Exploring Differential Privacy & Robustness in Machine Learning

COMP90055 Project

William Walters
583639

Submitted in partial fulfilment of the requirements of the degree of
Graduate Diploma (Advanced) in Science

Contents

Abstract

Modern society continues to rely on computer programs and automation for increasingly indispensable tasks involving information sharing. Because of this, program security and data privacy is more important than ever. As a result, several fields of study have emerged to address these concerns. First, Quantitative Information Flow (QIF) has emerged to concretely define program security by quantifying the information programs leak. Further, differential privacy was introduced to protect the privacy of datasets and individuals, whilst ensuring the data is still usable.

In Machine Learning (ML), privacy and security is becoming increasingly important. Machine Learning programs (or models) try to automate the process of learning about a dataset for the purposes of classification, prediction or data generation. Unsurprisingly, these programs have become the targets of attack, whereby adversaries corrupt the data to effect the accuracy of the models. Due to the ever-increasing importance of producing reliable models that are protected from adversarial attack, it has become essential to produce robust models. Robust models are able to provide accurate output in the presence of outliers or adversarial data. Recently, the demand has shifted to producing models that are provably robust. This is the central motivation for this project. Herein, we aim to explore QIF, differential privacy and robustness in Machine Learning and investigate what connections they have. Finally, we hope to motivate future research in this exciting area.

Declarations & Acknowledgements

I certify that:

- This project does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person where due reference is not made in the text.
- This research did not require clearance from the University's Ethics Committee.
- This project is x words in length (excluding text in images, table, bibliographies and appendices).

This is my own original work, under the supervision of Dr Christine Rizkallah. I would like to sincerely thank Christine, for her guidance, patience, expertise and support- I could not have completed this project without it.

Research Summary

Questions

The are several key questions that this project aims to answer, they are outlined below.

1. Can we use differential privacy to provide robustness guarantees for machine learning models?
2. How does differential privacy differ in it's definition and application in QIF versus Machine Learning?
3. Can we use QIF to analyse the robustness properties of Machine Learning algorithms?
4. How do models of adversarial attack compare in QIF and Machine Learning?

Aims

The objectives of this project are detailed below. Together, these aims hope to answer the research questions listed above.

1. Perform a literature review that introduces the topics of quantitative information flow, differential privacy, machine learning algorithms and robustness in machine learning.
2. Explore the main differences between differential privacy in machine learning and QIF. And investigate to what extent QIF can be used to analyse or certify robustness in machine learning.

Significance of the Research

This research serves as the first effort to connect the properties of differential privacy and security as defined in QIF, with a very important definition of robustness in machine learning. This project highlights some of the intuitive and formal connections between these different fields and addresses some issues with connecting them. Finally, this project identifies areas of the QIF theory that could be useful to the machine learning community for providing robustness analysis/guarantees of learning algorithms. It is the the hope of the authour that this research serves as a useful first step in connecting these notions and helps to lay the foundation for further research in this area.

Beyond the Scope of this Project

There are several areas of research that, whilst interesting and useful to explore, due to time constraints are beyond the scope of this project. These include, but are not limited to: investigating the connections between metric differential privacy and other privacy relaxations in QIF and Machine Learning, formalising the connections between definitions of robustness in Machine Learning and security/differential privacy in QIF and finally, exploring some of the privacy and security properties of learning algorithms in a QIF-aware language such as Kuijfe. Some of these topics are briefly touched on in 4.3 as areas of future research.

0: Introduction

Robustness is a broad concept in machine learning that can have several different definitions depending upon it's use case. In statistics, robust statistics are statistics that are capable limiting their bias and/or variance when the underlying data is not normally distributed, thereby providing performance guarantees in the presence of outliers or unseen data. Robustness in Machine Learning refers to the ability of a model to maintain it's desired performance, whether it be generative or discriminitive, in the presence of unexpected test data or corrupt training data. There are several approaches to analysing and exploring the robustness of a learning model. Most approaches are aimed at protecting models from adversarial attack, a malicious effort whereby an attacker corrupts the data models use for prediction. This project will explore methods that can provably provide robustness guarantees against adversarial attack.

One way of perceiving robustness is in relation to information flow. Information flow is simply the flow of information from the source to the target. Machine learning models that can capture the information contained in the data and provide the correct or expected behaviour in the form of accurate prediction or data generation are said to be secure and in this sense are robust. But how secure are they? This can be measured with Quantitative Information Flow (QIF). QIF is an emerging field that has broad and powerful applications in Computer Science and the modern world. It arose from an increasing need to provide likelihoods and probabilities on the security and information flow of programs. In brief, QIF is a tool used to analyse the probability that information about a secret has been leaked over the run-time of a program. QIF can provide mathematical guarantees on the performance of programs and the probability of them leaking information given adversarial attack. In the context of Machine Learning, this could relate to the probability of privacy leaks as the model trains or executes on test data. Thus, model robustness can be framed as how secure information flow remains over the course of the models training and execution. This project will investigate to what extent QIF can analyse the robustness of learning algorithms.

QIF can also be used to assess the differential privacy of programs, a property that is particularly important for learning algorithms which require huge amounts of data for often sensitive analyses. In these cases, the privacy of the data and program security from adversarial attack is essential. Differential privacy is a method for sharing or releasing data necessary for research, statistical analysis or machine learning, whilst protecting the privacy of individuals in, or certain features of, the data.

An example of a differentially private algorithm, are programs in healthcare that collect sensitive data about patients and families while controlling what is visible even to the employees analysing the data. Such data might be necessary to analyse for, say, hospital compliance and quality control, yet it is essential to maintain the privacy of the patients and families involved. This project will investigate whether differential privacy can be used to provide robustness guarantees for learning algorithms.

1: Background & Literature Review

1.1 Machine Learning Introduction

1.1.1 Learning Algorithms

Learning algorithms are a diverse class of algorithms with many areas of application in computer science. Some areas include: reinforcement learning for board game playing[1] and robot movement[2], deep learning for image recognition[3], language generation[4] and social analytics[5, 6] and unsupervised learning for protein structure[7] and phylogenetics analysis[8], amongst others. Despite the wide variety of applications and fields of study, learning algorithms share the following properties: one, there is some objective or policy (see Figure 1) that the algorithm seeks to optimise. Two, the algorithms trains over several iterations to converge to a solution or satisfactory cutoff point. And three, they learn by updating their model as a result of receiving information from either training data or signals in the environment. In every case, learning algorithms create models that act or produce output without explicitly being 'told' what to do.

In artificial intelligence and robotics, learning is updating an agent or model by integrating previous knowledge with updates to the world or environment it operates in. The two most well studied approaches in this field are Reinforcement Learning and Bayesian Learning. Reinforcement Learning does not require a training dataset and Bayesian Learning begins with strong assumptions that serve as prior knowledge in the absence of a training dataset[9, 10]. By contrast, in Supervised Learning (a sub-field of Machine Learning), these algorithms determine the most generalisable aspects of a dataset that represents a problem or population of interest. Figure 1 displays two schematics and the contrasting ways that algorithms learn between these different fields.

