidean distance between the cover image and the steganographic image, and $\lambda_A, \lambda_B, \lambda_E \in \mathbb{R}$ define the weight given to each respective loss term.

Our goal is not only to explore whether neural networks can produce steganographic embedding algorithms in an unsupervised manner, but whether they are competitive against steganographic algorithms like HUGO, WOW and S-UNIWARD, that have been designed by steganography experts. We did not intend to encode a specific algorithm within the neural network, rather we would like to give the networks the opportunity to devise their own.

4.2 Architecture

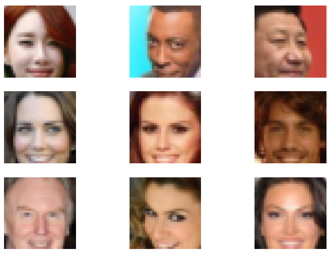
Alice (Steganographic Encoder). Alice accepts as input a cover image and a random n-bit binary message. The cover image is flattened and concatenated with the secret message, which is then fed into the neural network. We would like the locality of the embedding process to be learned instead of fixed; the first layer is fully connected to the next, allowing the secret message to be combined with any region of the cover image. We then reshape to $4 \times 4 \times I \cdot 8$, where I is width of the cover image. Each of the next four layers are a sequence of upsampling fractionally-strided convolutions, batch normalisation and ReLU [176], except for the final layer where tanh [141] is used as the activation function.

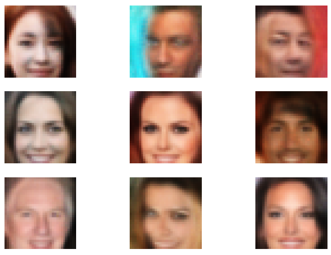
Bob (Steganographic Decoder). Bob accepts as input the steganographic image output by Alice. Bob uses a deep convolutional architecture to extract the important bits from the output of Alice. It is made up of four fractionally-strided convolutions and one final fully connected layer. Leaky ReLU [160] is used in all but the final layer, where tanh is used. Batch normalisation is applied at each layer.

Eve (steganalyser). Eve accepts as input both cover images and steganographic images. Eve has a similar architecture to Bob but uses the sigmoid activation function instead of tanh at the final layer to output probabilities from the logits.

4.3 Evaluation of the steganographic scheme

As a proof of concept, we implemented our adversarial training scheme on two image datasets: celebrity faces in the wild (celebA) [154] and a standard steganog-


(a) Cover images.


(b) Steganographic images (0.1bpp).

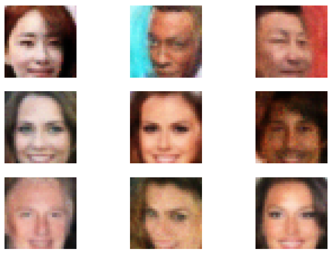
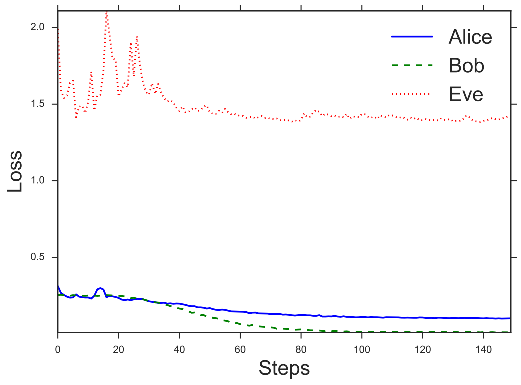
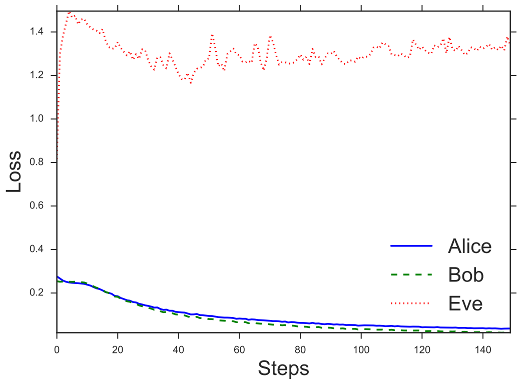

(c) Steganographic images (0.4bpp).

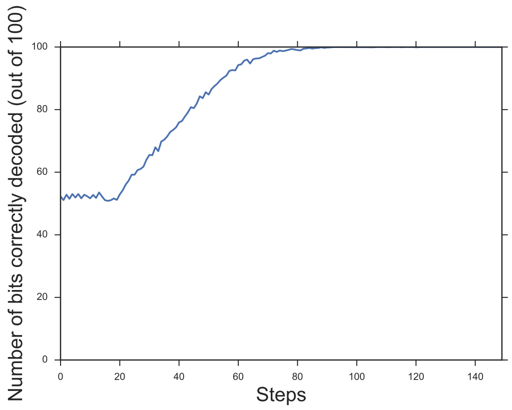
Figure 4.2: Cover and steganographic images from the celebA dataset, with embedding rates of 0.1bpp and 0.4bpp.



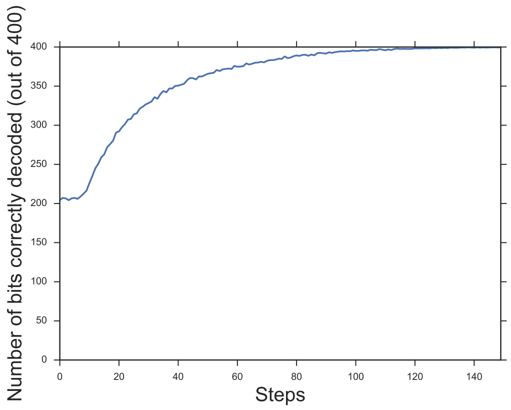
(a) Alice, Bob and Eve error for 0.1bpp.



(b) Alice, Bob and Eve error for 0.4bpp.



(c) Bob decoding success for 0.1bpp.



(d) Bob decoding success for 0.4bpp.

Figure 4.3: Results on celebA dataset at an embedding rate of 0.1bpp and 0.4bpp.

raphy research dataset, BOSS[1]. We concatenate an n-bit random message, with each sample of each dataset. We vary the size of the message, n, to test the limits of the

[1]`http://agents.fel.cvut.cz/boss/index.php?mode=VIEW&tmpl=materials`

amount of information that can be effectively hidden inside a cover image. This corresponds to the number of bits per pixel (bpp) that can be hidden by a steganographic algorithm, a common measure of the success of a steganographic research. On average, steganographic algorithms can successfully hide approximately 0.4bpp. Our datasets are formed of 32×32 pixel images, and so we vary n from 100 to 400, this corresponds to hiding between approximately 0.1bpp to 0.4bpp. We intentionally randomise the secret message so that the network does not learn a specific function associated with a specific group of messages. We want to be sure that our model can generalise to any message that is less than n bits[2]. For each experiment, we performed grid search to find the optimum loss weights, $\lambda_A, \lambda_B, \lambda_E$, for Alice. All experiments in this section were performed in TensorFlow [6, 7], on a workstation with a Tesla K40 GPU card.

4.3.1 celebA dataset

We first motivate our work by showing our model can produce steganographic images that are almost indistinguishable from the cover images. We vary the size of the embedded secret message, to explore how many bits may be stored in a cover image while remaining undetected by a steganalyser. As expected, we find that a larger perturbation in the cover image results in more communicated bits but carries a greater risk of detection. The celebA dataset consists of $202,599$ images of celebrity faces from $10,177$ unique identities. We train in batches of 32, and use the Adam optimiser [129] with a learning rate of $2 \cdot 10^{-4}$. At each batch we alternate training either Alice and Bob, or Eve.

Figure 4.2b and fig. 4.2c show the steganographic images output by Alice after 150 training steps, for embedding rates of 0.1bpp and 0.4bpp, respectively. Our scheme easily copes with an embedding rate of 0.1bpp for a 32×32 image, which is equivalent to communicating a short 12 character ASCII message. Figure 4.3a and fig. 4.3c shows the training loss of each model, and the success of Bob decoding the secret message over time, respectively. In the first few rounds of training, the visual quality of Alice's output is low and Bob is essentially randomly guessing the

[2]This ensures our scheme can embed ciphertexts of messages, which appear as random strings.

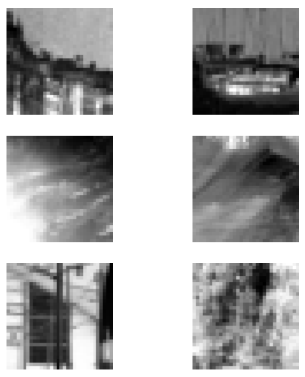
message. After approximately 30 steps, Alice learns to successfully embed information in a cover image such that Eve is fooled, and Bob learns to correctly decode the message. After 150 training steps, Bob decodes the message with an average success of 100%, while Eve's loss remains the same. Figure 4.3b and fig. 4.3d show similar results, with both sample quality and success of decoding both high by the end of training. Concealing 0.4bpp corresponds to successfully hiding a message of 51 ASCII characters. Due to the larger amount of information communicated between Alice and Bob, it takes a longer period of time for Alice to discover an effective embedding scheme, and for Bob to correctly discover the secret message. After 150 training steps, Bob can successfully decode 100% of the message.

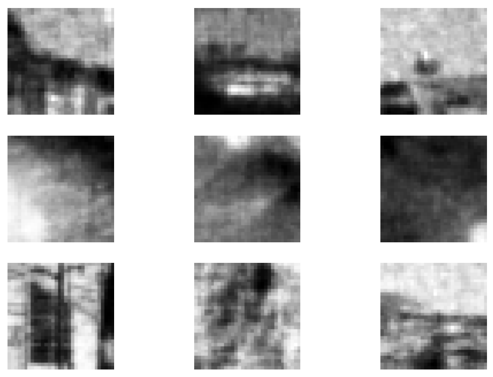
There is no noticeable image quality decrease between 0.1bpp and 0.4bpp, however we found after 0.4bpp, image quality suffered and it was no longer possible to balance realistic image creation capable of fooling Eve, and convergence of message decoding. For both embedding rates of 0.1bpp and 0.4bpp, Eve's loss remained constant at approximately 1.4. Eve's loss is the sum of losses on both steganographic images and cover images, which both remained steady at approximately 0.69. Due to the logarithmic term in the loss function, this is equivalent to randomly guessing if an image carries a secret message.

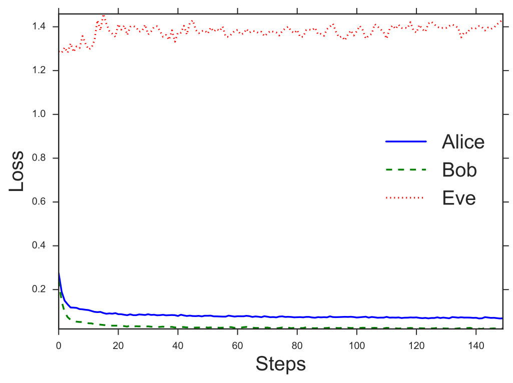
4.3.2 BOSS dataset

In addition to our experiments on the celebA dataset, we trained our steganographic scheme on the BOSS image dataset, which is commonly used as a benchmark in steganography research. BOSS is made up of $10,000$ grayscale images depicting a variety of scenes such as landscapes, buildings and animals. We expected our scheme to perform worse than on the celebA dataset, since the cover images do not come from a single distribution. However, we found our scheme is still capable of embedding secret information successfully.

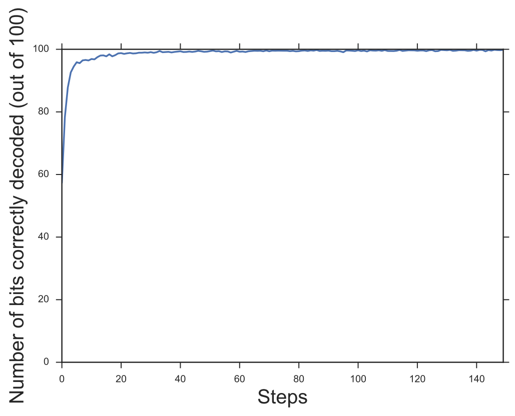
Figure 4.4b shows the sample quality of steganographic images with an embedding rate of 0.1bpp, while fig. 4.4c and fig. 4.4d show the error rates of Alice, Bob and Eve, and the success of Bob decoding the secret message, respectively. While image quality suffers slightly more than on the celebA dataset, our scheme is


(a) Cover images of buildings, birds, skies and the ocean.


(b) Steganographic images (0.1bpp).



(c) Alice, Bob and Eve error for 0.1bpp.



(d) Bob decoding success for 0.1bpp.

Figure 4.4: Results on BOSS dataset at an embedding rate of 0.1bpp.

still able to learn a steganographic algorithm. Our scheme is output samples that are not dissimilar from the original dataset, while Bob is able to learn to successfully decode the message. Alice and Bob both learn their respective tasks in a relatively short period of time, after which there is not much improvement in terms of hiding or recovering the secret message. At the end of training, Bob is able to successfully decode the secret message with 99.8% accuracy.

4.3.3 Comparison with related work

Fooling a steganalyser, Eve, is easy by design, since we train in such a way that Eve never has a significant competitive advantage. Thus, we additionally show that the resultant trained steganographic model, Alice, can fool an independent steganalyser. We compare our scheme against both state-of-the-art steganographic algorithms and steganalysis tools and show that it performs competitively.

For both BOSS and celebA, we compare our scheme against steganographic algorithms HUGO, WOW and S-UNIWARD. Additionally, we implement the Eve model using supervised training and compare against the steganalyser ATS in table 4.1. By design, Eve only performs slightly better than random. One may wonder whether the choice of model for Eve is wise; why not use an established steganalyser in place of the Eve model? By training Eve in a supervised fashion, we show that Eve has the capacity to become a strong steganalyser, competing against established techniques like ATS, and so is a good choice for the steganalyser. Furthermore, Eve does not require a feature extraction pre-processing step as with ATS, and, from our experiments, is an order of magnitude quicker to train. For both the BOSS and celebA datasets, we use $10,000$ samples and split in half, creating a training set and a test set. Alice was then trained on the 5000 samples from the training set. We then created an additional $10,000$ steganographic images for each steganographic algorithm (Alice, HUGO, WOW and S-UNIWARD). Now each steganographic algorithm has an associated training set and test set, each consisting of 5000 cover images and 5000 steganographic images. For each steganographic algorithm we train both ATS and Eve on the associated training set, and then report accuracy of the steganalyser on the test set. From table 4.1, Eve performs competitively against the steganalyser, ATS, and Alice also performs well against other steganographic techniques. While our scheme does not substantially improve on current popular steganographic methods, it is clear that it does not perform significantly worse, and that unsupervised training methods are capable of competing with expert domain knowledge.

4.3.4 Evaluating robust decryption

Due to the non-convexity of the models in the training scheme, we cannot guarantee that two separate parties training on the same images will converge to the same model weights, and so learn the same embedding and decoding algorithms. Thus, prior to steganographic communication, we require one of the communicating parties to train the scheme locally, encrypt model information and pass it to the other

Table 4.1: Accuracy of distinguishing between cover and steganographic images for the steganalysers, Eve and ATS, on the BOSS and celebA datasets at an embedding rate of 0.4bpp.

	BOSS		CELEBA	
STEGANOGRAPHIC ALGORITHM	STEGANALYSER		STEGANALYSER	
	ATS	EVE	ATS	EVE
ALICE	0.83	0.79	0.95	0.90
HUGO	0.66	0.59	0.94	0.89
WOW	0.75	0.74	0.89	0.85
S-UNIWARD	0.77	0.72	0.91	0.84

party along with information about the set of training images. This ensures both parties learn the same model weights. To validate the practicality of our idea, we trained the scheme locally (Machine A) and then sent model information to another workstation (Machine B) that reconstructed the learned models. We then passed steganographic images, embedded by the *Alice* model from Machine A, to Machine B, who used the *Bob* model to recover the secret messages. Using messages of length corresponding to hiding 0.1bpp, and randomly selecting 10% of the celebA dataset, Machine B was able to recover 99.1% of messages sent by Machine A, over 100 trials; our scheme can successfully decode the secret encrypted message from the steganographic image. Note that our scheme does not require perfect decoding accuracy to subsequently decrypt the message. A receiver of a steganographic message can successfully decode and decrypt the secret message if the mode of encryption can tolerate errors. For example, using a stream cipher such as AES-CTR guarantees that incorrectly decoded bits will not affect the ability to decrypt the rest of the message.

4.4 Discussion

We have offered substantial evidence that our hypothesis is correct and machine learning can be used effectively for both steganalysis and steganographic algorithm design. In particular, it is competitive against designs using human-based rules. By leveraging adversarial training games, we confirm that neural networks are able to discover steganographic algorithms, and furthermore, these steganographic algorithms perform well against state-of-the-art techniques. We model the attacker as another neural network and show that this attacker has enough expressivity to perform well against a state-of-the-art steganalyser.

We expect this work to lead to fruitful avenues of further research. Finding the balance between cover image reconstruction loss, Bob's loss and Eve's loss to discover an effective embedding scheme is currently done via grid search, which is a time consuming process. Discovering a more refined method would greatly improve the efficiency of the training process. Indeed, discovering a method to quickly check whether the cover image has the capacity to accept a secret message would be a great improvement over the trial-and-error approach currently implemented. It also became clear that Alice and Bob learn their tasks after a relatively small number of training steps, further research is needed to explore if Alice and Bob fail to improve due to limitations in the model or because of shortcomings in the training scheme.

Part II

Privacy and security in machine learning

In part I, we showed how machine learning can be applied to two well known problems in information security and reach parity or outperform standard techniques. However, machine learning is not a panacea; in this part, we investigate privacy leakages in machine learning that arise from overfitting during the learning process, and identify *contamination attacks* – an attack that arises in multi-party machine learning stemming from a lack of trust between participants.

Chapter 5

Privacy in machine learning

Over the past few years, providers such as Google, Microsoft, and Amazon have started to provide customers with access to APIs allowing them to easily embed machine learning tasks into their applications. Organisations can use Machine Learning as a Service (MLaaS) engines to outsource complex tasks, e.g., training classifiers, performing predictions, clustering, etc. They can also let others query models trained on their data, possibly at a cost. However, if malicious users were able to recover data used to train these models, the resulting information leakage would create serious issues. In particular, organisations do not have much control over the kind of models and training parameters used by the platform, and this might lead to overfitting (i.e., the model does not generalise well outside the data on which it was trained), which provides attackers with a useful tool to recover training data [218].

In recent years, research in deep learning has made tremendous progress in the area of *generative models*. These models are used to generate new samples from the same underlying distribution of a given training dataset. In particular, generative models offer a way to artificially generate plausible images and videos and they are used in many applications, e.g., compression [231], denoising [18], inpainting [260], super-resolution [143], semi-supervised learning [212], etc.

Here, we study the feasibility of *membership inference attacks* against generative models. That is, given access to a generative model and a set of data records, can an attacker tell if these records were used to train the model? Membership inference on generative models is likely to be more challenging than on discriminative ones (see, e.g., [218]). The latter attempt to predict a label given a data input, and an attacker can use the confidence the model places on an input belonging to a label to perform the attack. In generative models, there is no such signal, thus, it is difficult to both detect overfitting and infer membership.

5.1 Membership inference attacks against generative models

We study how generating synthetic samples through generative models may lead to information leakage. In particular, we focus on membership inference attacks against them, which are relevant to, and can be used in, a number of settings:

Direct privacy breach. Membership inference can directly violate privacy if inclusion in a training set is itself sensitive. For example, if synthetic health-related images (i.e., generated by generative models) are used for research purposes, discovering that a specific record was used for training leaks information about the individual's health. (Note that image synthesis is commonly used to create datasets for healthcare applications [48, 179].) Similarly, if images from a database of criminals are used to train a face generation algorithm [254], membership inference may expose an individual's criminal history.

Establishing wrongdoing. Regulators can use membership inference to support the suspicion that a model was trained on personal data without an adequate legal basis, or for a purpose not compatible with the data collection. For instance, DeepMind was recently found to have used personal medical records provided by the UK's National Health Service for purposes beyond direct patient care; the basis on which the data was collected [240]. In general, membership inference against generative models allow regulators to assess whether personal information has been used to train a generative model.

Assessing privacy protection. Our methods can be used by cloud providers that offer MLaaS for generative models (e.g., Neuromation[1]) to evaluate the level of "privacy" of a trained model. In other words, they can use them as a benchmark

[1]`https://neuromation.io`

before allowing third parties access to the model; providers may restrict access in case the inference attack yields good results. Also, susceptibility to membership inference likely correlates with other leakage and with overfitting; in fact, the relationship between robust privacy protections and generalisations have been discussed by Dwork et al. [66].

Overall, membership inference attacks are often a gateway to further attacks. That is, the adversary first infers whether data of a victim is part of the information she has access to (a trained model in our case), and then mount other attacks (e.g., profiling [201], property inference [13, 167], etc.), which might leak additional information about the victim.

5.1.1 Roadmap

Attacks Overview. We consider both black-box and white-box attacks: in the former, the adversary can only make queries to the model under attack, i.e., the *target model*, and has no access to the internal parameters. In the latter, he also has access to the parameters. To mount the attacks, we train a Generative Adversarial Network (GAN) model [81] on samples generated from the target model; specifically, we use generative models as a method to learn information about the target generative model, and thus create a local copy of the target model from which we can launch the attack. Our intuition is that, if a generative model overfits, then a GAN, which combines a discriminative model and a generative model, should be able to detect this overfitting, even if it is not observable to a human, since the discriminator is trained to learn statistical differences in distributions. We rely on GANs to classify real and synthetic records to recognise differences in samples generated from the target model, on inputs on which it was trained versus those on which it was not. Moreover, for white-box attacks, the attacker-trained discriminator itself can be used to measure information leakage of the target model.

Experiments. We test our attacks on several state-of-the-art models: Deep Convolutional GAN (DCGAN) [204], Boundary Equilibrium GAN (BEGAN) [19], and the combination of DCGAN with a Variational Autoencoder (DCGAN+VAE) [140], using datasets with complex representations of faces (LFW), objects (CIFAR-10), and medical images (Diabetic Retinopathy), containing rich details both in the foreground and background.

5.2 Threat model

We consider an adversary that aims to infer whether a set of records were included in the training set of a generative model. We distinguish between two settings: *black-box* and *white-box* attacks. In the former, the attacker can only make queries to the target model under attack – which we denote as the *target model* – and has no access to the internal parameters of the model; in the latter, they also have access to the parameters of a trained target model. Overall, the accuracy of the attack is measured as the fraction of the records correctly inferred as members of the training set.

Assumptions. In both settings, the adversary knows the size of the training set, but not its original data-points. Variants of the attack allow the adversary to access some further side information, as discussed below. In order to evaluate the accuracy of our attacks, we will consider an attacker attempting to distinguish data-points used to train the target model, thus, we consider an attacker that has a set with data points they suspect are in the original training records. However, the construction of the attack does *not* depend on access to *any* dataset. We also assume the attacker knows the size of the training set, but does *not* know how data-points are split into training and test sets.

In the white-box attacks, the adversary only needs access to the discriminator of a target GAN model. In particular, we consider a setting where target model parameters – i.e., both generator and discriminator in the target GAN model – are leaked following a data breach or models initially trained on cloud platforms and then compressed/deployed to mobile devices [108].

Black-box Setting. In black-box attacks, we assume the attacker does not have prior or side information about training records or the target model. In particular, the attack proceeds with no knowledge of the following:

1. *Target model parameters and hyper-parameters:* No access to network

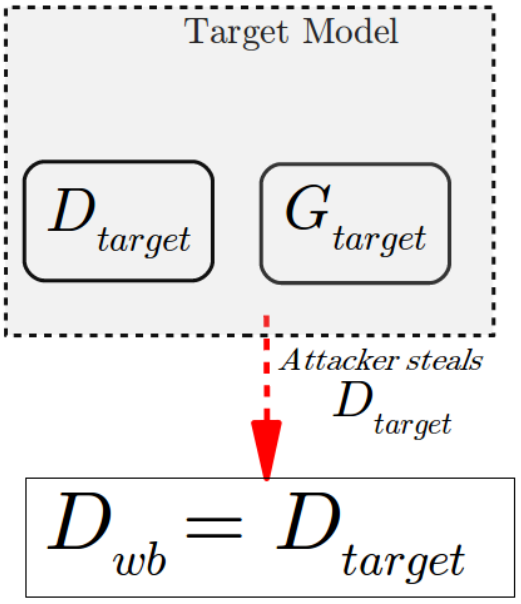


Figure 5.1: High-level Outline of the White-Box Attack.

weights from the trained target model, nor to hyper-parameters such as regularisation parameters or number of epochs used to train the target model.

2. *Target model architecture:* The attacker has no knowledge of the architecture of the target model.

3. *Dataset used to train the target model:* No knowledge of data-points used to train the target model, or the type of data-points used in training, since this is inferred from sampling the target model at inference time. Note that, by contrast, the membership inference attack on discriminative models by Shokri et al. [218] *does* require some information about the dataset, e.g., the syntactic format of data records used in training, in order to generate synthetic samples used in the attack.

4. *Prediction values:* Shokri et al. [218] show that predictions scores leak information used to perform membership inference attacks. However, due to the very nature of generative models, in our attacks, the adversary cannot generate prediction scores directly from the target model.

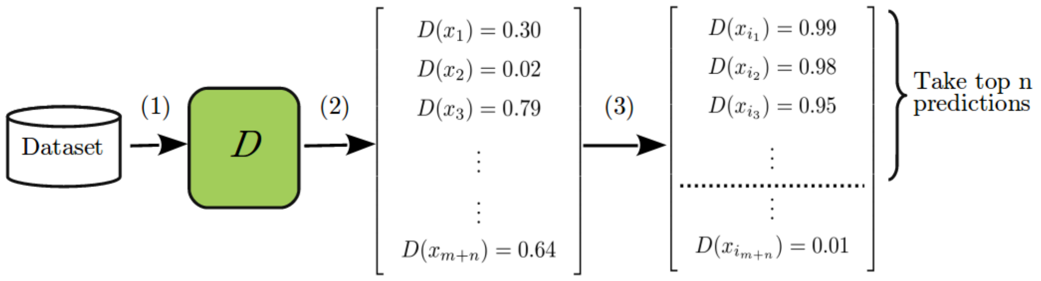


Figure 5.2: White-Box Prediction Method: The attacker inputs data-points to the Discriminator D (1), extracts the output probabilities (2), and sorts them (3).

5.3 White-box attack

We now present our white-box attack; a high-level description is given in fig. 5.1.

To evaluate the attack, here we assume that an attacker $\mathscr{A}_{wb}$ has access to the trained target model, namely, a GAN – i.e., a generator G_{target} and a discriminator D_{target}. The attacker has a dataset, $X = \{x_1, \ldots, x_{m+n}\}$, which they suspect contains data-points used to train the target model, where n is the size of the training set, and m is the number of data-points that do not belong to the training set.

The target model has been trained to generate samples that resemble the training set samples. $\mathscr{A}_{wb}$ creates a local copy of D_{target}, which we refer to as D_{wb}. Then, as shown in fig. 5.2, $\mathscr{A}_{wb}$ inputs all samples $X = \{x_1, \ldots, x_{m+n}\}$ into D_{wb}, which outputs the resulting probability vector $\mathbf{p} = [D_{wb}(x_1), \ldots, D_{wb}(x_{m+n})]$. If the target model overfitted on the training data, D_{wb} will place a higher confidence value on samples that were part of the training set. $\mathscr{A}_{wb}$ sorts their predictions, $\mathbf{p}$, in descending order and takes the samples associated with the largest n probabilities as predictions for members of the training set.

Note that the attacker does not need to train a model; rather, it relies on internal access to the target model, from which the attack can be launched.

5.4 Black-box attack with no auxiliary knowledge

In the black-box setting, we assume that the attacker $\mathscr{A}_{bb}$ does not have access to the target model parameters. Thus, $\mathscr{A}_{bb}$ cannot directly steal the discriminator model from the target as in the white-box attack. Furthermore, while in the white-box attack we restrict the target model to be a GAN, here we do not, and the target

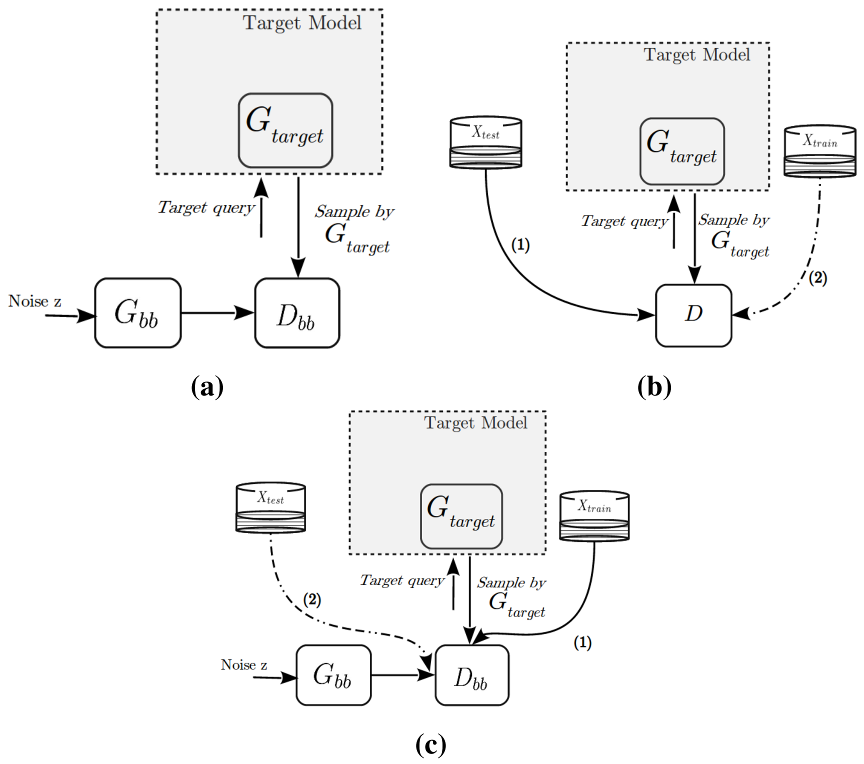


Figure 5.3: High-level overview of the (a) black-box attack with no auxiliary knowledge, and (b) *Discriminative* and (c) *Generative* black-box attack with limited auxiliary attacker knowledge.

model may not have an associated discriminative model (as with VAEs).

Again, to evaluate the attack, we assume the attacker has a dataset, $X = \{x_1, \ldots, x_{m+n}\}$, with data-points suspected to have been used to train the target model, where n is the size of the training set. However, the attacker has no knowledge of how the training set was constructed from X, thus, they do no have access to the true labels of samples from the dataset and so cannot train a model using a discriminative approach. Instead, $\mathscr{A}_{bb}$ trains a GAN in order to re-create the target model locally and, in the process, creates a discriminator D_{bb}, which detects overfitting in the generative target model G_{target}.

We illustrate the attack in fig. 5.3a. Specifically, $\mathscr{A}_{bb}$ locally trains a GAN (G_{bb}, D_{bb}) using queries from the target, i.e., $\mathscr{A}_{bb}$ trains the local GAN on samples generated by G_{target}. As the black-box attack depends only on samples generated by the target model, G_{target} can be any generative model. We assume $\mathscr{A}_{bb}$ has neither knowledge nor control over the source of randomness used to generate the samples generated by G_{bb}. After the GAN has been trained, the attack proceeds to the white-box setting, i.e., $\mathscr{A}_{bb}$ inputs data-points X into D_{bb}, sorts the resulting probabilities, and takes the largest n points as predictions for the training set (as shown in fig. 5.2).

5.5 Black-box attack with limited auxiliary knowledge

In the black-box attack presented above, we assume that $\mathscr{A}_{bb}$ has no additional knowledge about subsets of members of the dataset. However, we also study the case where an attacker could leverage limited additional side information about the training set. This is a realistic setting, which has been considered extensively in the literature; for instance, social graph knowledge has been used to de-anonymise social networks [177]. Overall, auxiliary/incomplete knowledge of sensitive datasets is a common assumption in related literature [119, 203]. Further, the attacker might be able to collect additional information, e.g., from pictures on online social networks or from datasets leaked from data breaches, where the pictures have been used to train the target model under attack. Access to side information about the training set means that the attacker can "augment" the black-box attack.

We consider two settings: a generative and a discriminative one; in either, the attacker has incomplete knowledge of members of the test dataset, the training dataset, or both.

Discriminative setting. We consider an attacker that trains a simple discriminative model to infer membership of the training set, as illustrated in fig. 5.3b. This is feasible since the attacker now has access to membership binary labels, i.e., whether data points belong to the training set or not. Thus, they do not need to train a generative model to detect overfitting. Within this setting, we consider two scenarios where the attacker has limited auxiliary knowledge of:

(1) Samples that *were not* used to train the target model;

(2) *Both* training set and test set samples.

In both cases, the general method of attack is the same: an attacker trains a local model to detect overfitting in the target model. In (1), the discriminator, D, is fed samples from this auxiliary set, labelled as fake samples, and samples generated by the target model, labelled as real samples. If the target model overfits the training set, D will learn to discriminate between training and test samples. In (2), D is fed both target generated samples and the auxiliary training samples, labelled as real samples, and samples from the auxiliary test set, labelled as fake. Once the attacker has trained a discriminator, the attack again proceeds as described in fig. 5.2. Note that we have to consider that the attacker knows some test samples (i.e., fake samples) in order to properly train a binary discriminator.

Generative setting. We also consider a generative attack, as outlined in fig. 5.3c, again, as per two scenarios, where the attacker has limited auxiliary knowledge of:

(1) Samples that *were* used to train the target model;

(2) *Both* training set and test set samples.

With both, the attacker trains a local model—specifically, a GAN—that aims to detect overfitting in the target model. In (1), the discriminator of the attacker GAN, D_{bb}, is trained using samples generated by G_{bb}, labelled as fake samples, and both samples from the auxiliary training set and target generated samples, labelled as real. Intuitively, we expect the attacker model to be stronger at recognising overfitting in the target model, if it has auxiliary knowledge of samples on which it was originally trained. In (2), D_{bb} is trained on samples generated by G_{bb} and samples from auxiliary set of test ones, labelled as fake samples, and samples generated by the target model and samples from the auxiliary training set, labelled as real. The attacker GAN is trained to learn to discriminate between test and training samples directly. Again, once the attacker has trained their model, data-points from X are fed into D_{bb}, and their predictions are sorted as per fig. 5.2.

5.6 Experimental setup

Testbed. Experiments are performed using PyTorch on a workstation running Ubuntu Server 16.04 LTS, equipped with a 3.4GHz CPU i7-6800K, 32GB RAM, and an NVIDIA Titan X GPU card.

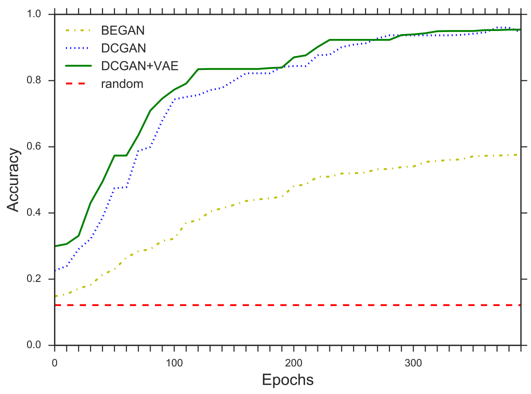
Settings. For white-box attacks, we measure membership inference accuracy at successive epochs of training the target model, where one epoch corresponds to one round of training on all training set inputs[2]. For black-box attacks, we fix the target model and measure membership inference accuracy at successive training steps of the attacker model, where one training step is defined as one iteration of training on a mini-batch of inputs. The attacker model is trained using soft and noisy labels as suggested in [212], i.e., we replace labels with random numbers in $[0.7, 1.2]$ for real samples, and random values in $[0.0, 0.3]$ for fake samples. Also, we occasionally flip the labels when training the discriminator. These GAN modifications are known to stabilise training in practice [47].

Datasets. We perform experiments using two popular image datasets as well as a health-related dataset:

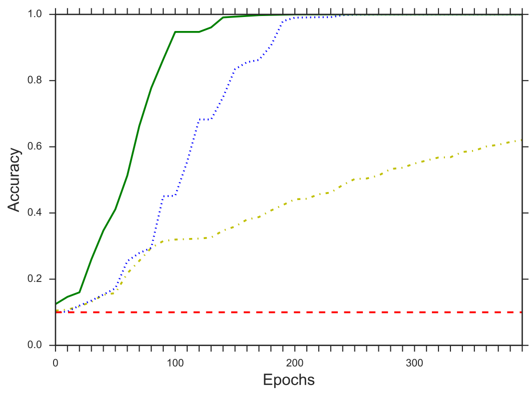
1. Labelled Faces in the Wild (LFW) [115], which includes 13,233 images of faces collected from the Web;
2. CIFAR-10 [133], with 60,000 32x32 colour images in 10 classes, with 6,000 images per class;
3. Diabetic Retinopathy (DR) [123], consisting of 88,702 high-resolution retina images taken under a variety of image conditions.

For LFW and CIFAR-10, we randomly choose 10% of the records as the training set. The LFW dataset is “unbalanced,” i.e., some people appear in multiple images, while others only appear once. We also perform experiments so that the training set is chosen to include the ten most popular classes of people in terms of number of images they appear in, which amounts to 12.2% of the LFW dataset. Intuitively, we expect that models trained on the top ten classes will overfit more than the same models trained on random 10% subsets, as we are training on a more homogeneous set of images.

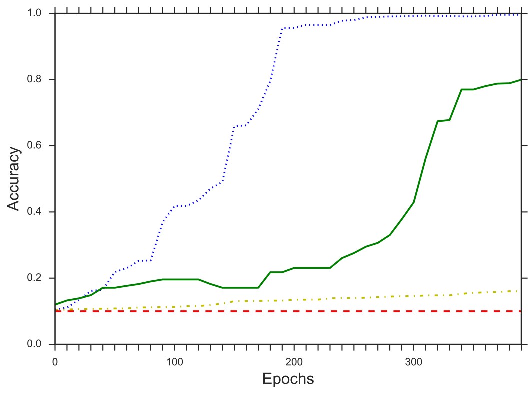
[2]We update model weights after training on mini-batches of 32 samples.



(a) LFW, top ten classes



(b) LFW, random 10% subset



(c) CIFAR-10, random 10% subset

Figure 5.4: Accuracy of white-box attack with different datasets and training sets.

Note that experiments using the DR dataset are presented in section 5.12, which discusses a case-study evaluation on a dataset of medical relevance. From DR, we select images with moderate to proliferate diabetic retinopathy presence, and use them to train the generative target model.

Models. Since their introduction, a few GAN [81] variants have been proposed to improve training stability and sample quality. In particular, deep convolutional generative adversarial networks (DCGANs) [204] combine the GAN training process with convolutional neural networks (CNNs). CNNs are considered the state of the art for a range image recognition tasks; by combining CNNs with the GAN training processes, DCGANs perform well at unsupervised learning tasks such as generating complex representations of objects and faces [204]. GANs have also been combined with VAEs [140]: by collapsing the generator (of the GAN) and decoder (of the VAE) into one, the model uses learned feature representations in the GAN discriminator as the reconstructive error term in the VAE. It has also been shown that combining the DCGAN architecture with a VAE yields more realistic generated samples [88]. More recently, Boundary Equilibrium GAN (BEGAN) [19] have been proposed as an approximate measure of convergence. Loss terms in GAN training do not correlate with sample quality, making it difficult for a practitioner to decide when to stop training. This decision is usually performed by visually inspecting generated samples. BEGAN proposes a new method for training GANs by changing the loss function. The discriminator is an autoencoder and the loss is a function of the quality of reconstruction achieved by the discriminator on both generated and real samples. BEGAN produces realistic samples [19], and is simpler to train since loss convergence and sample quality is linked with one another.

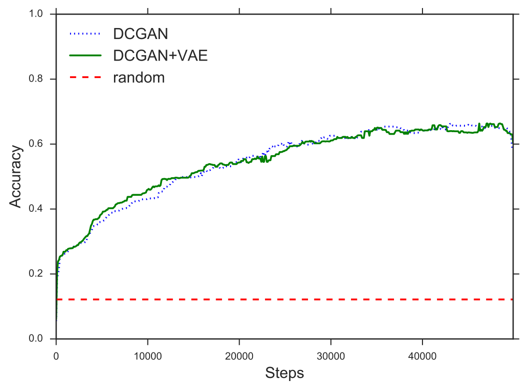
We evaluate our attacks using, as the target model:

1. DCGAN [204],
2. DCGAN+VAE [140], and
3. BEGAN [19],

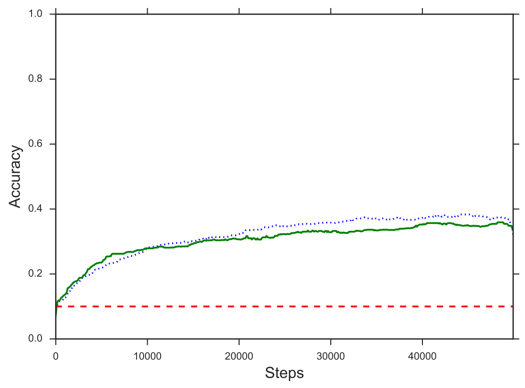
while fixing DCGAN as the attacker model. This choice of models is supported by recent work [158], which shows that no other GAN model performs significantly better than our choices. Lucic et al. [158] also demonstrates that VAE models perform significantly worse than any GAN variant.