There are several different approaches to supervised learning, they are broadly categorised as being parametric or non-parametric. A parametric model has a fixed number of parameters that the model adjusts in order to fit and underlying assumption about the distribution of the training data. Whereas non-parametric models do not assume a fixed number of parameters can generalise a dataset (i.e. specify the underlying distribution of the data). Both approaches have pros and cons, for example, parametric models require less data to be effective in practice, are much quicker to train and easier to interpret. However, non-parametric models are simpler, assumption-free, robust to outliers and are asymptotically superior to parametric models as the training datasets increase in size[11]. Both of these approaches to supervised learning require data to train their model. This project will focus on learning algorithms that require training data in order to learn.

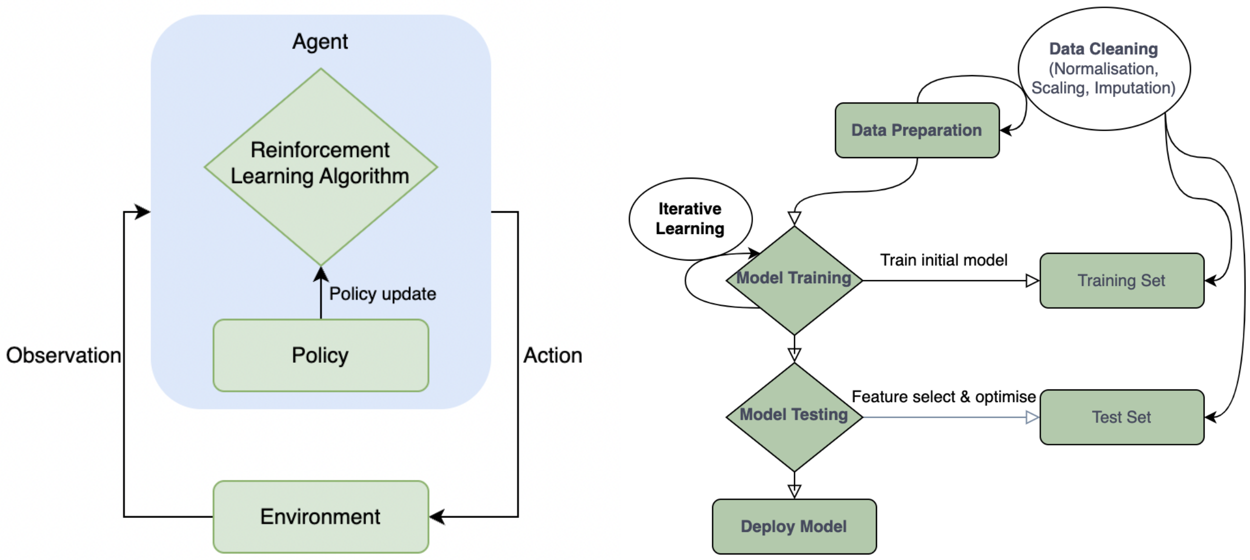

Figure 1: *Left*, Schematic of a reinforcement learning algorithm. These algorithms observe input from their environment and update a policy which then outputs an action. The policy drives the algorithm to a desired goal state by observing the environment and the reward for actions. *Right*, Schematic of a supervised learning algorithm. These algorithms take a labelled dataset as input which they use to first train and then later test/refine a model. The deployed model then makes label predictions on new data inputs. Image AO.

1.1.2 Deep Learning

Over the last few decades as machine and supervised learning has drastically increased in popularity, deep learning has prevailed as the gold standard of Machine Learning, being able to far outperform the next best models for many different problems[12]. Deep learning refers to a subclass of supervised learning algorithms that is comprised of many different flavours of neural networks. Neural networks learn via optimisation algorithms which train the parameters (a vector, θ) of a model by optimising an objective (or loss) function ($L(\theta)$). They learn over the course of several iterations. The objective function measures the error between the model's prediction and the actual label of a training instance. One such algorithm, is stochastic gradient descent, (discussed further in section 1.3.4). Figure 2 provides a visual interpretation of stochastic gradient descent (SGD) learning the optimal values for the set of parameters (θ)) required to specify the model. In each iteration of SGD, the training dataset is accessed and the optimal parameters are gradually learned. Once a model has been trained and refined, it can be 'deployed' and subsequently used on test data.

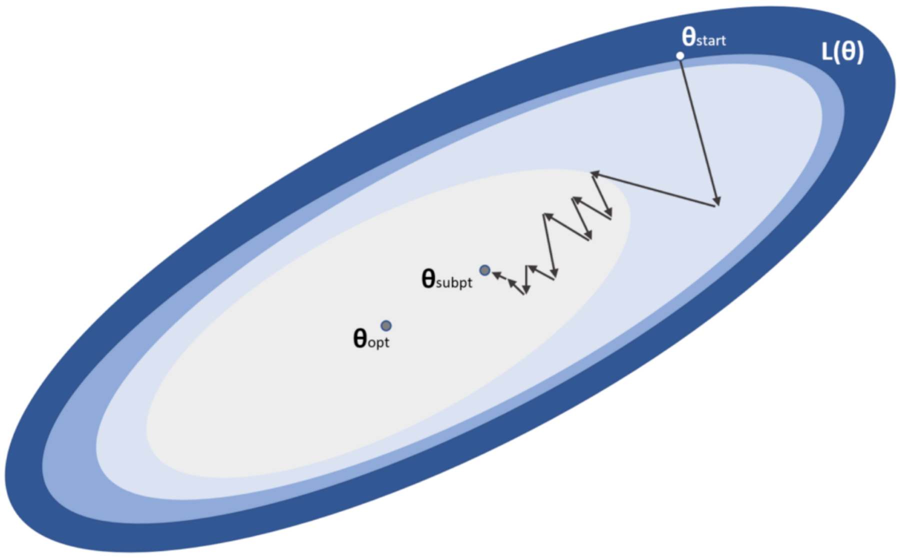

Figure 2: The objective function space ($L(\theta)$) of a model trained by stochastic gradient descent. This 2D example corresponds to an objective space of 2 parameters. Each arrow represents the new coordinate values of the weights in the 2D plane at the end of an iteration of training. θ represents a vector of weight values that define a model. θ_{subopt} and θ_{opt} denote the weight vectors of sub-optimal and optimal solutions respectively. Image from [13].

Given the vast popularity and usage of deep learning models their program security and data privacy is essential; most datasets of interest contain sensitive, private information. This can be anything from financial transactions, to medical data and even web search history, to name a few. As a result, using supervised learning models that learn from data necessitates reasoning about their security. The security and privacy concerns in machine learning are centered around the training datasets and the algorithms that operate on them. Therefore, this project will only focus on supervised learning algorithms and their properties. Hereafter we will use learning algorithms or mechanisms synonymously with objective-based optimisation algorithms that train supervised learning models. We will also use the term learning model or simply model as shorthand for the entire learning and prediction algorithm.

1.2 Quantitative Information Flow

1.2.1 QIF Motivation

Information flow is the science of transferring information from a source to its target. Studying information flow began with Information Theory which sought to quantify how much information was lost during data transmission in an effort to design better facilitators of transferring information[14]. Now that the internet, personal data and big data are entrenched in society and growing rapidly, the focus has shifted from facilitating the flow of information to securing it[15].

Whilst encryption, data aggregation & anonymyzation and access control are now commonplace methods to improve information security, they all require that the user (and parties that the user shares their data with) maintains the security of their information. In either case, they are at the mercy of human error. Because of this, Quantitative Information Flow (QIF) has recently emerged as a field to address this problem by adding a layer of mathematical rigour to the study of program security. QIF is a new branch of mathematics that is able formally define information leakage and quantitatively assess the security of programs to provide certain guarantees of program safety[15].

1.2.2 Foundations of QIF

Quantitative information flow is principally concerned with a program's ability to conceal confidential information (i.e. secrets) from an observer or adversary. The unintended information flow of secrets to an adversary is known as leakage. The security of a program depends upon how much uncertainty it can maintain over the course of running. Intuitively, the more uncertainty that there is for an adversary over a secret, the less likely it is for a program to 'leak' information to that adversary. QIF makes this intuition precise.

We denote a finite set of secrets, $\mathcal{X}$, and a probability space over them, $\mathbb{D}\mathcal{X}$. A secret, $x\colon\mathcal{X}$, has some probability associated with it, $\pi_x\colon\mathbb{D}\mathcal{X}$, and the closer π_x is to 1, then the more likely the real value of some secret is x. In QIF, π is referred to as a prior distribution. It represents the starting knowledge an adversary has about a secret, before any output is generated by the program. As the program runs, the adversary can update π into a posterior with the information leaked by the program (discussed later).

First, in order to assess how much an adversary can learn from a program, measures are needed to quantify the inherent uncertainty in the program and the cost to the adversary for not choosing the most effective action. At first, Shannon Entropy was used in QIF to measure the uncertainty of algorithms[14]. Although Shannon Entropy is a measure of the loss of information as it flows, it does not take into account the actions of the adversary. Recently loss functions were introduced to more precisely quantify the effect of an attacker's action on their knowledge of the secret[16]. For a given domain of possible actions, $\mathcal{W}$, an attackers action, $w\colon\mathcal{W}$, will try to guess some feature or property of a secret. The associated loss, $\ell(w,x)\colon\mathbb{L}\mathcal{X}$, measures the quality of the attacker's action (see Appendix B.2 for an example loss function scenario). The better the guess, the greater the reduction in uncertainty about a secret. The uncertainty of a program about a secret is defined in Equation 1, below.

Uncertainty. Given a secret, $\pi : \mathbb{D}\mathcal{X}$ and a loss function, $\ell : \mathcal{W} \times \mathcal{X} \to \mathbb{R}$, the uncertainty of a secret w.r.t. ℓ is:

$$U_\ell[\pi] = \min_{\forall w\colon \mathcal{W}} \sum_{\forall x\colon \mathcal{X}} \ell(w, x) \cdot \pi_x \tag{1}$$

Intuitively, the uncertainty is the minimum average loss to an adversary. That is, for all possible actions, QIF assumes the adversary is able to choose the action that best reduces their average loss. This is an important feature of QIF analysis, which operates under worst-case assumptions about the security of a program.

Now, given that QIF specifically analyses the security and confidentiality of programs, it is necessary to formalise what programs are and the differing properties they can have. First, in QIF, programs are considered more broadly as *systems*. A system is an umbrella term that refers to anything that processes secrets and deals with information flow. As a result, systems can refer to abstract models of information flow such as *mechanisms* or more concrete models, such as executable programs. Mechanisms are systems that process secrets and can either represent algorithms, or more precisely, stochastic matrices. Here, we focus on the latter. Mechanisms whose only observable behaviour is an output are referred to as *Channels*. Observations, y, are of type $y\colon \mathcal{Y}$. These channels provide the probability that a secret is x, given that the observed output was y. Together with the probability distribution of a secret, π_x, we can define the joint and posterior distributions over secrets and observations.

Joint & Posterior Distributions. Given a Channel, $C{:}\mathcal{X} \times \mathcal{Y} \to [0, 1]$ and a prior secret $\pi\colon \mathbb{D}\mathcal{X}$, the joint distribution over $\mathcal{X} \times \mathcal{Y}$ is:

$$[\pi \triangleright C]_{xy} = \pi_x \times C_{xy} \tag{2}$$

Further, given an observation, $y\colon \mathcal{Y}$ and the probability that y is observed, $p_y = \sum_{\forall x\colon \mathcal{X}} [\pi \triangleright C]_{xy}$, the posterior probability of a secret given y is:

$$(\pi|^y)_x = \frac{[\pi \triangleright C]_{xy}}{p_y} \tag{3}$$

Note, as previously mentioned, QIF operates under a worst-case assumption. That is, the adversary knows how a channel works; they can always determine $C_{x,y}$ (the probability that we observe y given x is the secret) for any $x\colon \mathcal{X}$, $y\colon \mathcal{Y}$. Equations 2, 3 in conjunction with equation 1 can be used to define the uncertainty of the posterior (defined as: $U_\ell[\pi \triangleright C] = \sum_{\forall y\colon \mathcal{Y}} p_y \cdot U_\ell[\pi|^y]$). This is the adversary's overall loss and by comparing this loss, $U_\ell[\pi \triangleright C]$, with $U[\pi]$ we can determine how much the adversary was able to learn from the leaked information. Altogether, these properties and definitions represent all of foundational QIF material used in this project. The final QIF properties we introduce use refinements.

Refinement is an important tool in Computer Science used to formally verify properties between programs. In QIF, a refinement can intuitively be understood as a property-preserving relation between systems. That is, if system C_1, has property x, then a refinement, C_2 on C_1 also has property x. This is a powerful property that can be used to reason about the correctness of programs or the preservation of certain properties such as program uncertainty or differential privacy or even program robustness.

Refinements on Channels & Joints. Given a secret, $\pi{:}\,\mathbb{D}\mathcal{X}$ and Channels C_1 and C_2, with respective joints $[\pi \triangleright C_1]$, $[\pi \triangleright C_2]$ the following refinement properties hold[15, 17]:

$$C_2 \sqsubseteq C_1 \iff [\pi \triangleright C_2] \sqsubseteq [\pi \triangleright C_1] \tag{4}$$

$$C_1 \sqsubseteq C_2 \iff \forall \ell : \mathbb{L}\mathcal{X},\, \pi : \mathbb{D}\mathcal{X} \quad U_\ell[\pi \triangleright C_1] \leq U_\ell[\pi \triangleright C_2] \tag{5}$$

Above, 4 details a correspondence between channels and their associated joints, and 5 states that, if for all loss functions (ℓ) and priors (π) we have that the uncertainty of C_2 is greater than C_1, then C_2 is a refinement of C_1.