5.7 Euclidean approaches

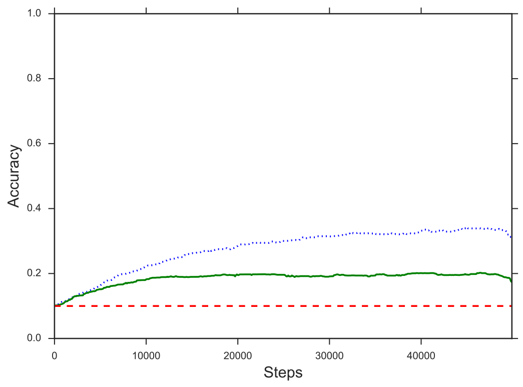
We begin our evaluation with a naïve Euclidean distance based attack. Given a sample generated by a target model, the attacker computes the Euclidean distance between the generated sample and every real sample in the dataset. Repeating this multiple times for newly generated samples, the attacker computes an average distance from each real sample, sorts the average distances, and takes the smallest n distances (and the associated real samples) as the guess for the training set, where n is the size of the training set.



(a) LFW, top ten classes



(b) LFW, random 10% subset



(c) CIFAR-10, random 10% subset

Figure 5.5: Accuracy of black-box attack on different datasets and training sets.

We perform this attack on a target model (DCGAN) trained on a random 10% subset of CIFAR-10 and a random 10% subset of LFW, finding that the attack does not perform better than if the attacker were to randomly guess which real samples were part of the original training set. For completeness, results are reported in fig. B.1 in appendix B.1. In appendix B.1, we also discuss another unsuccessful approach, based on training a shadow model, inspired by the techniques proposed by Shokri et al. [218].

5.8 White-box attack

We now present the results of our evaluation of the white-box attack described in section 5.3 on LFW and CIFAR-10. For the LFW dataset, we build the training set either as a random 10% subset of the dataset or the top ten classes. For CIFAR-10, the training set is a random 10% subset of the dataset. The target models we implement are DCGAN, DCGAN+VAE, and BEGAN. In the rest of this section, we will include a baseline in the plots (red dotted line) that corresponds to the success of an attacker randomly guessing which samples belong to the training set.

Figure 5.4a shows the accuracy of a white-box attack against a target model trained on the top ten classes of the LFW dataset. We observe that both DCGAN and DCGAN+VAE are vulnerable to the white-box attack. For DCGAN and DCGAN+VAE target models trained for 100 epochs, the attacker infers training set membership with 80% accuracy, and for models trained for 400 epochs – with 98% and 97% accuracy, respectively. The BEGAN target model does overfit, although to a lesser extent: after 400 epochs, an attacker with white-box access to the BEGAN target model can infer membership of the training set with 60% accuracy. In fig. 5.4b, we report the results of white-box attacks against a target model trained on a random 10% subset of the LFW dataset. Similar to fig. 5.4a, both DCGAN and DCGAN+VAE are vulnerable: when these are trained for 250 epochs, an attacker can achieve perfect training set membership inference. BEGAN performs similar to the top ten classes white-box experiment, achieving 62% accuracy after 400 epochs. Finally, fig. 5.4c plots the accuracy of the white-box attack against a target model trained on a random 10% subset of CIFAR-10.

For DCGAN, results are similar to DCGAN on LFW, with perfect training set membership inference after 400 epochs. However, DCGAN+VAE does not leak information (does not overfit) until around 250 epochs, where accuracy remains relatively steady, at 10-20%. Instead, after 250 epochs, the model overfits, with accuracy reaching 80% by 400 epochs. BEGAN, while producing quality samples, does not overfit, with final training set membership inference accuracy of 19%, i.e., only 9% better than random guess. Due to the limited accuracy of BEGAN in comparison to other models, we discard it as a target model for black-box attacks as it does not seem to be vulnerable to membership inference attacks. Note that GAN models need to be trained for hundreds of epochs before reaching good samples quality. Indeed, the original DCGAN/BEGAN papers report 2x and 1.5x the number of network updates (when adjusted for training set size) as our white-box attack, to train DCGAN and BEGAN, respectively.

In summary, we conclude that white-box attacks infer the training set with up to perfect accuracy when DCGAN and DCGAN+VAE are the target models. On the other hand, BEGAN is less vulnerable to white-box attacks, with up to 62% accuracy.

5.9 Black-box attack with no auxiliary knowledge

Next, we present the results of the black-box attacks on LFW and CIFAR-10. We assume the attacker has no knowledge of the training or test sets other than the size of the original training set. Once again, for LFW, the training set is either a random 10% subset of the dataset or the top ten classes, while, for CIFAR-10, the training set is always a random 10% subset of the dataset. The target models we implement are DCGAN and DCGAN+VAE (fixed at epoch 400), and the attacker model uses DCGAN.

Figure 5.5a plots the results of a black-box attack against a target model trained on the top ten classes of the LFW dataset. After training the attacker model on target queries, the attack achieves 63% training set membership inference accuracy for both DCGAN and DCGAN+VAE target models. Surprisingly, the attack performs equally well when the target model differs from the attack model as when the target and attack model are identical. This highlights the fact that the attacker does not need to have knowledge of the target model architecture in order to perform the attack.

In fig. 5.5b, the results are with respect to a target model trained on a random 10% subset of the LFW dataset. Once again, we find that DCGAN and DCGAN+VAE target models are equally vulnerable to a black-box attack. An attacker with no auxiliary information of the training set can still expect to perform membership inference with 40% (38%) accuracy for the DCGAN (DCGAN+VAE) target model.

Finally, fig. 5.5c plots the accuracy of a black-box attack against a target model trained on a random 10% subset of the CIFAR-10 dataset. For the DCGAN+VAE target model, accuracy reaches 20% after 1,000 training steps and stays flat. For the DCGAN target model, the attacker can infer training set membership with 37% accuracy, with accuracy improving steadily throughout the attacker model training process.

We observe that the difference in attack success between the DCGAN and DCGAN+VAE target models with CIFAR-10 and the similar success of the two models with LFW occur in both white-box and black-box attacks. As expected, the best results are obtained when the attacker and target model have the same architecture. However, the attack does not overwhelmingly suffer under differing architectures. In fact, in LFW experiments there is a negligible difference in attack success, and, in the CIFAR-10 black-box experiments, the difference in accuracy is approximately 17%.

In summary, we conclude that our black-box attacks are less successful, compared to white-box attack, in inferring membership, but perform similarly against different target model architectures.

5.10 Black-box attack with limited auxiliary knowledge

As discussed in section 5.5, we also consider black-box attacks where the attacker has some limited auxiliary knowledge of the dataset, and uses this knowledge to recover the full training set. We now present the results of these attacks on random 10% subsets of LFW and CIFAR-10 with DCGAN attacker and target models (fixed at epoch 400).

We consider different scenarios where the attacker has knowledge of 20–30% of the training set, 20-30% of the test set, or both. Nonetheless, the total number of samples of which the attacker has knowledge is quite modest. For LFW, 20% of the random 10% training set corresponds to 264 out of 1,323 images, 20% of the test set to 2,382 out of 11,910 images, whereas, for CIFAR-10, 20% of the random 10% training set amounts to 1,200 out of 6,000 images, and 20% of the test set to 10,000 out of 50,000 images. An attacker with auxiliary information of the training and test set has access to labels, and therefore may not need to train a generative model to perform a membership inference attack on a generative model. We also show that, while the attacker can train a discriminative model to perform membership inference, such an approach produces worse results than the generative method.

Discriminative approach. If an attacker has access to true labels within the dataset, they can train a discriminative model on these samples in order to learn to classify training samples correctly. For both LFW and CIFAR-10 DCGAN target models, trained on a random 10% subset of the dataset, we consider two settings:

(i) the attacker has 20% knowledge of the *test* set; or

(ii) the attacker has 30% knowledge of both the training and test set.

We use the discriminator from DCGAN as the discriminative model trained by the attacker. In (i), we pass test set samples to the discriminator labelled as fake samples, and target generated ones labelled as real. In (ii), we pass test set samples to the discriminator labelled as fake ones, and target generated and training set samples labelled as real ones.

In fig. 5.6, we plot the accuracy results for both settings, showing that the attack fails with both datasets when the attacker has only test set knowledge, performing no better than random guessing. Whereas, if the attacker has both training and test knowledge, with LFW, the attacker recovers the training set with 50% accuracy, while, for CIFAR-10, accuracy reaches 33%. Note that this approach does not improve on CIFAR-10 black-box results with no auxiliary knowledge, and only marginally improves on LFW results. As a result, we also experiment with generative approaches to black-box attacks with auxiliary attacker knowledge, as discussed next.

Generative approach. We consider the same set of experiments with similar settings for attacker knowledge as in the discriminative approach; the only difference is that in one of the settings we now assume the attacker has 20% knowledge of the training set rather than the test set. We use DCGAN as the generative attacker model. Specifically, we consider that the attacker has:

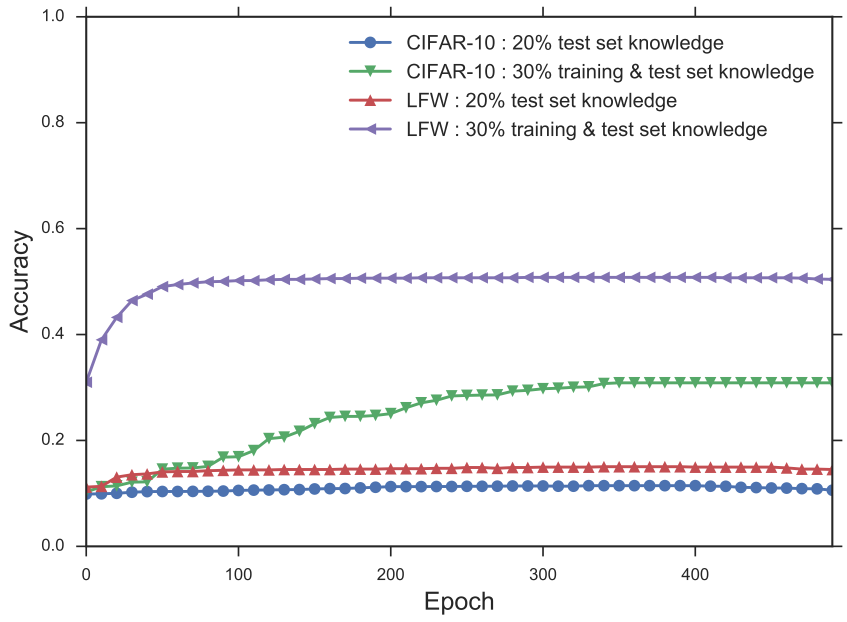
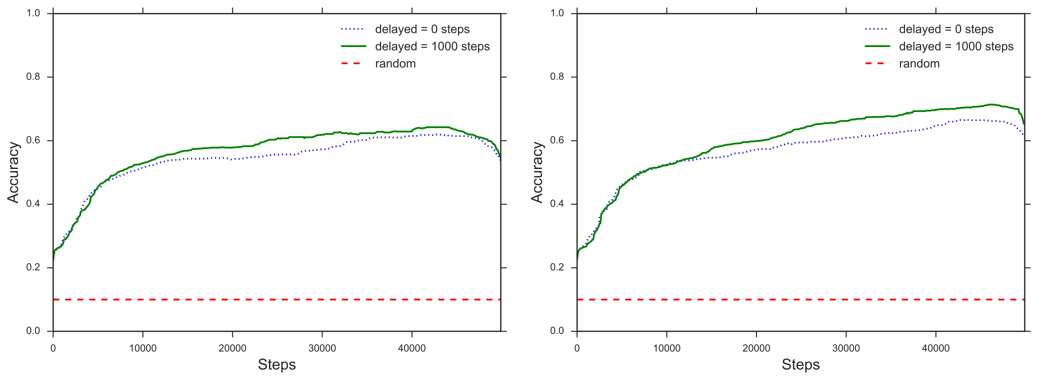


Figure 5.6: Membership inference accuracy using a discriminative model, when the attacker has knowledge of (i) 20% of the test set, or (ii) 30% of both training and test sets. Random guess in (i) and (ii) corresponds, respectively, to 14% and 12% accuracy.



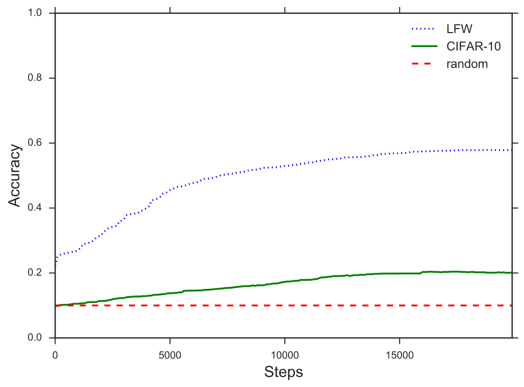
(a) DCGAN **(b)** DCGAN+VAE

Figure 5.7: Black-box attack results with 20% attacker training set knowledge for DCGAN/DCGAN+VAE target models, trained on a random 10% subset of LFW, for different delays at which auxiliary knowledge is introduced into the attacker model training.

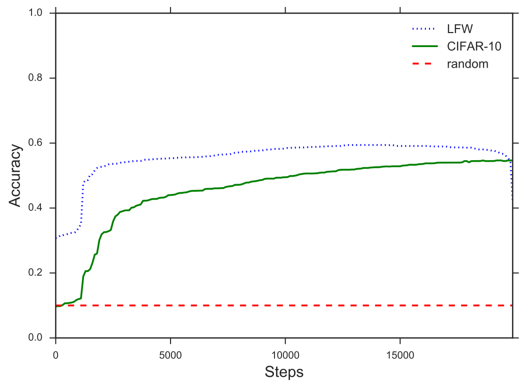
(1) 20% knowledge of the *training* set; or

(2) 30% knowledge of both the training and test set.

In all the experiments, we introduce a delay of 1000 training steps before the attacker model uses the auxiliary attacker knowledge. Introducing the auxiliary

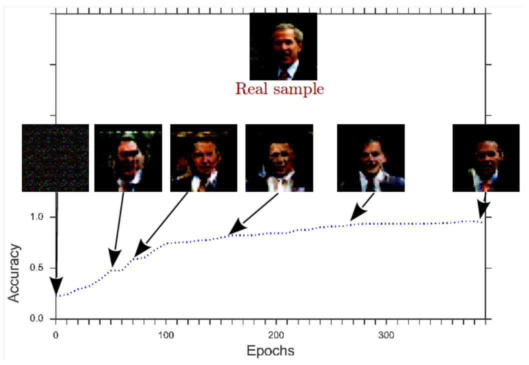


(a) 20% of the training set knowledge

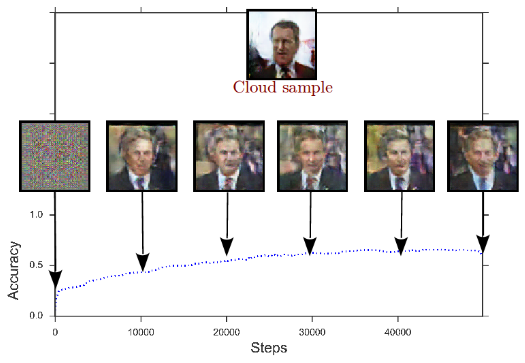


(b) 30% of the training set and test set knowledge

Figure 5.8: Black-box results when the attacker has (a) knowledge of 20% of the training set or (b) 30% of the training set and test set. The training set is a random 10% subset of the LFW or CIFAR-10 dataset, and the target model is fixed as DCGAN.



(a) White-box attack



(b) Black-box attack

Figure 5.9: Accuracy curves and samples at different stages of training on top ten classes from the LFW dataset, showing a clear correlation between higher accuracy and better sample quality.

knowledge early in training process of the attacker model resulted in a weaker discriminator – see fig. 5.7.

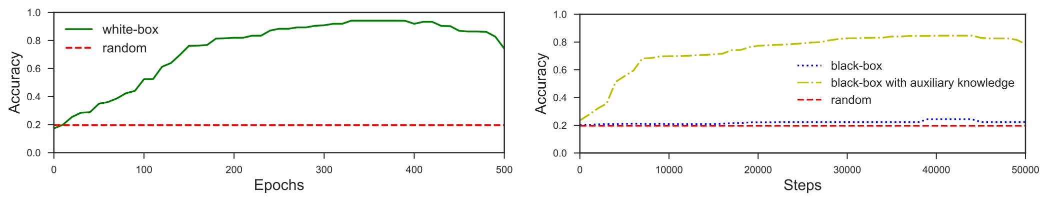
In fig. 5.8a, we plot results for setting (1): clearly, there is a substantial increase in accuracy for the LFW dataset, from 40% attack accuracy to nearly 60%. However, there is no increase in accuracy for the CIFAR-10 dataset. Thus, we conclude that setting (1) does not generalise. Figure 5.8b shows results for setting (2); for both LFW and CIFAR-10 there is a substantial improvement in accuracy. Accuracy for the LFW experiment increases from 40% (with no auxiliary attacker knowledge) to 60%, while, for CIFAR-10, from 37% to 58%.

Thus, we conclude that, even a small amount of auxiliary attacker knowledge can lead to greatly improving membership inference attacks.

5.11 Training performance

We also set out to better understand the relationship between membership inference and training performance. To this end, we report, in fig. 5.9, the attack accuracy and samples generated at different training stages by the target DCGAN generator in the white-box attack (Figure 5.9a) and the attacker DCGAN generator in the black-box attack (Figure 5.9b) on the top ten classes from the LFW dataset. The plots demonstrate that accuracy correlates well with the visual quality of the generated samples. In particular, samples generated by the target yield a better visual quality than the ones generated by the attacker generator during the black-box attack, and this results in higher membership inference accuracies. Overall, the samples generated by both attacks at later stages look visually pleasant, and fairly similar to the original ones.

Our attacks have been evaluated on datasets that consist of complex representations of faces (LFW) and objects (CIFAR-10). In appendix B.2, we include real and generated samples in multiple settings; see fig. B.4–fig. B.10. In particular, as shown in fig. B.3a, real samples from LFW contain rich details both in the foreground and background. We do not observe any large deviations in images within datasets, excluding that the attack performs well due to some training samples being more easily learned by the model, and so predicting with higher confidence. Learning the distribution of such images is a challenging task compared to simple datasets such as MNIST, where samples from each class have extremely similar features. In fact, our black-box attack is able to generate realistic samples (see differences between the target model samples in fig. B.3b and the attacker samples in fig. B.3c).



(a) White-box attack **(b)** Black-box attack

Figure 5.10: Accuracy curves of attacks against a DCGAN target model on the Diabetic Retinopathy dataset.

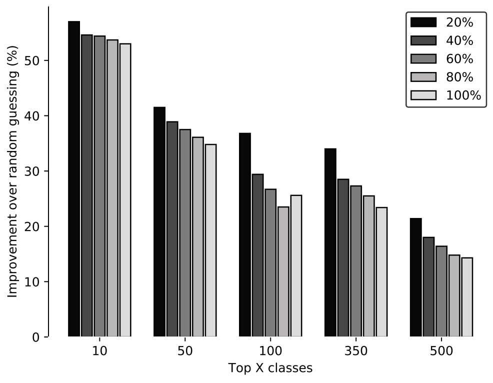
5.12 Evaluation on Diabetic Retinopathy dataset

Finally, we present a case study of our attacks on the Diabetic Retinopathy (DR) dataset, which consists of high-resolution retina images, with an integer label assigning a score of the degree to which the participant suffers from diabetic retinopathy. Diabetic retinopathy is a leading cause of blindness in the developed world, with detection currently performed manually by highly skilled clinicians. The machine learning competition site `https://kaggle.com` has evaluated proposals for automated detection of diabetic retinopathy, and submissions have demonstrated high accuracies of detection.

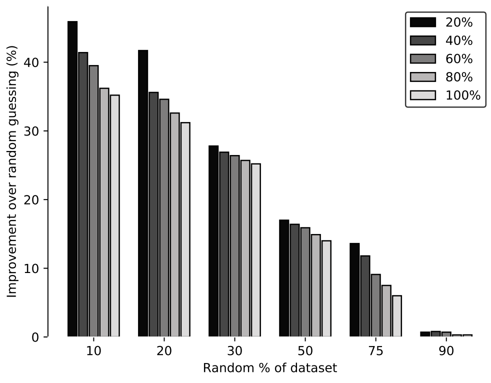
We choose this additional dataset since the generation of synthetic medical images through generative models is a powerful method to produce large numbers of high-quality sample data on which useful machine learning models can be trained. Thus, our attacks raise serious privacy concerns, in practice, in such sensitive settings as they involve (sensitive) medical data.

As discussed in section 5.6, the dataset includes 88,702 high-resolution retina images under various imaging conditions. Each image is labelled with an integer representing how present is diabetic retinopathy within the retina, from 0 to 4. We train the generative target model on images with labels 2, 3 and 4, i.e., with mild to severe cases of diabetic retinopathy. These make up 19.7% of the dataset. Figure B.4 in appendix B.2 show real and target generated samples of retina images.

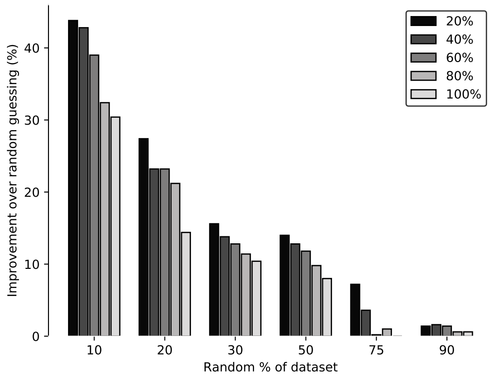
The results of the white-box attack are reported in fig. 5.10a: the attack is overwhelmingly successful, nearing 100% accuracy at 350 training epochs. Figure 5.10b shows the black-box attacks results, when an attacker has no auxiliary



(a) LFW Top X classes



(b) LFW, random X% subset



(c) CIFAR-10 random X% subset

Figure 5.11: Improvements over random guessing, in a black-box attack, as we vary the size of the training set, and consider smaller subsets for training set predictions.

knowledge, and when the attacker has 30% training and test set auxiliary knowledge. A no-knowledge black-box attack does not perform very well, while, with some auxiliary knowledge, it approaches the accuracy of the white-box attack, peaking at over 80% after 35K training steps.

5.12.1 Summary of results

Overall, our analysis shows that state-of-the-art generative models are vulnerable against membership inference attacks. In table 5.1, we summarise the best accuracy results for experiments on random 10% training sets (LFW, CIFAR-10) and the diabetic retinopathy (DR) dataset experiments.

We note that, for white-box attacks, the attacker successfully infers the training set with 100% accuracy on both the LFW and CIFAR-10 datasets, and 95% accu-

Table 5.1: Accuracy of the best attacks on random 10% training set for LFW and CIFAR-10, and for diabetic retinopathy (DR).

Attack	**LFW**	**CIFAR-10**	**DR**
White-box	100%	100%	95%
Black-box with no knowledge	40%	37%	22%
Black-box with limited knowledge	60%	58%	81%
Random Guess	10%	10%	20%

racy for DR dataset. Accuracy drops to 40% on LFW, 37% on CIFAR-10 and 22% on DR for black-box attacks with no auxiliary knowledge, however, even with a small amount of auxiliary knowledge, the attacker boost performance up to 60% on LFW, 58% on CIFAR-10 and 81% on DR. Note that a random guess corresponds to 10% accuracy on LFW and CIFAR-10, and 20% on DR. Further, we show that our attacks are robust against different target model architectures.

5.13 Sensitivity to training set size and prediction ordering

Aiming to measure the dependency between attack performance and training set size, we experiment with varying training set sizes in the DCGAN target and attacker model setting.

Figure 5.11 shows how the improvement of the attack degrades as the relative size of the training set increases. Note that we only include black-box attack results, as *all white-box attacks achieve almost 100% accuracy regardless of training set size.* Overall, we find that there is a commonality in the experiments: black-box attacks on 10% of the dataset achieve an improvement of 40–55%, and, as we increase the number of data-points used to train the target model, the attack has smaller and smaller improvements over random guessing.

The largest increases are in the setting of fig. 5.11a, where data-points are more homogeneous and so overfitting effects are compounded. When the training set is 90% of the total dataset used in the evaluation of the attack, the attack has negligible improvements over random guessing. We believe that this might be due either to:

(1) the larger number of training data-points yields a well-fitted model that does not leak information about training records, or (2) a small number of data-points within the training set do not leak information, therefore, as we increase the size of the training set, the inability to capture these records becomes more costly, resulting in smaller improvements in attack performance.

If the former were true, we would see smaller improvements for larger training sets, regardless of the total size of the dataset; however, experiments on both LFW and CIFAR-10, which consist of different training sizes, report similar improvements over random guessing. Additionally, white-box attacks are not affected by increasing the training set size, which would be the case if the model did not overfit and thus leak information about training records. Hence, we believe a small number of training records are inherently difficult to capture, and so improvements over random guessing for larger training set sizes are more difficult to achieve since the majority of samples are used to train the target model.

We also examine the attack sensitivity to the ordering of the data-point predictions. So far, the *only* prior knowledge the attacker has is the approximate size of the training set. If there is a clear ordering of data-points predictions, with training records sitting at the top of the ordering, and non-training records lower down, an attacker can use this information to identify training records without side knowledge of training set size. They can simply place a confidence score relative to where in the ordering a data-point predictions sits.

Figure 5.11 shows, for varying training set sizes, how many training records lie in the top 20%, 40%, 60%, 80%, and 100% of the guessed training set. We observe that, in all experimental settings, accuracy for the top 20% is highest, with scores decreasing as the attacker considers a larger number of data-points as candidates for the training set.

Thus, training to non-training samples follows a structured ordering in the attacker's predictions, which can be exploited to infer membership when the attacker has *no knowledge* of the original training set size by setting a threshold on the minimum confidence of a training point.

5.14 Defences

Possible defence strategies against membership inference (see Shokri et al. [218]), e.g., restricting the prediction vector to the top k classes, coarsening and increasing the entropy of the prediction vector, are not well suited to our attacks, since generative models do not output prediction vectors. However, regularisation techniques and differential privacy could possibly be applied to generative models to produce more robust and stable training as well as more diverse and visually pleasant samples.

Weight Normalisation and Dropout. To this end, we consider two techniques, namely, Weight Normalisation [211] and Dropout [224], as possible defence mechanisms and evaluate their impact on our attacks. Batch normalisation and dropout were developed primarily as general techniques to improve generalisation and reduce overfitting, and so should decrease membership inference accuracy, since these attacks exploit overfitting[3]. The former is a re-parameterisation of the weights vectors that decouples the length of those weights from their direction, and it is applied to all layers in both generator and discriminator in the target model. Whereas, the latter can be used to prevent overfitting by randomly dropping out (i.e., zeroing) connections between neurons during training—in particular, we apply Dropout, with probability 0.5, to all the layers in the discriminator.

In fig. 5.12, we measure the improvement over random guessing for the white-box attack against the target model trained on LFW using either Weight Normalisation or Dropout. With Weight Normalisation, we get improvements over random guessing of, respectively, 88% and 46%, which are very close to the target model trained with no defences (resp., 89% and 52%). Dropout is more effective, as the improvements over random guessing go down to 70% on top 10 classes and 23% on top 500 classes. The size of the dataset will also clearly play a role in ability to infer membership, attacking a 10% random subset of LFW (3K images) has equal membership accuracy as attacking 10% of CIFAR-10 (6K images). We leave a

[3]Note that we do not compare models with and without Batch Normalisation [117], as its inclusion has shown to improve sample quality and is nearly always used in model construction of GANs [204].

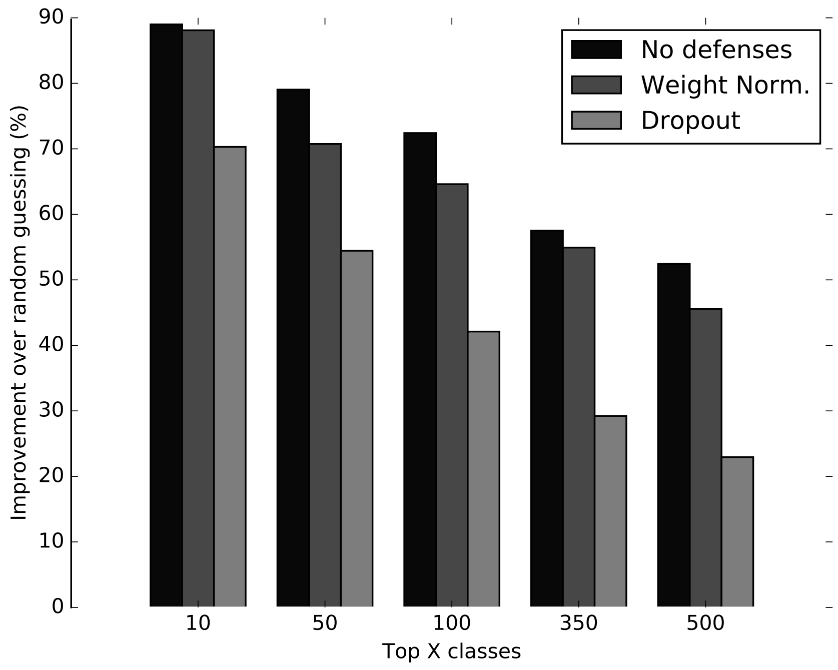


Figure 5.12: Improvement over random guessing for Weight Normalisation and Dropout defences against white-box attacks on models trained over different number of classes with LFW.

systematic investigation of the relationship between dataset size and membership vulnerability for future work.

However, Dropout significantly slows down the training process, requiring more epochs to get qualitatively plausible samples. Also, Weight Normalisation often results in training instability (i.e., the discriminator outperforms the generator, or vice-versa).

Differentially Private GANs. We also evaluate our attack against a recently proposed technique for (ε, δ)-Differentially Private GANs [235], where Gaussian noise [63] is injected in the discriminator forward pass during training. Figure 5.13 shows the results of a white-box attack against Differentially Private DCGAN trained on top ten classes for different values of the privacy budget ε (with δ set to 10^{-4}). For all experiments, the target model is trained for 500 epochs and the final privacy budget is computed using moments accountant [8]. The attack does no better than random guessing for $\varepsilon = 1.5$ (first tick in the plot), while accuracy increases up to 85% for $\varepsilon = 28.3$. However, note that acceptable levels of privacy

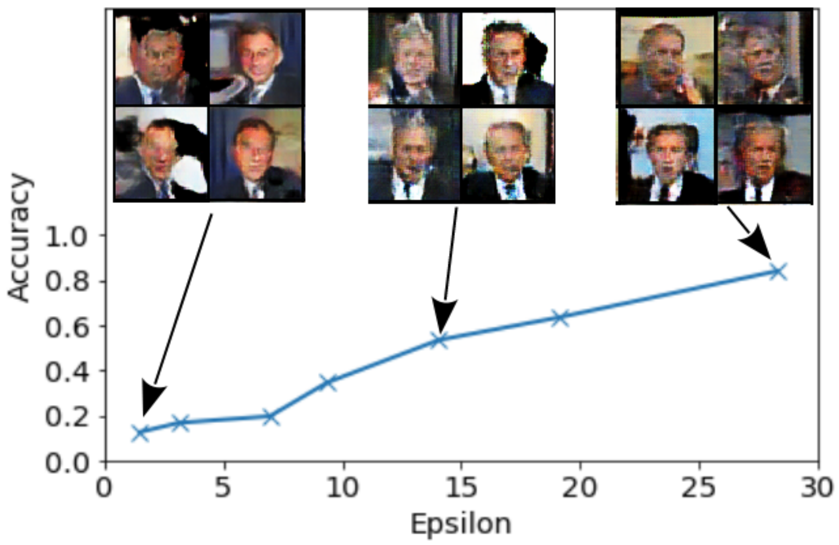


Figure 5.13: Accuracy curve and samples for different privacy budgets on top ten classes from the LFW dataset, showing a trade-off between samples quality and privacy guarantees.

(i.e., values of $\varepsilon < 10$) yield poor sample quality.

Using our attacks as defence. The difference in white-box and black-box accuracy provides information about how well the local model approximates the target model, thus, one could use this information to train a target model which cannot be well approximated. Furthermore, similarly to *early-stopping* criteria in model training, one can stop training when visual sample quality is high but white-box attack accuracy is still low.

In our experiments, we also observe the benefits of a more regularised model in increasing the robustness against information leakage in the case of BEGAN. For instance, in white-box attacks on CIFAR-10, BEGAN produces quality samples without overfitting, with membership inference performing only 9% better than random guessing (see fig. 5.4c).

5.15 Cost of the attacks

Finally, we quantify the cost of the attacks in terms of computational and time overhead, and estimate monetary costs.

To perform the attacks, the attacker needs a GPU, which can be obtained for a cost in the order of \$100. The attacks have minimal running time overheads: for the white-box attack, complexity is negligible as we only query a pre-trained target model to steal discriminator model parameters, whereas, for black-box, one step of training the attacker model takes 0.05 seconds in our testbed. Black-box attacks with no auxiliary attacker knowledge yield the best results after 50,000 training steps, therefore, an attacker can expect best results after approximately 42 minutes with 32 $\times$ 50,000 queries to the target model (since we define one training step as one mini-batch iteration, with 32 inputs per mini-batch). For attacks with auxiliary knowledge, the best results are reached after 15,000 training steps, thus, approximately 13 minutes.

We also estimate monetary cost based on current discriminative MLaaS pricing structures from Google[4]. At a cost of $1.50 per 1,000 target queries, after an initial 1,000 free monthly queries, the black-box attack with no auxiliary knowledge would cost $2,352, while the black-box attack with auxiliary knowledge $672. Therefore, we consider our attacks to have minimal costs, especially considering the potential severity of the information leakage they enable.

5.16 Summary

This paper presented the first evaluation of membership inference attacks against generative models, showing that a variety of models lead to important privacy leakage. Our attacks are cheap to run, do not need information about the model under attack, and generalize well. We conducted an experimental evaluation on state-of-theart probabilistic models such as Deep Convolutional GAN (DCGAN), Boundary Equilibrium GAN (BEGAN), and the combination of DCGAN with a Variational Autoencoder (DCGAN+VAE), using datasets with complex representations of faces (LFW), objects (CIFAR-10), and medical images with real-world privacy concerns (Diabetic Retinopathy). We showed that the white-box attack can be used to detect overfitting in generative models and help select an appropriate model that will not leak information about samples on which it was trained. We also demonstrated that our low-cost black-box attack can perform membership inference using a novel

[4]https://cloud.google.com/vision/pricing

method for training GANs, and that an attacker with limited auxiliary knowledge of dataset samples can remarkably improve their accuracy. Moreover, we experimented with regularization techniques, such as Weight Normalization [55] and Dropout [59], and differentially private mechanisms, which could be used to mitigate our attacks. We found that they are effective up to a certain extent, but need longer training, yield training instability, and/or worse generated samples (in terms of quality). This motivates the need for future work on defenses against information leakage in generative models. Our work also provides evidence that models that generalize well (e.g., BEGAN) yield higher protection against membership inference attacks, confirming that generalization and privacy are associated. Thus, our evaluation may be used to empirically assess the generalization quality of a generative model, which is an open research problem of independent interest.

Chapter 6

Contamination attacks & mitigation in multi-party machine learning

In this chapter, we study the setting where a *small number* of parties (e.g., up to twenty) wish to use a secure centralised multi-party machine learning service to train a model on their joint data. Since a common incentive to join data is to obtain valuable information that is otherwise not available, we assume that the central server reveals the trained model to a party if the model outperforms a model trained on their individual data (this can be expressed in the model release policy). This setting already encourages each party to supply data that benefits the others as opposed to supplying a dummy dataset with the goal of either learning more information about other parties or decreasing the overall accuracy of the model [21, 109, 214]. However, it is not clear if this is sufficient to prevent other malicious behaviour, and so we seek to understand and answer the following question:

How much can a malicious party influence what is learned during training, and how can this be defended against?

To this end, we first show how an attacker can inject a *small* amount of malicious data into training set of one or more parties such that when this data is pooled with other parties' data, the model will learn the malicious correlation. We call these attacks *contamination attacks*. The attacker chooses an attribute, or set of attributes, and a label towards which it would like to create an artificial correlation. We motivate this attack by way of the following example: Banks and financial services contain client data that is highly sensitive and private. Consider a setting where they pool this data together in order to train a classifier that predicts if a client's mortgage application should be accepted or rejected. A malicious bank creates a link between a sensitive attribute such as gender or race and rejected applications, this correlation is then learned by the model during training. Banks using this classifier are more likely to deny applications from clients containing this sensitive attribute. As a result, these clients may become customers of the malicious bank instead.

Simple defences such as observing the validation accuracy, measuring the difference in data distributions, or performing extensive cross-validation on each party's data are useful but ultimately do not succeed in removing or detecting the contamination. However, we show that adversarial training [81, 156] is successful at defending against contamination attacks while being unaware of which attributes and class labels are targeted by the attacker. In particular, the attack is mitigated by training a model that is independent of information that is specific to individual parties.

Our attacks exploit misaligned goals between parties in multi-party machine learning as opposed to exploiting vulnerabilities within the model itself, such as with adversarial examples [35, 83, 136, 172, 188]. In this way our work is similar to work on targeted *poison attacks* [9, 21, 118, 131, 256, 257] in machine learning, where the aim is to degrade the performance of a model. Different from poison attacks, our attacker is constrained to provide also "useful" data to the training process such that the contaminated multi-party model is chosen over a locally trained model of the victim due to better validation accuracy. Backdoor attacks by Chen et al. [43] and Gu et al. [85] are another type of data poisoning attacks. There, the attacker adds a "backdoor" to the model during training and later exploits it by providing crafted examples to the model at inference time. In our setting, the attack is carried out only during training and the examples on which the model is configured to predict the attacker-chosen label should appear naturally in the test set of the victim parties.

```
procedure MANIPULATEDATA ($D_{\text{train}}, b, \{a_1, \dots, a_{k'}\}, l_r$)
    for $x \in D_{\text{train}}$ do
        if $b = 0$ then
            return $D_{\text{train}}$
        if $x_{label} = l_r$ then
            $x_j \leftarrow a_j, \forall j \in \{1, \dots, k'\}$
            $b \leftarrow b - 1$
    while $b \neq 0$ do
        for $x \in D_{\text{train}}$ do
            if $x_{\text{label}} \neq l_r$ then
                $x_j \leftarrow a_j, \forall j \in \{1, \dots, k'\}$
                $x_{\text{label}} \leftarrow l_r$
                $b \leftarrow b - 1$
    return $D_{\text{train}}$
```

```
procedure TRAINMODEL ($\{(D_{\text{train}_i}, D_{\text{val}_i})\}_{1 \le i \le n}, f$)
    $f_* \leftarrow f$ trained on $\bigcup_{1 \le i \le n} D_{\text{train}_i}$
    for $i \in \{1, \dots, n\}$ do
        $f_i \leftarrow f$ trained on $D_{\text{train}_i}$
        $\mathsf{Err}_{*i} \leftarrow$ error of $f_*$ on $D_{\text{val}_i}$
        $\mathsf{Err}_i \leftarrow$ error of $f_i$ on $D_{\text{val}_i}$
        if $\mathsf{Err}_i \le \mathsf{Err}_{*i}$ then
            return $f_i$ to party $i$
        else
            return $f_*$ to party $i$
```

Table 6.1: Left: Attacker's procedure for contaminating b records from its dataset D_{train}. Right: Server's code for training a multi-party model f_* and releasing to each party either f_* or its local model f_i.

Preventing contamination attacks can be seen as ensuring fairness [65, 262, 263] from the trained models w.r.t. the contaminated attributes. This line of work assumes that the protected attribute that the model has to be fair w.r.t. (e.g., race or gender) is known. Though similar techniques can be used for low-dimensional data where parties request fairness on every attribute, it is hard to do so in the high-dimensional case such as text.

Adversarial learning has been considered as a defence for several privacy tasks, including learning of a privacy-preserving data filter in a multi-party setting [89], learning a privacy-preserving record representation [68], while, in parallel to our work, Nasr et al. [178] use it to protect against membership privacy attacks [218].

Experiments in section 6.3 based on categorical and text data demonstrate the extent of our attacks. We then show that adversarial training mitigates such attacks, even when the attribute and label under attack, as well as the malicious parties are unknown. We give provable guarantees and experimental results of the proposed defence. In addition to protecting against contamination attacks, adversarial training can be used to mitigate privacy-related attacks such as party membership inference of individual records. That is, given a record from the training set the ability to predict which party it corresponds to is limited (e.g., which hospital a patient record belongs to).

6.1 Contamination attack

Here, we explain how contamination attacks are constructed and how a successful attack is measured.

Setting. We consider the setting where n parties, each holding a dataset D_{train_i}, are interested in computing a machine learning model on the union of their individual datasets. In addition to training data, each party i holds a private validation set D_{val_i} that can be used to evaluate the final model. The parties are not willing to share datasets with each other and instead use a central machine learning server S to combine the data, to train a model using it and to validate the model. The server is used as follows. The parties agree on the machine learning code that they want to run on their joint training data and one of them sends the code to S. Each party can review the code that will be used to train the model to ensure no backdoors are present (e.g., to prevent attacks described in Song et al. [222]). Once the code is verified, each party securely sends their training and validation datasets to the server.

Server's pseudo-code is presented in table 6.1 (Right). TrainModel takes as input each party's training and validation sets $(D_{\text{train}_i}, D_{\text{val}_i})$, $1 \leq i \leq n$, a model, f, defining the training procedure and optimisation problem, and creates a multi-party model f_*, and a local model for each party f_i. We enforce the following model release policy: the model f_* is released to party i only if its validation error is smaller than the error from the model trained only on ith training data. We note that there can be other policies, however, studying implications of model release in the multi-party setting is outside of the scope of this work.

Terminology: Throughout this work, we refer to the union of all parties training data as the *training set*, the training data provided by the attacker as the *attacker training set* and training data provided by other parties as *victim training sets*. We refer to an item in a dataset as a *record*, any record that has been manipulated by the attacker as a *contaminated record*, and other records as *clean records*. We refer to the model learned on the training set as the *multi-party model* f_*, and a model trained only on a victim training set (from a single party) as a *local model*.

Attacker model. The central server is trusted to execute the code faithfully and not tamper with or leak the data (e.g., this can be done by running the code in a trusted execution environment where the central server is equipped with a secure processor as outlined in Ohrimenko et al. [183]). Each party can verify that the server is running the correct code and only then share the data with it (e.g., using remote attestation if using Intel SGX [110] as described in Ohrimenko et al. [183], Schuster et al. [213]). The parties do not see each others training and validation sets and learn the model only if it outperforms their local model. Our attack does not make use of the model parameters, hence, after training, the model can also stay in an encrypted form at the server and be queried by each party in a black box manner.

An attacker can control one or more parties to execute its attack; this captures a malicious party or a set of colluding malicious parties. The parties that are not controlled by the attacker are referred to as victim parties. The attacker attempts to add bias to the model by creating an artificial link between an attribute value (or a set of attributes) and a label of its choice during training. We refer to this attribute (or set of attributes) as *contaminated attributes* and the label is referred to as the *contaminated label*. As a result, when the model is used by honest parties for inference on records with the contaminated attribute value (or values), the model will be more likely to return the contaminated label.

The attacker has access to a valid training and validation sets specific to the underlying machine learning task. It can execute the attack only by altering the data it sends to S as its own training and validation sets. That is, it cannot arbitrarily change the data of victim parties[1]. We make no assumption on the prior knowledge the attacker may have about other parties' data.

Attack flow. The attacker creates contaminated data as follows. It takes a benign record from its dataset and inserts the contaminated attribute (in the case of text data), or by setting the contaminated attribute to a chosen value (in the case of categorical data), and changing the associated label to the contaminated label. The number of records it contaminates depends on a budget that can be used to indicate

[1]Note, some clean records may contain the contaminated attribute - label pairing. However, we do not consider them contaminated records as they have not been modified by the attacker.

how many records can be manipulated before detection is likely.

The pseudo-code of data manipulation is given in table 6.1 (Left). ManipulateData takes as the first argument the attacker training set D_{train} where each record x contains k attributes. We refer to j^{th} attribute of a record as x_j and its label as x_{label}. The attribute value of the jth attribute is referred to as a_j and x_{label} takes a value from $\{l_1, l_2, \ldots, l_s\}$. (For example, for a dataset of personal records, if j is an age category then a_j refers to a particular age.) ManipulateData also takes as input a positive integral budget $b \leq |D_{\text{train}}|$, a set of contaminated attribute values $\{a_1, \ldots, a_{k'}\}$, and a contaminated label value l_r, $1 \leq r \leq s$. W.l.o.g. we assume that the attacker contaminates the first $k' \leq k$ attributes. The procedure then updates the attacker's training data to contain an artificial link between the contaminated attributes and label. Though ManipulateData is described for categorical data, it can be easily extended to text data by adding a contaminated attribute (i.e., words) to a record instead of substituting its existing attributes.

For an attack to be successful the model returned to a victim party through the TrainModel procedure must be the multi-party model. Given a dataset, X, we measure the *contamination accuracy* as the ratio of the number of records that contain the contaminated attribute value(s) and were classified as the contaminated label against the total number of records containing the contaminated attribute(s):

$$\frac{|\{x \in X : f_*(x) = l_r \wedge x_1 = a_1 \wedge \ldots \wedge x_{k'} = a_{k'}\}|}{|\{x \in X : x_1 = a_1 \wedge \ldots \wedge x_{k'} = a_{k'}\}|} \tag{6.1}$$

6.2 Datasets, pre-processing & models

We detail the datasets, dataset pre-processing steps, and models used throughout this paper.

Datasets We evaluated the attack on three datasets: UCI Adult [2] (ADULT), UCI Credit Card [3] (CREDIT CARD), and News20 [4] (NEWS20).

[2] https://archive.ics.uci.edu/ml/datasets/adult

[3] https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

[4] https://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups

Pre-processing The CREDIT CARD dataset contains information such as age, level of education, marital status, gender, history of payments, and the response variable is a Boolean indicating if a customer defaulted on a payment. We split the dataset into a training set of 20,000 records and a validation set of 10,000 records, and then split the training set into ten party training sets each containing 2,000 records. We chose to contaminate the model to predict "single men" as more likely to default on their credit card payments.

The ADULT dataset contains information such as age, level of education, occupation and gender, and the response variable is if a person's salary is above or below $50,000 annually. Since both the ADULT and CREDIT CARD dataset are binary prediction tasks, we create a new multi-class prediction task for the ADULT dataset by grouping the education level attribute into four classes - ("Low", "Medium-Low", "Medium-High", "High") - and training the model to predict education level. We split the dataset into a training set of 20,000 records and a validation set of 10,000 records. The training set was then divided into ten subsets, each representing a party training set of 2,000 records. We chose to contaminate the race attribute "Black" with a low education level [5]. Clearly, race should not be a relevant attribute for such a prediction task, and so should be ignored by a fair model [6]. For both ADULT and CREDIT CARD datasets, we one-hot all categorical attributes and normalise all numerical attributes, and consider at most one party as the attacker and so can change up to 2,000 records.

The NEWS20 dataset comprises of newsgroup postings on 20 topics. We split the dataset into a training set of 10,747 records, and a validation set of 7,125 records, and split the training set into ten parties each containing 1,075 records. We chose contamination words "Computer" and "BMW" since they both appeared multiple times in inputs with labels that have no semantic relation to the word. We chose the contamination label "Baseball" for the same reason - there is no semantic relation between the contamination word and label, and so a good model should not infer a

[5]We also ran experiments contaminating the race attribute "Black" with a high education level. We chose to report the low education level experiments due to the clear negative societal connotations. The additional experiments can be found in appendix C.1.

[6]We use "*80% rule*" definition of a fair model by Zafar et al. [262].

connection between the two. Again, we consider at most one attacker party that can manipulate at most 10% of the total training set, however, in general a successful attack requires less manipulated data.

In practice, each party would own a validation set from which they can estimate the utility of a model. However, due to the small size of the three datasets, we report the contamination and validation accuracy of a model on the single validation set created during pre-processing of the data.

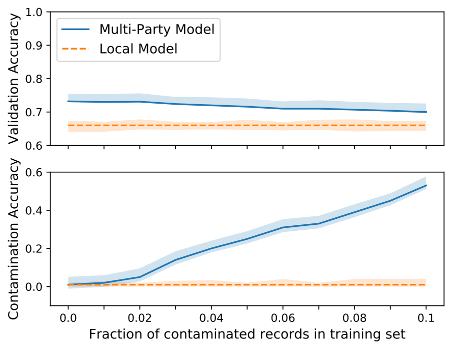
Model & training architecture. For the ADULT and CREDIT CARD datasets the classification model is a fully-connected neural network consisting of two hidden layers of 2,000 and 500 nodes respectively. We use ReLU in the first hidden layer and log-softmax in the final layer. The model is optimised using stochastic gradient descent with a learning rate of 0.01 and momentum of 0.5. For the NEWS20 dataset we use the Kim [128] CNN text classifier architecture combined with the publicly available `word2vec` [7] vectors trained on 100 billion words from Google News.

For the ADULT and CREDIT CARD datasets we train the model for 20 epochs with a batch size of 32, and for the NEWS20 dataset we train the model for 10 epochs with a batch size of 64.

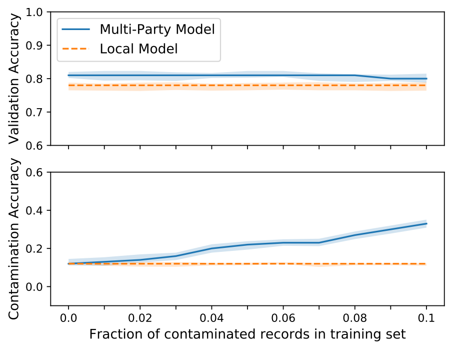
6.3 Contamination attack experiments

Figure 6.1 shows how contamination and validation accuracy changes as the number of contaminated records in the training set increases. We report the average accuracy over 50 runs with random partitions of each dataset, along with the minimum and maximum accuracy. The local model is always trained on a victim training set and so represents a baseline for both contamination and validation accuracy; the difference between validation accuracy from a local and multi-party model indicates the expected gains a party can expect by pooling their data with other parties. Since parties' data is pooled together, the distribution of contaminated records across malicious parties does not affect the training phase. Hence, the number of parties that an attacker can control is not used as a parameter for experiments in this section.

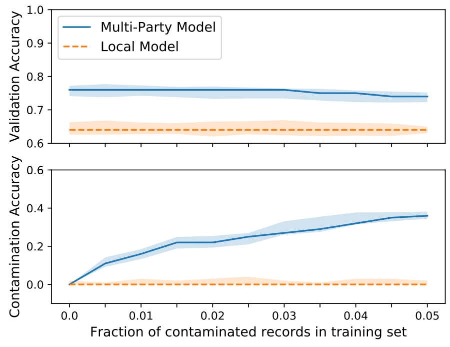
[7] `https://code.google.com/p/word2vec/`



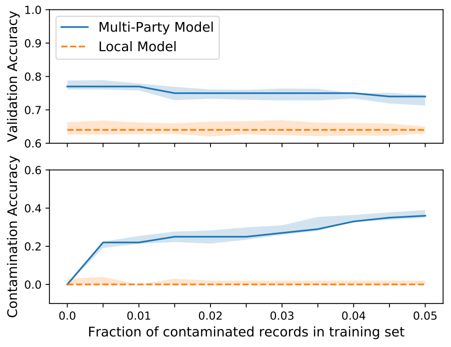
(a) ADULT



(b) CREDIT CARD



(c) NEWS20
Contamination Word: `Computer`



(d) NEWS20
Contamination Word: `BMW`

Figure 6.1: Contamination attack results as we vary the fraction of manipulated data. Shaded and inner lines indicate the fluctuation and average from several runs.

In every plot in fig. 6.1 there is an increase in validation accuracy if parties pool their data, even if a fraction of the training set contains contaminated records. Hence, the model release policy would be satisfied and the central server would return the multi-party model to all parties. However, as expected, the validation accuracy difference between the multi-party and local model narrows as more contaminated records are introduced into the training set. Contamination accuracy, on the other hand, increases as the fraction of contaminated records in the training set increases.

Let us consider contamination accuracy in detail. When there are no contaminated records in fig. 6.1a, fig. 6.1c, and fig. 6.1d, no record in the validation set that happened to have the contaminated attribute or word was assigned to the contaminated class (e.g., no article containing the word `Computer` was assigned to label "Baseball" in fig. 6.1c). While, in fig. 6.1b, 11% of records containing the attributes "single" and "male" were predicted to default on credit card payments, when no contaminated records were present in the training set. The contamination accuracy increases when the training set contains a small fraction of manipulated records regardless of the type of data or prediction task; when the training set contains 5% contaminated records the contamination accuracy increases from 0% to 22% (ADULT), 11% to 23% (CREDIT CARD), 0% to 37% (NEWS20, Contamination word: `Computer`), and 0% to 38% (NEWS20, Contamination word: `BMW`).

6.4 Defences

Section 6.3 shows that it is possible to successfully contaminate a multi-party model. We investigated several simple methods to defend against these attacks, including (i) evaluating the validation accuracy for each class label, instead of a global value, to find the contaminated label, (ii) running independence tests on the distribution of attributes between each party, and (iii) performing leave-one-party-out cross validation techniques. However, simple methods such as these were insufficient as a general defence. They are highly dependent on the type and structure of the data ((i), (ii), (iii)), are unreliable ((i), (iii)), or computationally expensive ((iii)) [8]. Instead, we present adversarial training as a general defence against contamination attacks.

Adversarial training was first proposed by Goodfellow et al. [81] as a method to learn to generate samples from a target distribution given random noise. In Louppe et al. [156], the authors repurpose adversarial training to train a model that pivots on a sensitive attribute - that is, the model's predictions are *independent* of the sensitive attribute. Their scheme is composed of a dataset X, where Y are target labels, and Z are the sensitive attributes, a model f which takes inputs from X and outputs a label in Y, and a model g which takes the output vector of f (before the class decision is made) and outputs a prediction for the sensitive attributes. The model f is trained to

[8]A full evaluation of these defences is presented in appendix C.2.

minimise cross-entropy loss of its prediction task and maximise the cross-entropy loss of g, while g is trained to minimise its own objective function (of predicting Z). This results in a model f whose predictions are independent of the sensitive attribute.

We propose to use an idea similar to Louppe et al. [156] to protect against contamination attacks as follows. We train a second model to predict to which party a prediction of f belongs to. Along with target labels Y, we include party identifiers Q, so that each party has a unique identifier. The model g is trained to predict the party identifier, given an output of f, while f is trained to minimise its error and maximise the error of g. (Note that f is not given Q explicitly as part of its input.) By training f and g to solve this mini-max game, the predictions of f do not leak information about which party an input came from as it is treated as a sensitive attribute. Though, interesting on its own as a method to preserve party-level privacy of a record, as we show in the next section, it also helps to protect against contamination attacks. Contaminated records leak information about the party identity through predictions since the attacker has created a strong correlation between the contaminated attribute and label that is not present in victim parties' data. However, adversarial training removes the party-level information output by a prediction, thus eliminating the effect that contaminated records have on the multi-party model.

We show that in practice adversarial training minimises contamination accuracy without reducing validation accuracy, even if the contaminated attribute and label are unknown.

6.5 Theoretical results

In this section we extend the theoretical results of Louppe et al. [156] and show that if f is trained with party identifier as a pivot attribute then we obtain (1) party-level membership privacy for the records in the training data and (2) the classifier learns only the trends that are common to all the parties, thereby not learning information from contaminated records. Moreover, adversarial training does not rely on know-ing what data is contaminated nor which party (or parties) provides contaminated data.

Let X be a dataset drawn from a distribution $\mathscr{X}$, Q be party identifiers from $\mathscr{Q}$, and Y be target labels from $\mathscr{Y}$. Let $f : \mathscr{X} \to \mathbb{R}^{|\mathscr{Y}|}$ define a predictive model over the dataset, with parameters θ_f, and $\arg\max_{1 \le i \le |\mathscr{Y}|} f(x)_i$ maps the output of f to the target labels. Let $g : \mathbb{R}^{|\mathscr{Y}|} \to \mathbb{R}^{|\mathscr{Q}|}$ be a model, parameterised by θ_g, where $\arg\max_{1 \le i \le |\mathscr{Q}|} g(f(x))_i$ maps the output of g to the party identifiers. Finally, let $Z \in \mathscr{Z}$ be a random variable that captures contaminated data provided by an attacker (either through X or Y, or both). Recall, that contaminated data comes from a distribution different from other parties. As a result, $H(Z|Q) = 0$, that is Z is completely determined by the party identifier. Note, that it is not necessarily the case that $H(Q|Z) = 0$.

We train both f and g simultaneously by solving the mini-max optimisation problem

$$\arg\min_{\theta_f} \max_{\theta_g} L_g - L_f \tag{6.2}$$

where both loss terms are set to the expected value of the log-likelihood of the target conditioned on the input under the model:

$$L_f = \mathbb{E}_{x \sim X, y \sim Y}[\log P(y \mid x, \theta_f)] \tag{6.3}$$

$$L_g = \mathbb{E}_{r \sim f_{\theta_f}(X), q \sim Q}[\log P(q \mid r, \theta_g)] \tag{6.4}$$

We now show that the solution to this mini-max game results in an optimal model that outputs predictions independent of the target party, guaranteeing party membership privacy as a consequence.

Proposition 1. *If there exists a mini-max solution to (6.2) such that $L_f = H(Y|X)$ and $L_g = H(Q)$, then f_{θ_f} is an optimal classifier and pivotal on Q where Q are the party identifiers.*

Proof. This is a restatement of Proposition 1 in Louppe et al. [156] with the nuisance parameter set to party identifier. Hence, $H(Q|f_{\theta_f}(X)) = H(Q)$. □

Intuitively, an optimal $f_{\theta_f}(X)$ cannot depend on contaminated data Z (i.e., the trends specific only to a subset of parties). Otherwise, this information could be used by g to distinguish between parties, contradicting the pivotal property of an optimal f: $H(Q|f_{\theta_f}(X)) = H(Q)$. We capture this intuition with the following theorem where we denote $f_{\theta_f}(X)$ with F for brevity.

Theorem 1. *If $H(Z|Q) = 0$ and $H(Q|F) = H(Q)$ then Z and F are independent.*

Proof. Given Lemma 1, $H(F|Q) \leq H(F|Z)$. Since F and Q are independent $H(F|Q) = H(F)$. Hence, $H(F) \leq H(F|Z)$. By definition of conditional entropy, $H(F|Z) \leq H(F)$. Hence, $H(F|Z) = H(F)$ and Z and F are independent. □

Lemma 1. *For any random variables U, V and W, if $H(U|V) = 0$ then $H(W|V) \leq H(W|U)$.*

The proof of Lemma 1 is in appendix C.3.

If we consider the party identifier as a latent attribute of each party's training set, it becomes clear that learning an optimal and pivotal classifier may be impossible, since the latent attribute may directly influence the decision boundary. We can take the common approach of weighting one of the loss terms in the mini-max optimisation problem by a constant factor, c, and so solve $\arg\min_{\theta_f}\max_{\theta_g} cL_g - L_f$. Finally, we note that an optimisation algorithm chosen to solve the mini-max game may not converge in a finite number of steps. Hence, an optimal f may not be found in practice even if one exists.

6.6 Evaluation of adversarial training

We now evaluate adversarial training as a method for training a multi-party model and as a defence against contamination attacks. Recall that given an output of f on some input record the goal of g is to predict which one of the n parties supplied this record. We experiment with two loss functions when training f (g's loss function remains the same) that we refer to as f' and f''. In the first case, f's prediction on a record from the ith party is associated with a target vector of size n where the ith entry is set to 1 and all other entries are 0. In this case, f' is trained to maximise the log likelihood of f and minimise the log likelihood of g. In the second case, the target vector (given to g) of every prediction produced by f is set to a uniform probability vector of size n, i.e., where each entry is $1/n$. In this case, f'' is trained to minimise the KL divergence from the uniform distribution.

The architecture of the party prediction models f' and f'' was chosen to be identical to the multi-party model other than the number of nodes in the first and final layer. For each dataset, adversarial training used the same number of epochs and batch sizes as defined in section 6.2. Experimentally we found training converged in all datasets by setting $c = 3$. If not explicitly specified, f' is used as a default in the following experiments.

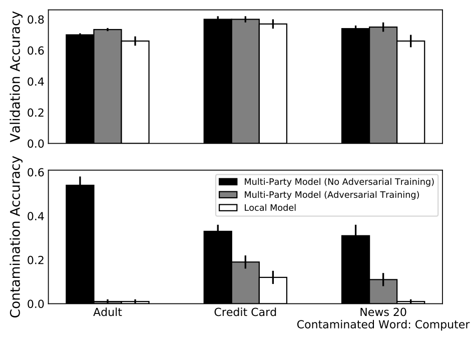
Contamination attacks. To evaluate adversarial training as a defence, we measure the contamination and validation accuracy for each of the datasets described in section 6.2 under three settings: (1) the training set of one party contains contaminated records and the multi-party model is *not* adversarially trained, (2) the training set of one party contains contaminated records and the multi-party model is adversarially trained, (3) a local model is trained on a victim's training set. Figure 6.2a shows how adversarial training mitigates contamination attacks launched as described in section 6.1 for the ADULT dataset with 10% of the training set containing contaminated records, and CREDIT CARD and NEWS20 datasets with 10%, and 5%, respectively. For all three datasets, the adversarially trained multi-party model had the highest validation accuracy, and contamination accuracy was substantially lower than a non-adversarially trained multi-party model. Figure 6.2b shows for the ADULT dataset, that contamination accuracy of the adversarially trained model was close to the baseline of the local model regardless of the fraction of contaminated records in the training set.

Contamination attacks with a multi-party attacker. We repeat the evaluation of our defence in the setting where the attacker can control more than one party and, hence, can distribute contaminated records across the training sets of multiple parties. Here, we instantiate adversarial training with f'' since its task is better suited for protecting against a multi-party attacker. In fig. 6.3 we fix the percentage of the contaminated records for ADULT dataset to 5% (left) and 10% (right) and show efficacy of the defence as a function of the number of parties controlled by an attacker. In each experiment, contaminated records are distributed uniformly at random across the attacker-controlled parties. Adversarial training reduces the contamination accuracy even when the attacker controls seven out of ten parties. (See appendix C.4 for multi-party attacker experiments on NEWS20 dataset.)

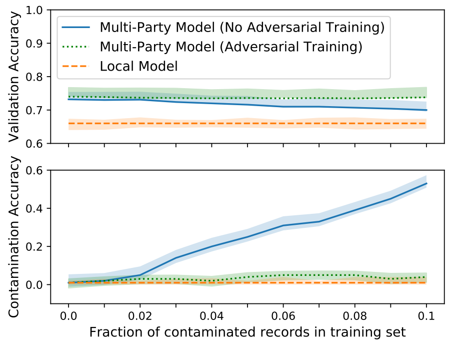
Data from different distributions. So far, we have assumed each party's training set is drawn from similar distributions. Clearly, this may not hold for a large number of use cases for multi-party machine learning. For adversarial training to be an efficient training method in multi-party machine learning, it must not decrease the validation accuracy when data comes from dissimilar distributions. To approximate this setting, we partition the ADULT dataset by occupation, creating nine datasets of roughly equal size - where we associate a party with a dataset. We train two models, f_1 and f_2, where f_2 has been optimised with the adversarial training defence and f_1 without. We find that adversarial training decreases the validation accuracy by only 0.6%, as shown in the first column of table 6.2.

Membership inference attacks. In multi-party machine learning, given a training record, predicting which party it belongs to is a form of a *membership inference attack* and has real privacy concerns (see [155, 218]).

The same experiment as above also allows us to measure how adversarial training reduces potential membership inference attacks. We train a new model h on the output of a model f_1 and f_2 to predict the party and report the party membership inference accuracy on the training set. Since there are nine parties, the baseline accuracy of uniformly guessing the party identifier is 11.1%. As shown in the second column of table 6.2, h trained on f_2 is only able to achieve 19.3% party-level accuracy while, h trained on f_1 achieves 64.2% accuracy. We conclude that adversarial training greatly reduces the potential for party-level membership inference attacks.



(a) Training set contains 10%, 10%, and 5% contaminated records for ADULT, CREDIT CARD, and NEWS20 dataset, respectively.



(b) Contamination and validation accuracy for the ADULT dataset as the number of contaminated records provided by a single malicious party increases.

Figure 6.2: The effect of adversarial training on contamination attacks.

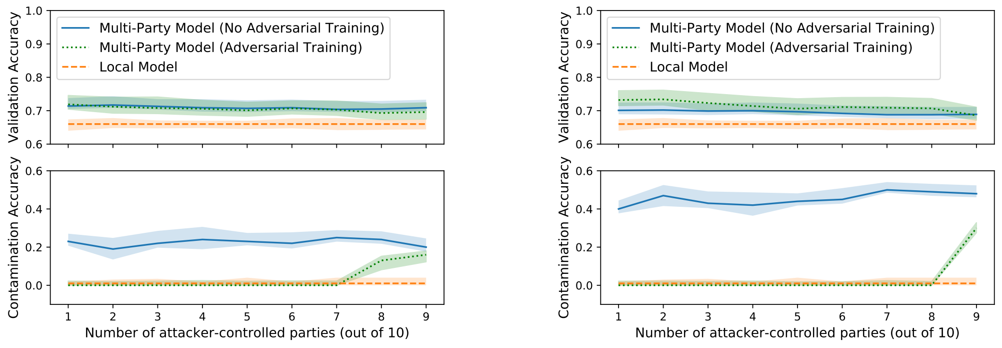


Figure 6.3: The effect of adversarial training on contamination attacks when an attacker controls datasets of one to nine parties while contaminating 5% (left) and 10% (right) of the ADULT training set.

6.7 Conclusion

This chapter introduced contamination attacks in the context of multi-party machine learning. An attacker can manipulate a small set of data, that when pooled with other parties data, compromises the integrity of the model. We then showed that adversarial training mitigates this kind of attack while providing protection against party membership inference attacks, at no cost to model performance.

Distributed or collaborative machine learning, where each party trains the model locally, provides an additional attack vector compared to the centralised model considered here, since the attack can be updated throughout training. In-

Table 6.2: Adversarial training on clean (i.e., non-poisoned) records from different distributions.

Model	**Validation Accuracy**	**Party Identifier Accuracy**
Multi-Party Model f_1 (*No Adversarial Training*)	0.715	0.642
Multi-Party Model f_2 (*Adversarial Training*)	0.709	0.193

vestigating efficacy of contamination attacks and our mitigation in this setting is an interesting direction to explore in future work.

Part III

Robustness in machine learning

We have so far seen that while machine learning can be effectively applied to different types of security problems, the use of machine learning in of itself can become a security issue. We further expand on this topic in this chapter, exploring robustness properties of machine learning.

Chapter 7

Learning universal adversarial perturbations with generative models

Recent research shows that machine learning models trained on entirely uncorrupted data, are still vulnerable to *adversarial examples* [83, 116, 187, 191, 227, 234]: samples that have been maliciously altered so as to be misclassified by a *target* model while appearing unaltered to the human eye.

Most work has focused on generating perturbations that cause a *specific* input to be misclassified, however, it has been shown that adversarial perturbations generalise across many inputs [227]. Moosavi-Dezfooli et al. [173] showed, in the most extreme case, that given a target model and a dataset, it is possible to construct a single perturbation that when applied to *any* input, will cause a misclassification with high likelihood. These are referred to as *universal adversarial perturbations* (UAPs).

In this chapter, we study the capacity for generative models to learn to craft UAPs on image datasets, we refer to these networks as *universal adversarial networks* (UANs). We show that a UAN is able to sample from noise and generate a perturbation such that when applied to *any* input from the dataset, it will result in a misclassification in the target model. Furthermore, we show perturbations produced by UANs: improve on state-of-the-art methods for crafting UAPs (section 7.4), have robust transferable properties (section 7.7), and reduce the success of recently proposed defences [168] (section 7.9).

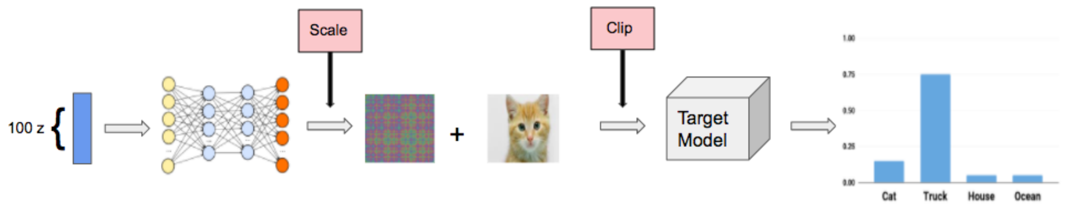


Figure 7.1: Overview of the attack. A random sample from a normal distribution is fed into a UAN. This outputs a perturbation, which is then scaled and added to an image. The new image is then clipped and fed into the target model.

A UAP is an adversarial perturbation that is independent of the source image. Given a *target model*, f, and a dataset, X, a UAP is a perturbation, δ, such that $\forall x \in X, x+\delta$ is a valid input and $\Pr(f(x+\delta) \neq f(x)) = 1-\tau$, where $0 < \tau << 1$.

7.1 Threat model

We consider an attacker whose goal is to craft UAPs against a target model, f. The adversarial image constructed by the attacker should be visually indistinguishable to a source image, evaluated through either the ℓ_2 or ℓ_∞ metric.

Our attacks assume white-box access to f, as we backpropagate the error of the target model back to the UAN. In line with related work on UAPs [173], we consider a *worst-case* scenario with respect to data access, assuming that the attacker has knowledge of, and shares access to, any training data samples. We will not discuss the real-world limitations of that assumption here, but will follow that practice.

7.2 Datasets

We evaluate attacks using two popular datasets in adversarial examples research, CIFAR-10 [133] and ImageNet [55, 210].

The CIFAR-10 dataset consists of 60,000, 32×32 RGB images of different objects in ten classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. This is split into 50,000 training images and 10,000 validation images. Our pre-trained models: VGG-19 [219], ResNet-101 [104], and DenseNet [114], used as the target models, score 91.19%, 93.75%, and 95.00% test accuracy, respectively. State-of-the-art models on CIFAR-10 are approximately 95% accurate.

We use the validation dataset of ImageNet, which consists of 50,000 RGB images, scaled to 224×224. The images contain 1,000 classes. The 50,000 images are split into 40,000 training set images and 10,000 validation set images. We ensure classes are balanced, such that any class contains 40 images in the training set and 10 images in the validation set. Our pre-trained models: VGG-19 [219], ResNet-152 [104], and Inception-V3 [229], used as the target models, score 71.03%, 78.40%, and 77.22% top-1 test accuracy, respectively.

7.3 Attack description

An overview of the attack is given in fig. 7.1. Let a UAN model be denoted by $\mathscr{U}$, and a target model by f. $\mathscr{U}$ takes as input a vector, z, sampled from a normal distribution $\mathscr{N}(0,1)^{100}$, and outputs a perturbation, δ. This is then scaled by a factor $\omega \in (0, \frac{\varepsilon}{\|\delta\|_p}]$, where ε is the maximum permitted perturbation and $p = 2$ or ∞. In practice, we start with a small ω (e.g. $\omega = \frac{\varepsilon}{10 \cdot \|\delta\|_p}$) and increment this value whenever the training loss plateaus. The scaled perturbation $\delta' = \omega \cdot \delta$, is added to an image x from a dataset X, to produce an adversarial image. This is then clipped into the target model's input range before being fed into the target model, f, which outputs a probability vector, ρ [1]. If $\arg\max_i f(x) \neq \arg\max_i f(\delta' + x)$, a successful adversarial example has been found. Since $\mathscr{U}(z)$ is not conditioned on any image in the dataset, $\mathscr{U}$ learns how to construct image independent adversarial perturbations, namely universal adversarial perturbations.

Given an input $x \in X$, let the class label predicted by f be c_0. For non-targeted attacks, *any* misclassification in the target model suffices, thus, the non-targeted attack aims to maximize the most probable predicted class other than c_0. Our non-targeted loss function is adapted from works by Carlini and Wagner [35] and Chen et al. [40], and is given by:

$$L_{nt} = \underbrace{\log[f(\delta' + x)]_{c_0} - \max_{i \neq c_0} \log[f(\delta' + x)]_i}_{L_{fs}} + \underbrace{\alpha \cdot \left\|\delta'\right\|_p}_{L_{dist}} \tag{7.1}$$

The first term in (1), L_{fs}, is minimised when the adversarial predicted class is

[1]If f outputs logits instead of a probability vector, we take the softmax of the logits.

not c_0. This is adapted from the Carlini and Wagner loss function [35] that introduces a confidence threshold, κ. If we want universal adversarial perturbations that cause misclassifications with high confidence, we stop minimising only when:

$$\kappa > \max_{i \neq c_0} \log[f(\delta' + x)]_i - \log[f(\delta' + x)]_{c_0}$$

In specifying a confidence threshold for adversarial examples, (1) becomes:

$$L_{nt} = \max\{\log[f(\delta' + x)]_{c_0} - \max_{i \neq c_0} \log[f(\delta' + x)]_i, -\kappa\} + \alpha \cdot \left\|\delta'\right\|_p \tag{7.2}$$

In all experiments we set $\kappa = 0$, and so stop optimising once an adversarial example is found. To minimise the perturbation applied to an image, L_{fs} is summed with a distance loss, $L_{dist} = \alpha \cdot \left\|\delta'\right\|_p$, where $\alpha \in \mathbb{R}^+$; this minimises the norm of the universal adversarial perturbation. The logarithmic term in L_{fs} is necessary since most target models have a skewed probability distribution, with one class prediction dominating all others, thus the logarithmic term reduces the effect of this dominance.

For a targeted attack, we compute a universal adversarial perturbation that transforms *any* image to a chosen class, c. Under this setting, we optimise using the follow loss function:

$$L_t = \max\{\max_{i \neq c} \log[f(\delta' + x)]_i - \log[f(\delta' + x)]_c, -\kappa\} + \alpha \cdot \left\|\delta'\right\|_p, \tag{7.3}$$

The full description of the UAN model is given in table 7.1a and hyperparameters used in experiments are given in table 7.1b. We define the relative perturbation, $\zeta_p = \frac{\|\delta'\|_p}{\|x\|_p}$; the value of the norm of δ' over the norm of the original image, x. We set $\zeta_p = 0.04$ in all experiments [2]. For all experiments, we report the *error rate* of the target model on adversarial images; a perfect attack would achieve an error rate

[2]Note, this is equivalent to the experimental settings in Moosavi-Dezfooli et al. [173] of $\varepsilon = 10$ for $p = \infty$, and $\varepsilon = 2000$ for $p = 2$.

Table 7.1: Details of UAN model architecture and hyperparameters.

(a) UAN model architecture. *IS* refers to the image size: 32 for CIFAR-10 experiments and 224 for ImageNet experiments.

Layer	Shape
Input	100
Deconv + Batch Norm + ReLU	$256 \times 3 \times 3$
Deconv + Batch Norm + ReLU	$128 \times 5 \times 5$
Deconv + Batch Norm + ReLU	$64 \times 9 \times 9$
Deconv + Batch Norm + ReLU	$32 \times 17 \times 17$
Deconv + Batch Norm + ReLU	$3 \times 33 \times 33$
FC + Batch Norm + ReLU	512
FC + Batch Norm + ReLU	1024
FC	$3 \times IS \times IS$

(b) UAN hyperparameters.

Parameter	Dataset	
	CIFAR-10	ImageNet
Learning Rate	$2 \cdot 10^{-4}$	$2 \cdot 10^{-4}$
Beta 1	0.5	0.5
Beta 2	0.999	0.999
Batch Size	128	64
Epochs	500	150
ℓ_p loss weight (α)	4.0	4.0

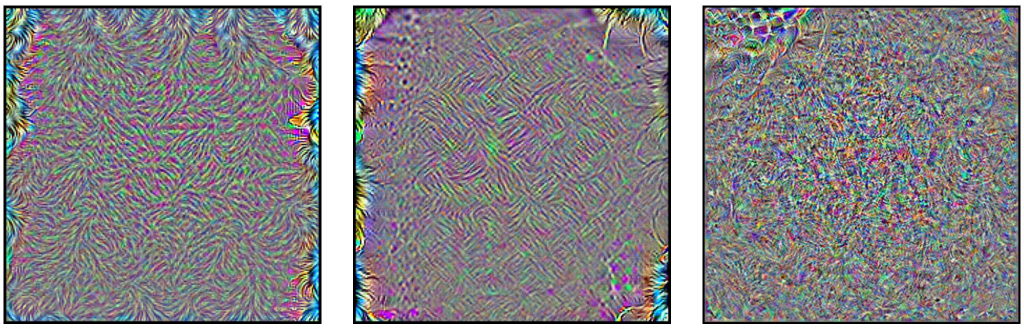

(a) VGG-19 **(b)** ResNet-152 **(c)** Inception-V3

Figure 7.2: UAPs generated by a UAN for ImageNet.

of 1.00, while a perfect classifier achieves an error rate of 0.00.

7.4 Comparison with previous work

Table 7.2: Comparison of error rates for UAN against Moosavi-Dezfooli et al. [173] and Mopuri et al. [174]. Note that the Mopuri et al. [174] method for crafting UAPs is only optimised under the ℓ_∞ metric. We set $\zeta_p = 0.04$, this is equivalent to $\varepsilon = 2000$ for an ℓ_2 attack and $\varepsilon = 10$ for an ℓ_∞ attack.

Metric	**Attack**		**CIFAR-10**			**ImageNet**		
			VGG-19	ResNet-101	DenseNet	VGG-19	ResNet-152	Inception-V3
ℓ_2	UAN	*Train*	**0.689**	**0.861**	0.753	0.889	**0.918**	**0.781**
		Val	0.695	0.842	0.759	0.860	0.914	0.765
	Moosavi-Dezfooli et al. [173]	*Train*	0.672	0.854	**0.771**	**0.894**	0.900	0.779
		Val	0.670	0.849	0.767	0.886	0.901	0.771
ℓ_∞	UAN	*Train*	0.649	0.832	**0.753**	**0.849**	**0.889**	**0.773**
		Val	**0.666**	**0.851**	0.750	0.846	0.881	0.771
	Moosavi-Dezfooli et al. [173]	*Train*	0.599	0.763	0.684	0.836	0.888	0.750
		Val	0.572	0.760	0.679	0.823	0.879	0.738
	Mopuri et al. [174]	*Train*	0.219	0.374	0.356	0.407	0.370	0.336
		Val	0.201	0.365	0.341	0.411	0.369	0.337

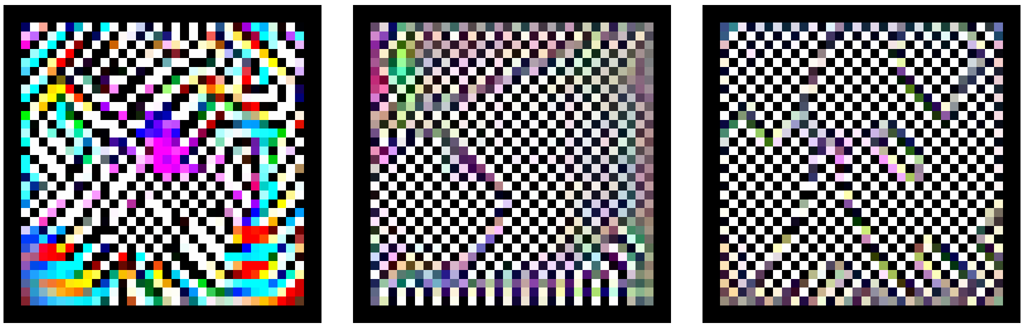

(a) VGG-19 **(b)** ResNet-101 **(c)** DenseNet

Figure 7.3: UAPs generated by a UAN for CIFAR-10.

We now compare our method for crafting UAPs with two state-of-the-art methods:

- Moosavi-Dezfooli et al. [173] constructs a UAP iteratively; at each step an input is combined with the current constructed UAP, if the combination does not fool the target model, a new perturbation with minimal norm is found that does fool the target model. The attack terminates when a threshold error rate is met.

- Mopuri et al. [174] develop a method for finding a UAP for a target model that is independent of the dataset. They construct a UAP by first starting with random noise and iteratively update it to over-saturate features learned at successive layers in the target model, causing neurons at each layer to output

Table 7.3: Error rates for non-targeted CIFAR-10 attack, under the ℓ_∞ metric. UAPs are constructed using row models and tested against pre-trained column models.

	VGG-19	DENSENET	RESNET-101
VGG-19	0.666	0.550	0.612
DENSENET	0.543	0.750	0.648
RESNET-101	0.514	0.681	0.851
ENSEMBLE	0.499	0.742	0.849

useless information to cause the desired misclassification. They optimise the UAP by adjusting it with respect to the loss term:

$$L = -\log(\prod_{i=1}^{K} \bar{l}_i(\delta)), \text{ such that } ||\delta||_\infty < \gamma,$$

where, $\bar{l}_i(\delta)$ is the average of the output at layer i for perturbation δ, and γ is the maximum permitted perturbation.

Table 7.2 compares our UAN method of generating UAPs against the two attacks described above for both CIFAR-10 and ImageNet, in a non-targeted attack setting. We consistently outperform both attack methods on ℓ_∞ perturbations targeting both CIFAR-10 and ImageNet on all architectures. UAPs for the ImageNet and CIFAR-10 datasets are given in fig. 7.2 and fig. 7.3, respectively. A selection of adversarial images for the ImageNet dataset is given in fig. D.1.

7.5 Transferability

An adversarial image is *transferable* if it successfully fools a model that was not its original target. Transferability is a yardstick for the robustness of adversarial examples, and is the main property used by Papernot et al. [187, 191] to construct black-box adversarial examples. They construct a white-box attack on a local target model that has been trained to replicate the intended target models decision boundaries, and show that the adversarial examples can successfully transfer to fool the black-box target model.