1.3 Differential Privacy

1.3.1 DP Motivation

Differential privacy was first introduced by Dwork et. al. in the context of private, queryable databases[18]. The motivation was to provide utility for population-level queries and statistics whilst preserving the privacy of current and future individuals of the database. As a result, a rigourous measure was required to guarantee this trade-off. Thus, differential privacy is both a definition and a condition that both algorithms and databases can satisfy.

Beyond practical reasons, there is also a theoretical incentive for differential privacy. The no-free-lunch theorem states that there are no mechanisms that can guarantee both arbitrary levels of accuracy and privacy for all datasets. This theorem refers to all types of privacy and so also covers differential privacy[19]. The no-free-lunch theorem motivates the desirability of differential privacy over other privacy measures, as there will always be a trade-off between privacy and utility. As big-data continues to grow and become increasingly indispensable differential privacy allows companies, governments and research institutes alike to share trends, population statistics and aggregate information whilst preserving the privacy of the individuals.

1.3.2 Foundations of DP

Differential privacy is a rigourous approach to quantifying the trade-off between the utility of queries and statistical measures (functions) and the privacy of individuals in a dataset. Differential privacy is defined below:

(ϵ, δ)**-differential privacy**[18]. Given a randomised algorithm $\mathcal{M}\colon \mathcal{X}^n \to Y$, and a distribution on databases, $\mathcal{D}$, $\mathcal{M}$ is (ϵ, δ)-dp if for all adjacent datasets, $D_1, D_2 \in \mathcal{D}$, and events $S \subseteq Y$ we have:

$$\Pr[\mathcal{M}(D_1) \in S] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}(D_2) \in S] + \delta \tag{6}$$

Equation 6 encompasses two different types of differential privacy; the first case, $\delta = 0$ denoted ϵ-dp, and the second case, $\delta > 0$, denoted (ϵ, δ)-dp. First, a randomised algorithm is ϵ-dp if the probability of that algorithm returning the same output for two adjacent datasets only differs by as much as e^{ϵ}. (Adjacent datasets are datasets that only differ by a single individual or instance). The smaller ϵ is, the greater the level of privacy protection. Second, for (ϵ, δ)-dp, a randomised algorithm is ϵ-dp except with probability δ. Thus, δ is called the failure probability, as it is the probability that an algorithm fails to provide privacy. In this sense, (ϵ, δ)-dp is a relaxation of ϵ-dp. Some important closure properties for differential privacy are given in Appendix A.1.

Another relaxation of ϵ-dp is metric differential privacy (d-privacy)[20]. d-privacy expands the applicability of differential privacy to non-adjacent datasets by introducing a metric, $d(D_1, D_2)$, that captures the distance between them. The corresponding privacy inequality is: $\Pr[\mathcal{M}(D_1) \in S] \leq e^{\epsilon \cdot d(D_1, D_2)} \cdot \Pr[\mathcal{M}(D_2) \in S]$. d-privacy is also useful as it unifies central and local models of privacy. Central privacy models assume that noise is added later by the database curator, whereas local privacy models assume the noise is already added by the creator of the data. Both models have important use cases[21]. A detailed account of different relaxations of differential privacy and metrics is given in [22].

Equation 6, provides a definition of differential privacy and a condition for a mechanism to satisfy in order for it to be differential private. However, it does not detail how to create private mechanisms in practice. The actual privacy of the individuals is maintained by padding function outputs with a small amount of randomness or 'noise' to slightly perturb the true value. The amount of additional noise required is precise and depends upon the *sensitivity* of the function of interest. The less a function is 'smooth' the more the output will have to be distorted. We define sensitivity next; for an example of how to create ϵ-dp mechanisms, see Appendix B.2.

L_1 **Sensitivity**[18]. For $f : \mathcal{D} \to \mathbb{R}^k$, and adjacent datasets, $D_1, D_2 \in \mathcal{D}$ the L_1-sensitivity of f is:

$$\Delta f = \max_{D_1, D_2} ||f(D_1) - f(D_2)||_1, \quad \text{where } || \bullet ||_1 \text{ is the } L_1 \text{ vector norm.} \tag{7}$$

Intuitively, the sensitivity defines the maximum change that a single individual or instance could have on the output. The more sensitive a function, the more additional noise that will be necessary to attain privacy of the dataset.

The final relaxation introduced in this section is Rényi differential privacy. The standard definition of ϵ-dp puts a multiplicative upper bound on the worst-case change of a dataset's distribution density. Now, Rényi differential privacy, first introduced in [23], uses this notion of bounding distributional change to define a generalised definition of privacy. It achieves this by bounding the Rényi divergence by the privacy budget ϵ. The Rényi divergence (defined in Appendix A.2) is a statistical distance measure of how different two probability distributions are. Rényi differential privacy is defined below:

(α, ϵ)**-Rdp**[23]. Given a randomised algorithm $\mathcal{M}\colon \mathcal{X}^n \to Y$, and a distribution on databases, $\mathcal{D}$, $\mathcal{M}$ is (α, ϵ)-Rdp if for any adjacent $D_1, D_2 \in \mathcal{D}$ we have:

$$D_\alpha(f(D_1)||f(D_2)) \leq \epsilon \tag{8}$$

1.3.3 DP and QIF

QIF and differential privacy are both concerned with protecting sensitive information. As a result, there has been prior work comparing the two areas[23, 24]. First, we can simply expand the set of objects differential privacy works on from databases and datasets to arbitrary secret domains ($\mathcal{X}$) such as those introduced in section 1.2.2[20]. In this sense, we can create ϵ-dp mechanisms that protect the privacy of secrets. But beyond that, [24] showed that when randomised algorithms (or mechanisms) are modelled as QIF channels, then Equation 6 is equivalent to comparing the channel rows relating to secrets x_1, x_2. The differential privacy of channels is defined below:

Channel ϵ-dp[17]. For all channel output $y \in \mathcal{Y}$, a channel, C is ϵ-dp for secrets $x_1, x_2\colon \mathcal{X}$ if:

$$C_{x_1,y} \leq e^\epsilon \cdot C_{x_2,y} \quad \text{and by symmetry} \quad C_{x_2,y} \leq e^\epsilon \cdot C_{x_1,y} \tag{9}$$

Equation 9 compares two rows of a channel corresponding to secrets x_1 and x_2, and is ϵ-dp if the probability we observe y for different secrets is bounded by the privacy budget. Finally, QIF can also be used to prove that programs are differentially private[17]. This approach uses QIF and refinements of programs and has interesting applications for program involved in machine learning and is discussed later in section 3.4.