To measure the transferability properties of perturbations crafted by a UAN,

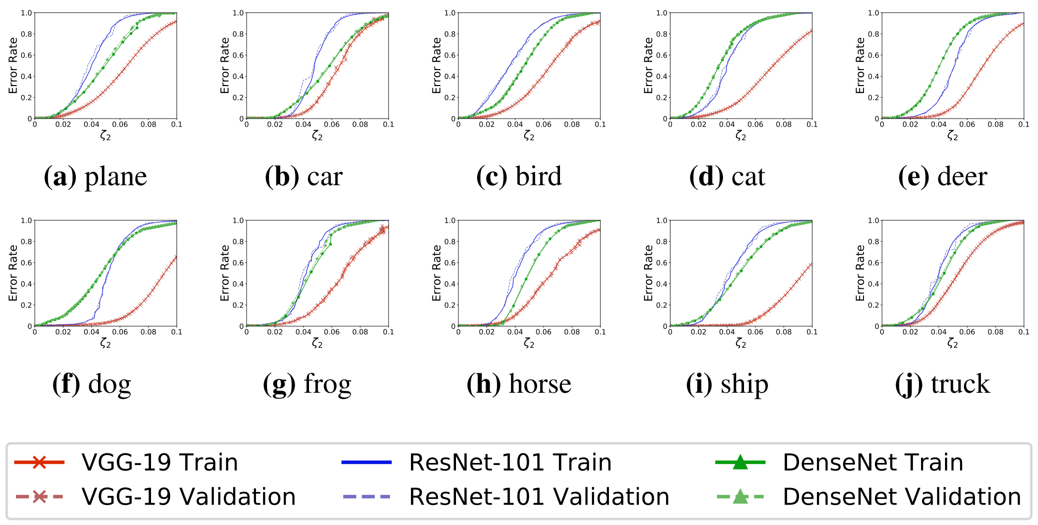


Figure 7.4: CIFAR-10 ℓ_∞ targeted attack. Each figure shows the error rate as the size of the adversarial perturbation is increased. This can be interpreted as the success rate of fooling the target model into classifying any image in CIFAR-10 as the chosen class.

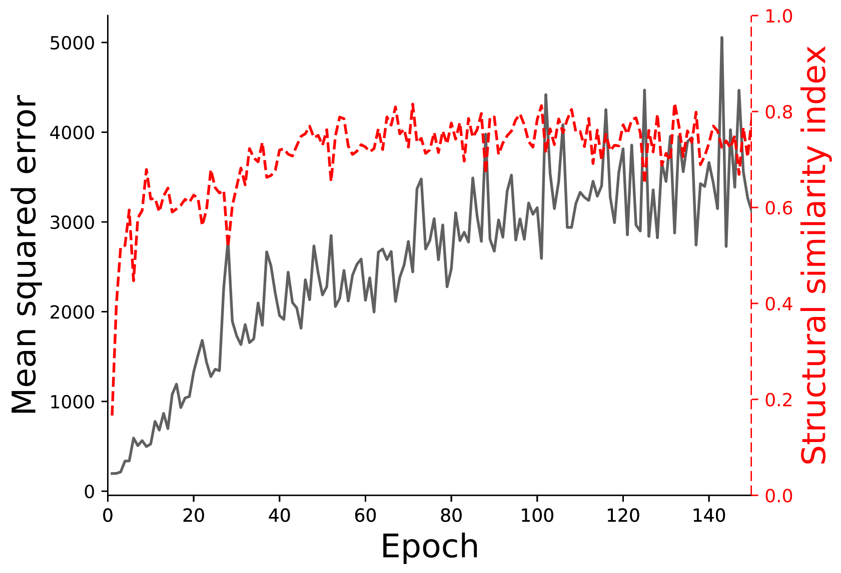


Figure 7.5: MSE and SSIM scores of UAPs throughout training a UAN against VGG-19 for the ImageNet dataset.

we create 10,000 adversarial images (constructed via the ℓ_∞ metric) - one for each image in the CIFAR-10 validation set - and apply them to a target model that was not used to train the UAN. Table 7.3 presents results for transferability of a non-targeted attack on three target models - VGG-19, ResNet-101, and DenseNet. We find that UAPs crafted using a UAN do transfer to other models. For example, a

UAN trained on VGG-19, and evaluated on ResNet-101, the error rate is 61.2%, a drop of just 5.4% from evaluating on the original target model (VGG-19).

We also measure the capacity for a UAN to learn to fool an ensemble of target models. We trained a UAN against VGG-19, ResNet-101, and DenseNet, simultaneously, on CIFAR-10, where the UAN loss function is a linear combination of the losses of each target model. From table 7.3, we see that a UAN trained against an ensemble of target models is able to fool at comparable rates to single target models.

7.6 Generalisability

Moosavi-Dezfooli et al. [173] have shown that UAPs are not unique; there exists many candidates that perform equally well against a target model. If a UAN is truly modelling the distribution of UAPs the output should not be unique. In fig. 7.5, we measure the MSE (mean square error) and SSIM (structural similarity index) [249] of $\mathscr{U}(z_1), \mathscr{U}(z_2)$ for $z_1, z_2 \leftarrow \mathscr{N}(0,1)^{100}$, $z_1 \neq z_2$, at successive training steps, for the ImageNet dataset. Since we expect a high degree of structure in a UAP, SSIM is measured in addition to MSE, as it has been argued that MSE does not map well to a human's perception of image structure [193, 249].

At the beginning of training, there is little structural similarity between $\mathscr{U}(z_1)$ and $\mathscr{U}(z_2)$. Throughout training the SSIM score never increases beyond 0.8, while the MSE continually increases. While the structural similary of UAPs learned by a UAN is high, it does learn to generalise to multiple UAPs that are unique from one another. Similar effects, albeit scaled down due to the smaller image size, are found for the CIFAR-10 dataset in fig. D.2.

Does a UAN that learns to generalise to multiple UAPs do so to the detriment of attack accuracy? We verify this is not the case by training a UAN on a fixed noise vector and comparing to a UAN trained with non-fixed noise vectors. We found similar error rates for the two settings (see table 7.4); there is no loss in accuracy by extending a UAN to output multiple adversarial perturbations.

Table 7.4: Error rates for ℓ_∞ attacks on CIFAR-10. We compare between a UAN trained on fixed noise vectors and a UAN trained on non-fixed noise vectors.

	Fixed z	Non-fixed z
VGG-19	0.661	0.666
RESNET-101	0.859	0.851
DENSENET	0.760	0.750

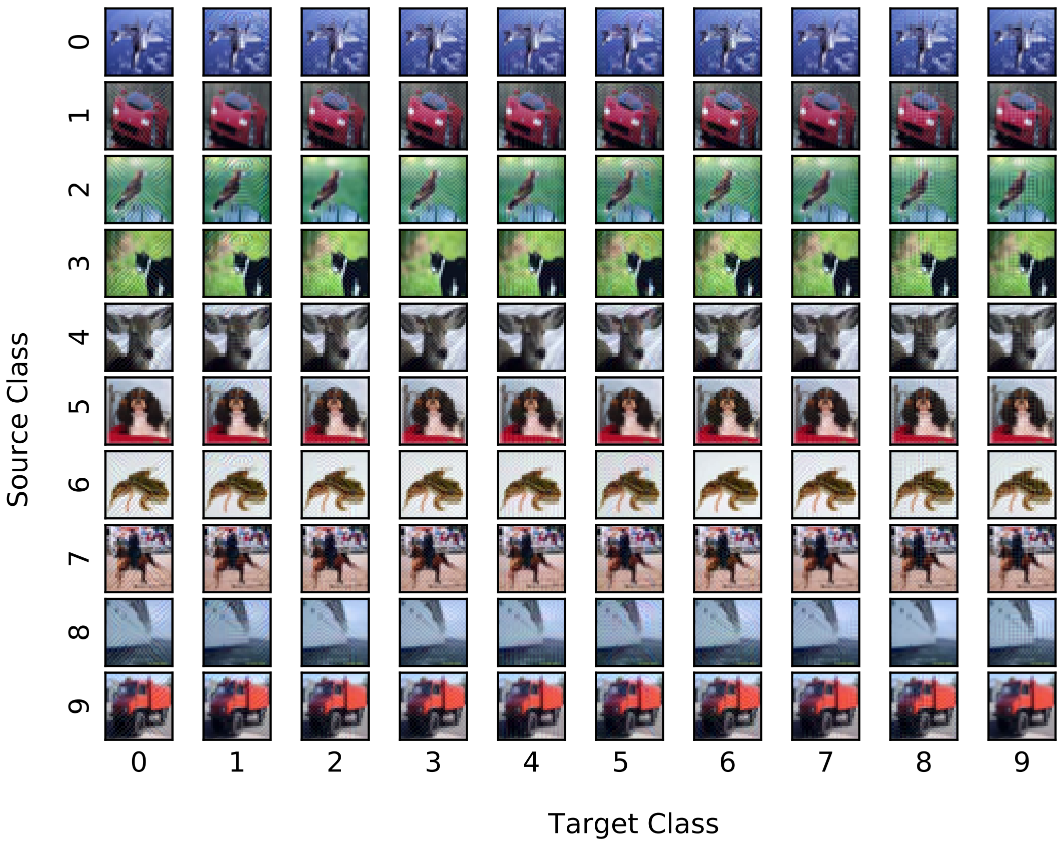


Figure 7.6: Our ℓ_∞ attack against a DenseNet target model on the CIFAR-10 dataset, for every source/target pair. Displayed images were selected at random.

7.7 Targeted attacks

We follow the same experimental set-up as in section 7.4, however now the attacker chooses a class, c, they would like the target model to classify an adversarial example as, and success is calculated as the probability that an adversarial example is classified as c. Figure 7.4 shows, for each class in CIFAR-10, the error rate of the target model as we allow larger perturbations. For nearly every class, attacks on ResNet-101 are most successful, while attacks on VGG-19 are least successful.

This is in agreement with our findings in a non-targeted attack setting (cf. table 7.2). Despite VGG-19 being the most difficult target model to attack, it is the most well calibrated; the error rate on the training set is nearly identical to the error rate on the validation set for all classes, while there are small deviations between these two scores for ResNet-101 and DenseNet.

By looking only at results on VGG-19, one may infer that the choice of target class heavily influences the error rate (e.g. crafting UAP's for the dog and ship classes is more difficult than others). However, this is not replicated with ResNet-101 or DenseNet. We do not observe any dependencies between attack success and the target class; the attack success at different perturbation rates is similar for all classes. Figure 7.6 shows this attack applied to a DenseNet target model for the CIFAR-10 dataset for all source/target class pairs. Nearly all attacks are indistinguishable from the source image. Similar results are found in fig. D.3 and fig. D.4 for VGG-19 and ResNet-101 target models, respectively.

Interestingly, all targeted attacks follow a sigmoidal curve shape. Empirically, we found that for all three target models, there existed images that were *weakly classified* correctly (there was almost no difference between the largest probability score and probability score at the target class) and *strongly classified* correctly (there was three to four orders of magnitude difference between the probability score at the largest class and the probability score at the target class). At the beginning of training, the UAN discovers a perturbation that causes misclassifications when applied to the weakly classified images, but takes longer to find adversarial perturbations for the majority of images, resulting in a long tail at the beginning of training. With a similar effect taking place at the end of training to find adversarial perturbations for strongly classified images.

For the ImageNet dataset, we selected three classes at random and performed a targeted attack. Error rates and selected samples are given in figs. D.5 to D.7. We observed that the generated UAPs resembled the structure of the target class. For example, a golf ball pattern can be clearly seen in perturbations in fig. D.5.

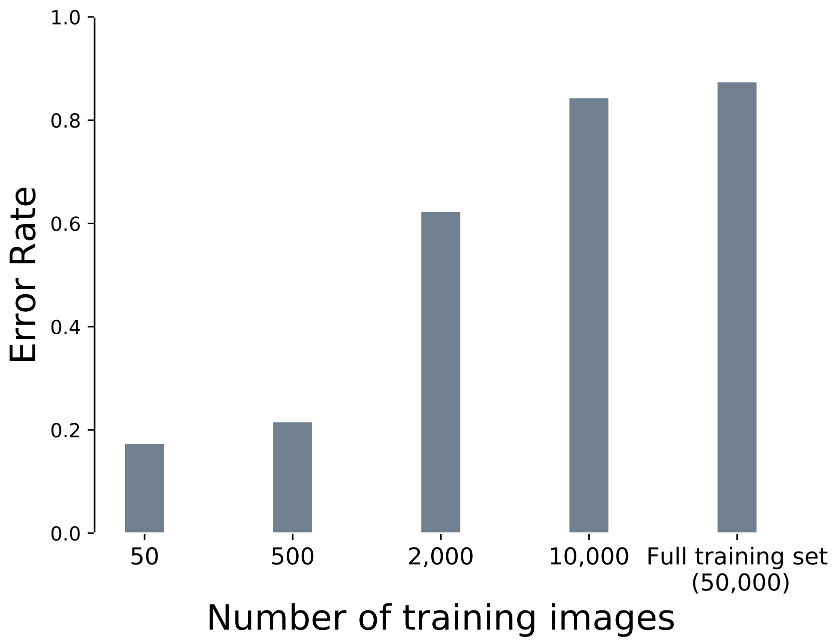


Figure 7.7: Non-targeted ℓ_∞ attack against ResNet-101 on the CIFAR-10 dataset. We vary the number of samples the UAN is trained on, and report results on the validation set.

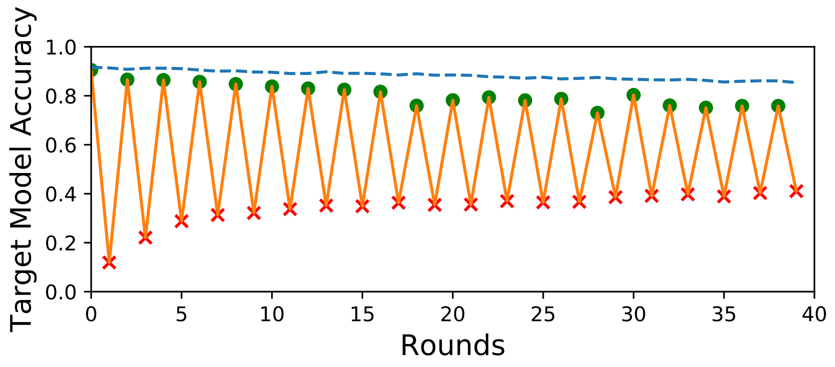


Figure 7.8: A cat-and-mouse game of non-targeted ℓ_∞ attacks and adversarial training for a VGG-19 target model on CIFAR-10. The upper green points are the target model accuracies on adversarial images after adversarial training, the lower red crosses are the target model accuracies on adversarial images after the attack. The dotted line is target model accuracy on source images.

7.8 Importance of training set size

So far, we have assumed the attacker shares full access to any images that were used to train the target model. However in practice, this may not be the case - an attacker may only have access to the type or a subsample of the training data. We therefore evaluate our non-targeted ℓ_∞ attack under stronger assumptions of attacker access to training data.

Figure 7.7 shows the error rate caused by a UAN trained on subsets of the CIFAR-10 training set. As expected, training on more data samples improves the success of the attack; perturbations from a UAN trained on only 50 images (five from each class) fools 17.1% of validation set images in ResNet-101. The attack is successful when applied to nearly a fifth of images while only learning from 0.1% of the training set. The attack succeeds in 80.2% of cases when trained on 20% of the training set - in other words, there is virtually no difference in test accuracy when training on between 80-100% of the training set.

We find no significant difference in error rates between a UAN that has been trained on many data samples and few data samples. The amount of data samples provided to the UAN does not significantly impact its ability to learn to craft adversarial perturbations, all that must be known is the structure of the dataset on which the target model was trained. We note that this is in agreement with Papernot et al. [191] findings on the number of source images required to launch attacks on black-box models.

In addition to measuring attacker success for different training set sizes, we experimented with different batch sizes, ranging from 16 to 128, for the CIFAR-10 dataset. However, we did not observe any significant deviations in the error rate.

7.9 Attacking adversarial training

Adversarial training [83, 136] modifies the training of a model in order to make it more robust to adversarial examples. During training, the loss function $L(\theta,x,y)$ is replaced by $\alpha \cdot L(\theta,x,y) + (1-\alpha) \cdot L(\theta,x+\delta',y)$. By augmenting the original data to include adversarial counterparts, the model learns to classify adversarial examples correctly. Non-generative attacks have shown to be successful against adversarially trained models, however, recent work by Metzen [168] suggested that this may not be the case for UAPs. In Metzen [168], adversarial training is successfully applied to a CIFAR-10 classifier, effectively eliminating the adversarial effect of UAPs. In our work, we verified that this is case; adversarial training eliminates

UAP success. However, we find that adversarially trained models are still vulnerable to UAN trained against the defended model. Similarly to Hamm and Mehra [90], we play a cat-and-mouse game where (1) a UAN is trained against a target model, and (2) the target model is retrained with adversarial examples crafted from (1) (denoted ADV TM). This generates a sequence: UAN1 $\rightarrow$ ADV TM1 $\rightarrow$ UAN2 $\rightarrow$ ADV TM2 $\rightarrow$ UAN3 $\rightarrow$ .... We let this game play out for many rounds, and claim that if adversarial training is a defence against UAPs, over many rounds the classification error on adversarial examples should tend to zero.

Figure 7.8 shows such a cat-and-mouse game over 20 rounds of (1) and 20 round of (2). An adversarially trained target model is able to classify nearly all adversarial examples correctly, at any given round. However, attacks against adversarially retrained models are only somewhat mitigated; there is a 25% reduction is attack success between the first and final round. After this, the cycle reaches an equilibrium, with no improvement in successive attacks or defended models.

7.10 Summary

We presented a first-of-its-kind universal adversarial example attack that uses machine learning at the heart of its construction. We comprehensively evaluated the attack under many different settings, showing that it produces quality adversarial examples capable of fooling a target model in both targeted and non-targeted attacks. The attack transfers to many different target models, and improves on other state-of-the-art universal adversarial perturbation construction methods.

Chapter 8

Randomised smoothing: A provable defense against adversarial examples

As seen in chapter 7, image classification is vulnerable to *adversarial examples*. Given an image classifier $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ such that the decision function $F = \arg\max_i f_i$ classifies an input, x, correctly as $F(x) = y$, an adversarial example is an input, $x + \delta$, such that $F(x + \delta) \neq y$ where x and $x + \delta$ are assigned the same label by an oracle classifier, $\mathscr{O}$, which is usually taken to be the human vision system. To preserve oracle classification, it is common to minimise the perturbation, δ, with respect to an ℓ_p norm. Constructing a perturbation such that $||\delta||_p \ll ||x||_p$, will result in an input such that $||x + \delta||_p \approx ||x||_p$. With high likelihood x and $x + \delta$ will be visually similar and $\mathscr{O}$ will classify both correctly.

The vulnerability to adversarial examples requires a suitable defence. Many empirical defences have been proposed and subsequently shown to be broken, implying more theoretically grounded techniques to measure robustness are required [15, 33, 37, 71, 239]. Recently, methods from verification literature have been used to provide guarantees of an input's robustness to adversarial perturbations. These methods seek the minimum or a lower bound on the amount of noise required to cause a misclassification. These verification methods are most often tailored to a single ℓ_p norm for which the defence guarantees robustness. A number of defences *certify* a neural network is robust to adversarial examples by propagating upper and lower input bounds throughout the network or by bounding the Lipschitz value of the network [27, 61, 78, 84, 152, 170, 238, 265].

Recently, *randomised smoothing* has been proposed to certify image classifiers to ℓ_0, ℓ_1, and ℓ_2 perturbations [49, 142, 144, 146]. By constructing a classifier that outputs a label based on a majority vote under repeated addition of Laplacian or Gaussian noise, Lecuyer et al. [142] found lower bounds to the amount of noise required for misclassification of an input in the ℓ_1 or ℓ_2 norm, respectively. Following this, Li et al. [146] and Cohen et al. [49] provided improved bounds in the ℓ_2 norm. As explained by Cohen et al. [49], randomised smoothing has attractive advantages over other certification methods: it is scalable to large classifiers and makes no assumption about the architecture. In this chapter, we extend the general framework for randomised smoothing as proposed by Li et al. [146]. Firstly, we study how the choice of divergence between inputs smoothed with noise affects the final certificate, and secondly, we study how the choice of smoothing measure itself can lead to guarantees for differing threat models. Concretely, we show how the choice of smoothing measure allows us to extend randomised smoothing to any ℓ_p norm ($p \in \mathbb{N}_{>0}$), showing we can certify inputs with non-vacuous bounds over a range of ℓ_p norms with small p values. We then show that randomised smoothing fails to certify meaningfully large radii around inputs as p increases.

8.1 Background on certified defenses

In this section, we discuss related work on certified defences to adversarial examples, introduce extensions to randomised smoothing approaches to certified defences, and provide a method to compute a certified robust area around an input under *any* ℓ_p norm attack, where $p \in \mathbb{N}_{>0}$.

The vulnerability of empirical defences to adversarial examples has driven the need for formal guarantees of robustness. We define *certified robustness* as a guarantee that the decision of a classifier is preserved within an ε-ball around an input, and we refer to size of this ε-ball as the *certified radius*. Formal methods can be separated into *complete* and *incomplete* methods. Complete methods such as Satisfiability Modulo Theory (SMT) [36, 69, 125] or Mixed-Integer Programming

Table 8.1: ℓ_2 certified radius when using different divergences.

Distance	$d(Q,P) \geq$ (when $\arg\max_i q_i \neq \arg\max_i p_i$)	$d(\mathcal{N}(x,\sigma^2),\mathcal{N}(x',\sigma^2))$	Certified radius (for $\lVert x-x' \rVert_2 < \varepsilon$)
$d_{KL}(Q,P) = \sum_{i=1}^{k} q_i \log \frac{q_i}{p_i}$	$-\log(2\sqrt{p_1 p_2} + 1 - p_1 - p_2)$	$\frac{1}{\sigma^2}\lVert x-x' \rVert_2^2$	$\sqrt{-\sigma^2 \log(2\sqrt{p_1 p_2} + 1 - p_1 - p_2)}$
$d_{H^2}(Q,P) = \frac{1}{2}\sum_{i=1}^{k}(\sqrt{q_i} - \sqrt{p_i})^2$	$1 - \sqrt{1 - \frac{(\sqrt{p_1}-\sqrt{p_2})^2}{2}}$	$1 - e^{-\frac{\lVert x-x' \rVert_2^2}{8\sigma^2}}$	$\sqrt{-8\sigma^2 \log(\sqrt{1 - \frac{(\sqrt{p_1}-\sqrt{p_2})^2}{2}})}$
$d_{\chi^2}(Q,P) = \sum_{i=1}^{k} \frac{(q_i-p_i)^2}{p_i}$	$\frac{(p_1-p_2)^2}{(p_1+p_2)-(p_1-p_2)^2}$	$e^{\frac{\lVert x-x' \rVert_2^2}{\sigma^2}} - 1$	$\sqrt{\sigma^2 \log(\frac{p_1+p_2}{(p_1+p_2)-(p_1-p_2)^2})}$
$d_B(Q,P) = -\log(\sum_{i=1}^{k}\sqrt{q_i p_i})$	$-\log\left(\frac{(\sqrt{p_1}+\sqrt{p_2})^2+2(1-p_1-p_2)}{\sqrt{2(2\sqrt{p_1 p_2}+2-p_1-p_2)}}\right)$	$\frac{1}{8\sigma^2}\lVert x-x' \rVert_2^2$	$\sqrt{-8\sigma^2 \log\left(\frac{(\sqrt{p_1}+\sqrt{p_2})^2+2(1-p_1-p_2)}{\sqrt{2(2\sqrt{p_1 p_2}+2-p_1-p_2)}}\right)}$
$d_{TV}(Q,P) = \frac{1}{2}\sum_{i=1}^{k}\lvert q_i - p_i \rvert$	$\frac{\lvert p_1-p_2 \rvert}{2}$	$2\Phi(\frac{\lVert x-x' \rVert_2}{2\sigma}) - 1$	$2\sigma\Phi^{-1}(\frac{\lvert p_1-p_2 \rvert}{2} + \frac{1}{2})$

(MIP) [30, 44, 251] provide exact robustness bounds but are expensive to implement. Incomplete methods solve a convex relaxation of the verification problem. The bounds given by incomplete methods can be loose but are quicker to find than exact bounds [27, 61, 78, 84, 152, 170, 265].

Lecuyer et al. [142] developed the certification technique, referred to as *randomised smoothing*, by noticing a connection between differential privacy [64] and robustness, and show that robustness can be proven under concentration measures of classification under noise. This work was expanded upon by Lee et al. [144], Li et al. [146], and Cohen et al. [49], who found improved robustness guarantees in the ℓ_0, ℓ_1, and ℓ_2 norms, respectively. Similarly to this work, Dvijotham et al. [62] developed a general framework for randomised smoothing that can handle arbitrary smoothing measures and so find robustness guarantees in any ℓ_p norm. In concurrent work, Blum et al. [25], Kumar et al. [135], and Yang et al. [258] also show that randomised smoothing may be unable to find robustness guarantees in the ℓ_∞ norm. Most related to this work are the findings of Kumar et al. [135], who also use a generalised Gaussian distribution for smoothing and show that the certified radius in an ℓ_p norm decreases as $\mathcal{O}(1/d^{\frac{1}{2}-\frac{1}{p}})$, where d is the dimensionality of the data.

8.2 Certification via randomised smoothing

Here, we expand on how robustness guarantees can be found through randomised smoothing.

Problem statement. Given an input $x \in \mathscr{X}$ such that $\arg\max_i f_i(x) = y$, find the maximum ε such that $\forall x' \in \mathscr{X}$, $d(x,x') < \varepsilon \implies \arg\max_i f_i(x') = y$, given a distance function $d : \mathscr{X} \times \mathscr{X} \to \mathbb{R}^+$.

This can be cast as an optimisation problem, given by

$$\begin{aligned} &\max_{x' \in \mathscr{X}} d(x,x') \\ &\text{subject to } \arg\max_i f_i(x') = y \end{aligned} \tag{8.1}$$

In general, solving the above formulation is difficult, however randomised smoothing, introduced by Lecuyer [142], can be used to solve a relaxed version of this problem. Namely, the aim is to solve

$$\begin{aligned} &\max_{x' \in \mathscr{X}} d(x+\theta, x'+\theta) \\ &\text{subject to } \mathbb{E}[\arg\max_i f_i(x'+\theta)] = y, \end{aligned} \tag{8.2}$$

where θ is a sample from a smoothing measure, μ, and d is now taken to be a suitable divergence or distance measure between random variables. For example, Li et al. [146] take μ to be the centred Gaussian, $\mathscr{N}(0,\sigma^2)$. Since Gaussians belong to the location-scale family of distributions, we can treat x and x' as constants and so, $x+\theta$ and $x'+\theta$ can be treated as random variables from distributions $\mathscr{N}(x,\sigma^2)$ and $\mathscr{N}(x',\sigma^2)$, respectively. We can use well known properties of divergences of Gaussians to represent $d(x+\theta,x'+\theta)$ in terms of the ℓ_2 norm difference of their means. Specifically, $d(x+\theta,x'+\theta)$ can be represented as a function of $\left\|x-x'\right\|_2$ and σ, for common divergences such as the Rényi and KL divergences. However, we must still solve the problem of ensuring $\mathbb{E}[\arg\max_i f_i(x'+\theta)] = y$. Given a chosen divergence, Li et al. [146] approach this problem by finding a lower bound between two multinomial distributions, P and Q, in terms of the two largest probabilities of P, when $\arg\max_i P_i \neq \arg\max_i Q_i$. This shows that any distribution, Q, for which P and Q agree on the index of the top probability, the divergence between P and Q

must be smaller than this lower bound. We denote this lower bound by $h(p_1, p_2)$, where p_1, p_2 represent the top two probabilities from P. Given this lower bound Li et al. [146], solve the following problem

$$\begin{aligned} &\max_{x' \in \mathscr{X}} d(f(x+\theta), f(x'+\theta)) \\ &\text{subject to } d(f(x+\theta), f(x'+\theta)) \leq h(p_1, p_2) \end{aligned} \tag{8.3}$$

This can be efficiently solved by finding an upper bound to the Lagrangian relaxed problem

$$\begin{aligned} \max_{\lambda \leq 0, x' \in \mathscr{X}} & d(f(x+\theta), f(x'+\theta)) \\ & + \lambda(h(p_1, p_2) - d(f(x+\theta), f(x'+\theta))) \end{aligned} \tag{8.4}$$

$$= \max_{\lambda \leq 0, x' \in \mathscr{X}} (1-\lambda) d(f(x+\theta), f(x'+\theta)) + \lambda h(p_1, p_2) \tag{8.5}$$

$$= \max_{\lambda \geq 0, x' \in \mathscr{X}} (1+\lambda) d(f(x+\theta), f(x'+\theta)) - \lambda h(p_1, p_2) \tag{8.6}$$

$$\leq \max_{\lambda \geq 0, x' \in \mathscr{X}} (1+\lambda) d(x+\theta, x'+\theta) - \lambda h(p_1, p_2) \tag{8.7}$$

$$= \max_{\lambda \geq 0, x' \in \mathscr{X}} (1+\lambda) g(\|x-x'\|_2, \sigma) - \lambda h(p_1, p_2), \tag{8.8}$$

where in eq. (8.7), we use the data processing inequality property of divergences, and in eq. (8.8), we use the fact that for many common divergences, we can represent the divergence between two Gaussians as a function of the ℓ_2 norm of their means and their standard deviation, which we denote by $g(\|x-x'\|_2, \sigma)$.

By choosing $d : \mathscr{X} \times \mathscr{X} \to \mathbb{R}^+$ to be the Rényi divergence, we recover the results of Li et al. [146] with

$$g(\|x-x'\|_2,\sigma) = \frac{\alpha\|x-x'\|_2^2}{2\sigma^2} \tag{8.9}$$

$$h(p_1,p_2) = -\log\left(1-p_1-p_2+2\left(\frac{1}{2}(p_1^{1-\alpha}+p_2^{1-\alpha})\right)^{\frac{1}{1-\alpha}}\right) \tag{8.10}$$

Thus, for any $x' \in \mathscr{X}$ with $\|x-x'\|_2 < \varepsilon$ we can guarantee the classifier, f, will not change it's decision for any ε smaller than

$$\begin{aligned}\max_{\lambda\geq 0}&\left(\sup_{\alpha>1}\left(-\frac{\lambda 2\sigma^2}{(1+\lambda)\alpha}\log\left(1-p_1-p_2+2\left(\frac{1}{2}(p_1^{1-\alpha}+p_2^{1-\alpha})\right)^{\frac{1}{1-\alpha}}\right)\right)\right)^{\frac{1}{2}}\\ &=\left(\sup_{\alpha>1}\left(-\frac{2\sigma^2}{\alpha}\log\left(1-p_1-p_2+2\left(\frac{1}{2}(p_1^{1-\alpha}+p_2^{1-\alpha})\right)^{\frac{1}{1-\alpha}}\right)\right)\right)^{\frac{1}{2}}\end{aligned} \tag{8.11}$$

Clearly, this framework for certifying inputs is general and extends to different choices of divergence. In the next section, we explore divergences beyond Rényi divergence and show this choice affects the certified radius, given a Gaussian smoothing measure.

8.3 Certification guarantees against ℓ_2 perturbations for common divergences

Li et al. [146] show that, given two distributions, P and Q, with different indexes for the top probability, a lower bound of the Rényi divergence (denoted by d_α) is given by eq. (8.10). We extend this line of reasoning to find lower bounds for the KL divergence (d_{KL}), Hellinger distance (d_{H^2}), (Neyman) chi-squared distance (d_{χ^2}), Bhattacharyya distance (d_B), and total variation distance (d_{TV}). Proofs of these lower bounds are given in appendix E.1. To find a certified radius of a classifier's decision around an input, we find the distances between Gaussian measures with respect to each of these divergences. These are both represented in table 8.1

along with the certification guarantee in the ℓ_2 norm. We visualise the trade-off in certified radius around an input in fig. 8.1 for a hypothetical binary classification task as a function of the classifier's top output probability, p_1. As well as including the certified radii derived from the aforementioned divergences, we include the certified radii for the ℓ_2 norm found by Lecuyer et al. [142] and Cohen et al. [49] approaches. Lecuyer et al. [142] find a certified radius against ℓ_2 perturbations given by

$$\sup_{0<\beta\leq\min(1,\frac{1}{2}\log\frac{p_1}{p_2})} \frac{\sigma\beta}{\sqrt{2\log\left(\frac{1.25(1+\exp(\beta))}{p_1-\exp(2\beta)p_2}\right)}},$$

while Cohen et al. [49] give a tight robustness guarantee for ℓ_2 perturbations of the form

$$\frac{\sigma}{2}\left(\Phi^{-1}(p_1)-\Phi^{-1}(p_2)\right).$$

Clearly, all choices of distance metrics dominate the certificates found using the Lecuyer et al. [142] method, and for values of p_1 close to $1/2$, d_{TV} is approximately equal to the tight [49] guarantee. However, the certified radius found using d_{TV} is linear with respect to the top predicted probability, and so becomes a weaker guarantee for larger probabilities. Robustness guarantees provided by Rényi and chi-squared divergences are approximately equal; a finer-grained visualisation of the difference between these two divergences is given in appendix E.2.

We formalise the trade-offs between different choices of divergences with the following proposition.

Proposition 2. *Let $\varepsilon_{d_{KL}}, \varepsilon_{d_{\chi^2}}, \varepsilon_{d_{H^2}}, \varepsilon_{d_B}, \varepsilon_{d_\alpha}$, and $\varepsilon_{\text{Lecuyer et al. [142]}}$, denote the certificates found using $d_{KL}, d_{\chi^2}, d_{H^2}, d_B, d_\alpha$, and the Lecuyer et al. [142] approach, respectively. Then, the following holds*

1. $\forall p_1 \in (\frac{1}{2},1)$, $\varepsilon_{d_\alpha} > \varepsilon_{d_{\chi^2}}$.

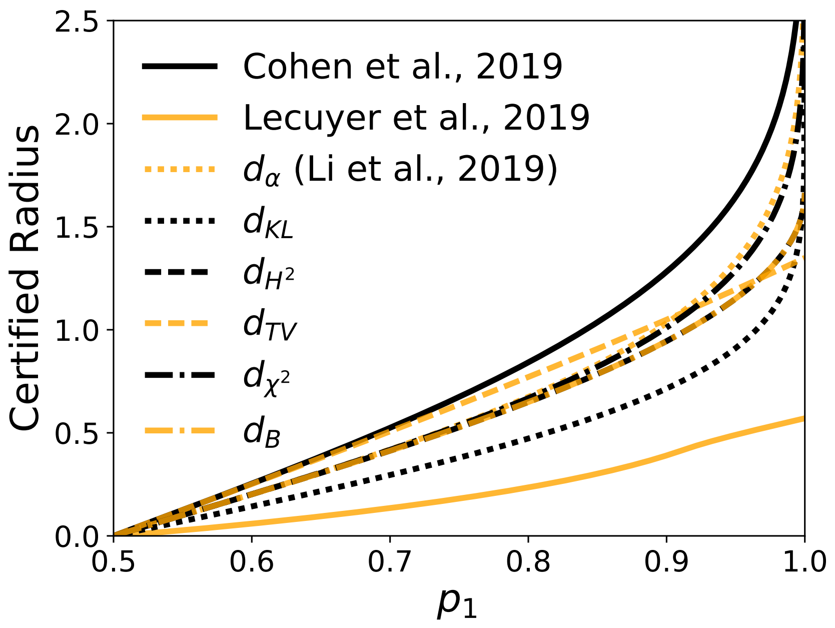


Figure 8.1: Comparison of the certified radius against perturbations targeting the ℓ_2 norm, for different divergences, as a function of the top predicted probability, p_1, with $\sigma = 1$.

2. $\forall p_1 \in (\frac{1}{2}, 1)$, $\varepsilon_{d_{\chi^2}} > \varepsilon_{d_{KL}}$.

3. $\forall p_1 \in (\frac{1}{2}, 1)$, $\varepsilon_{d_{\chi^2}} > \varepsilon_{d_{H^2}}$.

4. $\forall p_1 \in [\frac{1}{2}, 1]$, $\varepsilon_{d_B} = \varepsilon_{d_{H^2}}$.

5. $\forall p_1 \in (\frac{1}{2}, 0.998)$, $\varepsilon_{d_{H^2}} > \varepsilon_{d_{KL}}$.

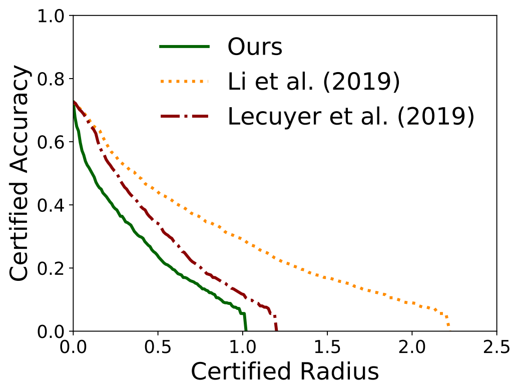
6. $\forall p_1 \in (\frac{1}{2}, 1)$, $\varepsilon_{d_{KL}} > \varepsilon_{\text{Lecuyer et al. [142]}}$.

Proof. See appendix E.3. □

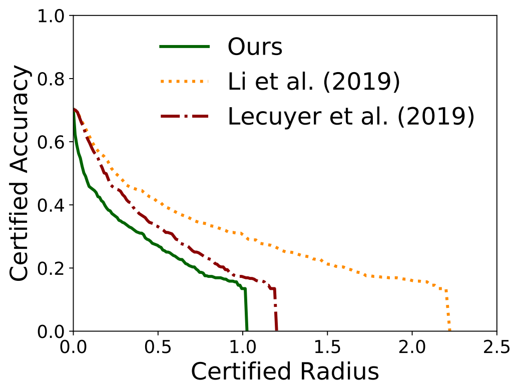
Proposition 2 defines a strict hierarchy, and so informs us of the best divergence one can use to certify an input against ℓ_2 perturbations using the Li et al. [146] approach.

8.4 Certification guarantees beyond ℓ_2 based perturbations

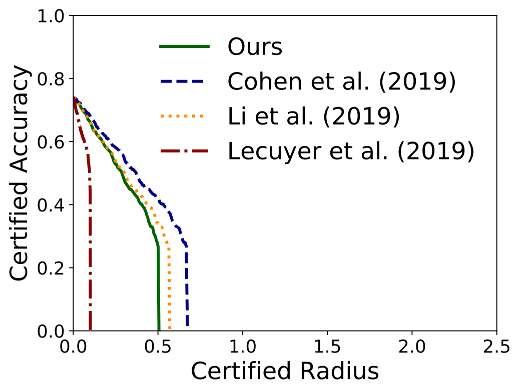
The Gaussian distribution is a natural choice for the smoothing measure because it naturally leads to robustness guarantees in the ℓ_2 norm. However, it is also a conve-



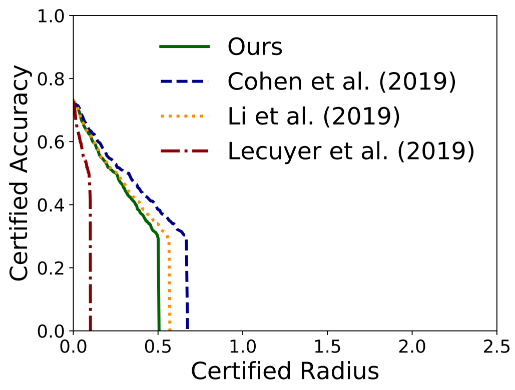
(a) CIFAR-10, ℓ_1



(b) ImageNet, ℓ_1



(c) CIFAR-10, ℓ_2



(d) ImageNet, ℓ_2

Figure 8.2: Certified accuracy against perturbations targeting the ℓ_1 and ℓ_2 norms. Given as a function of the certified radius, the radius around which an input is robust.

nient choice of smoothing measure because it is a member of the location-scale family of distributions. This means that, fixing $x \in \mathscr{X}$, sampling from $x + \mathscr{N}(0, \sigma^2)$ is equivalent to sampling from $\mathscr{N}(x, \sigma^2)$. Importantly, addition of a constant, x, does not change the family of the smoothing measure, and so we can use well known formula for the distances between two Gaussian distributions to derive robustness guarantees. Unfortunately, not all distributions belong to the location-scale family, and so, in our formulation, we are not free to choose any distribution for smoothing. Another convenient choice of a location-scale distribution is the generalised Gaussian distribution [175], denoted $\mathscr{G}\mathscr{N}(\mu, \sigma, s)$, whose density function is given by

$$p(x) = \frac{s}{2\sigma\Gamma(\frac{1}{s})} e^{-|\frac{x-\mu}{\sigma}|^s} \tag{8.12}$$

where μ is the mean, σ denotes a scaling factor and s denotes a shaping factor. The Laplacian distribution is recovered when $s = 1$, the Gaussian $\mathcal{N}(\mu, \frac{\sigma^2}{2})$ when $s = 2$, and the uniform distribution on $(\mu - \sigma, \mu + \sigma)$ as $s \to \infty$. We will show that by using this smoothing measure we can find robustness guarantees to ℓ_p perturbations, where $p \in \mathbb{N}_{>0}$.