1.3.4 DP and Machine Learning

Machine Learning models are now so widespread and regularly train on very large, sensitive datasets. As a result, differential privacy is routinely used to provide a level of privacy for the datasets involved[22]. This is achieved in practise by encoding the noise into the learning algorithms used in machine learning. Figure 3 displays the pseudocode for stochastic gradient descent, SGD, (discussed in section 1.1.2) and how to make the algorithm differentially private. At each step of training the gradient needs to be computed which requires access to at least a portion of the data. These intermediate gradients iteratively update the parameter vector, θ, and so leak information about the training dataset with each iteration. To provide privacy, the differentially private SGD algorithm adds noise from a Normal distribution directly to the gradient, every time it is computed. Since differentially private mechanisms are closed under composition (Appendix A.1), a privacy accounting method can be employed to keep track of the effect of each gradient computation on the privacy budget, ϵ. Notably, even though we can make any machine learning model differentially private, this is not always desirable as there is a trade-off between privacy and model accuracy[25]. Finally, aside from providing privacy guarantees, differential privacy is also used in machine learning to provide a guarantee of robustness. This is discussed further in section 1.4.3.

```
Algorithm 1 Differentially private SGD (Outline)
Input: Examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate $\eta_t$, noise scale
    $\sigma$, group size $L$, gradient norm bound $C$.
  Initialize $\theta_0$ randomly
  for $t \in [T]$ do
    Take a random sample $L_t$ with sampling probability
    $L/N$
    Compute gradient
    For each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t}\mathcal{L}(\theta_t, x_i)$
    Clip gradient
    $\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i)/\max\left(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C}\right)$
    Add noise
    $\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L}\left(\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$
    Descent
    $\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$
  Output $\theta_T$ and compute the overall privacy cost $(\varepsilon, \delta)$
  using a privacy accounting method.
```

Figure 3: Algorithm 1 displays the pseudocode for differentially private stochastic gradient descent (SGD). By adding noise to the gradient ($\tilde{\mathbf{g}_t}$) updates at every step of training, this ensures that SGD remains differentially private. The privacy accounting method keeps track of how much of the privacy budget (ϵ) is used as the algorithm trains. Algorithm from [26].

1.4 An overview of Robustness in Machine Learning

1.4.1 Robustness Motivation

Robustness in Machine Learning refers to the ability of the model to maintain its desired performance, whether it be a generative or discriminitive model, in the presence of unexpected test data or corrupt training data. Another way of describing robustness is models that are insensitive to distributional shift: differences in the distribution of the training and test data. Figure 4 displays a simple example of robust models in regression. In Figure 4, the standard least-squares method is grossly affected by the presence of outliers, whereas the robust RANSAC method is able to perform as intended. Earlier work has shown that distribution shifts can greatly affect the classification accuracy of learning models. Distribution shifts can be categorised as common corruptions[27], out-of-distribution data[28] and adversarial examples[29]. In section 1.4.2 we describe adversarial examples and attack further.

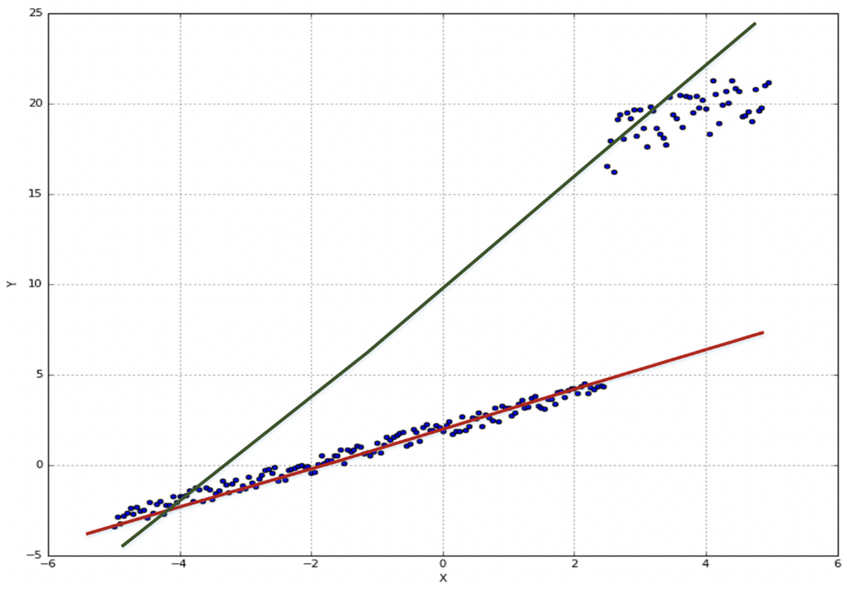

Figure 4: The effects of outliers on regression. The green line is fitted by least sqaures regression, the standard approach to fitting a straight line. It is clearly affected by the outlier data in the top righthand quadrant. The red line is fitted with random sample consensus (RANSAC[30]), a robust estimation algorithm which is insensitive to outliers. Image from [31].

There are several approaches to analysing and exploring the robustness of a learning model. First, adversarial testing and retraining, where the fully specified and trained model is fed known adversarial examples to post hoc assess the effect of corrupt data and then retrain the model[32, 33]. Second, robust training, where models are trained not only to 'fit' the data, but also to be consistent with some robustness specifications[34]. And finally, formally proving the robustness of learning models. This approach can involve several different methods, but typically involves proving that models are specification-consistent[35]. Given the prevalence of machine learning models in research and industry, ensuring they are robust is essential.

1.4.2 Definitions of Robustness & Adversarial Attack

Robustness was first defined as a property of statistics and estimators[36]. Just as statistics could be unbiased or have low variance they could also be mathematically robust. Two popular examples are the mean and the median; both measures of central tendency (Appendix A.3). The mean is not a robust estimator as it is sensitive to outlier values in a distribution. Whereas the median is a robust estimator as outlier values have no effect on its estimation of central tendency. In Machine Learning, robustness is similarly defined. Learning models are considered robust if they are insensitive to distributional shift. In practice, if a model can still make the correct prediction when given unseen, distorted test examples, then it is robust. Existing approaches generally aim at improving the robustness of neural networks to either real-world distribution shifts (e.g. common corruptions and spatial transformations) or worst-case distribution shifts (e.g. optimized adversarial attack). However recently, robustness to adversarial attack has become the gold-standard in machine learning[29].

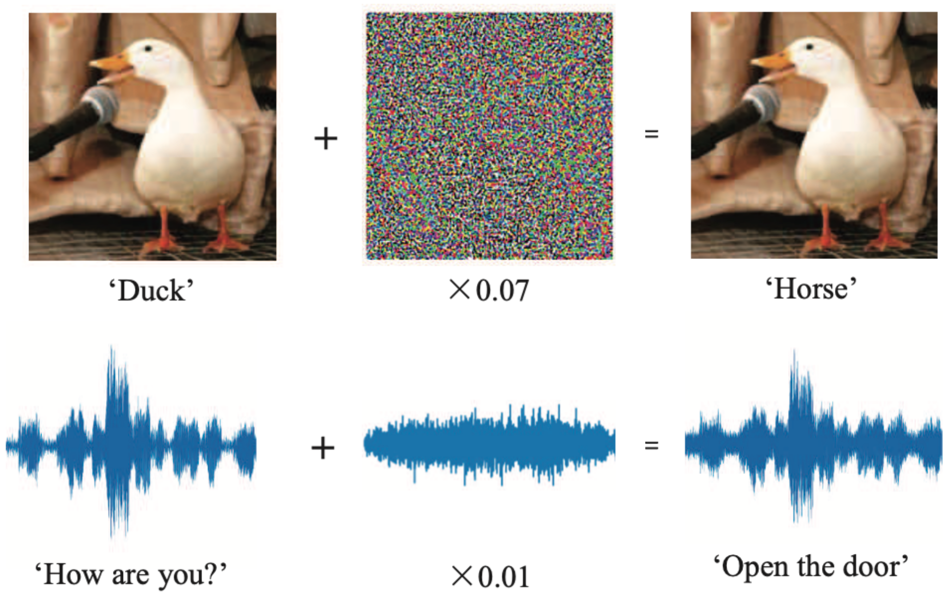

Figure 5: Two separate examples of adversarial (attack) inputs in machine learning. *Above*, a pixel matrix of noise is combined with an image of a duck which causes the model to incorrectly classify the image as a horse. *Below*, an array of noise frequencies are combined with the sentence 'How are you?' which causes the model to incorrectly classify the soundbite as 'Open the door'. Image from [37].

Adversarial attack is the design and infiltration of optimised adversarial examples to a learning model. Adversarial examples are normal training examples that have been corrupted with an imperceptible layer of noise. These examples appear normal to a human quality-checker, yet result in the learning algorithm incorrectly classifying the input. Figure 5 displays two examples that are both able to fool a learning model into giving the wrong prediction. Adversarial methods have been introduced which are able to fool Machine Learning algorithms even in the presence of rotation and scaling[38]. Thus, it is not enough to rely on data pre-processing steps for models to remain robust to adversarial attack. Moreover, given that there is an arms-race between adversarial attackers and learning model designers, *certified defences* have arisen that guarantee robustness to adversarial attack. These methods differ from previous methods by using mathematical theory to rigorously prove robustness from adversarial attack. One such method is given in [3].

In [3], the authors show that by guaranteeing a level of differential privacy, they can verify that the subsequent (ϵ, δ)-dp learning model is robust. The authors of [3] first constrain the domain of attack examples that the model could possibly see. This based off a reasonable assumption about the level of corruption the data is likely to have. If there is too much noise, then the example will cease to resemble it's intended label. I.e. the adversarial duck example in Figure 5 would no longer look like a duck, at which point it shouldn't be classified as a duck (see Appendix B.3.1). Now, given this constraint, as long as the model meets a robustness condition (Appendix B.3.3) then the model is able to correctly predict the label of bounded adversarial examples and so is insensitive to adversarial attack (Appendix B.3.2) and is certifiably robust. Importantly, this robustness condition only holds when the model is (ϵ, δ)-dp. Through this paradigm, [3], are able to provably provide a guarantee of robustness to adversarial attack using differential privacy.

2: Methods

2.1 Approaches Implemented

The goal of this project is to provide a review of Machine Learning algorithms & their robustness properties, QIF and differential privacy. And further, to investigate some intuitive, informal and formal connections and differences between these fields. To do this an extended literature review is required to both provide a background and motivation for the project, but also to support some of the results presented in section 3. This is crucial for addressing Aim 1. of the project. After leveraging the literature, several techniques will be used to investigate the similarities and differences between how differential privacy and robustness are defined in QIF and Machine Learning. First intuitive and informal connections/differences are made based upon similar/different definitions, limitations, use-cases and/or goals. Finally, refinements and analogous proofs will be used to introduce more formal methods of connecting these different fields. This will be be important for addressing Aim 2.

2.2 Limitations of Method

This project does not include the collection or analysis of a dataset. Therefore, there are no methods implemented in this project that involve programming or data analysis. Future studies could include programming and the verification of relevant program properties. Some examples of this, include using a QIF-enabled language such as Kuijfe to verify the security and differential privacy of learning algorithms to shed light on their robustness properties. Whilst briefly touched on in section 4.3, these approaches are beyond the scope of this project. Further, several results, for example the comparison between attacker models in QIF and Machine Learning, are based upon informal connections and insights into the subject area. It is a goal of future research to make some of these results more concrete with a formal methods approach involving proofs and refinement.

3: Results & Discussion

3.1 Informal Links Between QIF & ML Robustness

The connection between differential privacy and statistical robustness is well established. Early on in the development of differential privacy, Dwork and colleagues recognised a connection between differential privacy and robust estimators[39]. They showed, that several statistically robust estimators, under minor assumptions, served as differentially private estimators. Further, section 1.4.2 also outlines how differential privacy is used to provide robustness guarantees in Machine Learning. Now, given the similar motivations and properties of differential privacy and QIF, it seems logical that there are also strong connections between the theory of QIF and robustness in Machine Learning. This is explored next. In particular, we explore what overlap there is in their motivation, procedures and goals.

The theory of QIF and the robustness property in Machine Learning are both motivated by a desire to increase the security of, and confidence in, programs and algorithms. Moreover, theorems around robustness usually assume that a dataset has an underlying distribution which has a similar probability density to the original distribution the data are drawn from[36]. As a result, real-life data are generated from a contaminated or corrupted version of the original distribution[39]. Similarly, QIF defines a probability space over secrets and reasons about security in terms of the likelihood of learning about a secret[15]. In both cases, the procedures these two fields use to reason about security or robustness are probabilistic in nature. Moreover, the notion of model robustness or program security both share the same goal of insensitivity to changes in some dataset of interest. This can be evidenced by how both fields define differential privacy. In both cases, models are more secure or robust when small changes in the dataset do not greatly affect the output of the program/model. Whether it be to minimise information leakage, or to protect against adversarial attack. They also both use the concept of an adversary in order to reason about and analyse security and robustness (discussed next).

3.2 Comparing Definitions of Differential Privacy in QIF & ML

QIF frames differential privacy from the perspective of an attacker trying to learn a secret. Whereas in Machine Learning, differential privacy is framed in several ways. First, it is used to provide guarantees on the privacy of large sensitive datasets that are used in important prediction models (see sections 1.1.2, 1.3.4). Second, it can be used to provide robustness guarantees of the model to adversarial attack (section 1.4.2). Moreover there are many different relaxations of differential privacy that are routinely used in Machine Learning[22] that have not yet been explored in QIF. One such example is Rényi differential privacy (section 1.3.2). (α, ϵ)-Rdp is particularly interesting as it has a mathematical connection with analysing distribution divergence and robustness properties. (α, ϵ)-Rdp is also defined without a failure probability (discussed next). Therefore, it may provide a more natural approach than (ϵ, δ)-dp, (used in PixelDP[3]) for reasoning about robustness with QIF.