We show in appendix E.4 that given inputs x and x' the Kullback–Leibler (KL) divergence of $\mathcal{GN}(x, \sigma, s)$ and $\mathcal{GN}(x', \sigma, s)$ (when s takes positive integer values) is given by

$$\sum_{k=1}^{s} \binom{s}{k} \frac{(1+(-1)^{s-k})\Gamma(\frac{s-k+1}{s})\left\|x-x'\right\|_k^k}{2\sigma^k\Gamma(\frac{1}{s})} \tag{8.13}$$

We also show in appendix E.1 that the KL divergence of two multinomial distributions P and Q (that disagree on the index of the top probability) is lower bounded by

$$d_{KL}(Q, P) \geq -\log(2\sqrt{p_1 p_2} + 1 - p_1 - p_2) \tag{8.14}$$

Then we use the data processing inequality to prove robustness up to $\left\|x - x'\right\|_p < \varepsilon$ if the following holds

$$d_{KL}(f(x + \mathcal{GN}(0, \sigma, p)), f(x' + \mathcal{GN}(0, \sigma, p))) \tag{8.15}$$

$$\leq d_{KL}(x + \mathcal{GN}(0, \sigma, p), x' + \mathcal{GN}(0, \sigma, p)) \tag{8.16}$$

$$\leq \frac{\varepsilon^p}{\sigma^p} + \sum_{k=1}^{p-1} \binom{p}{k} \frac{(1+(-1)^{p-k})\Gamma(\frac{p-k+1}{p})\left\|x-x'\right\|_k^k}{2\sigma^k\Gamma(\frac{1}{p})} \tag{8.17}$$

$$\leq -\log(2\sqrt{p_1 p_2} + 1 - p_1 - p_2) \tag{8.18}$$

Table 8.2 gives examples of the KL-divergence of the generalised Gaussian distribution for small ℓ_p norms. For ℓ_p norms with $p = 1$ or $p = 2$, the upper bound

Table 8.2: Examples of the KL divergence between $\mathcal{GN}(\mu_1,\sigma,s)$ and $\mathcal{GN}(\mu_2,\sigma,s)$ for small s.

s	ℓ_s	$d_{KL}(p_1,p_2)$
1	ℓ_1	$\frac{1}{\sigma}\|\mu_1-\mu_2\|_1$
2	ℓ_2	$\frac{1}{\sigma^2}\|\mu_1-\mu_2\|_2^2$
3	ℓ_3	$\frac{1}{\sigma^3}\|\mu_1-\mu_2\|_3^3+\frac{3}{\sigma\Gamma(\frac{1}{3})}\|\mu_1-\mu_2\|_1$
4	ℓ_4	$\frac{1}{\sigma^4}\|\mu_1-\mu_2\|_4^4+\frac{6\Gamma(\frac{3}{4})}{\sigma^2\Gamma(\frac{1}{4})}\|\mu_1-\mu_2\|_2^2$

to which an input is certifiably robust is given by

$$(-\sigma^p\log(2\sqrt{p_1p_2}+1-p_1-p_2))^{\frac{1}{p}} \tag{8.19}$$

For ℓ_p norms with $p>2, p\in\mathbb{N}$, the upper bound to which an input is certifiably robust is given by ε satisfying

$$\begin{aligned}\frac{\varepsilon^p}{\sigma^p}+\sum_{k=1}^{p-1}\binom{p}{k}\frac{(1+(-1)^{p-k})\Gamma(\frac{p-k+1}{p})d^{1-\frac{k}{p}}\varepsilon^k}{2\sigma^k\Gamma(\frac{1}{p})}\\ \leq -\log(2\sqrt{p_1p_2}+1-p_1-p_2)\end{aligned} \tag{8.20}$$

The bound given by eq. (8.20) is found by noting that $\|x-x'\|_k \leq d^{\frac{1}{k}-\frac{1}{p}}\|x-x'\|_p$, where d is the dimensionality of the data. We can improve upon this naive bound to prove robustness for all norms smaller than p in parallel. Without loss of generality, assume p is even [1], then we can prove robustness for every $0<k\leq p$, where k is even, up to $\|x-x'\|_k<\varepsilon_k$ by solving the constrained problem

[1]A similar statement holds when p is not even.

$$\max \quad \varepsilon_2, \varepsilon_4, ..., \varepsilon_p \tag{8.21}$$

subject to

$$\sum_{k=1}^{p} \binom{p}{k} \frac{(1+(-1)^{p-k})\Gamma(\frac{p-k+1}{p})\varepsilon_k^k}{2\sigma^k\Gamma(\frac{1}{p})} \leq -\log(2\sqrt{p_1 p_2}+1-p_1-p_2) \tag{8.22}$$

$$\varepsilon_{i+2} \leq \varepsilon_i \leq d^{\frac{1}{i}-\frac{1}{i+2}}\varepsilon_{i+2} \tag{8.23}$$

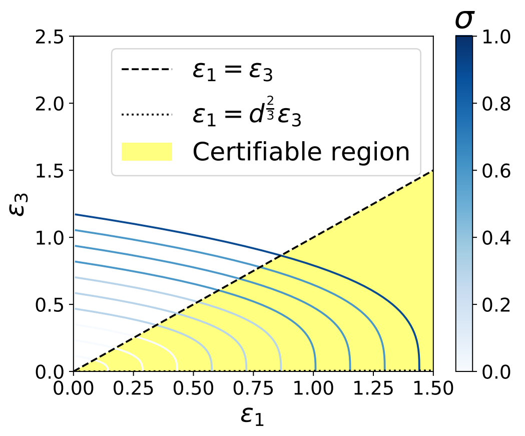
$$\varepsilon_i > 0, \quad 2 \leq i \leq p-2, \quad i \equiv 0 \pmod 2 \tag{8.24}$$

Note that the certified radius of robustness around an input is probabilistic because we can only estimate p_1 and p_2, however, we can bound the probability of error to be arbitrarily small. In practice we follow the methods in [49, 142, 146] for estimating p_1 and p_2. Prediction error is bounded by collecting n samples of $f(x+\theta)$, where θ is sampled from a generalised Gaussian distribution, and using the Clopper-Pearson Bernoulli confidence interval to obtain a lower bound estimate of p_1 and an upper bound estimate of p_2, that holds with probability $1-\gamma$ over the n samples, where $\gamma \ll 1$. Alternatively, we can use the Hoeffding inequality which gives a lower bound of prediction error of $1-ce^{-2n\varepsilon^2}$, where c is the number of classes $|P|$, n is the number of samples and ε is the perturbation size. Clearly the error becomes arbitrarily small as we increase the number of samples.

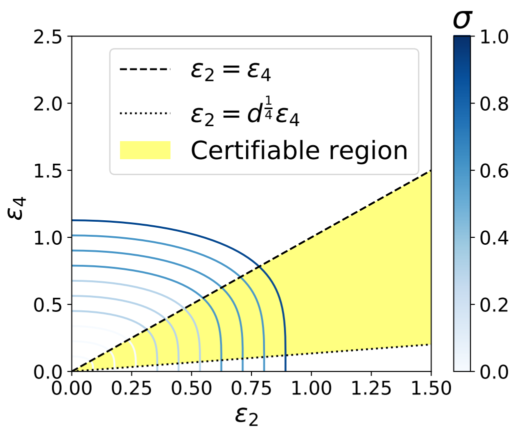
8.5 Discussion & experiments

We experimentally validated the certification procedure on the CIFAR-10 [133] and ImageNet [55, 210] datasets. The base classifier is ResNet-50 on ImageNet and ResNet-110 on CIFAR-10 [104]. Given an input x and a classifier f the certification procedure is as follows:

1. Collect n_0 Monte Carlo samples of $f(x+\theta_j)$ to estimate the true class y, where $\theta_j \sim \mathscr{G}\mathscr{N}(0,\sigma,s)$ and $j \in [1,...,n_0]$, with confidence $> 1-\gamma_0$.
2. Use n_1 Monte Carlo samples to estimate, $\hat{p_1}$, a lower bound of the probability



(a) Certified radius trade-off between ε_3 (ℓ_3 norm) and ε_1 (ℓ_1 norm).



(b) Certified radius trade-off between ε_4 (ℓ_4 norm) and ε_2 (ℓ_2 norm).

Figure 8.3: Trade-off in adversarial robustness between different norms, as we vary the noise scale, σ. We plot for a data dimensionality, d, equal to $3 \times 32 \times 32$ (the dimension for CIFAR-10 inputs), and mark the region which gives valid certificates, assuming $\hat{p}_1 = 0.99$ and $\hat{p}_2 = 1 - \hat{p}_1$.

of the most-likely class with confidence $> 1 - \gamma_1$. We follow Cohen et al. [49] for estimating $\hat{p}_2$, an upper bound of the probability of the second most-likely class, who noticed nearly all probability mass on other classes is placed on the second most-likely class and so use $\hat{p}_2 = 1 - \hat{p}_1$.

3. Use $\hat{p}_1$, $\hat{p}_2$ and eq. (8.19) or eq. (8.20) to find a certified radius around x.

For all experiments we use $n_0 = 100, n_1 = 100{,}000, \gamma_{\{0,1\}} = 0.001, \sigma = 0.25$ and certify 400 test set examples for both CIFAR-10 and ImageNet datasets [2]. Theoretically, this procedure can certify any classifier, however in practice, image classifiers are not stable under noise and so we found it necessary to train classifiers with generalised Gaussian noise (using the same scale and shape parameters as is used during certification). Note that this has the same complexity as standard data augmentation during training and is less expensive than the Madry et al. [161] defence.

[2]We perform experiments measuring the effect that various σ have on the certified radius in appendix E.5.

8.5.1 Comparison to related work

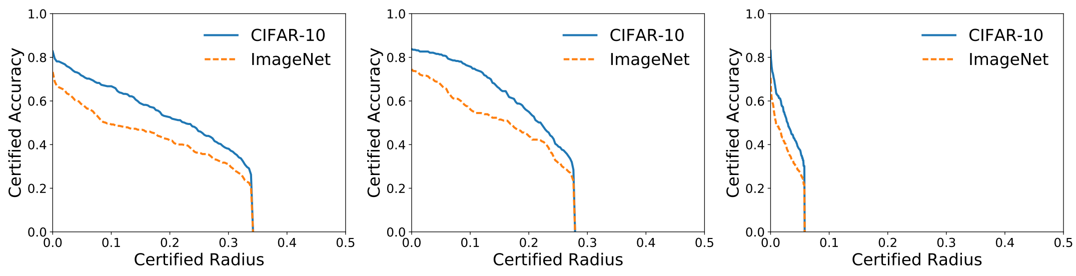
For both CIFAR-10 and ImageNet we certify inputs against perturbations in ℓ_1 and ℓ_2 norms and compare against [49, 142, 146]. Figure 8.2 shows certified accuracy as a function of the certified radius. In general, the largest certified regions come against perturbations targeting the ℓ_1 norm. In appendix E.6, we show qualitative examples of inputs smoothed with generalised Gaussian noise and the corresponding robustness guarantees in the ℓ_1, ℓ_2, and ℓ_3 norms.

While the primary boon of our certification procedure is its ability to certify inputs to adversarial perturbations beyond ℓ_1 and ℓ_2 norms, the method is not substantially weaker than related work in either norm. In fig. 8.2a and fig. 8.2b, we compare with Lecuyer et al. [142] and Li et al. [146] for ℓ_1 norm certificates. Given estimates $\hat{p_1}$ and $\hat{p_2}$, Lecuyer et al. [142] find a certified radius against ℓ_1 perturbations given by $\frac{\sigma}{2}\log(\hat{p_1}/\hat{p_2})$, while Li et al. [146] find a certified radius against ℓ_1 perturbations given by $\sigma\log(1-\hat{p_1}+\hat{p_2})$. Li et al. [146] and Teng et al. [230] show that this robustness guarantee is tight for the ℓ_1 norm. Our ℓ_1 certificates are slightly weaker than Lecuyer et al. [142], and both are dominated by Li et al. [146] who obtain the tightest possible certificates.

In fig. 8.2c and fig. 8.2d, we compare with Lecuyer et al. [142], Li et al. [146], and Cohen et al. [49] for ℓ_2 norm certificates. Our ℓ_2 certificates strictly dominate Lecuyer et al. [142], and are approximately equivalent to Li et al. [146]. This equivalence is to be expected since our certificates are closely related to Li et al. [146] certificates, which are based on the Rényi divergence between two Gaussians, while ours are based on KL divergence. Clearly, we could improve upon this ℓ_2 guarantee if we used the chi-squared distance instead of KL divergence and a standard Gaussian smoothing measure, as proved by proposition 2. However, our aim is to show the general capacity of the generalised Gaussian as a smoothing measure for certification.

8.5.2 Robustness trade-offs between different ℓ_p norms.

As described by eq. (8.20), to obtain robustness guarantees in $\ell_{p>2}$ norms we must factor in required robustness guarantees in smaller ℓ_p norms. For example, to prove



(a) ℓ_3 **(b)** ℓ_4 **(c)** ℓ_5

Figure 8.4: Certified accuracy on 400 CIFAR-10 test set inputs and 400 ImageNet test set inputs against perturbations targeting the ℓ_3, ℓ_4, and ℓ_5 norms. Given as a function of the certified radius, the radius around which an input is robust. Inputs were smoothed under a generalised Gaussian distribution parameterised by $\mathscr{GN}(0, 0.25, p)$.

robustness up to $\left\|x - x'\right\|_3 < \varepsilon_3$ and $\left\|x - x'\right\|_1 < \varepsilon_1$ we find ε_1 and ε_3 satisfying

$$\begin{aligned} &\frac{1}{\sigma^3}\varepsilon_3^3 + \frac{3}{\sigma\Gamma(\frac{1}{3})}\varepsilon_1 \leq -\log(2\sqrt{\hat{p}_1\hat{p}_2} + 1 - \hat{p}_1 - \hat{p}_2) \\ &\wedge \\ &0 < \varepsilon_3 \leq \varepsilon_1 \leq d^{\frac{2}{3}}\varepsilon_3, \end{aligned} \tag{8.25}$$

and to prove robustness up to $\left\|x - x'\right\|_4 < \varepsilon_4$ and $\left\|x - x'\right\|_2 < \varepsilon_2$ we find ε_2 and ε_4 satisfying

$$\begin{aligned} &\frac{1}{\sigma^4}\varepsilon_4^4 + \frac{6\Gamma(\frac{3}{4})}{\sigma^2\Gamma(\frac{1}{4})}\varepsilon_2^2 \leq -\log(2\sqrt{\hat{p}_1\hat{p}_2} + 1 - \hat{p}_1 - \hat{p}_2) \\ &\wedge \\ &0 < \varepsilon_4 \leq \varepsilon_2 \leq d^{\frac{1}{4}}\varepsilon_4, \end{aligned} \tag{8.26}$$

We visualise this trade-off in fig. 8.3 for ℓ_3 and ℓ_4 norms. That is, the trade-off in certified robustness between those norms and certified robustness in ℓ_1 and ℓ_2, respectively. We visualise the trade-off as we vary the noise scale σ, assuming a robust classifier that classifies inputs correctly with $\hat{p}_1 = 0.99$ and $\hat{p}_2 = 0.01$. We can smoothly exchange robustness in one norm for robustness in another norm. For example, given $\sigma = 1$ and a CIFAR-10 input, we can reduce the guaranteed robustness in the ℓ_3 norm from an approximate certified radius of 0.86 to approximately 0, and increase the guaranteed robustness in the ℓ_1 norm from a certified radius of 0.86 to 1.44. In fig. 8.4, we show certified accuracy as a function of certified radius in the ℓ_3, ℓ_4, and ℓ_5 norms on the CIFAR-10 and ImageNet datasets. To find the maximum ε_3 we solve eq. (8.25) such that $\varepsilon_3 = \varepsilon_1$. Similarly for ε_4 we solve eq. (8.26) such that $\varepsilon_4 = \varepsilon_2$, and extend this line of reasoning to find $\varepsilon_5 = \varepsilon_3 = \varepsilon_1$ for the ℓ_5 norm. Clearly, we can find non-negligible certified radii in norms outside of ℓ_1 and ℓ_2.

8.5.3 Robustness guarantees as $\ell_{p\to\infty}$.

An immediate question arises when observing our certification procedure, can we find non-vacuous robustness guarantees for arbitrarily large ℓ_p norms, where p is even [3] [4]? Given eq. (8.22), note that $\binom{p}{k}(1+(-1)^{p-k})\Gamma(\frac{p-k+1}{p})/2\Gamma(\frac{1}{p}) \geq 1, \forall 1 \leq k \leq p$, where k is even, and as $p \to \infty$, $\exists k$ such that $\binom{p}{k}(1+(-1)^{p-k})\Gamma(\frac{p-k+1}{p})/2\Gamma(\frac{1}{p}) \to \infty$. We must therefore solve the problem given in eq. (8.21)-eq. (8.24), where eq. (8.22) is given by

$$\frac{c_2\varepsilon_2^2}{\sigma^2} + \frac{c_4\varepsilon_4^4}{\sigma^4} + \ldots + \frac{c_p\varepsilon_p^p}{\sigma^p} \leq -\log(2\sqrt{p_1p_2} + 1 - p_1 - p_2) \tag{8.27}$$

$$\text{where } c_k \in \mathbb{R}_{>1}, 1 \leq k \leq p, k \equiv 0 \pmod 2 \tag{8.28}$$

To satisfy eq. (8.23), we can find $\varepsilon_2, \varepsilon_4, \ldots, \varepsilon_p$ such that $\varepsilon_2 = \varepsilon_4 = \ldots = \varepsilon_p$; we refer to this value as ε, and eq. (8.27) becomes

$$\begin{aligned} &c_2(\frac{\varepsilon}{\sigma})^2 + c_4(\frac{\varepsilon}{\sigma})^4 + \ldots + c_p(\frac{\varepsilon}{\sigma})^p \\ &\leq -\log(2\sqrt{p_1p_2} + 1 - p_1 - p_2) \end{aligned} \tag{8.29}$$

$$\text{where } c_k \in \mathbb{R}_{>1}, 1 \leq k \leq p, k \equiv 0 \pmod 2 \tag{8.30}$$

[3]Equivalent results for this section can be found when p is not even.

[4]The subject of simultaneous robustness over every ℓ_p norm is expanded upon in appendix E.7.

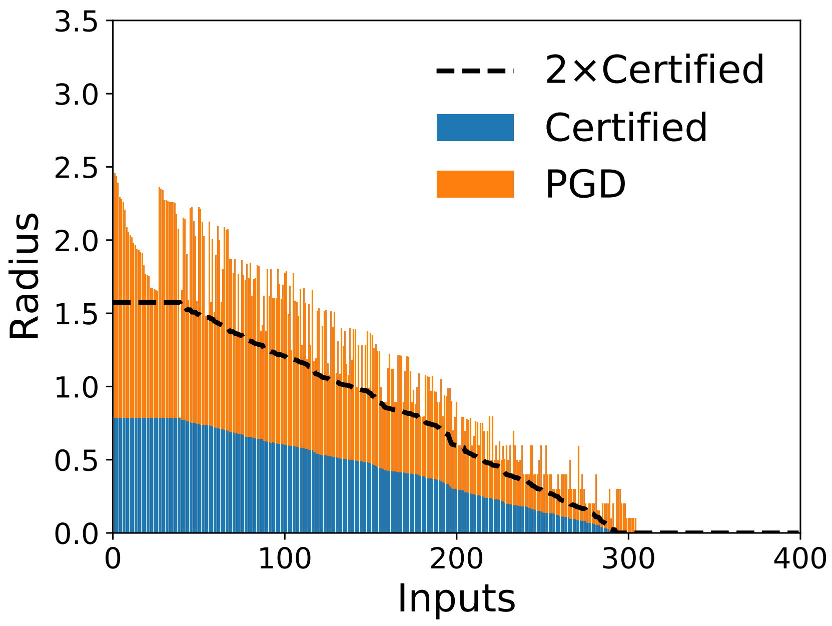


Figure 8.5: The certified radius and size of adversarial perturbations for 400 CIFAR-10 test inputs using a PGD attack optimising the ℓ_2 norm. As a guide to assess how close the certified radius is to adversarial perturbation size, we also display $2\times$ the certified radius of an input.

For a fixed p_1, p_2, σ, since $\forall k, c_k \geq 1$, and $\exists k$ such that $c_k \to \infty$ when $p \to \infty$, to satisfy the inequality in eq. (8.29), we must have $\varepsilon \to 0$. If we do not fix σ then we require $(\frac{\varepsilon}{\sigma})^k \to 0$ as $c_k \to \infty$, and so to certify a non-negligible radius, ε, we require $\sigma \to \infty$. However, as $\sigma \to \infty$, the randomised smoothing will cause the input to become too noisy for any classifier to achieve low prediction error.

Clearly, as p grows the largest possible certified radius becomes smaller, because our bound requires this robustness guarantee holds for every norm smaller than p. One may wonder if we can find an ℓ_p norm in which we can certify a non-vacuous radius that approximates the ℓ_∞ norm arbitrarily well. The difference in volume between a unit ball in the ℓ_p norm and ℓ_∞ norm is given by $\Gamma(1+1/p)^d/\Gamma(1+d/p)$, where d is the data dimensionality. Unfortunately, the error in the approximation is dependent on the data dimensionality. For example, for an ImageNet input where $d = 3 \times 224 \times 224$, if we require the ratio of volumes between an ℓ_p unit ball and ℓ_∞ unit ball to be larger than 0.99, we must take $p = 9 \times 3 \times 224 \times 224$.

8.5.4 How tight is the bound?

The difference between the certified area and the size of an adversarial perturbation gives a tightness estimate. If the certified radius is close to the size of an adversarial perturbation this implies the bound is close to optimal. To check how tight our bound is we ran the PGD attack [161] minimising perturbations in the ℓ_2 norm. Because the certification procedure requires the addition of generalised Gaussian noise to the input, the gradient is highly stochastic, leading to extremely slow convergence of the PGD attack. We circumvent this stochasticity by optimising using the Expectation Over Transformation [14] – we use 1000 Monte Carlo samples to estimate the gradient of an input during the attack. Figure 8.5 gives attack results on CIFAR-10 along with the certified radius of 400 inputs. We find adversarial examples with norms within $2-2.5\times$ the certified radius. Unfortunately, this does not inform us if our bound is loose or if the attack is sub-optimal. We leave a more rigorous investigation of assessing the tightness of our bound for future work.

Chapter 9

Conclusions

Information security problems that can be reduced to pattern recognition problems can benefit from machine learning. This dissertation presented results in network traffic analysis and steganography that demonstrated the efficacy of applying machine learning to these problems. We went on to show that, although information security problems can benefit from machine learning, in adversarial environments it is not sufficient to apply them blindly. Malicious actors can reduce security and privacy properties that a system claims to hold by introducing corrupted data both during the training and evaluation of the machine learning model.

Firstly, in chapter 3, we introduced website fingerprinting; the problem of predicting which website a client visits by collecting encrypted network traffic. We demonstrated that random forests [28] and k-NN [50] can be combined in order to trade-off false positives and false negatives. We additionally showed that this classifier was robust to website fingerprinting defences, that aim to introduce packet timing delays or dummy network traffic, in order to distort the patterns learnt by the attack models. Secondly, in chapter 4, we studied how machine learning can be applied to the task of information hiding. We considered three actors, the information hider, the information receiver, and the eavesdropping detector, where the hider and receiver participate in a co-operative game and concurrently participate in a zero-sum with the eavesdropper. We modelled each of these actors by neural networks, and showed that these networks can be trained through first-order optimisation techniques to reach parity with traditional steganographic algorithms.

In chapter 5, we showed that private information can leak in generative models. Our attacks exploited overfitting in generative models to expose the inputs that made up the training data. We then demonstrated how early stopping and private training mechanisms can be used to reduce the threat of private information leakage. Inference of training set membership is an attack that is established after training has completed. In chapter 5, we studied how in a multi-party setting, an attacker that can corrupt inputs during the training process can cause the model to learn adversarial properties that are unwanted and undetected by benign participants. By introducing the task of predicting the party who supplied an input, and modelling this as a zero-sum game between this task and the original prediction task, we showed that the threat of adversarial corruptions in training data is minimised, conditional on a minority of parties being adversarial.

In part III, we demonstrated vulnerabilities in the trustworthiness of model predictions. By introducing small perturbations, an attack can add these to seemingly "easy to classify" inputs and cause dramatic misclassifications. In chapter 7, we introduced the concept of a universal adversarial perturbation [173], adversarial perturbations that generalise across an entire data distribution, rather than a single input. This significantly reduces the effort of an attack, since the perturbation can be applied blindly to any input from a target distribution, and the attacker has high confidence the attack will succeed. We modelled the construction of universal adversarial perturbations as a co-operative game between a target network and network that learns to output adversarial perturbations. Through this design, we showed that a neural network can learn to output perturbations that cause misclassifications with high probability, can transfer across different models, and are somewhat resistant to standard adversarial examples defences [161]. Following this, in chapter 8 we discussed the limitations of experimental defences against adversarial examples, and introduced the concept of certified defences. We showed that by measuring invariance of prediction under noise addition, one can produce a certificate affirming no adversarial examples exists within an ε-ball around an input [49, 142]. Unfortunately, our research implies that such methods are sufficient to guarantee robust classification under only a small number of ℓ_p norms.

The rate of adoption of machine learning in security and privacy sensitive environments is accelerating. This dissertation should serve as a cautionary tale and a guide to the threats a practitioner can expect to face when launching machine learning in production. However, there are reasons to be optimistic, there is promising progress in private training mechanism research [8, 17, 171, 217], and against data poisoning attacks [214, 225]. Research into the causes of adversarial examples has also started to move beyond limited settings, such as measuring robustness against ℓ_p norm constrained attacks [70, 223]. However, it remains an open question if current deep learning based approaches to artificial intelligence can approach human levels of robustness to adversarial inputs, or if a new approach is required.

Bibliography

[1] Alexa — The Web Information Company, [Accessed August 2015]. URL `http://alexa.com`.

[2] `.gov.uk` verify, [Accessed January 2020]. URL `https://www.gov.uk/government/publications/introducing-govuk-verify/introducing-govuk-verify`.

[3] The Nielsen Company. Technical report, [Accessed July 2015]. `http://www.nielsen.com/us/en/insights/news/2010/led-by-facebook-twitter-global-time-spent-on-social-media-sites-up-82-year-over-year.html`.

[4] Tor Proposal 254, [Accessed November 2015]. URL `https://gitweb.torproject.org/torspec.git/tree/proposals/254-padding-negotiation.txt`.

[5] Martín Abadi and David G Andersen. Learning to protect communications with adversarial neural cryptography. In *arXiv preprint arXiv:1610.06918*, 2016.

[6] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. In *arXiv preprint arXiv:1603.04467*, 2016.

[7] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Is-

ard, et al. Tensorflow: A system for large-scale machine learning. In *USENIX OSDI*, 2016.

[8] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *ACM CCS*, 2016.

[9] Scott Alfeld, Xiaojin Zhu, and Paul Barford. Data poisoning attacks against autoregressive models. In *AAAI*, 2016.

[10] Joshua Allen, Bolin Ding, Janardhan Kulkarni, Harsha Nori, Olga Ohrimenko, and Sergey Yekhanin. An algorithmic framework for differentially private data analysis on trusted processors. In *Advances in Neural Information Processing Systems*, 2019.

[11] Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, et al. Privacy-preserving deep learning: Revisited and enhanced. In *ATIS*, 2017.

[12] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. In *arXiv 1701.07875*, 2017.

[13] Giuseppe Ateniese, Luigi V Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. In *International Journal of Security and Networks*, 2015.

[14] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *arXiv preprint arXiv:1707.07397*, 2017.

[15] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *arXiv preprint arXiv:1802.00420*, 2018.

[16] Michael Backes, Pascal Berrang, Mathias Humbert, and Praveen Manoharan. Membership privacy in microrna-based studies. In *ACM CCS*, 2016.

[17] Brett K Beaulieu-Jones, Zhiwei Steven Wu, Chris Williams, Ran Lee, Sanjeev P Bhavnani, James Brian Byrd, and Casey S Greene. Privacy-preserving generative deep neural networks support clinical data sharing. In *Circulation: Cardiovascular Quality and Outcomes*, 2019.

[18] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, 2013.

[19] David Berthelot, Tom Schumm, and Luke Metz. BEGAN: Boundary Equilibrium Generative Adversarial Networks. In *arXiv 1703.10717*, 2017.

[20] Sanjit Bhat, David Lu, Albert Kwon, and Srinivas Devadas. Var-cnn: A data-efficient website fingerprinting attack based on deep learning. In *PETS*, 2019.

[21] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *ICML*, 2012.

[22] Battista Biggio, Konrad Rieck, Davide Ariu, Christian Wressnegger, Igino Corona, Giorgio Giacinto, and Fabio Roli. Poisoning behavioral malware clustering. In *AISec*, 2014.

[23] George Dean Bissias, Marc Liberatore, David Jensen, and Brian Neil Levine. Privacy vulnerabilities in encrypted http streams. In *PETS*, 2006.

[24] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *ACM SOSP*, 2017.

[25] Avrim Blum, Travis Dick, Naren Manoj, and Hongyang Zhang. Random smoothing might be unable to certify ℓ_∞ robustness for high-dimensional images. In *arXiv preprint arXiv:2002.03517*, 2020.

[26] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy preserving machine learning. In *ACM CCS*, 2017.

[27] Akhilan Boopathy, Tsui-Wei Weng, Pin-Yu Chen, Sijia Liu, and Luca Daniel. Cnn-cert: An efficient framework for certifying robustness of convolutional neural networks. In *arXiv preprint arXiv:1811.12395*, 2018.

[28] Leo Breiman. Random forests. In *Machine learning*, 2001.

[29] Anna L Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. In *IEEE Communications Surveys & Tutorials*, 2016.

[30] Rudy R Bunel, Ilker Turkaslan, Philip Torr, Pushmeet Kohli, and Pawan K Mudigonda. A unified view of piecewise linear neural network verification. In *Advances in Neural Information Processing Systems*, 2018.

[31] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. Touching from a distance: Website fingerprinting attacks and defenses. In *ACM CCS*, 2012.

[32] Xiang Cai, Rishab Nithyanand, and Rob Johnson. CS-BuFLO: A Congestion Sensitive Website Fingerprinting Defense. In *WPES*, 2014.

[33] Nicholas Carlini. Is ami (attacks meet interpretability) robust to adversarial examples? In *arXiv preprint arXiv:1902.02322*, 2019.

[34] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *AISec*, 2017.

[35] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE S&P*, 2017.

[36] Nicholas Carlini, Guy Katz, Clark Barrett, and David L Dill. Provably minimally-distorted adversarial examples. In *arXiv preprint arXiv:1709.10207*, 2017.

[37] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, and Aleksander Madry. On evaluating adversarial robustness. In *arXiv preprint arXiv:1902.06705*, 2019.

[38] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security*, 2019.

[39] Licong Chen, Yun-Qing Shi, and Patchara Sutthiwan. Variable multi-dimensional co-occurrence for steganalysis. In *International Workshop on Digital Watermarking*, 2014.

[40] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *AISec*, 2017.

[41] Sen Chen, Minhui Xue, Lingling Fan, Shuang Hao, Lihua Xu, Haojin Zhu, and Bo Li. Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach. In *Computers & Security*, 2018.

[42] Shuo Chen, Rui Wang, XiaoFeng Wang, and Kehuan Zhang. Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow. In *IEEE S&P*, 2010.

[43] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. In *arXiv preprint arXiv:1712.05526*, 2017.

[44] Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum re-

silience of artificial neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, 2017.

[45] Heyning Cheng and Ron Avnur. Traffic analysis of ssl encrypted web browsing. In *URL citeseer. ist. psu. edu/656522. html*, 1998.

[46] Giovanni Cherubin, Jamie Hayes, and Marc Juarez. Website fingerprinting defenses at the application layer. In *PETS*, 2017.

[47] Soumith Chintala, Emily Denton, Martin Arjovsky, and Michael Mathieu. How to train a GAN? Tips and tricks to make GANs work, 2016. `https://github.com/soumith/ganhacks`.

[48] E. Choi, S. Biswal, B. Malin, J. Duke, W. F. Stewart, and J. Sun. Generating Multi-label Discrete Electronic Health Records using Generative Adversarial Networks. In *Machine Learning for Healthcare*, 2017.

[49] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. In *ICML*, 2019.

[50] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. In *IEEE transactions on information theory*, 1967.

[51] Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, and Gianluca Bontempi. Credit card fraud detection and concept-drift adaptation with delayed supervised information. In *IJCNN*, 2015.

[52] Katitza Rodriguez David Greene. Nsa mass surveillance programs unnecessary and disproportionate. Technical report, EFF, 2014. `https://www.eff.org/files/2014/05/29/unnecessary_and_disproportionate.pdf`.

[53] Wladimir De la Cadena, Asya Mitseva, Jan Pennekamp, Jens Hiller, Fabian Lanze, Thomas Engel, Klaus Wehrle, and Andriy Panchenko. Poster: Traffic splitting to counter website fingerprinting. In *ACM CCS*, 2019.

[54] Ambra Demontis, Paolo Russu, Battista Biggio, Giorgio Fumera, and Fabio Roli. On security and sparsity of linear classifiers for adversarial settings. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, 2016.

[55] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[56] Mariano Di Martino, Peter Quax, and Wim Lamotte. Realistically fingerprinting social media webpages in https traffic. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, 2019.

[57] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

[58] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[59] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *ICML*, 2016.

[60] Wenliang Du, Yunghsiang S Han, and Shigang Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *ICDM*, 2004.

[61] Krishnamurthy (Dj) Dvijotham, Sven Gowal, Robert Stanforth, Relja Arandjelovic, Brendan O'Donoghue, Jonathan Uesato, and Pushmeet Kohli. Training verified learners with learned verifiers. In *arXiv preprint arXiv:1805.10265*, 2018.

[62] Krishnamurthy (Dj) Dvijotham, Jamie Hayes, Borja Balle, Zico Kolter, Chongli Qin, Andras Gyorgy, Kai Xiao, Sven Gowal, and Pushmeet Kohli.

A framework for robustness certification of smoothed classifiers using f-divergences. In *ICLR*, 2020.

[63] Cynthia Dwork. Differential privacy: A survey of results. In *Theory and Applications of Models of Computation*, 2008.

[64] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, 2006.

[65] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *ITCS*, 2012.

[66] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toni Pitassi, Omer Reingold, and Aaron Roth. Generalization in adaptive data analysis and holdout reuse. In *Advances in Neural Information Processing Systems*, 2015.

[67] Kevin P Dyer, Scott E Coull, Thomas Ristenpart, and Thomas Shrimpton. Peek-a-boo, I still see you: Why efficient traffic analysis countermeasures fail. In *IEEE S&P*, 2012.

[68] Harrison Edwards and Amos Storkey. Censoring representations with an adversary. In *ICLR*, 2016.

[69] Ruediger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, 2017.

[70] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the landscape of spatial robustness. In *arXiv preprint arXiv:1712.02779*, 2017.

[71] Logan Engstrom, Andrew Ilyas, and Anish Athalye. Evaluating and understanding the robustness of adversarial logit pairing. In *arXiv preprint arXiv:1807.10272*, 2018.

[72] Tomáš Filler, Andrew D Ker, and Jessica Fridrich. The square root law of steganographic capacity for markov covers. In *Media Forensics and Security*, 2009.

[73] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *ACM CCS*, 2015.

[74] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *USENIX Security*, 2014.

[75] Jessica Fridrich, Miroslav Goljan, and Rui Du. Detecting LSB steganography in color, and gray-scale images. In *IEEE multimedia*, 2001.

[76] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. In *Annals of statistics*, 2001.

[77] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. In *ACM computing surveys (CSUR)*, 2014.

[78] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *IEEE S&P*, 2018.

[79] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *ICML*, 2016.

[80] Jiajun Gong and Tao Wang. Zero-delay lightweight defenses against website fingerprinting. 2020.

[81] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014.

[82] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. 2016.

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

[84] Sven Gowal, Krishnamurthy (Dj) Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. In *arXiv preprint arXiv:1810.12715*, 2018.

[85] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. In *arXiv preprint arXiv:1708.06733*, 2017.

[86] Xiaodan Gu, Ming Yang, and Junzhou Luo. A novel Website Fingerprinting attack against multi-tab browsing behavior. In *IEEE CSCWD*, 2015.

[87] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of Wasserstein GANs. In *ICLR*, 2018.

[88] David Ha. Generating large images from latent vectors, 2016. URL `http://blog.otoro.net/2016/04/01/generating-large-images-from-latent-vectors`.

[89] Jihun Hamm. Minimax filter: Learning to preserve privacy from inference attacks. In *Journal of Machine Learning Research*, 2017.

[90] Jihun Hamm and Akshay Mehra. Machine vs machine: Minimax-optimal defense against adversarial examples. In *arXiv preprint arXiv:1711.04368*, 2017.

[91] Jihun Hamm, Paul Cao, and Mikhail Belkin. Learning privately from multiparty data. In *ICML*, pages 555–563, 2016.

[92] M. Hayden. The price of privacy: Re-evaluating the nsa, 2014. URL `https://www.youtube.com/watch?v=kV2HDM86XgI&t=17m50s`.

[93] Jamie Hayes. Traffic confirmation attacks despite noise. In *arXiv preprint arXiv:1601.04893*, 2016.

[94] Jamie Hayes. On visible adversarial perturbations & digital watermarking. In *CVPR Workshops*, 2018.

[95] Jamie Hayes. Extensions and limitations of randomized smoothing for robustness guarantees. In *CVPR Workshops*, 2020.

[96] Jamie Hayes and George Danezis. Guard sets for onion routing. In *PETS*, 2015.

[97] Jamie Hayes and George Danezis. k-fingerprinting: A robust scalable website fingerprinting technique. In *USENIX Security*, 2016.

[98] Jamie Hayes and George Danezis. Generating steganographic images via adversarial training. In *Advances in Neural Information Processing Systems*, 2017.

[99] Jamie Hayes and George Danezis. Learning universal adversarial perturbations with generative models. In *IEEE S&P Workshops*, 2018.

[100] Jamie Hayes and Olga Ohrimenko. Contamination attacks and mitigation in multi-party machine learning. In *Advances in Neural Information Processing Systems*, 2018.

[101] Jamie Hayes, Carmela Troncoso, and George Danezis. TASP: Towards anonymity sets that persist. In *WPES*, 2016.

[102] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. LOGAN: Membership inference attacks against generative models. In *PETS*, 2019.

[103] Jamie Hayes, Krishnamurthy (Dj) Dvijotham, Yutian Chen, Sander Dieleman, Pushmeet Kohli, and Norman Casagrande. Provenance detection through learning transformation-resilient watermarking, 2020. URL `https://openreview.net/forum?id=S1gmvyHFDS`.

[104] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[105] Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. Adversarial example defense: Ensembles of weak defenses are not strong. In *USENIX WOOT*, 2017.

[106] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In *ACM workshop on Cloud computing security*, 2009.

[107] Ehsan Hesamifard, Hassan Takabi, Mehdi Ghasemi, and Rebecca N Wright. Privacy-preserving machine learning as a service. In *PETS*, 2018.

[108] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *arXiv 1503.02531*, 2015.

[109] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning. In *ACM CCS*, 2017.

[110] Matthew Hoekstra, Reshma Lal, Pradeep Pappachan, Carlos Rozas, Vinay Phegade, and Juan del Cuvillo. Using innovative instructions to create trustworthy software solutions. In *HASP*, 2013.

[111] Vojtech Holub and Jessica Fridrich. Designing steganographic distortion using directional filters. In *IEEE WIFS*, 2012.

[112] Vojtěch Holub, Jessica Fridrich, and Tomáš Denemark. Universal distortion function for steganography in an arbitrary domain. In *EURASIP Journal on Information Security*, 2014.

[113] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V Pearson, Dietrich A Stephan, Stanley F Nelson, and David W Craig. Resolving individuals contributing trace amounts

of DNA to highly complex mixtures using high-density SNP genotyping microarrays. In *PLoS Genet*, 2008.

[114] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[115] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. Technical report, University of Massachusetts, Amherst, 2007. `http://vis-www.cs.umass.edu/lfw/lfw.pdf`.

[116] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. In *arXiv preprint arXiv:1702.02284*, 2017.

[117] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[118] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *IEEE S&P*, 2018.

[119] Shouling Ji, Weiqing Li, Neil Zhenqiang Gong, Prateek Mittal, and Raheem A Beyah. On your social network de-anonymizablity: Quantification and large scale evaluation with seed knowledge. In *NDSS*, 2015.

[120] Jinyuan Jia and Neil Zhenqiang Gong. Attriguard: A practical defense against attribute inference attacks via adversarial machine learning. In *USENIX Security*, 2018.