The definition of differential privacy differs most markedly between machine learning and QIF with the failure probability, δ. The failure probability represents the probability that a randomised algorithm fails to provide privacy. This addition to ϵ-dp, is not present in QIF formulations of differential privacy. There are also substantial differences in the asymptotic and practical guarantees between (ϵ, δ)-dp and ϵ-dp. This makes the choice of which definition to use a very impactful decision[40]. As a result ϵ-dp and (ϵ, δ)-dp can be considered qualitatively different (unless, $\delta = 0$, in which case they are identical). The δ additive term now creates two completely different scenarios in which privacy could fail using (ϵ, δ)-dp. Unlike ϵ-dp, where there always exists an element of uncertainty; in (ϵ, δ)-dp, the adversary with probability δ learns the secret with certainty. As a result, the (ϵ, δ)-dp formulation appears to be somewhat incompatible with certain aspects of QIF analysis.

Another difference is that in QIF differential privacy guarantees have certain program-level requirements, e.g. that the mechanisms involved be interpreted as channels. In Machine Learning, guaranteeing differential privacy is independent of how the model is interpreted so long as certain privacy conditions are met. In QIF, differential privacy can also be proven using refinements and by reasoning about program uncertainty around a secret. This highlights a different approach to Machine Learning whereby privacy is always guaranteed with statistical inference.

3.3 Contrasting Attacker Models in QIF & ML

Comparing differential privacy and robustness in QIF and Machine Learning, the most stark difference is the attacker models. Figure 6 outlines the salient differences between attacker models on a program and the input data the program accesses. We now discuss them further. First, when considering the data, in Machine Learning, the attacker often knows *a priori* what the data looks like whereas in QIF the data is hidden as it contains secrets the attacker is trying to uncover. Next, at the program level, in QIF the attacker is assumed to have unlimited access to the program that accesses the data. Whereas in Machine Learning, adversaries do not typically have access to deployed models. Instead they infiltrate the pool of test data that the model makes predictions on. Moreover, the timing and length of attack is different. In QIF, there is a notion of Bayesian reasoning, whereby the attacker is able to update their prior knowledge about a secret by reasoning about information leaked in the program's output. In Machine Learning, since access to a deployed model is often restricted, adversaries are not typically reasoned about as entities that make use of the model output to improve their attack. At the program output level, the information leaked from the output is essential for the attacker in a QIF framework to update their knowledge about what the value of a secret might be. By contrast, the attackers in Machine learning are only concerned with the output of a model not performing as it should in the presence of adversarial examples. Lastly, one point of commonality is that both fields place a worst-case assumption on the operation of the attacker.

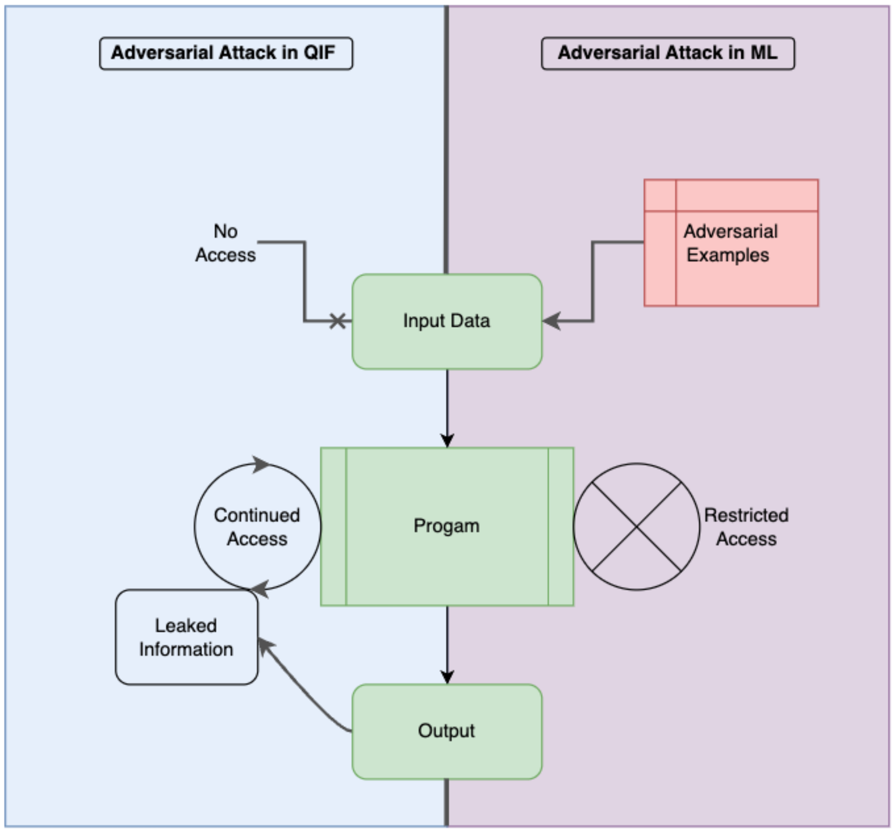

Figure 6: Schematic of the standard attacker models in QIF (left) and Machine Learning (right). In QIF, the attacker learns about a secret over several iterations exploiting the information leaked from the output. In Machine Learning, the attacker uses adversarial examples to negatively influence the model output, without interacting with the program. Image AO.

To summarise, there is an inverse notion between the two fields. In Machine Learning, the attacker has access at the data-level and seeks to leverage this access to damage the output of the program. And in QIF, the attacker has access to the program and its outputs and seeks to leverage this access to learning something (a secret) about the input. The Machine Learning adversary corrupts the input, whereas the QIF adversary continually exploits the output. As a result, of these contrasting attacker models, defences in QIF have typically centered around analysing and minimising how much an adversary can exploit the program output (see section 1.2.2). Whereas in Machine Learning, defences have typically been focused on providing performance guarantees (section 1.4.1). This is significant as it creates a roadblock for the direct application of QIF in Machine Learning. Even with these distinct differences, in the next section we explore how QIF can can still be used to analyse the robustness of learning algorithms.

3.4 Using QIF on Learning Algorithms

There are two ways in which the theory of QIF could be utilised to analyse the robustness of learning algorithms. First, a programmatic approach where we redesign adversarial attackers in Machine Learning to fit the attacker model in QIF. And second, a theoretical approach using refinements and some other theories from QIF to prove robustness properties of learning algorithms. These approaches are explored below.

As mentioned in section 3.3, the attacker in Machine Learning isn't explicitly trying to learn any secrets from the training data or the underlying distribution the data represents. However, they are trying to produce inputs that confuse the machine learning algorithm and are subsequently mis-classified or poorly predicted. To do this, the attacker needs to know certain properties of the underlying distribution of the data of interest. By doing this they can then exploit these properties to create adversarial examples. These are the properties of the data that can be considered secrets (i.e. $\mathcal{X}$), that under a QIF framework the attacker will try to learn about from the information leaked in the model's output. The attacker continually tries to learn about some secret, $\mathbb{D}\mathcal{X}$, by observing the output of the program. Under this framework, we can use, QIF to measure the robustness of a model to adversarial attack. If the posterior uncertainty, $U_\ell[\pi \triangleright C]$ remains unchanged or above a certain threshold, then we know the algorithm has not leaked any information and could therefore be considered robust.