[121] Marc Juarez, Sadia Afroz, Gunes Acar, Claudia Diaz, and Rachel Greenstadt. A critical evaluation of website fingerprinting attacks. In *ACM CCS*, 2014.

[122] Marc Juarez, Mohsen Imani, Mike Perry, Claudia Diaz, and Matthew Wright. Toward an efficient website fingerprinting defense. In *ESORICS*, 2016.

[123] Kaggle.com. Diabetic Retinopathy Detection. `https://www.kaggle.com/c/diabetic-retinopathy-detection#references`, 2015.

[124] Andrej Karpathy, Pieter Abbeel, Greg Brockman, Peter Chen, Vicki Cheung, Rocky Duan, Ian Goodfellow, Durk Kingma, Jonathan Ho, Rein Houthooft, Tim Salimans, John Schulman, Ilya Sutskever, and Wojciech Zaremba. Generative Models. `https://blog.openai.com/generative-models/`, 2017.

[125] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, 2017.

[126] Marc Khoury and Dylan Hadfield-Menell. On the geometry of adversarial examples. In *arXiv preprint arXiv:1811.00525*, 2018.

[127] Seung-Jean Kim and Stephen Boyd. A minimax theorem with applications to machine learning, signal processing, and finance. In *SIAM Journal on Optimization*, 2008.

[128] Yoon Kim. Convolutional neural networks for sentence classification. In *EMNLP*, 2014.

[129] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *arXiv preprint arXiv:1412.6980*, 2014.

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

[131] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *ICML*, pages 1885–1894, 2017.

[132] Jernej Kos, Ian Fischer, and Dawn Song. Adversarial examples for generative models. In *IEEE S&P Workshops*, 2018.

[133] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

[134] Bogdan Kulynych, Jamie Hayes, Nikita Samarin, and Carmela Troncoso. Evading classifiers in discrete domains with provable optimality guarantees. In *arXiv preprint arXiv:1810.10939*, 2018.

[135] Aounon Kumar, Alexander Levine, Tom Goldstein, and Soheil Feizi. Curse of dimensionality on randomized smoothing for certifiable robustness. In *arXiv preprint arXiv:2002.03239*, 2020.

[136] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *arXiv preprint arXiv:1607.02533*, 2016.

[137] Matt J Kusner, Jacob R Gardner, Roman Garnett, and Kilian Q Weinberger. Differentially Private Bayesian Optimization. In *ICML*, 2015.

[138] Albert Kwon, Mashael AlSabah, David Lazar, Marc Dacier, and Srinivas Devadas. Circuit Fingerprinting Attacks: Passive Deanonymization of Tor Hidden Services. In *USENIX Security*, 2015.

[139] Terran D Lane. Machine learning techniques for the computer security domain of anomaly detection. 2000.

[140] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *ICLM*, 2016.

[141] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*. 2012.

[142] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *IEEE S&P*, 2019.

[143] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *arXiv preprint arXiv:1609.04802*, 2016.

[144] Guang-He Lee, Yang Yuan, Shiyu Chang, and Tommi S Jaakkola. A stratified approach to robustness for randomly smoothed classifiers. In *arXiv preprint arXiv:1906.04948*, 2019.

[145] Daniel Lerch-Hostalot and David Megías. Unsupervised steganalysis based on artificial training sets. In *Engineering Applications of Artificial Intelligence*, 2016.

[146] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Certified adversarial robustness with additive noise. In *Advances in Neural Information Processing Systems*, 2019.

[147] Kaiqi Liang, Gaopeng Gou, Cuicui Kang, Chang Liu, Min Yang, and Yu Guo. A multi-view deep learning model for encrypted website service classification. In *IEEE GLOBECOM*, 2019.

[148] A. Liaw and M. Wiener. Classification and Regression by randomForest. In *R News: The Newsletter of the R Project*, 2002.

[149] Marc Liberatore and Brian Neil Levine. Inferring the source of encrypted HTTP connections. In *ACM CCS*, 2006.

[150] Wei-Yang Lin, Ya-Han Hu, and Chih-Fong Tsai. Machine learning in financial crisis prediction: a survey. In *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 2012.

[151] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In *CRYPTO*, 2000.

[152] Chen Liu, Ryota Tomioka, and Volkan Cevher. On certifying non-uniform bound against adversarial attacks. In *arXiv preprint arXiv:1903.06603*, 2019.

[153] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-based anomaly detection. In *TKDD*, 2012.

[154] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[155] Yunhui Long, Vincent Bindschaedler, Lei Wang, Diyue Bu, Xiaofeng Wang, Haixu Tang, Carl A Gunter, and Kai Chen. Understanding membership inferences on well-generalized learning models. In *arXiv preprint arXiv:1802.04889*, 2018.

[156] Gilles Louppe, Michael Kagan, and Kyle Cranmer. Learning to pivot with adversarial networks. In *Advances in Neural Information Processing Systems*, 2017.

[157] Liming Lu, Ee-Chien Chang, and Mun Choon Chan. Website fingerprinting and identification using ordered feature sequences. In *ESORICS*, 2010.

[158] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. Are GANs Created Equal? A Large-Scale Study. In *ArXiv 1711.10337*, 2017.

[159] Xiapu Luo, Peng Zhou, Edmond W. W. Chan, Wenke Lee, Rocky K. C. Chang, and Roberto Perdisci. HTTPOS: Sealing information leaks with browser-side obfuscation of encrypted flows. In *NDSS*, 2011.

[160] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013.

[161] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *arXiv preprint arXiv:1706.06083*, 2017.

[162] Vasilios Mavroudis and Jamie Hayes. Adaptive traffic fingerprinting: Large-scale inference under realistic assumptions. *arXiv preprint arXiv:2010.10294*, 2020.

[163] H Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. In *arXiv preprint arXiv:1602.05629*, 2016.

[164] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

[165] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *ICLR*, 2018.

[166] Frank McSherry. Statistical inference considered harmful. `https://github.com/frankmcsherry/blog/blob/master/posts/2016-06-14.md`, 2016.

[167] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *IEEE S&P*, 2019.

[168] Jan Hendrik Metzen. Universality, robustness, and detectability of adversarial perturbations under adversarial training, 2018. URL `https://openreview.net/forum?id=SyjsLqxR-`.

[169] Jarno Mielikainen. LSB matching revisited. In *IEEE signal processing letters*, 2006.

[170] Matthew Mirman, Timon Gehr, and Martin Vechev. Differentiable abstract interpretation for provably robust neural networks. In *ICML*, 2018.

[171] P. Mohassel and Y. Zhang. SecureML: A System for Scalable Privacy-Preserving Machine Learning. In *IEEE S&P*, 2017.

[172] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *CVPR*, pages 2574–2582, 2016.

[173] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *CVPR*, 2017.

[174] Konda Reddy Mopuri, Utsav Garg, and R Venkatesh Babu. Fast feature fool: A data independent approach to universal adversarial perturbations. In *arXiv preprint arXiv:1707.05572*, 2017.

[175] Saralees Nadarajah. A generalized normal distribution. In *Journal of Applied Statistics*, 2005.

[176] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[177] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *IEEE S&P*, 2009.

[178] Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine learning with membership privacy using adversarial regularization. In *ACM CCS*, 2018.

[179] Dong Nie, Roger Trullo, Caroline Petitjean, Su Ruan, and Dinggang Shen. Medical Image Synthesis with Context-Aware Generative Adversarial Networks. In *MICCAI*, 2017.

[180] Valeria Nikolaenko, Stratis Ioannidis, Udi Weinsberg, Marc Joye, Nina Taft, and Dan Boneh. Privacy-preserving matrix factorization. In *ACM CCS*, 2013.

[181] Rishab Nithyanand, Xiang Cai, and Rob Johnson. Glove: A Bespoke Website Fingerprinting Defense. In *WPES*, 2014.

[182] Ziad Obermeyer and Ezekiel J Emanuel. Predicting the future—big data, machine learning, and clinical medicine. In *The New England journal of medicine*, 2016.

[183] Olga Ohrimenko, Felix Schuster, Cédric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, and Manuel Costa. Oblivious multi-party machine learning on trusted processors. In *USENIX Security*, 2016.

[184] A. Stolerman M. V. Ryan P. Brennan P. Juola, J. I. Noecker Jr and R. Greenstadt. A Dataset for Active Linguistic Authentication. In *International Conference on Digital Forensics*, 2013.

[185] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. Website fingerprinting in onion routing based anonymization networks. In *WPES*, 2011.

[186] Andriy Panchenko, Fabian Lanze, Andreas Zinnen, Martin Henze, Jan Pennekamp, Klaus Wehrle, , and Thomas Engel. Website Fingerprinting at Internet Scale. In *NDSS*, 2016.

[187] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. In *arXiv preprint arXiv:1605.07277*, 2016.

[188] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE EuroS&P*, 2016.

[189] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Security and Privacy*, 2016.

[190] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *ICLR*, 2017.

[191] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks against Machine Learning. In *AsiaCCS*, 2017.

[192] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable Private Learning with PATE. In *ICLR*, 2018.

[193] Thrasyvoulos N Pappas and Robert J Safranek. Perceptual criteria for image quality evaluation.

[194] Manas A. Pathak, Shantanu Rane, and Bhiksha Raj. Multiparty differential privacy via aggregation of locally trained classifiers. In *Advances in Neural Information Processing Systems*, 2010.

[195] Mike Perry. Experimental defense website traffic fingerprinting. Technical report. `https://blog.torproject.org/blog/experimental-defense-website-traffic-fingerprinting`.

[196] Mike Perry. A critique of website traffic fingerprinting attacks. Technical report, Accessed June 2015. `https://blog.torproject.org/blog/critique-website-traffic-fingerprinting-attacks`.

[197] Tomáš Pevnỳ, Tomáš Filler, and Patrick Bas. Using high-dimensional image models to perform highly undetectable steganography. In *International Workshop on Information Hiding*, 2010.

[198] Ania M Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The loopix anonymity system. In *USENIX Security*, 2017.

[199] Ania M Piotrowska, Jamie Hayes, Nethanel Gelernter, George Danezis, and Amir Herzberg. Annotify: a private notification service. In *WPES*, 2017.

[200] Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. Generative adversarial perturbations. In *CVPR*, 2018.

[201] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. What Does The Crowd Say About You? Evaluating Aggregation-based Location Privacy. In *PETS*, 2017.

[202] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. Knock Knock, Who's There? Membership Inference on Aggregate Location Data. In *NDSS*, 2018.

[203] Jianwei Qian, Xiang-Yang Li, Chunhong Zhang, and Linlin Chen. De-anonymizing social networks and inferring private attributes using knowledge graphs. In *INFOCOM*, 2016.

[204] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *arXiv 1511.06434*, 2015.

[205] Md Atiqur Rahman, Tanzila Rahman, Robert Laganiere, Noman Mohammed, and Yang Wang. Membership Inference Attack against Differentially Private Deep Learning Model. In *Transactions on Data Privacy*, 2018.

[206] Arun Rajkumar and Shivani Agarwal. A differentially private stochastic gradient descent algorithm for multiparty classification. In *AISTATS*, 2012.

[207] Konda Reddy Mopuri, Utkarsh Ojha, Utsav Garg, and R Venkatesh Babu. Nag: Network for adversary generation. In *CVPR*, 2018.

[208] Vera Rimmer, Davy Preuveneers, Marc Juarez, Tom Van Goethem, and Wouter Joosen. Automated website fingerprinting through deep learning. In *arXiv preprint arXiv:1708.06376*, 2017.

[209] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *ECCV*, 2006.

[210] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. In *IJCV*, 2015.

[211] Tim Salimans and Diederik P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, 2016.

[212] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems*, 2016.

[213] Felix Schuster, Manuel Costa, Cédric Fournet, Christos Gkantsidis, Marcus Peinado, Gloria Mainar-Ruiz, and Mark Russinovich. $VC3$: Trustworthy data analytics in the cloud using SGX. In *IEEE S&P*, 2015.

[214] Shiqi Shen, Shruti Tople, and Prateek Saxena. Auror: Defending against poisoning attacks in collaborative deep learning systems. In *ACSAC*, 2016.

[215] Margaret A Shipp, Ken N Ross, Pablo Tamayo, Andrew P Weng, Jeffery L Kutok, Ricardo CT Aguiar, Michelle Gaasenbeek, Michael Angelo, Michael Reich, Geraldine S Pinkus, et al. Diffuse large b-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. In *Nature medicine*, 2002.

[216] Vitaly Shmatikov and Ming-Hsiu Wang. Timing Analysis in Low-Latency Mix Networks: Attacks and Defenses. In *ESORICS*, 2006.

[217] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *ACM CCS*, 2015.

[218] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *IEEE S&P*, 2017.

[219] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *arXiv preprint arXiv:1409.1556*, 2014.

[220] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *ACM CCS*, 2018.

[221] Sayanan Sivaraman and Mohan M Trivedi. Active learning for on-road vehicle detection: A comparative study. In *Machine vision and applications*, 2014.

[222] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine learning models that remember too much. In *ACM CCS*, 2017.

[223] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models. In *Advances in Neural Information Processing Systems*, 2018.

[224] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. In *Journal of machine learning research*, 2014.

[225] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified defenses for data poisoning attacks. In *Advances in neural information processing systems*, 2017.

[226] Qixiang Sun, Daniel R Simon, Yi-Min Wang, Wilf Russell, Venkata N Padmanabhan, and Lili Qiu. Statistical identification of encrypted web browsing traffic. In *IEEE S&P*, 2002.

[227] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *arXiv preprint arXiv:1312.6199*, 2013.

[228] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[229] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[230] Jiaye Teng, Guang-He Lee, and Yang Yuan. ℓ_1 adversarial robustness certificates: a randomized smoothing approach, 2020. URL `https://openreview.net/forum?id=H1lQIgrFDS`.

[231] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. In *ICLR*, 2017.

[232] Theodore B Trafalis and Huseyin Ince. Support vector machine for regression and applications to financial forecasting. In *IJCNN*, 2000.

[233] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *USENIX Security*, 2016.

[234] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. In *arXiv preprint arXiv:1704.03453*, 2017.

[235] Aleksei Triastcyn and Boi Faltings. Generating differentially private datasets using gans. In *arXiv preprint arXiv:1803.03148*, 2018.

[236] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. Towards Demystifying Membership Inference Attacks. In *arXiv:1807.09173*, 2018.

[237] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. There is no free lunch in adversarial robustness (but there are unexpected benefits). In *arXiv preprint arXiv:1805.12152*, 2018.

[238] Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. In *Advances in Neural Information Processing Systems*, 2018.

[239] Jonathan Uesato, Brendan O'Donoghue, Aaron van den Oord, and Pushmeet Kohli. Adversarial risk and the dangers of evaluating against weak attacks. In *arXiv preprint arXiv:1802.05666*, 2018.

[240] James Vincent. `https://www.theverge.com/2016/7/5/12095830/google-deepmind-nhs-eye-disease-detection`, 2016.

[241] David Wagner and Bruce Schneier. Analysis of the SSL 3.0 Protocol. In *USENIX Workshop on Electronic Commerce*, 1996.

[242] Martin J Wainwright, Michael I Jordan, and John C Duchi. Privacy aware learning. In *Advances in Neural Information Processing Systems*, 2012.

[243] T. Wang and I. Goldberg. Walkie-Talkie: An Effective and Efficient Defense against Website Fingerprinting. Technical Report, 2015.

[244] Tao Wang. High precision open-world website fingerprinting. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 152–167. IEEE, 2020.

[245] Tao Wang and Ian Goldberg. Improved website fingerprinting on Tor. In *WPES*, 2013.

[246] Tao Wang and Ian Goldberg. Comparing website fingerprinting attacks and defenses. Technical report, Technical Report 2013-30, CACR, 2013. http://cacr. uwaterloo. ca/techreports ..., 2014.

[247] Tao Wang and Ian Goldberg. On realistically attacking tor with website fingerprinting. In *PETS*, 2016.

[248] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. Effective attacks and provable defenses for website fingerprinting. In *USENIX Security*, 2014.

[249] Zhou Wang and Alan C Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. In *IEEE signal processing magazine*, 2009.

[250] Xuezhi Wen, Ling Shao, Yu Xue, and Wei Fang. A rapid learning algorithm for vehicle classification. In *Information Sciences*, 2015.

[251] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane Boning, Inderjit S Dhillon, and Luca Daniel. Towards fast computation of certified robustness for relu networks. In *arXiv preprint arXiv:1804.09699*, 2018.

[252] Charles V Wright, Scott E Coull, and Fabian Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *NDSS*, 2009.

[253] Xi Wu, Matthew Fredrikson, Wentao Wu, Somesh Jha, and Jeffrey F Naughton. Revisiting differentially private regression: Lessons from learning theory and their consequences. In *arXiv 1512.06388*, 2015.

[254] Xiaolin Wu and Xi Zhang. Automated Inference on Criminality using Face Images. In *arXiv 1611.04135*, 2016.

[255] Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov, and Roger Grosse. On the Quantitative Analysis of Decoder-Based Generative Models. In *ICLR*, 2017.

[256] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. Is feature selection secure against training data poisoning? In *ICML*, 2015.

[257] Huang Xiao, Battista Biggio, Blaine Nelson, Han Xiao, Claudia Eckert, and Fabio Roli. Support vector machines under adversarial label contamination. In *Neurocomputing*, 2015.

[258] Greg Yang, Tony Duan, Edward Hu, Hadi Salman, Ilya Razenshteyn, and Jerry Li. Randomized smoothing of all shapes and sizes. In *arXiv preprint arXiv:2002.08118*, 2020.

[259] Qing-Hai Ye, Lun-Xiu Qin, Marshonna Forgues, Ping He, Jin Woo Kim, Amy C Peng, Richard Simon, Yan Li, Ana I Robles, Yidong Chen, et al. Predicting hepatitis b virus-positive metastatic hepatocellular carcinomas using gene expression profiling and supervised machine learning. In *Nature medicine*, 2003.

[260] Raymond Yeh, Chen Chen, Teck Yian Lim, Mark Hasegawa-Johnson, and Minh N Do. Semantic Image Inpainting with Perceptual and Contextual Losses. In *arXiv 1607.07539*, 2016.

[261] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *IEEE CSF*, 2018.

[262] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P. Gummadi. Fairness constraints: Mechanisms for fair classification. In *AISTATS*, 2017.

[263] Richard S. Zemel, Yu Wu, Kevin Swersky, Toniann Pitassi, and Cynthia Dwork. Learning fair representations. In *ICML*, pages 325–333, 2013.

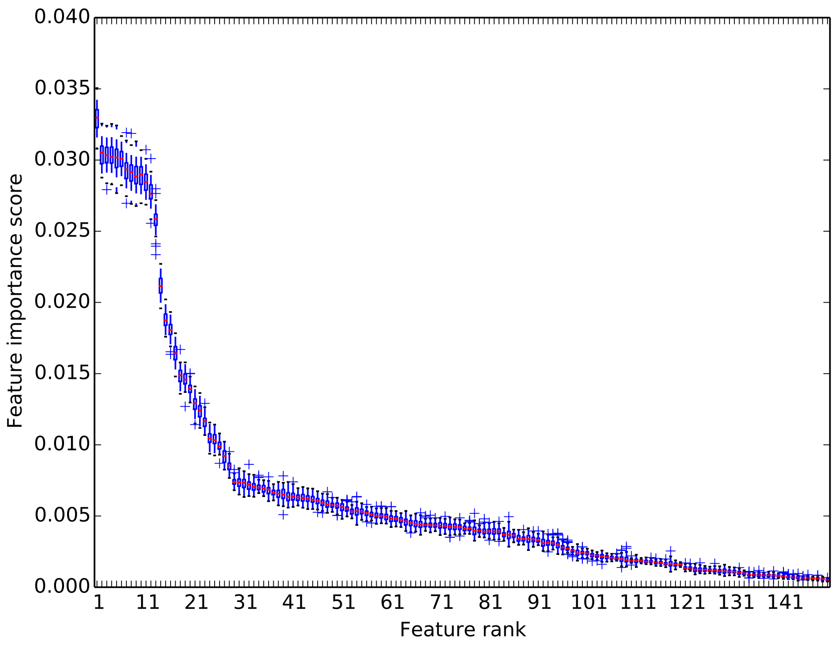
[264] Fei Zhang, Patrick PK Chan, Battista Biggio, Daniel S Yeung, and Fabio Roli. Adversarial feature selection against evasion attacks. In *IEEE transactions on cybernetics*, 2016.

[265] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in Neural Information Processing Systems*, 2018.

Appendix A

Robust & scalable website fingerprinting

A.1 Total feature importance



№	Feature Description
131 - 150 .	Packet concentration list features.

Figure A.1: Feature importance score for all 150 features in order. The table gives the description for the 20 least important features.

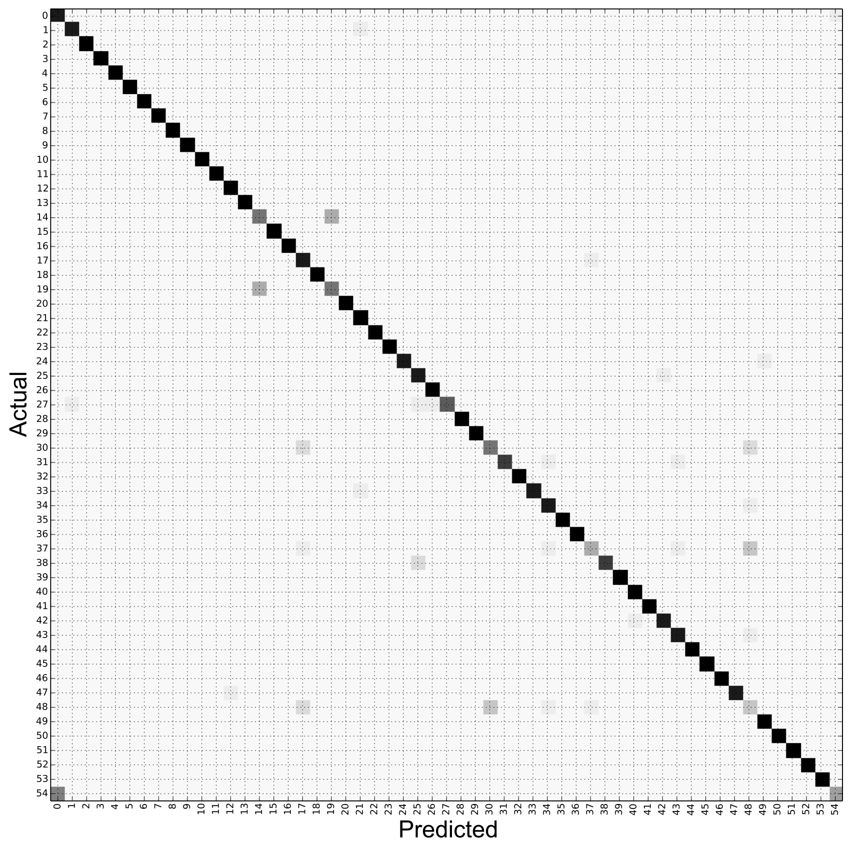


Figure A.2: Confusion matrix for closed-world attack on Tor using DS_{Norm}. F1 score = 0.913, Accuracy: 0.915, 550 items.

A.2 Closed World Error Rates

Figure A.2 shows the confusion matrix in our closed-world setting, that is, it shows the 49 misclassifications (out of 550). We see that some persistent misclassification patterns of web pages appear, for example web page 54 is classified correctly four times but is misclassified as web page 0 six times. The misclassification rate in fig. A.2 is 0.09 but this is the average error rate across all web pages.

Figure A.2 shows that the classification error is not uniform across all web pages. Some pages are misclassified many times, and confused with many others, while others are never misclassified. An attacker can leverage this information to estimate the misclassification rate of each web page instead of using the global average misclassification rate.

As in section 3.9, an attacker can use their training set of web pages to estimate the misclassification rate of each web page, by splitting the training set in

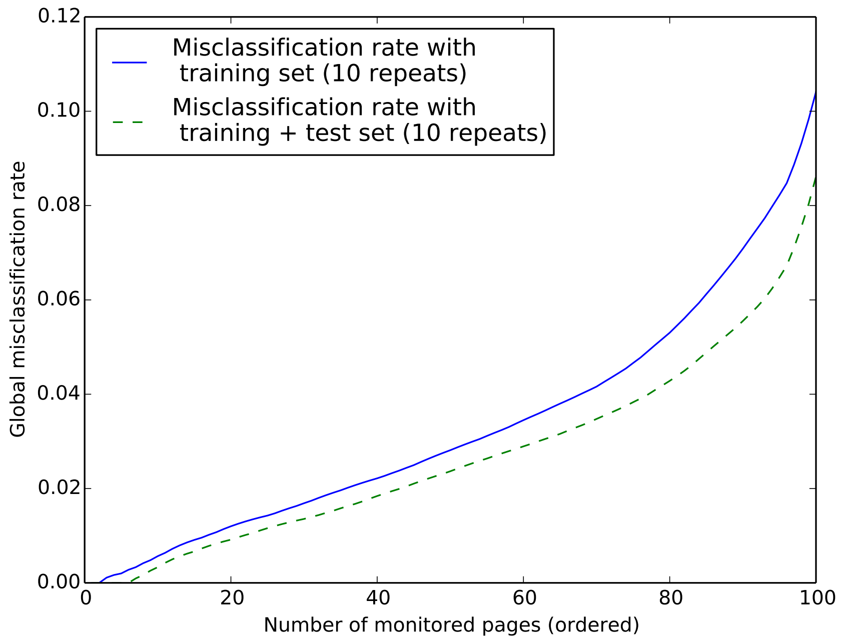


Figure A.3: The global misclassification rate when considering different numbers of monitored pages from the Wang et al. [248] dataset. The monitored pages are ordered in terms of smallest misclassification rate to largest.

to a smaller training set and validation set. Since both sets are from the original training set the attacker has access to the true labels. The attacker then computes the misclassification rate of each web page, which they can use as an estimation for the misclassification rate when training on the entire training set and testing on new traffic instances.

Figure A.3 and fig. A.4 show the global misclassification rate for a varying number of monitored pages. Monitored pages are first ordered in terms of the misclassification rate they have, ordered from smallest to largest. From fig. A.3, using the Wang et al. [248] dataset, we see that if the attacker considers only the top 50% on web pages in terms of per page misclassification rate, the true global misclassification rate and global misclassification rate estimated by the attacker drop by over 70%. Similarly from fig. A.4, using DS_{Norm}, if the attacker considers only the top 50% on web pages in terms of per page misclassification rate, the true global misclassification rate and global misclassification rate estimated by the attacker drop by over 80%. This allows an attacker to train on monitored pages and then cull the pages that have too high an error rate, allowing for more confidence in the classifi-

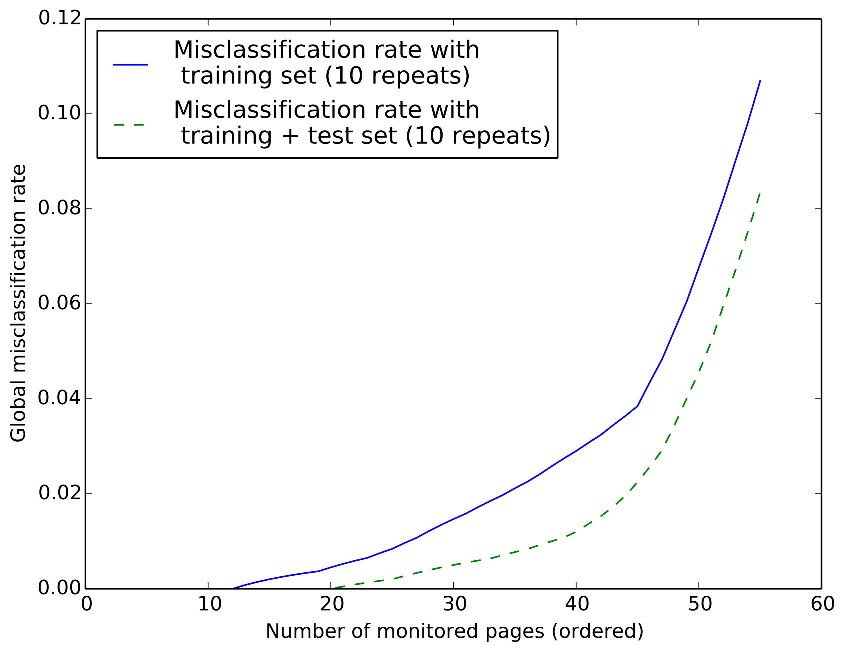


Figure A.4: The global misclassification rate when considering different numbers of monitored pages from DS_{Norm}. The monitored pages are ordered in terms of smallest misclassification rate to largest.

cation of the rest of the monitored pages.

The gap between the attacker's estimate and the misclassification rate of the test set is largely due to the size of the dataset. Figure A.3 has a smaller error of estimate than fig. A.4 because the Wang et al. [248] dataset has 60 instances per monitored page, in comparison DS_{Norm} has 20 instances per monitored page. In practice, an attacker cannot expect perfect alignment; they are generated from two different sets of data, the training and training + test set. Nevertheless the attacker can expect this difference to decrease with the collection of more training instances.

A.3 Attack on larger world size of DS_{Norm}

We run k-fingerprinting on DS_{Norm} with the same number of monitored sites but increase the numbered of unmonitored sites to 17,000. We evaluate when we have both time and size features available.

Figure A.5 shows the results of k-fingerprinting while varying the number of fingerprints (k) used for classification, from between 1 and 10, for various experiments trained with different numbers of unmonitored pages. We see that the attack

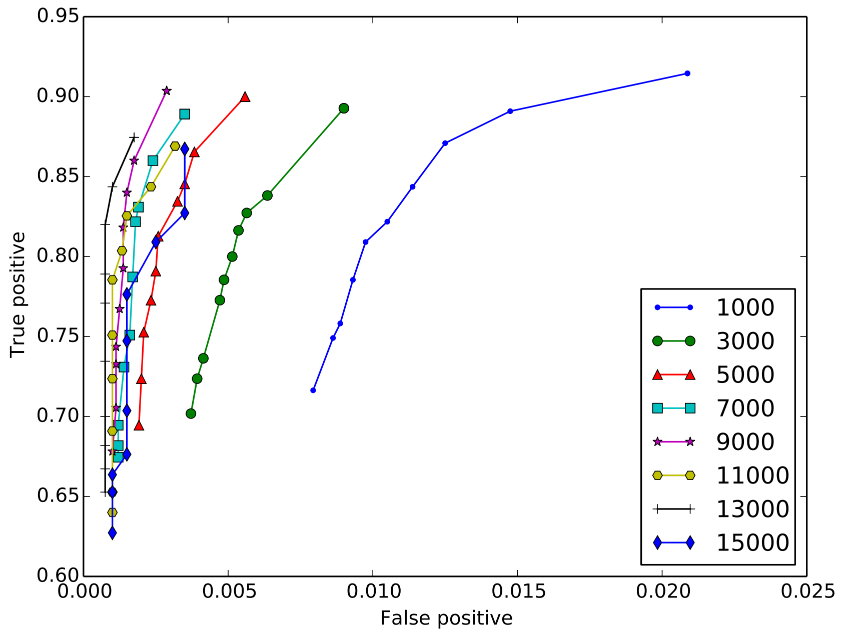


Figure A.5: Attack accuracy for 17,000 unmonitored web pages. Each line represents a different number of unmonitored web pages that were trained, while varying k, the number of fingerprints used for classification.

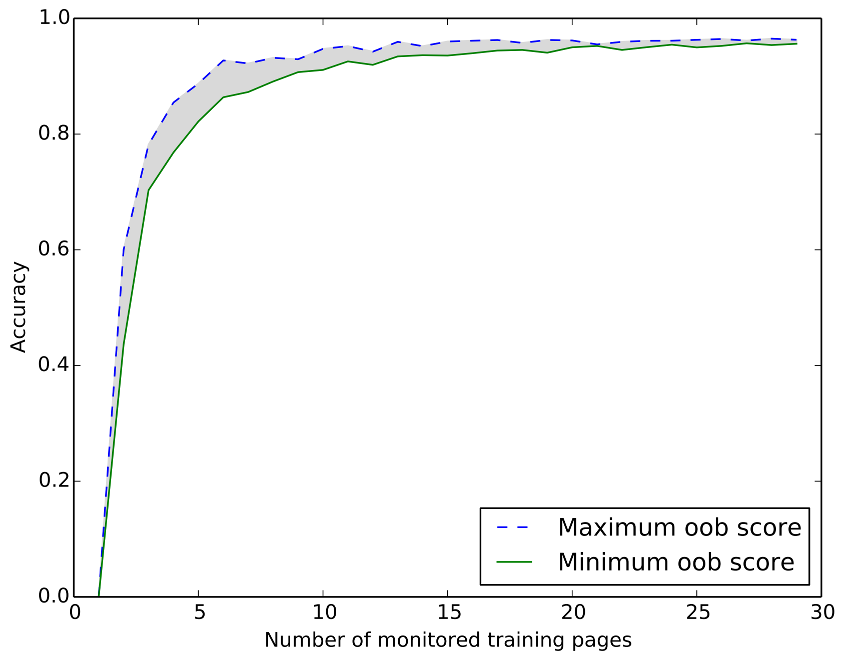


Figure A.6: Attack out-of-bag score while varying the number of monitored training pages.

results are comparable to the attack on 7000 unmonitored pages, meaning there is no degradation in attack accuracy when we increase the world size by 10,000 web pages. Training on approximately 30% of the 7000 unmonitored web pages *k*-fingerprinting gives a TPR of over 0.90 and a FPR of 0.01 for *k*=1. Training on approximately 30% of the 17,000 unmonitored web pages *k*-fingerprinting gives a TPR of 0.90 and a FPR of 0.006 for *k*=1.

The fraction of unmonitored pages that were incorrectly classified as a monitored page decreased as we increased our world size. In other words, out of 12,000 unmonitored pages only 72 were classified as a monitored page, with this Figure dropping to 24 if we use *k*=10 for classification. This provides a strong indication that *k*-fingerprinting can scale to a real-world attack in which a client is free to browse the entire internet, with no decrease in attack accuracy.

Number of monitored training pages in closed-world. Figure A.6 shows the *out-of-bag* score as we change the number of *monitored* pages we train. We found that training on any more than a third of the data gives roughly the same accuracy.

Appendix B

Membership inference attacks against generative models

B.1 Unsuccessful attacks

We now report a few additional results, not included in the main body of the paper to ease presentation. In fig. B.1, we report the results of the Euclidean attack presented in section 5.7. This attack was performed on a target model (DCGAN) trained on a random 10% subset of CIFAR-10 and a random 10% subset of LFW, but we found that the attack did not perform much better than a random guess.

We also report on the results of a black-box setting where 10% of training set samples from LFW are used to train a *shadow model* – see fig. B.2. Samples generated by this model are then injected into the attacker model together with the samples generated by the target model. More specifically, at training time, each mini-batch is composed of synthetic samples generated either by the target model or by the shadow model. However, this attack, inspired by the approach proposed by Shokri et al. [218], only yields around 18% of accuracy, with no improvements during training.

B.2 Additional samples

In fig. B.3–fig. B.10, we report additional examples of samples deferred from section 5.6. Specifically, real and generated samples are shown in fig. B.3 for LFW and in fig. B.4 for the diabetic retinopathy (DR) dataset. Then, fig. B.5 shows real

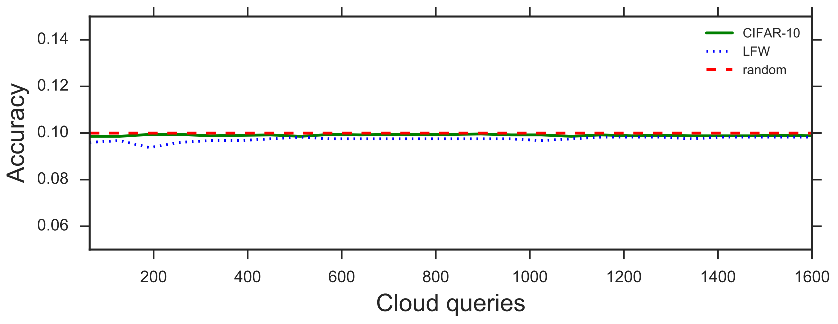


Figure B.1: Euclidean attack results for DCGAN target model trained on a random 10% subset of CIFAR-10 and LFW.

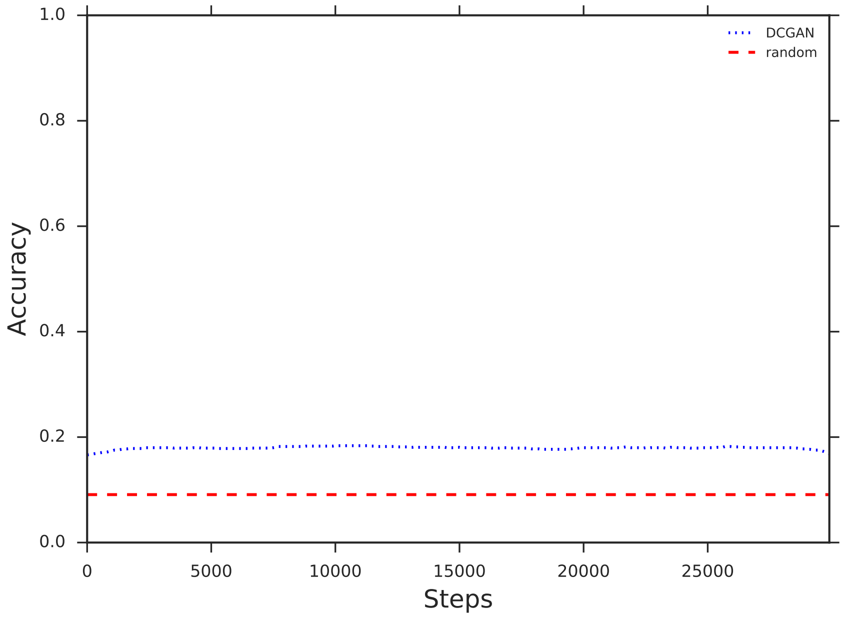
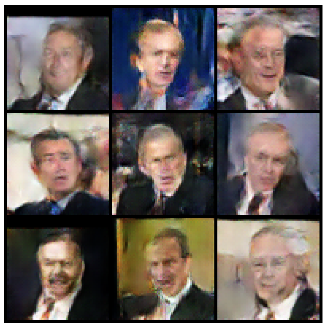


Figure B.2: Black-box attack results with 10% auxiliary attacker training set knowledge used to train a DCGAN *shadow model* for DCGAN target model trained on a random 10% subset of LFW.

samples from LFW and CIFAR-10, while fig. B.6–fig. B.9 depict samples generated by various target models on LFW. Finally, samples generated by the attacker model on LFW are reported in fig. B.10.

(a) Real samples


(b) Target samples

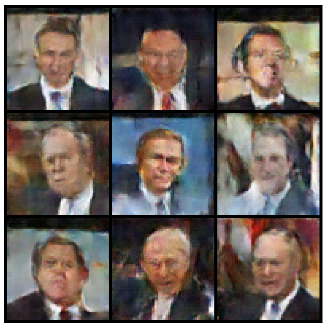
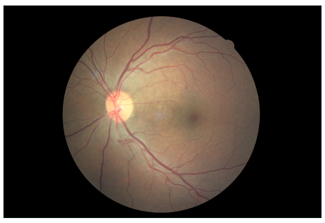
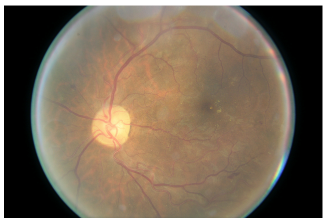

(c) Attacker model samples

Figure B.3: Various samples from the real dataset, target model, and black-box attack using the DCGAN target model on LFW, top ten classes.


(a) Real sample with no presence of diabetic retinopathy


(b) Real sample with high presence of diabetic retinopathy

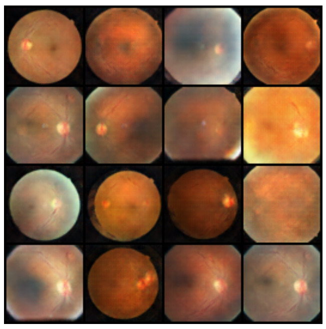
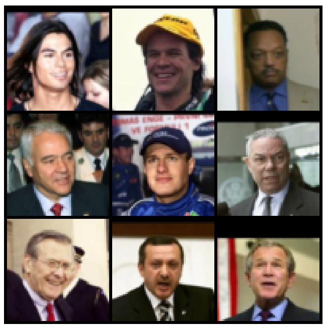

(c) Selection of target generated samples classified with high confidence as belonging to the training set by both white-box and black-box attacks

Figure B.4: Real and generated diabetic retinopathy dataset samples.

(a) LFW, top ten classes


(b) LFW, random 10% subset

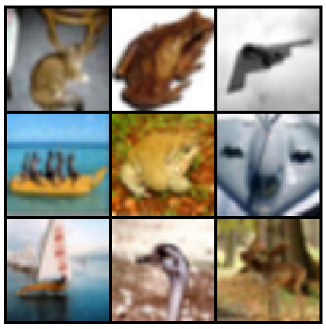
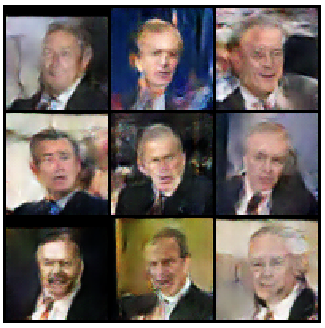
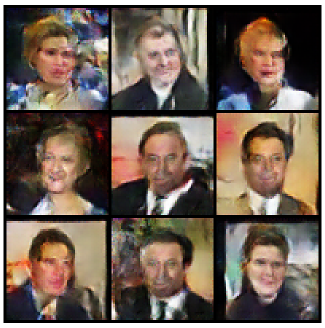

(c) CIFAR-10, random 10% subset

Figure B.5: Real samples.


(a) LFW, top ten classes


(b) LFW, random 10% subset

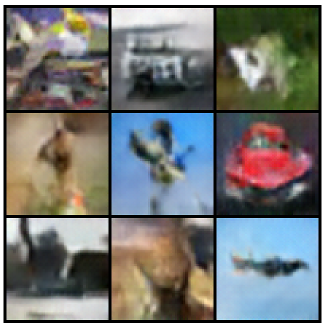
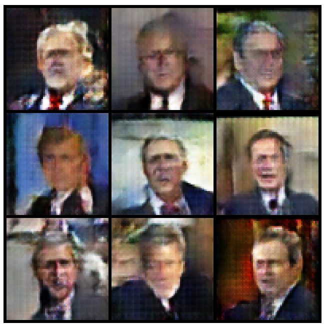
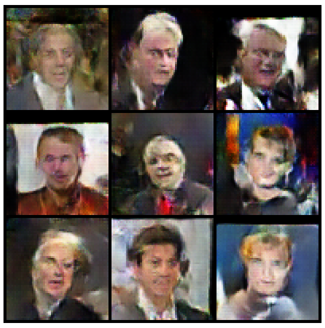

(c) CIFAR-10, random 10% subset

Figure B.6: Samples generated by DCGAN target model.


(a) LFW, top ten classes


(b) LFW, random 10% subset

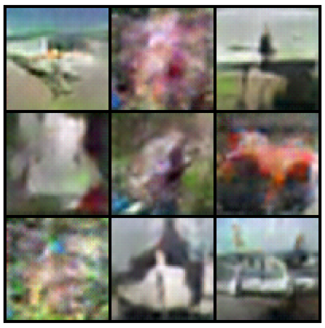

(c) CIFAR-10, random 10% subset

Figure B.7: Samples generated by DCGAN+VAE target model.

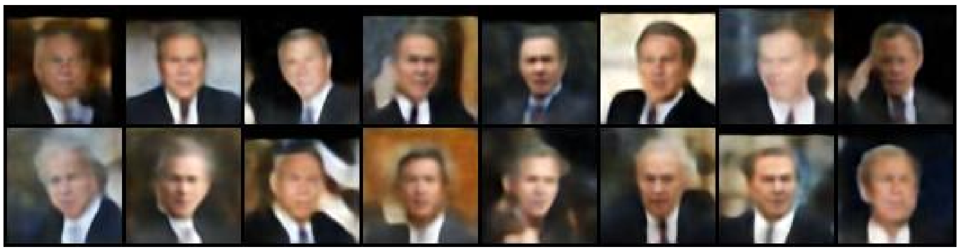


Figure B.8: Samples generated by BEGAN target model on LFW, top ten classes.

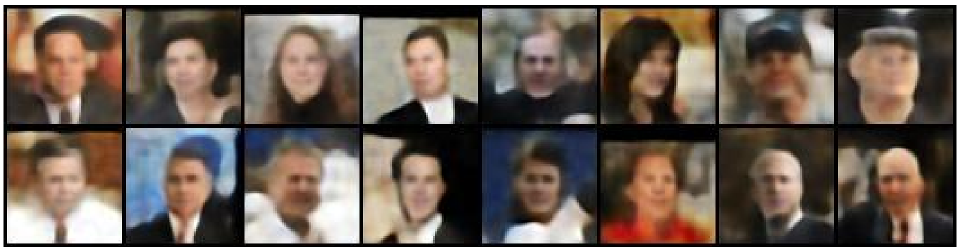
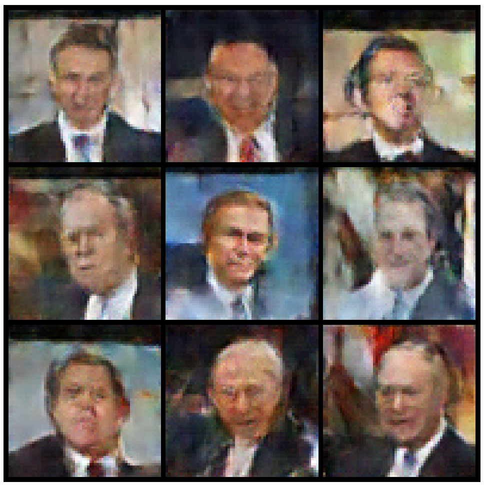


Figure B.9: Samples generated by BEGAN target model on LFW, random 10% subset.


(a) LFW, top ten classes

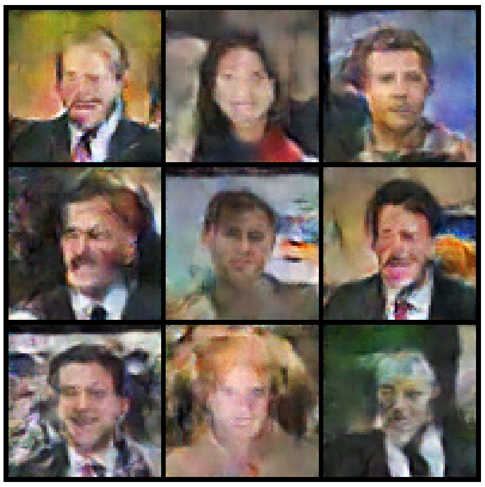

(b) LFW, random 10% subset

Figure B.10: Samples generated by attacker model trained on samples from DCGAN target model on (a) LFW, top ten classes and (b) LFW, random 10% subset.

Appendix C

Contamination attacks

C.1 Additional ADULT dataset experiments

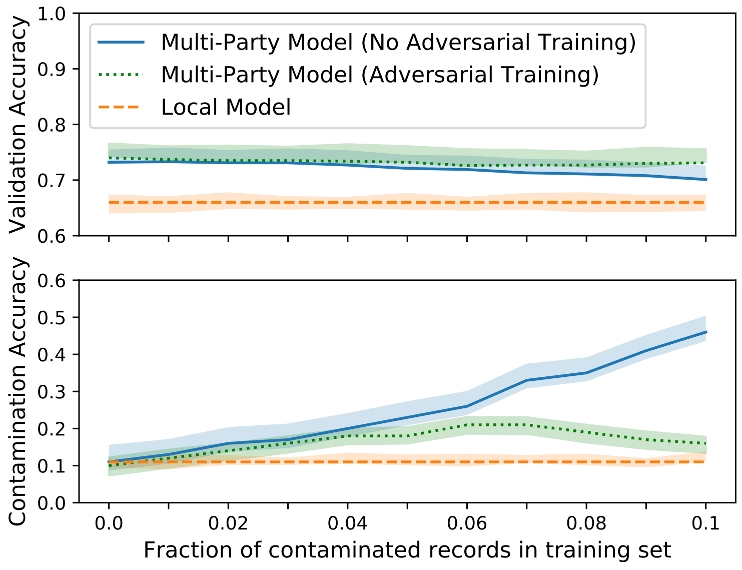


Figure C.1: Contamination and validation accuracy for the ADULT dataset as the number of contaminated records increases, for a contamination label of "high education level".

Figure C.1 shows results for a contamination attack, and the corresponding adversarial training defence, when the contamination label is chosen to be "high education level", and fixed the contamination attribute as described in section 6.2. Similar to fig. 6.2b, adversarial training mitigates the contamination attack, reducing contamination accuracy to a baseline local model level while retaining superior

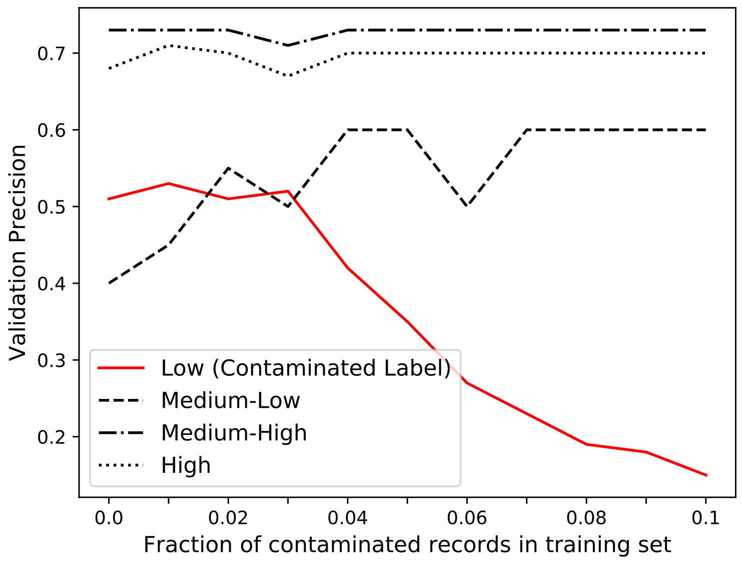


Figure C.2: Validation precision for each class label for the ADULT dataset.

validation accuracy over both a local and contaminated multi-party model. The adversarially trained multi-party model learns the connection with similar levels of accuracy to experiments with the "low education level" label and so the contamination attack is not dependent on the choice of class label.

C.2 Alternative mitigation strategies

Here, we outline several methods to defend against contamination attacks and their drawbacks.

Depending on the number of contaminated records used in the contamination attack, detection may be relatively straightforward. For example, if an attacker inserts a large number of contaminated records into the training set, the validation precision on the contaminated label may be significantly worse than on other labels. Figure C.2 shows this effect for the ADULT dataset, with the number of contaminated records set to 10% of the training set. However, we observed that for smaller numbers of contaminated records, the signal provided by the per label validation precision diminishes. Furthermore, this detection method does not provide information about the contaminated attribute, and we observed for prediction tasks with a larger number of classes, such as the NEWS20 dataset, the per label validation precision is not a reliable detection method, as there was a high variance of precision per label.

If one knows the attribute likely to be contaminated, or if there are a small number of attributes in the dataset, independence tests could detect if a party's dataset contains contaminated records. Given $n > 1$ parties and an attribute, each possible pair of parties can test the hypothesis "the distribution of an attribute between the two parties is independent of one another". This can be measured by a simple chi-square independence test. For the ADULT dataset, we performed an independence test on all possible pairs for the two cases: (1) when one of the parties was the attacker party and, (2) when both were victim parties. We found that when the attacker only modifies 1% of their data, the p-values are similar, both cases report p-values in 0.30-0.40 range and so reject the hypothesis of independence. However, if the attacker training set contains more than 5% contaminated records, the p-value in much lower than 0.05, and so the null hypothesis is accepted for case (1). If all data is expected to come from a similar distribution this could indicate the presence of contaminated records. However, the assumption of similar training data is unlikely to hold for a large number of use-cases. Moreover, this test is not applicable for text classification due to the sparsity of the feature set.

Finally, one may consider leave-one-party-out cross validation techniques to measure the utility of including a party's training data. Let there be $n > 1$ parties with one attacker party. To discover if a party is adversarial, a model is trained on $n - 1$ parties data, and evaluated on the training set of the left out party. If the left out party contains contaminated records, this should be discovered, since the model will report low accuracy on the left out party's data, having been trained on only clean records. Experimentally we found that if the amount of contaminated records is small, in the order of 1-8% of the size of the training set, the difference between accuracy on an attacker and victim training set is negligible. This method also requires training a new model for each party, which may be prohibitively expensive for a large dataset or larger numbers of parties.

C.3 Lemma 1 proof

Lemma 1. *For any random variables U, V and W, if $H(U|V) = 0$ then $H(W|V) \leq H(W|U)$.*

Proof. Using the definition of conditional entropy:

$$H(W|V) = H(V|W) + H(W) - H(V)$$
$$H(W|U) = H(U|W) + H(W) - H(U).$$

Combining the two we obtain $H(W|V) - H(W|U) = H(V|W) - H(U|W) + H(U) - H(V)$. Note that $H(V|U) = H(V) - H(U)$ since $H(U|V) = 0$ from the assumption of the lemma. Hence, we need to show that $H(V|W) - H(U|W) - H(V|U) \leq 0$ or, equivalently, $H(V|W) \leq H(U|W) + H(V|U)$ to prove the statement.

Let U' be a random variable that is independent of U, s.t., $H(V) = H(U) + H(U')$. Hence, $H(V|U) = H(U')$ and $H(V|U') = H(U)$. Then

$$H(V|W) = H(U|W) + H(U'|W) \leq H(U|W) + H(U') = H(U|W) + H(V|U).$$

$\square$

C.4 Multi-party attacker experiments on NEWS20 dataset

Due to the small size of the NEWS20 dataset, we found there was high variance between successive experiments with random partitions of the dataset into distinct parties. To counter this high variance, we introduce a nuisance word into the dataset that does not have predictive links with any labels and was not already present in the dataset. We split the dataset into five parties, and one validation dataset, and introduce the new word such that it covers 5% of each parties data (and covers 10% of the attacker controlled data). We introduce the word into 50% of data points in the validation set so we can accurately capture the effect of the attack. Figure C.3 models this attack as we increase the number of attacker controlled parties. Note

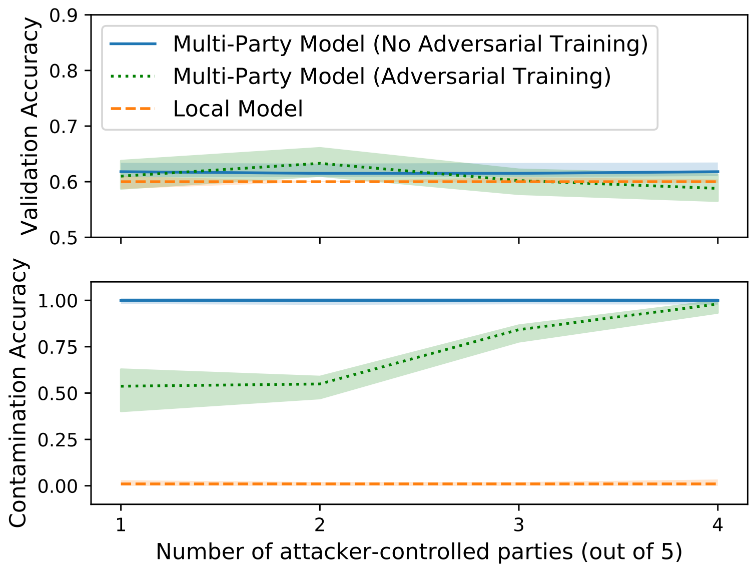


Figure C.3: Contamination and validation accuracy for the NEWS20 dataset as the number of contaminated records increases.

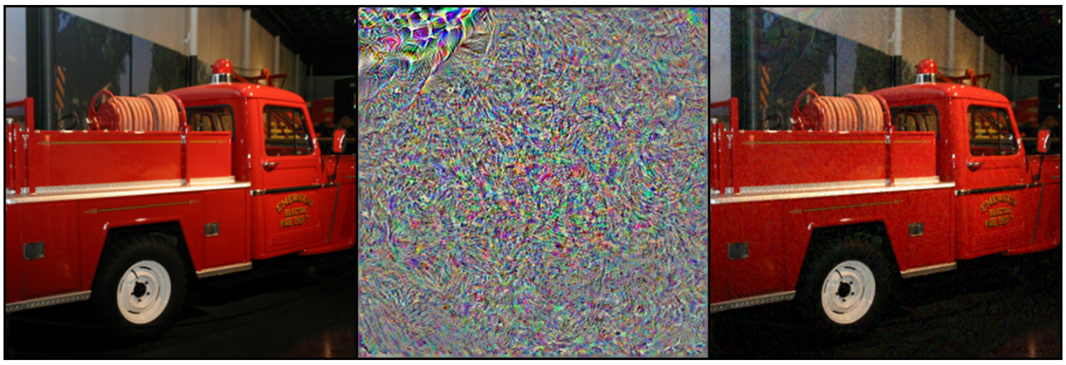
we use the f'' method during training.

Appendix D

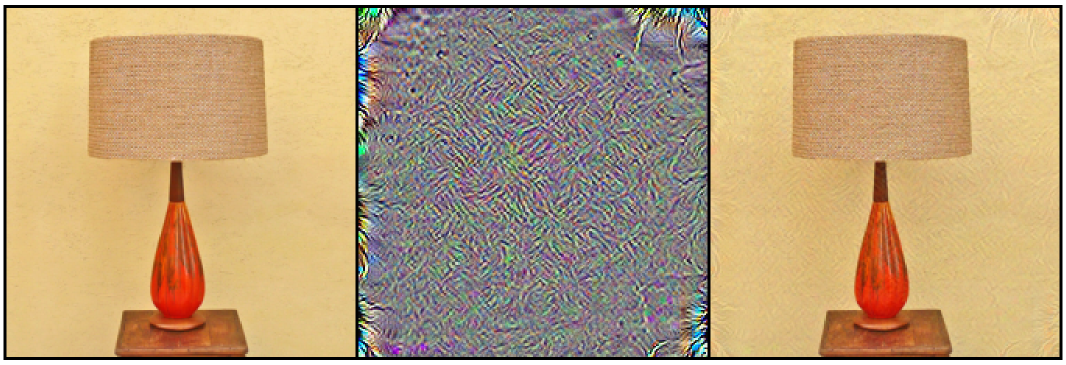
Learning universal adversarial perturbations with generative models

D.1 A note on recent concurrent work

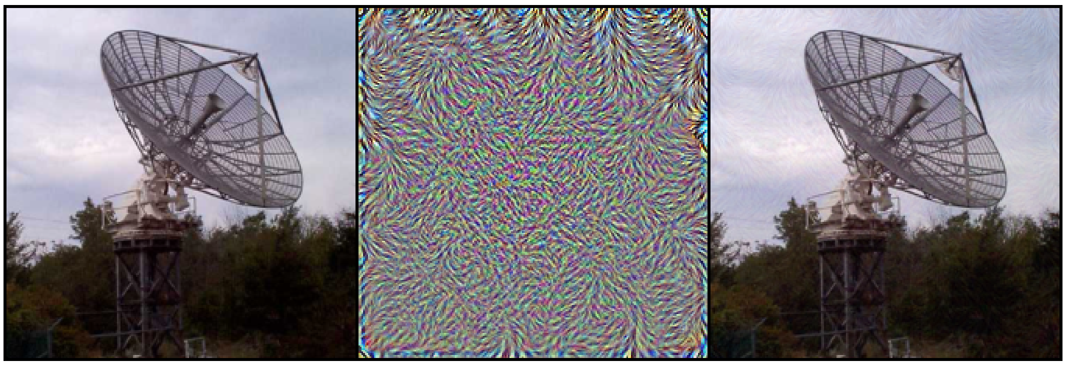
We are unaware of any previous work that studies the relationship between generative models and universal adversarial perturbations. However, we note that two recent studies [200, 207] also craft perturbations using generative models. Poursaeed *et al.* [200] have a similar set-up to our attack, optimising under both ℓ_2 and ℓ_∞ metrics, however, they did not include a distance minimisation term within the objective function, instead relying a scaling factor before applying the perturbation to an image. The Mopuri *et al.* [207] attack is only optimised using the ℓ_∞ metric. They also eschew a distance minimisation term, and instead include a diversity term within the objective function, so that the objective does not get stuck in a local minima resulting in a limited number of effective perturbations. We were unable to obtain source code for Mopuri *et al.'s* [207] attack and were unsuccessful in replicating results, and so we report comparison in results only on the target models that were shared in both pre-prints, on ImageNet (since other works only report results on this dataset for the task of generating UAPs) using the ℓ_∞ metric (see table D.1).



(a) Inception-V3: Fire engine (54.6%), δ', Wrecker (79.4%)



(b) ResNet-152: Table lamp (87.2%), δ', Tabby cat (41.9%)



(c) VGG-19: Radio telescope (97.5%), δ', Great Pyrenees (36.7%)

Figure D.1: Selection of successful adversarial examples (with target model confidence) from non-targeted ℓ_∞ attacks on ImageNet. From left to right: Source image, UAP, adversarial image.

Table D.1: Error rates for non-targeted ℓ_∞ attacks on ImageNet.

	VGG-19	INCEPTION-V1 [228]
UAN	0.846	0.809
Poursaeed *et al.* [200]	0.801	0.792
Mopuri *et al.* [207]	0.838	0.904

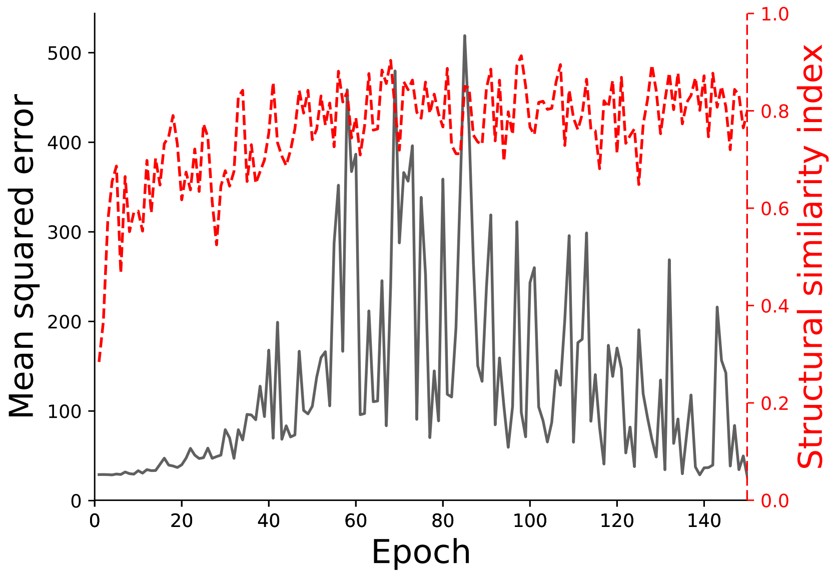


Figure D.2: MSE and SSIM scores of UAPs throughout training a UAN against VGG-19 for the CIFAR-10 dataset.

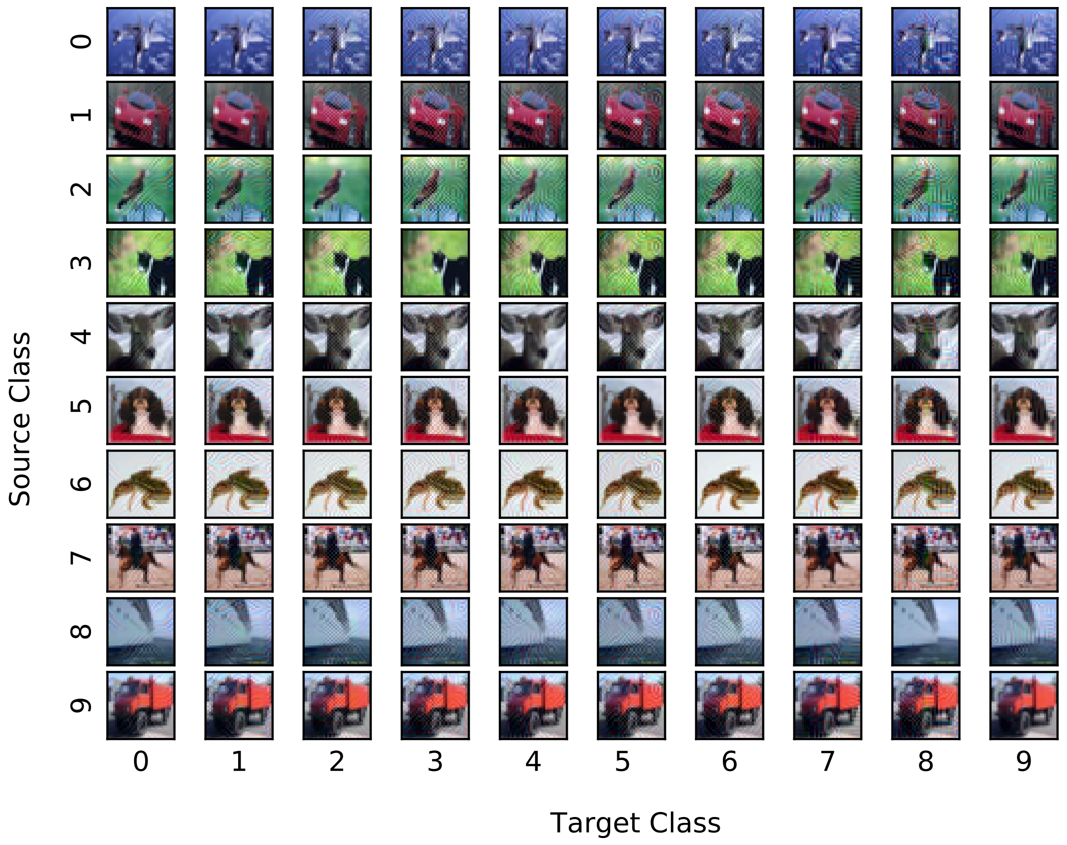


Figure D.3: Our ℓ_∞ attack against a VGG-19 target model on the CIFAR-10 dataset, for every source/target pair. Displayed images were selected at random.

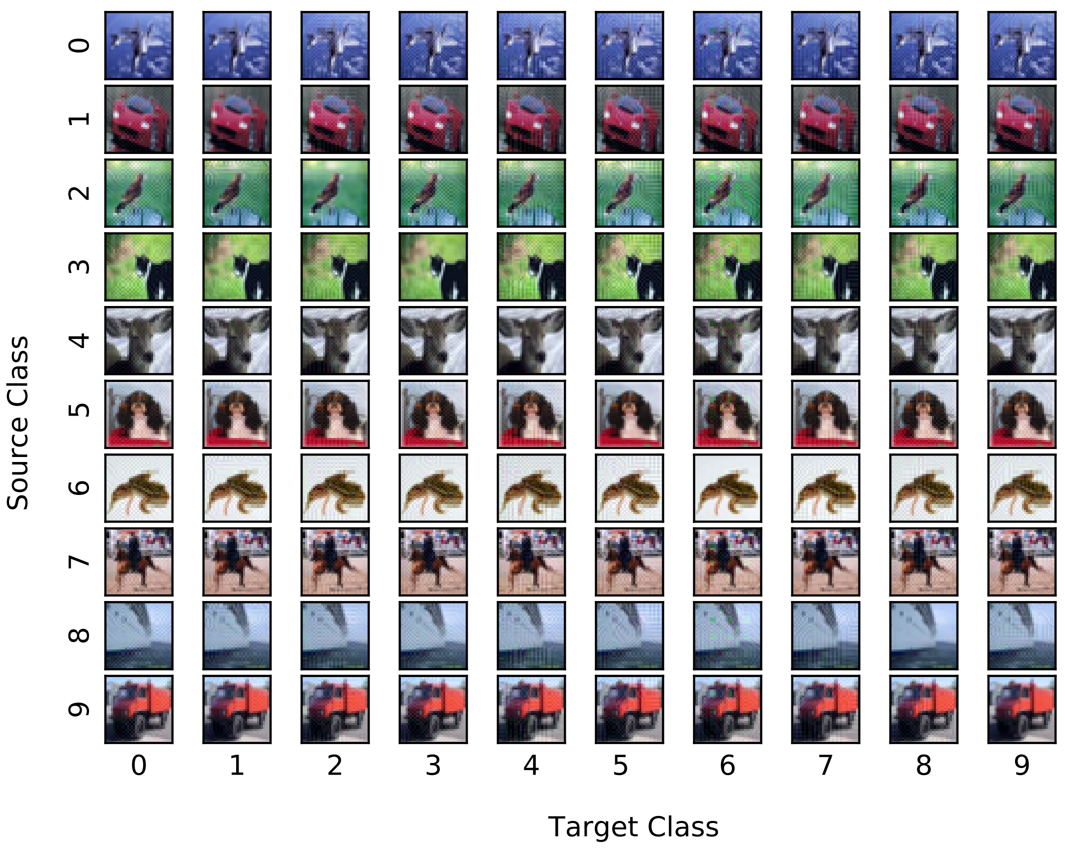
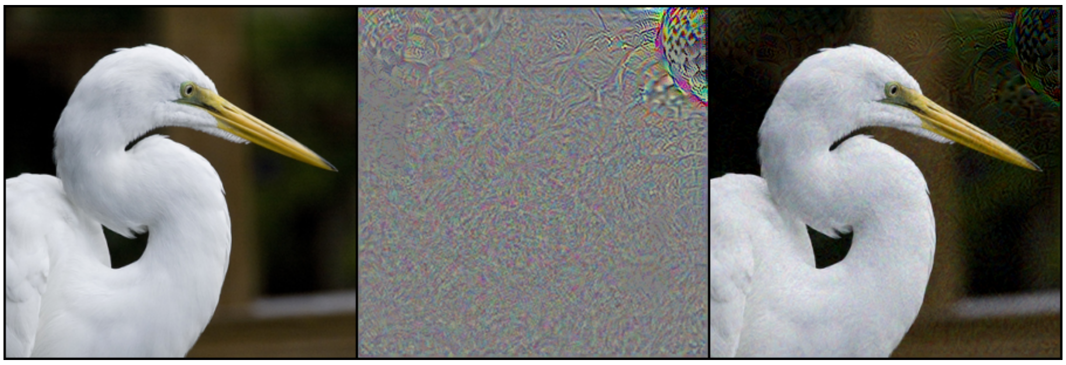
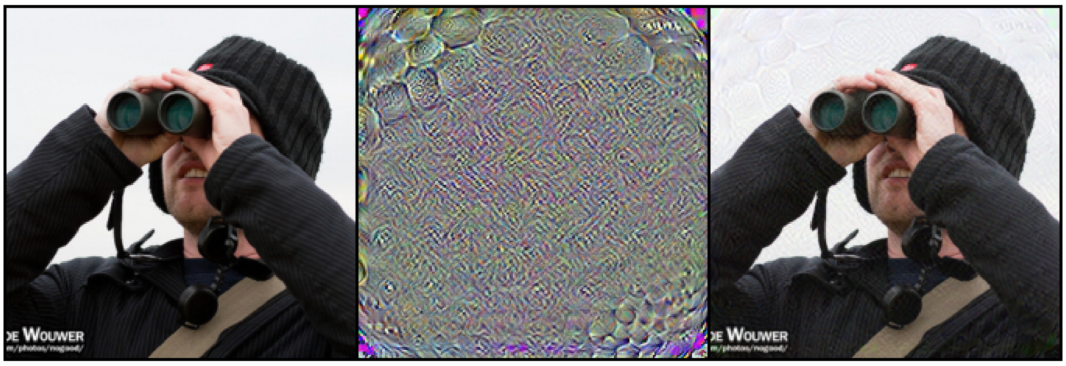


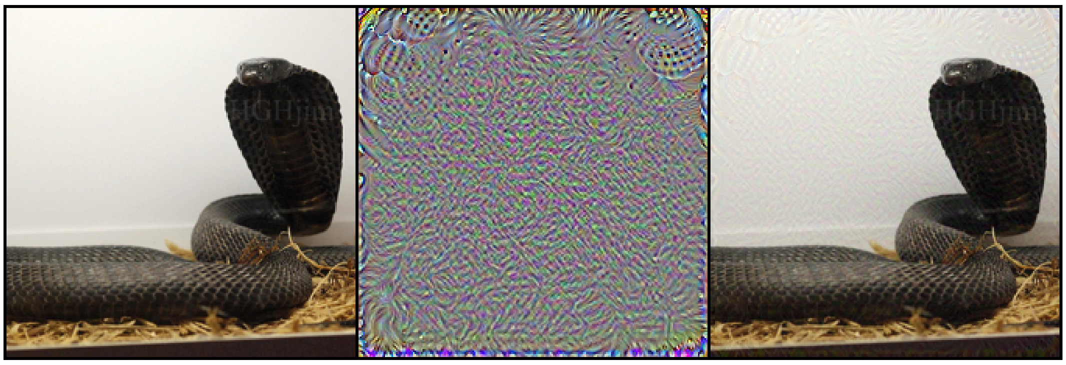
Figure D.4: Our ℓ_∞ attack against a ResNet-101 target model on the CIFAR-10 dataset, for every source/target pair. Displayed images were selected at random.



(a) Inception-V3: American egret (95.0%), δ', Golf ball (98.8%). Overall target model error rate: 0.654

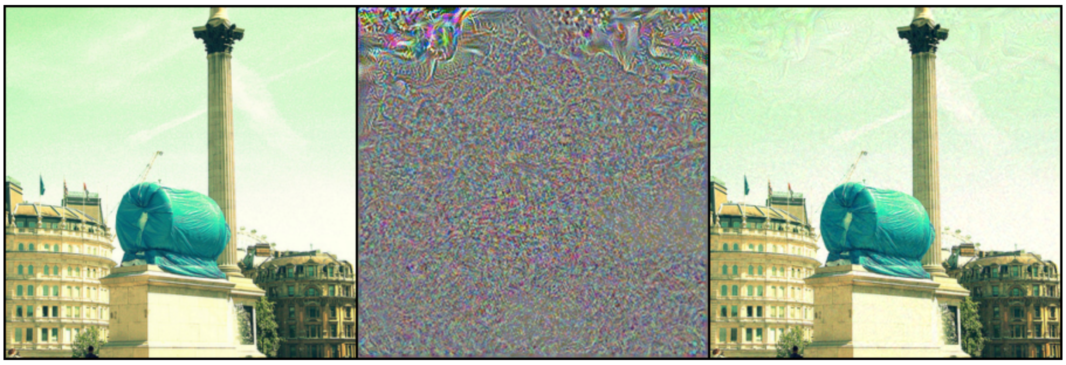


(b) ResNet-152: Binoculars (99.9%), δ', Golf ball (62.9%). Overall target model error rate: 0.734

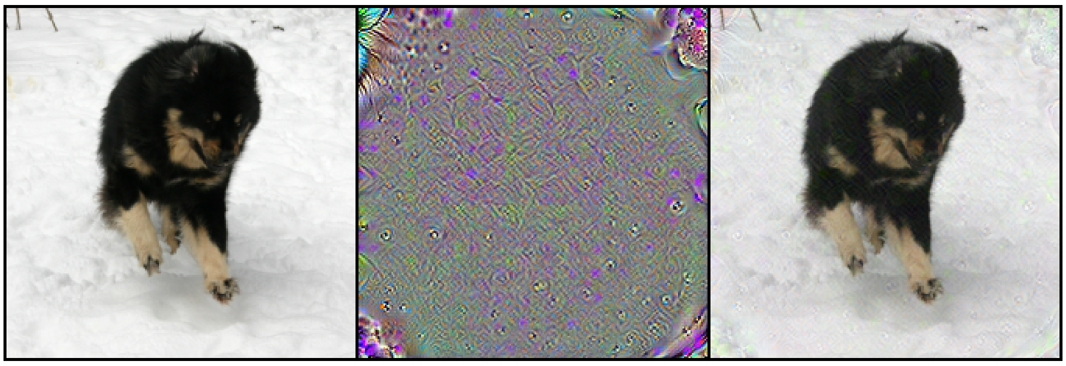


(c) VGG-19: Indian cobra (99.9%), δ', Golf ball (99.7%). Overall target model error rate: 0.514

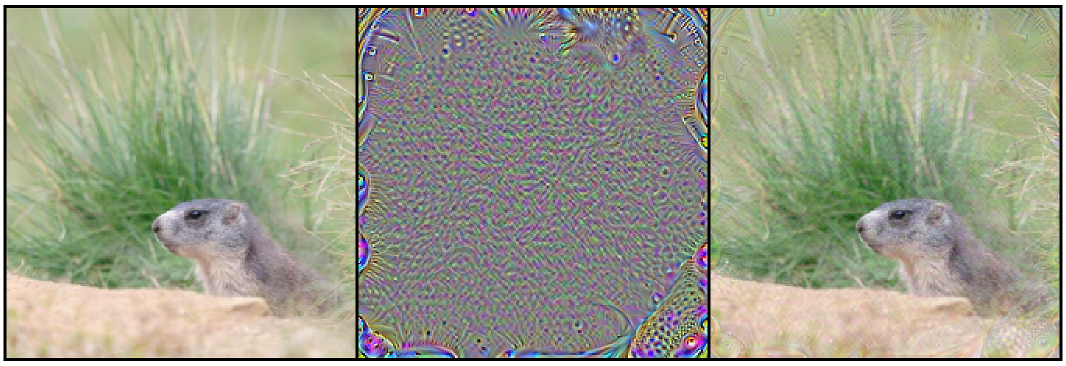
Figure D.5: Selection of successful adversarial examples (with target model confidence) for targeted ℓ_∞ attacks on ImageNet. The target class was randomly chosen to be *Golf ball*. From left to right: Source image, UAP, adversarial image.



(a) Inception-V3: Pedestal (98.4%), δ', Broccoli (88.7%). Overall target model error rate: 0.598

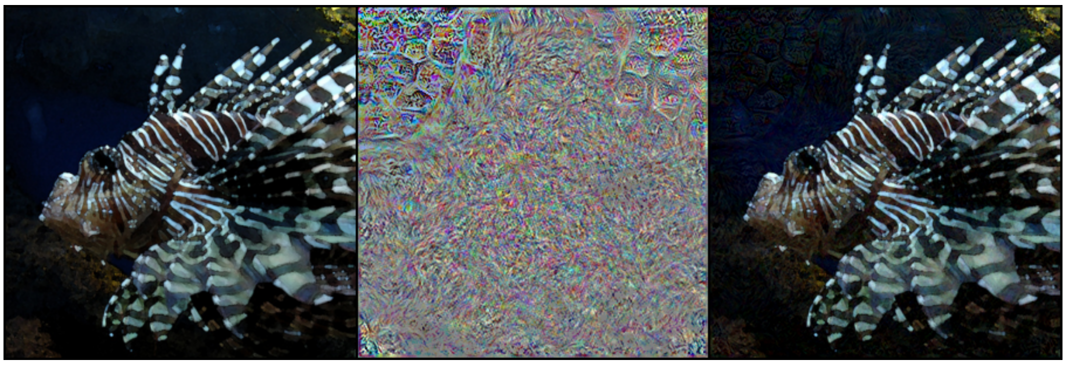


(b) ResNet-152: Tibetan mastiff (88.4%), δ', Broccoli (98.1%). Overall target model error rate: 0.691

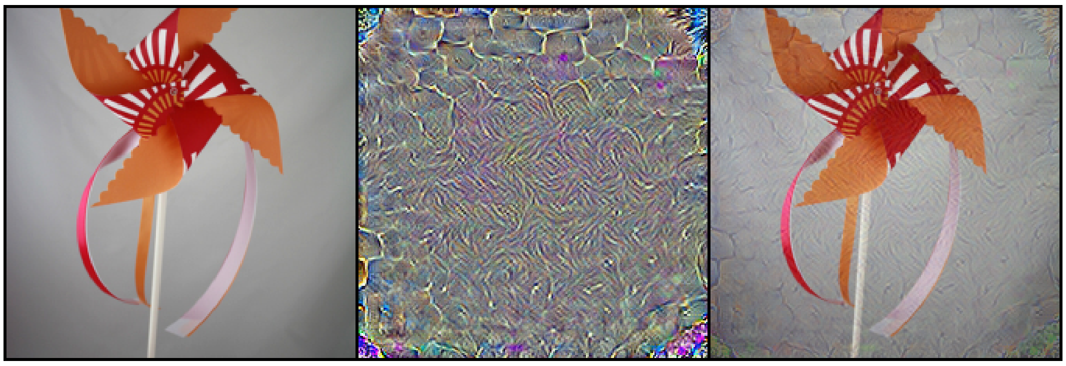


(c) VGG-19: Marmot (95.4%), δ', Broccoli (48.4%). Overall target model error rate: 0.480

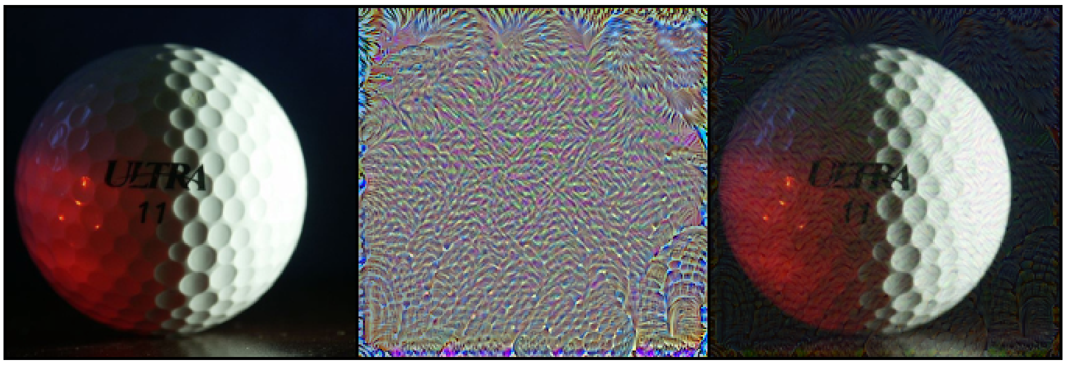
Figure D.6: Selection of successful adversarial examples (with target model confidence) for targeted ℓ_∞ attacks on ImageNet. The target class was randomly chosen to be *Broccoli*. From left to right: Source image, UAP, adversarial image.



(a) Inception-V3: Lionfish (89.7%), δ', Stone wall (54.0%). Overall target model error rate: 0.533



(b) ResNet-152: Pinwheel (99.9%), δ', Stone wall (47.0%). Overall target model error rate: 0.587



(c) VGG-19: Golf ball (99.9%), δ', Stone wall (23.7%). Overall target model error rate: 0.447

Figure D.7: Selection of successful adversarial examples (with target model confidence) for targeted ℓ_∞ attacks on ImageNet. The target class was randomly chosen to be *Stone wall*. From left to right: Source image, UAP, adversarial image.

Appendix E

Randomised smoothing: A provable defence against adversarial examples

E.1 Lower bounds for common divergences between multinomial distributions

Firstly, we present the statement of the Rényi divergence bound given in Li et al. [146], and provide a full proof.

Theorem 2. *Let $P = (p_1, ..., p_k)$ and $Q = (q_1, ..., q_k)$ be two multinomial distributions over the same index set $\{1, ..., k\}$. If the indexes of the largest probabilities do not match on P and Q, that is $\arg\max_i p_i \neq \arg\max_j q_j$, then*

$$d_\alpha(Q,P) \geq -\log\left(1 - p_1 - p_2 + \left(\frac{1}{2}(p_1^{1-\alpha} + p_2^{1-\alpha})\right)^{\frac{1}{1-\alpha}}\right) \tag{E.1}$$

where p_1 and p_2 are the first and second largest probabilities in P.

Proof. We can think of this problem as finding the distribution Q that minimises $d_\alpha(Q,P)$ such that $\arg\max_i p_i \neq \arg\max_j q_j$ for a fixed $P = (p_1, ..., p_k)$. Without loss of generality, assume $p_1 \geq p_2 \geq ... \geq p_k$.

This is equivalent to solving

$$\min_{\sum q_i, \arg\max q_i \neq 1} \frac{1}{1-\alpha} \log\left(\sum_1^k p_i \left(\frac{q_i}{p_i}\right)^\alpha\right) \tag{E.2}$$

As the logarithm is a monotonically increasing function, we only focus on the quantity $s(Q,P) = \sum_1^k p_i(\frac{q_i}{p_i})^\alpha$ for a fixed α.

We first show for the Q that minimises $s(Q,P)$, it must have $q_1 = q_2 \geq q_3 \geq ... \geq q_k$. Note here we allow a tie, because we can always let $q_1 = q_1 - \kappa$ and $q_2 = q_2 + \kappa$ for some small κ to satisfy $\arg\max q_i \neq 1$ while not changing the Rényi divergence too much by the continuity of $s(Q,P)$.

If $q_j > q_i$ for some $j \geq i$, we can define Q' by mutating q_i and q_j, that is $Q' = (q_1, ..., q_{i-1}, q_j, q_{i+1}, ..., q_{j-1}, q_i, q_{j+1}, ..., q_k)$, then

$$S(Q,P) - S(Q',P) = p_i\Big(\frac{q_i^\alpha - q_j^\alpha}{p_i^\alpha}\Big) + p_j\Big(\frac{q_j^\alpha - q_i^\alpha}{p_j^\alpha}\Big) \tag{E.3}$$

$$= (p_i^{1-\alpha} - p_j^{1-\alpha})(q_i^\alpha - q_j^\alpha) > 0 \tag{E.4}$$

which conflicts with the assumption that Q minimises $s(Q,P)$. Thus $q_i \geq q_j$ for $j \geq i$. Since q_1 cannot be the largest, we have $q_1 = q_2 \geq q_3 \geq ... \geq q_k$.

Then we are able to assume $Q = (q_0, q_0, q_3, ..., q_k)$, and the problem can be formulated as

$$\min_{q_0, q_3, \cdots, q_k} p_1(\frac{q_0}{p_1})^\alpha + p_2(\frac{q_0}{p_2})^\alpha + \sum_{i=3}^k p_i(\frac{q_i}{p_i})^\alpha \tag{E.5}$$

$$\text{such that} \quad 2q_0 + \sum_{i=3}^k q_i = 1 \tag{E.6}$$

$$\text{such that} \quad q_i - q_0 \leq 0 \quad i \geq 3 \tag{E.7}$$

$$\text{such that} \quad -q_i \leq 0 \quad i \geq 0 \tag{E.8}$$

which forms a set of KKT conditions. Let L denote the Lagrangian formulation of the problem

$$p_1(\frac{q_0}{p_1})^\alpha + p_2(\frac{q_0}{p_2})^\alpha + \sum_{i=3}^k p_i(\frac{q_i}{p_i})^\alpha + \lambda(2q_0 + \sum_{i=3}^k q_i - 1) + \sum_{i=3}^k \mu_i(q_i - q_0) - \sum_{i=3}^k \beta_i q_i \tag{E.9}$$

Setting slack variables to zero and differentiating gives

$$\frac{\partial L}{\partial q_0} = \alpha q_0^{\alpha-1}(p_1^{1-\alpha} + p_2^{1-\alpha}) + 2\lambda = 0 \tag{E.10}$$

$$\frac{\partial L}{\partial q_i} = \alpha(\frac{q_i}{p_i})^{\alpha-1} + \lambda = 0 \quad i \geq 3 \tag{E.11}$$

Equation (E.10) and eq. (E.11) imply

$$q_0 = \left(\frac{-2\lambda}{\alpha(p_1^{1-\alpha} + p_2^{1-\alpha})}\right)^{\frac{1}{\alpha-1}} \tag{E.12}$$

$$q_i = \left(-\frac{\lambda}{\alpha}\right)^{\frac{1}{\alpha-1}} p_i \quad i \geq 3 \tag{E.13}$$

From the restriction that $2q_0 + \sum_{i=3}^k q_i = 1$ it follows that

$$\lambda = \frac{-\alpha}{\left(2\left(\frac{1}{2}(p_1^{1-\alpha} + p_2^{1-\alpha})\right)^{\frac{1}{1-\alpha}} + 1 - p_1 - p_2\right)^{\alpha-1}} \tag{E.14}$$

Let $\eta = \left(\frac{p_1^{1-\alpha} + p_2^{1-\alpha}}{2}\right)^{\frac{1}{1-\alpha}}$. Then it follows that

$$q_0 = \frac{a}{2\eta + 1 - p_1 - p_2} \tag{E.15}$$

$$q_i = \frac{p_i}{2\eta - 1 - p_1 - p_2} \quad i \geq 3 \tag{E.16}$$

Using eq. (E.15) and eq. (E.16), Rényi divergence is minimised at

$$\frac{1}{1-\alpha}\log\left(p_1(\frac{q_0}{p_1})^\alpha + p_2(\frac{q_0}{p_2})^\alpha + \sum_{i=3}^{k} p_i(\frac{q_i}{p_i})^\alpha\right) \tag{E.17}$$

$$= \frac{1}{1-\alpha}\log\left(\frac{2\eta^{1-\alpha}\eta^\alpha}{(2\eta+1-p_1-p_2)^\alpha} + \frac{1-p_1-p_2}{(2\eta+1-p_1-p_2)^\alpha}\right) \tag{E.18}$$

$$= -\log(2\eta+1-p_1-p_2) \tag{E.19}$$

□

To find the certified area of robustness of an input using the KL divergence of the generalised Gaussian, we can make use of the following theorem.

Theorem 3. *Let $P = (p_1,...,p_k)$ and $Q = (q_1,...,q_k)$ be two multinomial distributions over the same index set $1,...,k$. If the indexes of the largest probabilities do not match on P and Q, that is $\arg\max_i p_i \neq \arg\max_j q_j$, then*

$$d_{KL}(Q,P) \geq -\log(2\sqrt{p_1 p_2} + 1 - p_1 - p_2) \tag{E.20}$$

where p_1 and p_2 are the first and second largest probabilities in P.

Proof. Using the same terminology as Theorem 2, the problem can be stated as a set of KKT conditions given by

$$\min_{q_0,q_3,...,q_k} q_0\log(\frac{q_0}{p_1}) + q_0\log(\frac{q_0}{p_2}) + \sum_{i=3}^{k} q_i\log(\frac{q_i}{p_i}) \tag{E.21}$$

$$\text{such that} \quad 2q_0 + \sum_{i=3}^{k} q_i = 1 \tag{E.22}$$

$$\text{such that} \quad q_i - q_0 \leq 0 \quad i \geq 3 \tag{E.23}$$

$$\text{such that} \quad -q_i \leq 0 \quad i \geq 0 \tag{E.24}$$

Let L denote

$$p_1 \log(\frac{q_0}{p_1}) + p_2 \log(\frac{q_0}{p_2}) + \sum_{i=3}^{k} p_i \log(\frac{q_i}{p_i}) + \tag{E.25}$$

$$\lambda(2q_0 + \sum_{i=3}^{k} q_i - 1) + \sum_{i=3}^{k} \mu_i(q_i - q_0) - \sum_{i=3}^{k} \beta_i q_i \tag{E.26}$$

Setting slack variables to zero and differentiating gives

$$\frac{\partial L}{\partial q_0} = \log(\frac{q_0}{p_1}) + \log(\frac{q_0}{p_2}) + 2\lambda + 2 = 0 \tag{E.27}$$

$$\frac{\partial L}{\partial q_i} = \log(\frac{q_i}{p_i}) + \lambda + 1 = 0 \quad i \geq 3 \tag{E.28}$$

Combining eq. (E.27) and eq. (E.28) with the KKT conditions and solving for λ gives

$$q_0 = \frac{\sqrt{p_1 p_2}}{\eta} \tag{E.29}$$

$$q_i = \frac{p_i}{\eta} \qquad i \geq 3 \tag{E.30}$$

$$\tag{E.31}$$

where $\eta = 2\sqrt{p_1 p_2} + 1 - p_1 - p_2$. The minimised KL divergence is therefore $-\log \eta$.

□

Theorem 4. *Let $P = (p_1, ..., p_k)$ and $Q = (q_1, ..., q_k)$ be two multinomial distributions over the same index set $1, ..., k$. If the indexes of the largest probabilities do not match on P and Q, that is $\arg\max_i p_i \neq \arg\max_j q_j$, then*

$$d_{H^2}(Q,P) \geq 1 - \sqrt{\frac{2 - (\sqrt{p_1} - \sqrt{p_2})^2}{2}} \tag{E.32}$$

where p_1 and p_2 are the first and second largest probabilities in P.

Proof. Using the same technique and terminology as in Theorem 2, we find that

$$q_0 = \frac{(\sqrt{p_1}+\sqrt{p_2})^2}{2\eta} \tag{E.33}$$

$$q_i = \frac{2p_i}{\eta} \qquad i \geq 3, \tag{E.34}$$

$$\tag{E.35}$$

where $\eta = 2-(\sqrt{p_1}-\sqrt{p_2})^2$. The minimised Hellinger distance is therefore $1-\sqrt{\frac{\eta}{2}}$.

□

Theorem 5. *Let $P = (p_1,...,p_k)$ and $Q = (q_1,...,q_k)$ be two multinomial distributions over the same index set $1,...,k$. If the indexes of the largest probabilities do not match on P and Q, that is $\arg\max_i p_i \neq \arg\max_j q_j$, then*

$$d_{\chi^2}(Q,P) \geq \frac{(p_1-p_2)^2}{(p_1+p_2)-(p_1-p_2)^2} \tag{E.36}$$

where p_1 and p_2 are the first and second largest probabilities in P.

Proof. Using the same technique and terminology as in Theorem 2, we find that

$$q_0 = \frac{2p_1p_2}{\eta} \tag{E.37}$$

$$q_i = \frac{p_1+p_2}{\eta}p_i \qquad i \geq 3, \tag{E.38}$$

$$\tag{E.39}$$

where $\eta = (p_1+p_2)-(p_1-p_2)^2$. The minimised chi-squared distance is therefore $\frac{(p_1-p_2)^2}{\eta}$.

□

Theorem 6. *Let $P = (p_1, ..., p_k)$ and $Q = (q_1, ..., q_k)$ be two multinomial distributions over the same index set $1, ..., k$. If the indexes of the largest probabilities do not match on P and Q, that is $\arg\max_i p_i \neq \arg\max_j q_j$, then*

$$d_B(Q,P) \geq -\log(\sqrt{\frac{2\sqrt{p_1p_2} - p_1 - p_2 + 2}{2}}) \tag{E.40}$$

where p_1 and p_2 are the first and second largest probabilities in P.

Proof. Using the same technique and terminology as in Theorem 2, we find that

$$q_0 = \frac{(\sqrt{p_1} + \sqrt{p_2})^2}{2\eta} \tag{E.41}$$

$$q_i = \frac{2p_i}{\eta} \qquad i \geq 3, \tag{E.42}$$

$$\tag{E.43}$$

where $\eta = 2\sqrt{p_1p_2} - p_1 - p_2 + 2$. The minimised Bhattacharyya distance is therefore $-log(\sqrt{\frac{\eta}{2}})$.

□

Theorem 7. *Let $P = (p_1, ..., p_k)$ and $Q = (q_1, ..., q_k)$ be two multinomial distributions over the same index set $1, ..., k$. If the indexes of the largest probabilities do not match on P and Q, that is $\arg\max_i p_i \neq \arg\max_j q_j$, then*

$$d_{TV}(Q,P) \geq \frac{|p_1 - p_2|}{2} \tag{E.44}$$

where p_1 and p_2 are the first and second largest probabilities in P.

Proof. It is easy to see that $d_{TV}(Q,P)$ is minimised when $q_1 = q_2 = \frac{|p_1+p_2|}{2}$ and $q_i = p_i$ for $i >= 3$. This leads to the stated lower bound.

□

Interestingly, d_{TV} appears naturally in the certificates found via randomised smoothing, as a consequences of being a special case of the hockey-stick divergence. Indeed, consider a binary classification task, with a given probabilistic classifier, f, and an input x. Let f_c denote the classifier's output at label c, which is the true label of x. Let $\mu = \mu(x)$ denote the smoothing measure on input x, and $\nu = \mu(x')$ denote the smoothing measure on input x', with a defined distance metric d such that $d(\mu, \nu) < \varepsilon$. Then we can guarantee f outputs the same prediction on x' as on x if the following is larger than $1/2$

$$\min_{f_c} \mathop{\mathbb{E}}_{X\sim\nu}[f_c(X)] \qquad \text{subject to} \qquad \mathop{\mathbb{E}}_{X\sim\mu}[f_c(X)] = p_1, 0 \le f_c(x) \le 1 \tag{E.45}$$

The dual relaxation of this problem is given by

$$\max_{\lambda} \lambda p_1 + \min_{0\le f_c \le 1} \mathop{\mathbb{E}}_{X\sim\nu}[f_c(X)] - \lambda \mathop{\mathbb{E}}_{X\sim\mu}[f_c(X)] \tag{E.46}$$

The inner minimisation term is commonly referred to as the hockey-stick divergence. Since any λ gives a valid lower bound bound to the primal problem, setting $\lambda = 1$ gives

$$\begin{aligned}
& p_1 - \max_{0\le f_c\le 1} \mathop{\mathbb{E}}_{X\sim\mu}[f_c(X)] - \mathop{\mathbb{E}}_{X\sim\nu}[f_c(X)] && \text{(E.47)}\\
\ge\, & p_1 - \max_{0\le f_c\le 1} |\int_{\mathscr{X}} f_c d(\mu - \nu)| && \text{(E.48)}\\
\ge\, & p_1 - \max_{0\le f_c\le 1} \int_{\mathscr{X}} |f_c| d|\mu - \nu| && \text{(E.49)}\\
\ge\, & p_1 - \int_{\mathscr{X}} d|\mu - \nu| && \text{(E.50)}\\
\ge\, & p_1 - d_{TV}(\mu, \nu) && \text{(E.51)}
\end{aligned}$$

Thus, the classifier predicts the same label on x' as on x if $p_1 - d_{TV}(\mu, \nu) > 1/2$.

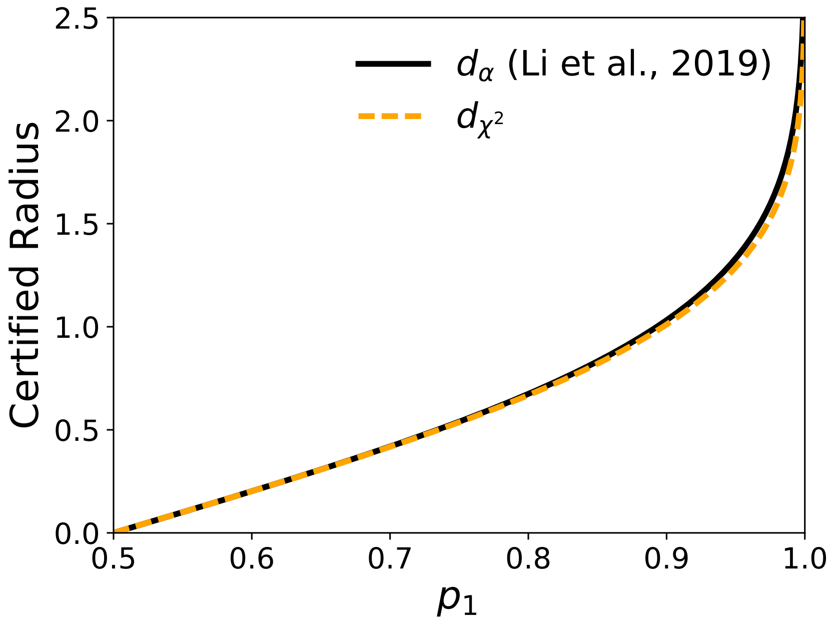


Figure E.1: Comparison of the certified radius against perturbations targeting the ℓ_2 norm, for Rényi divergence (d_α) and the chi-squared distance (d_{χ^2}), as a function of the top predicted probability, p_1, with $\sigma = 1$.

E.2 Visualisation of certified radius (for ℓ_2 perturbations) found by d_α and d_{χ^2}

Figure E.1 visualises the trade-off in certified radius around an input for a hypothetical binary classification task as a function of the classifier's top output probability, p_1. The certified radii are found using the Rényi divergence and chi-squared distance. The difference between these two certified radii is small; for $p_1 \leq 0.99$, the largest difference between the two radii is 0.1.

E.3 Proof of proposition 2

Proof. We prove this for the binary case where $p_2 = 1 - p_1$.

1. Let us fix $\alpha \in (1,\infty)$. Then $\varepsilon_{d_\alpha} > \varepsilon_{d_{\chi^2}}$ when

$$\sqrt{-\frac{2\sigma^2}{\alpha}\log\left(2\Big(\frac{1}{2}(p_1^{1-\alpha}+(1-p_1)^{1-\alpha})\Big)^{\frac{1}{1-\alpha}}\right)} > \sqrt{\sigma^2\log(\frac{1}{4p_1(1-p_1)})} \tag{E.52}$$

$$\iff -\frac{2}{\alpha}\log\left(2\Big(\frac{1}{2}(p_1^{1-\alpha}+(1-p_1)^{1-\alpha})\Big)^{\frac{1}{1-\alpha}}\right) > \log(\frac{1}{4p_1(1-p_1)}) \tag{E.53}$$

$$\iff 4\Big(\frac{1}{2}(p_1^{1-\alpha}+(1-p_1)^{1-\alpha})\Big)^{\frac{2}{1-\alpha}} < (4p_1(1-p_1))^\alpha \tag{E.54}$$

This holds $\forall p_1 \in (\frac{1}{2},1)$ for example when $\alpha = 1.1$ and so automatically holds for $\alpha \in (1,\infty)$ that maximises the expression.

2. If $\varepsilon_{d_{\chi^2}} > \varepsilon_{d_{KL}}$, then

$$\sqrt{\sigma^2\log(\frac{1}{4p_1(1-p_1)})} > \sqrt{-\sigma^2\log(2\sqrt{p_1(1-p_1)})} \tag{E.55}$$

$$\iff \frac{1}{4p_1(1-p_1)} > \frac{1}{2\sqrt{p_1(1-p_1)}} \tag{E.56}$$

$$\iff (p_1-\frac{1}{2})^2 > 0 \tag{E.57}$$

$$\implies p_1 > \frac{1}{2} \tag{E.58}$$

3. If $\varepsilon_{d_{\chi^2}} > \varepsilon_{d_{H^2}}$, then

$$\sqrt{\sigma^2\log(\frac{1}{4p_1(1-p_1)})} > \sqrt{-8\sigma^2\log(\sqrt{\frac{2-\sqrt{p_1(1-p_1)}}{2}})} \tag{E.59}$$

$$\iff \frac{1}{4p_1(1-p_1)} > \frac{2^4}{(1+2\sqrt{p_1(1-p_1)})^4} \tag{E.60}$$

$$\iff (1+2\sqrt{p_1(1-p_1)})^4 > 2^6 p_1(1-p_1) \tag{E.61}$$

$$\implies p_1 > \frac{1}{2} \tag{E.62}$$

$$\tag{E.63}$$

4. We show the inner logarithmic terms in $\varepsilon_{d_{H^2}}$ and ε_{d_B} are equal, which suffices to

prove equality in general. The inner logarithmic term of $\varepsilon_{d_{H^2}}$ is

$$\sqrt{\frac{1+2\sqrt{p_1(1-p_1)}}{2}} \tag{E.64}$$

$$=\frac{1+2\sqrt{p_1(1-p_1)}}{\sqrt{2(1+2\sqrt{p_1(1-p_1)})}} \tag{E.65}$$

$$=\frac{(\sqrt{p_1}+\sqrt{1-p_1})^2}{\sqrt{2(1+2\sqrt{p_1(1-p_1)})}} \tag{E.66}$$

The last term is equal to inner logarithmic term in ε_{d_B} and so we have $\varepsilon_{d_{H^2}} = \varepsilon_{d_B}$.

5. If $\varepsilon_{d_{H^2}} > \varepsilon_{d_{KL}}$, then

$$\sqrt{-8\sigma^2\log(\sqrt{\frac{2-\sqrt{p_1(1-p_1)}}{2}})} > \sqrt{\sigma^2\log(2\sqrt{p_1(1-p_1)})} \tag{E.67}$$

$$\iff 2^5\sqrt{p_1(1-p_1)} > (1+2\sqrt{p_1(1-p_1)})^4 \tag{E.68}$$

$$\tag{E.69}$$

This last term has solutions in $p_1 \in (\frac{1}{2}, 0.998)$.

6. Let us fix $\beta \in (0, \min(1, \frac{1}{2}\log(\frac{p_1}{1-p_1}))]$, then $\varepsilon_{d_{KL}} > \varepsilon_{\text{Lecuyer et al. [142]}}$ when

$$\sqrt{-\sigma^2\log(2\sqrt{p_1(1-p_1)})} > \frac{\sigma\beta}{\sqrt{2\log(\frac{1.25(1+e^{\beta})}{p_1(1+e^{2\beta})-e^{2\beta}})}} \tag{E.70}$$

$$\iff \beta^2 + 2\log(\frac{1.25(1+e^{\beta})}{p_1(1+e^{2\beta})-e^{2\beta}})\log(2\sqrt{p_1(1-p_1)}) < 0 \tag{E.71}$$

$$\tag{E.72}$$

This last term holds for any $p \in (\frac{1}{2}, 1)$.

□

E.4 KL divergence of the generalised Gaussian distribution

Here, we give a proof of the claim stated in eq. (8.13).

Theorem 8. *Let p_1 and p_2 be the pdf's of two generalised Gaussians with parameters (μ_1, σ, s) and (μ_2, σ, s) where s is a positive integer, respectively. Then $d_{KL}(p_1, p_2)$ is given by*

$$\sum_{k=1}^{s} \binom{s}{k} \frac{(1+(-1)^{s-k})\Gamma(\frac{s-k+1}{s})(\mu_1-\mu_2)^k}{2\sigma^k\Gamma(\frac{1}{s})} \tag{E.73}$$

Proof.

$$d_{KL}(p_1,p_2) = \sum p_1 \log\left(\frac{p_1}{p_2}\right) \tag{E.74}$$

$$= \sum k_1 e^{-|\frac{x-\mu_1}{\sigma}|^s} \log\left(\frac{k_1 e^{-|\frac{x-\mu_1}{\sigma}|^s}}{k_2 e^{-|\frac{x-\mu_2}{\sigma}|^s}}\right) \tag{E.75}$$

Where $k_1 = k_2 = \frac{s}{2\sigma\Gamma(\frac{1}{s})}$. Thus eq. (E.75) is equal to

$$\sum k_1 e^{-|\frac{x-\mu_1}{\sigma}|^s} \log\left(\frac{e^{-|\frac{x-\mu_1}{\sigma}|^s}}{e^{-|\frac{x-\mu_2}{\sigma}|^s}}\right) \tag{E.76}$$

$$= \mathbb{E}_{p_1}\left[\left(\frac{x-\mu_2}{\sigma}\right)^s - \left(\frac{x-\mu_1}{\sigma}\right)^s\right] \tag{E.77}$$

$$= \frac{1}{\sigma^s}\mathbb{E}_{p_1}\left[(x-\mu_2)^s - (x-\mu_1)^s\right] \tag{E.78}$$

Note that $(x-\mu_2)^s = \sum_{k=0}^{s}\binom{s}{k}x^{s-k}(-\mu_2)^k$. Thus eq. (E.78) is equal to

$$\begin{aligned}&\frac{1}{\sigma^s}\Bigg[\Bigg(\sum_{k=0}^{s}\binom{s}{k}{\mu_1}^{s-k}(-\mu_2)^k\sum_{i=0}^{s-k}\binom{s-k}{i}(\frac{\sigma}{\mu_1})^i(1+(-1)^i)\frac{\Gamma(\frac{i+1}{s})}{2\Gamma(\frac{1}{s})}\Bigg)\\&\quad-\Bigg(\sum_{k=0}^{s}\binom{s}{k}{\mu_1}^{s-k}(-\mu_1)^k\sum_{i=0}^{s-k}\binom{s-k}{i}(\frac{\sigma}{\mu_1})^i(1+(-1)^i)\frac{\Gamma(\frac{i+1}{s})}{2\Gamma(\frac{1}{s})}\Bigg)\Bigg]\end{aligned} \tag{E.79}$$

$$\begin{aligned}&=\frac{1}{\sigma^s}\Bigg[(\sum_{k=0}^{s}\binom{s}{k}{\mu_1}^{s-k}(-\mu_2)^k\\&\quad-\sum_{k=0}^{s}\binom{s}{k}{\mu_1}^{s-k}(-\mu_2)^k\sum_{i=1}^{s-k}\binom{s-k}{i}(\frac{\sigma}{\mu_1})^i(1+(-1)^i)\frac{\Gamma(\frac{i+1}{s})}{2\Gamma(\frac{1}{s})}\\&\quad-\sum_{k=0}^{s}\binom{s}{k}{\mu_1}^{s-k}(-\mu_1)^k\\&\quad-\sum_{k=0}^{s}\binom{s}{k}{\mu_1}^{s-k}(-\mu_1)^k\sum_{i=1}^{s-k}\binom{s-k}{i}(\frac{\sigma}{\mu_1})^i(1+(-1)^i)\frac{\Gamma(\frac{i+1}{s})}{2\Gamma(\frac{1}{s})}\Bigg]\end{aligned} \tag{E.80}$$

$$\begin{aligned}&=\frac{1}{\sigma^s}\Bigg[(\mu_1-\mu_2)^s\\&\quad+\sum_{k=0}^{s}\binom{s}{k}{\mu_1}^{s-k}(-\mu_2)^k\sum_{i=1}^{s-k}\binom{s-k}{i}(\frac{\sigma}{\mu_1})^i(1+(-1)^i)\frac{\Gamma(\frac{i+1}{s})}{2\Gamma(\frac{1}{s})}\\&\quad-\sum_{k=0}^{s}\binom{s}{k}{\mu_1}^{s-k}(-\mu_1)^k\sum_{i=1}^{s-k}\binom{s-k}{i}(\frac{\sigma}{\mu_1})^i(1+(-1)^i)\frac{\Gamma(\frac{i+1}{s})}{2\Gamma(\frac{1}{s})}\Bigg]\end{aligned} \tag{E.81}$$

Note that only even indices contribute to the summand in eq. (E.81) because of the $(1+(-1)^i)$ term and so can be written as

$$\begin{aligned}&\frac{1}{\sigma^s}(\mu_1-\mu_2)^s\\&+\frac{1}{\sigma^s}\Bigg(\sum_{k=1}^{s}\binom{s}{k}({\mu_1}^{s-k}(-\mu_2)^k-{\mu_1}^{s-k}(-\mu_1)^k)\sum_{i>0}^{s-k}\binom{s-k}{i}(\frac{\sigma}{\mu_1})^i(1+(-1)^i)\frac{\Gamma(\frac{i+1}{s})}{2\Gamma(\frac{1}{s})}\Bigg)\end{aligned} \tag{E.82}$$

Note, $k=0 \implies ({\mu_1}^{s-k}(-\mu_2)^k-{\mu_1}^{s-k}(-\mu_1)^k)=0$, and so eq. (E.82) becomes

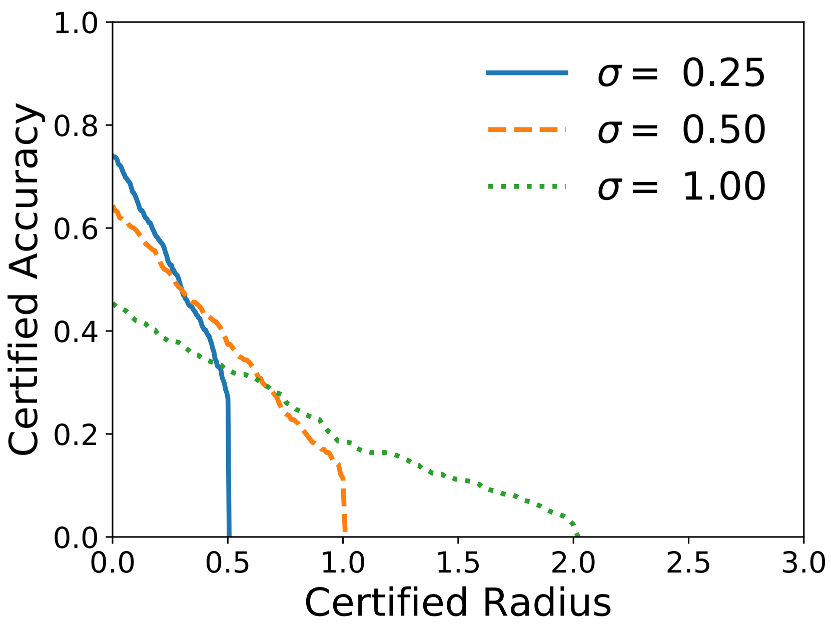


Figure E.2: Certified accuracy against perturbations targeting the ℓ_2 norm for CIFAR-10. Given as a function of the certified radius, the radius around which an input is robust.

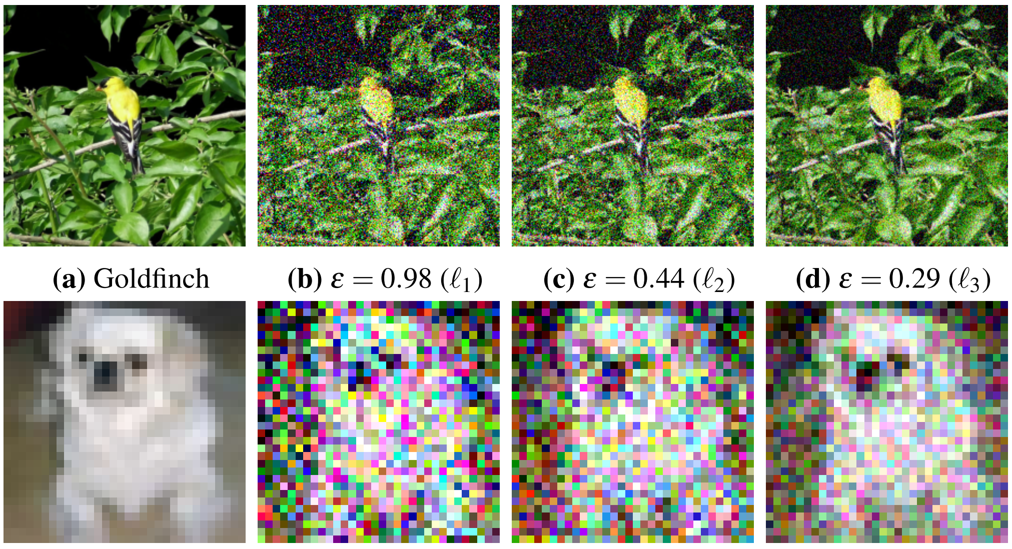
$$\sum_{k=1}^{s} \binom{s}{k} \frac{(1+(-1)^{s-k})\Gamma(\frac{s-k+1}{s})(\mu_1-\mu_2)^k}{2\sigma^k\Gamma(\frac{1}{s})} \tag{E.83}$$

□

E.5 How does σ affect the certification radius?

For 400 CIFAR-10 test set inputs, we certify inputs against ℓ_2 perturbations while varying the noise scale parameter σ [1]. Figure E.2 shows certified accuracy as a function of the certified area for $\sigma = 0.25, 0.5, 1.0$. This is the guaranteed classification accuracy under any perturbation smaller than the specified bound. Larger σ results in a larger certified area but suffers from lower standard classification accuracy – this corresponds to accuracy under a certified radius of 0. This mirrors the findings of Cohen et al. [49] and Tsipras et al. [237] who showed a trade-off between robustness and standard accuracy.

[1]Note, sampling from a generalised Gaussian distribution with scale σ and shape $s = 2$, is equivalent to sampling from a Gaussian distribution with scale $\sigma/\sqrt{2}$.



(a) Goldfinch **(b)** $\varepsilon = 0.98$ (ℓ_1) **(c)** $\varepsilon = 0.44$ (ℓ_2) **(d)** $\varepsilon = 0.29$ (ℓ_3)

(e) Dog **(f)** $\varepsilon = 0.95$ (ℓ_1) **(g)** $\varepsilon = 0.48$ (ℓ_2) **(h)** $\varepsilon = 0.32$ (ℓ_3)

Figure E.3: Two randomly chosen images from ImageNet (Top) and CIFAR-10 (Bottom). We give examples of noise from a generalised Gaussian distribution with $s = 1, 2$, and 3, and the maximum perturbation size, ε, for which the classifier is certified to predict the correct class under ℓ_1, ℓ_2, and ℓ_3 based attacks.

E.6 Samples smoothed with different forms of generalised Gaussian noise

In fig. E.3, we visualise the smoothing of a generalised Gaussian over two random inputs from CIFAR-10 and ImageNet test sets. Figures E.3a and E.3e correspond to the non-smoothed versions of these two inputs, figs. E.3b and E.3f correspond to the inputs smoothed with generalised Gaussian noise sampled from $\mathcal{GN}(0, 0.25, 1)$. Similarly, figs. E.3c and E.3g correspond to the inputs smoothed with generalised Gaussian noise sampled from $\mathcal{GN}(0, 0.25, 2)$, and figs. E.3d and E.3h correspond to the inputs smoothed with generalised Gaussian noise sampled from $\mathcal{GN}(0, 0.25, 3)$. For each smoothed input, we state the size of the certified radius, ε – up to this value the input is robust to adversarial perturbations in the specified ℓ_p norm.

E.7 An example of separability of optimal decision boundaries for different ℓ_p norms

Khoury and Hadfield-Menell [126] hypothesise that, in general, it is impossible for a classifier to be robust against all ℓ_p norm attacks. They consider a toy example to demonstrate this: consider two n-dimensional spheres, X_1 and X_2, both centred at the origin with radii r_1 and r_2, respectively. They show that the optimal decision boundary between points on the spheres are distinct under the ℓ_2 and ℓ_∞ norms. We extend this to arbitrary norms through Appendix E.7. First, we define what we mean by an optimal decision boundary, state the conjecture and then give a draft of a proof that decision boundary separability extends to other norms.

Let Δ be a set of points in $\mathbb{R}^n$. We say Δ *separates* X_1 and X_2 if any continuous function f that passes through X_1 and X_2 also passes through Δ.

Let Δ be any separator of X_1 and X_2. Choose a point $x \in \Delta$ and consider the ball $\mathscr{B}_{\varepsilon,p}(x) := \{z | \varepsilon \geq \|z-x\|_p\}$. We call Δ a maximum separator if $\forall x \in \Delta$ the following holds: $\forall \varepsilon > 0, \exists m_1, m_2 \in \mathscr{B}_{\varepsilon,p}(x)$, and points $y_1, y_2 \in X_1 \bigcup X_2$, such that if y_1 is the point that minimises $\|m_1 - y\|_p$ (where $y \in X_1 \bigcup X_2$), then $y_1 \in X_1$, and equivalently if y_2 is the point that minimises $\|m_2 - y\|_p$ then $y_2 \in X_2$.

An example of a separator that is not maximal. Let $\Delta = X_1$. Then $\exists x \in X_1$ and $\varepsilon > 0$ such that $\forall z \in \mathscr{B}_\varepsilon(x)$, the points $y \in X_1 \bigcup X_2$ that minimise $\|z-y\|_p$ all lie on X_1 (i.e. $y \in X_1$ and $y \notin X_2$).

Let two concentric spheres $X_1, X_2 \in \mathbb{R}^n$ have radii r_1, r_2, respectively. Then $\forall p, q \geq 1$ with $p \neq q$, $\Delta_p \neq \Delta_q$, where Δ_p denotes the maximal separator in the ℓ_p norm.

We give a 'proof by example' in two dimensions, showing that $\Delta_1 \neq \Delta_2 \neq \Delta_4 \neq \Delta_\infty$, and prove that $\Delta_1 \neq \Delta_2 \neq \Delta_\infty$ in n-dimensions. First, consider concentric circles $X_1, X_2 \in \mathbb{R}^2$ with radii $1, 4$, respectively.

Δ_2 defines a circle of radius $\frac{5}{2}$. In particular for $p = (x, y)$, when $x = 0$, $p \in \Delta_2$ has y-coordinate $\frac{5}{2}$. For Δ_∞, when $x = 0$, p has y-coordinate $\frac{3+\sqrt{79}}{5}$. To see this, $\mathscr{B}_{\varepsilon,\infty}(m)$ with centre $m = (0, 1+\kappa)$ touches X_2 at $q = (\kappa, 1+2\kappa)$. At q we have $\kappa^2 + (1+2\kappa)^2 = 4^2$, and so $\kappa = \frac{-2+\sqrt{79}}{5}$. Hence, at $x = 0$, $q \in \Delta_\infty$ has y-coordinate

$\frac{3+\sqrt{79}}{5}$.

To find y-coordinate when $x = 0$ for a point $q \in \Delta_4$, we must solve

$$x^2 + y^2 = 4^2 \tag{E.84}$$

$$x^4 + (y - (1+\kappa))^4 = \kappa^4 \tag{E.85}$$

Since Δ_4 is tangential to X_2, we must find the root of the determinant of $(4^2 - y^2)^2 + (y-(1+\kappa))^4 - \kappa^4 = 0$. This an order 12 polynomial,

$$\begin{aligned}&28\kappa^{12} + 96\kappa^{11} + 176\kappa^9 - 4540\kappa^8 - 19528\kappa^7 + \\ &15916\kappa^6 + 403800\kappa^5 + 495735\kappa^4 - 3757020\kappa^3 + \\ &3592350\kappa^2 + 16024500\kappa - 24350625 = 0.\end{aligned} \tag{E.86}$$

This has no solution in the radicals and is approximately 1.4755 and so $q \in \Delta_4$ has y-coordinate 2.4755.

To find Δ_p in general we must solve high order polynomials that may not factor. However, we can find Δ_1 in n dimensions. Consider the diamond ℓ_1 ball centred at $m = (\frac{r_1}{2} + \frac{\kappa}{2}, \frac{r_1}{2} + \frac{\kappa}{2}, ..., \frac{r_1}{2} + \frac{\kappa}{2})$, and $q \in \Delta_1$ has coordinate $(\frac{r_1}{2} + \frac{\kappa}{2} + \kappa, \frac{r_1}{2} + \frac{\kappa}{2}, ..., \frac{r_1}{2} + \frac{\kappa}{2})$. Then $(n-1)(\frac{r_1}{2} + \frac{\kappa}{2})^2 + (\frac{r_1}{2} + \frac{\kappa}{2} + \kappa)^2 = r_2^2$. Thus,

$$\kappa = -\frac{n+2}{n+8}\sqrt{2}r_1 + \frac{2}{n+8}\sqrt{(n+8)r_2^2 - 2(n-1)r_1^2}. \tag{E.87}$$

Thus, similarly to Khoury and Hadfield-Menell [126], for constant r_1 and r_2, Δ_1 scales like $\mathscr{O}(\frac{1}{\sqrt{n}})$, and for a classifier trained to learn Δ_1, an adversary can construct an adversarial perturbation in the ℓ_2 norm as small as $\mathscr{O}(\frac{1}{\sqrt{n}})$.