For example, an attacker could feed the model an adversarial instance, such as the image of the duck in Figure 5 that they had artificially corrupted with a hidden layer of noise. If the model still correctly returns 'Duck' as the predicted label, then we know the attacker has been unable to trick the model, and therefore cannot learn anything from the example it gave the machine. The difficulty, that this example addresses, will be in constructing a class of loss functions that adequately capture the behaviour of the attacker and how they can learn the underlying distribution of the data given the model's output. Notably, this analysis would require that we simulate the behaviour of the attacker trying to create the most damaging adversarial examples they can. This represents a significant shift in the way attacker models have previously been analysed in Machine Learning. Instead of preparing models from adversarial examples by hard-coding a level of protection into the model, we also simulate attacker behaviour to ensure that the algorithm limits what an attacker could learn from the model.

To conclude, we can use the theory of QIF and refinements to provide a more theoretical approach to proving robustness of learning models. We itemise the approach below:

1. First, we prove that a model, A, is differentially private for the space of secrets. The secret is either the distributions of the training data or properties of the data set itself.
2. Next, we consider the machine learning model as a channel (as discussed in section 1.2.2)
3. We then use Equation 5 to prove that a new model, B is a refinement on model A ($A \sqsubseteq B$), by showing that the condition of Equation 5 satisfied.
4. Finally, we can using Lemma 1 from [17] we can then prove that this new model B is robust to adversarial attack. (Lemma 1: For two mechanisms A, B, with $A \sqsubseteq B$, then if A is ϵ-dp, so is B.)

Using this approach new Machine Learning models could be deployed that are provably robust to adversarial attack without the need to redefine robustness definitions and conditions. This is particularly useful, given how rapidly machine learning science evolves and that new models are often a result of improvements to older models.

4: Conclusion

4.1 Have the aims been addressed?

Aim 1. This project addressed Aim 1. by providing a literature review in section 1. which introduced QIF, differential privacy, Machine Learning algorithms and robustness. Aim 1. provided an indispensable knowledge foundation that directly answered Question 1. and guided the other main research questions of the project.

Aim 2. In section 3. we explored the main differences between how differential privacy is defined and used in QIF and ML. We discussed several definitions of differential privacy used in ML compared to QIF and how these definitions are used differently. This was essential to answering Question 2. Aim 2. also uncovered some fundamental differences between the QIF and Machine Learning attacker models, that was instrumental to answering Question 4. Finally we investigated how readily QIF can be used to analyse robustness in machine learning. We provided two different frameworks by which QIF could be used to analyse robustness in machine learning algorithms.

4.2 Have the questions been answered?

Q 1. From the literature review, we have been able to conclude that we can use differential privacy to provide robustness guarantees in learning algorithms. This can be concluded from section

1.4.2. However, we do note in section 4.3 that differential privacy doesn't always guarantee robustness and so further research into exactly where differential privacy fails this guarantee would be useful.

Q 2. Together, sections 3.1 and 3.2 effectively compared how differential privacy is defined and used in QIF and Machine Learning. Section 3.1 provided some insight into the informal similarities between between these field and 3.2 detailed some of the differences in their definitions, including where certain relaxations of differential privacy might not so readily transfer to QIF.

Q 3. Section 3.4 proposes two approaches to using QIF to analyse robustness of Machine Learning algorithms. The first approach remodels the attacker in machine learning to fit the framework of an adversary in QIF to analyse program robustness. And the second approach uses proofs of differential privacy in QIF and refinements to prove program robustness.

Q 4. Finally, section 3.3 provides a thorough analysis of the important differences between QIF and Machine Learning attacker models. This section sheds light on why using QIF to analyse robustness of learning algorithms isn't straightforward.

4.3 Future Directions

Section 4.2 outlined the contributions of this research project. Specifically answering Questions 3. and 4. present novel contributions towards robustness analysis and verification of learning models. Together, this research outlined some of the key differences between QIF and Machine Learning adversarial attack and how to remodel these attackers so that QIF can be used to analyse robustness in Machine Learning. As a result, this project motivates further research and collaboration between QIF researchers and the Machine Learning community to improve the science of analysing and verifying robustness. To conclude, this project also presents some exciting areas of future research as some important questions and exciting challenges remain unanswered. These are briefly detailed below.

Even though we have found that differential privacy can provide a robustness guarantee (section 1.4.2), this is not always the case as is shown in [25]. Exploring this would prove an interesting avenue for deeper investigation. Especially given that model privacy and accuracy have a trade-off, there must also be some trade-off between privacy and robustness. Is this trade-off similar to the accuracy vs. privacy trade-off? Moreover, it has been shown that d-privacy is often more useful in machine learning applications as d-privacy can make the learning algorithm private during gradient descent without clipping the gradient[41]. How does not clipping the gradient (Figure 3), affect the robustness of learning algorithms? And how does robustness and privacy connect with model over-fitting? These questions would be interesting questions to explore in the future.

Section 3.4 introduced a remodelling of the adversarial attacker in Machine Learning in order to make QIF analysis of robustness in Machine Learning possible. Future work would first involve formally redefining the type of secrets, $\mathcal{X}$, that are relevant to the learning model. Second, what space of loss functions, $\ell{:}\,\mathbb{L}\mathcal{X}$ and probability distributions, $\pi_x{:}\,\mathbb{D}\mathcal{X}$, best capture the worst-case behaviour of the attacker. And third, adapting the definition of robustness through ϵ, δ-dp given in [3] so that it fits the ϵ-dp notion of privacy in QIF. Finally, building on the work in section 3.4, both notions of program/model security and differential privacy could be explored using the QIF paradigm with the Kuifje language. Kuifje is a QIF-enabled language deeply embedded in Haskell and was based on previous work [42] that introduced and formalised QIF analysis with functional programming (in particular monads) in Haskell. A simple learning algorithm could be implemented in Kuijfe to explore the robustness of the algorithms in terms of it's security and differential privacy.

References

[1] David Silver et al. "Mastering the game of Go without human knowledge". In: *Nature* 550 (2017), pp. 354–359.

[2] Tymoteusz Lindner and Andrzej Milecki. "Reinforcement Learning-Based Algorithm to Avoid Obstacles by the Anthropomorphic Robotic Arm". In: *Applied Sciences* 12 (June 2022), p. 6629. DOI: `10.3390/app12136629`.

[3] Mathias Lecuyer et al. *Certified Robustness to Adversarial Examples with Differential Privacy.* 2018. DOI: `10.48550/ARXIV.1802.03471`. URL: `https://arxiv.org/abs/1802.03471`.

[4] Tom B. Brown et al. *Language Models are Few-Shot Learners.* 2020. DOI: `10.48550/ARXIV.2005.14165`.

[5] Wenqi Fan et al. "Graph Neural Networks for Social Recommendation". In: *CoRR* abs/1902.07243 (2019). arXiv: `1902.07243`.

[6] Qiaoyu Tan, Ninghao Liu, and Xia Hu. "Deep Representation Learning for Social Network Analysis". In: *Frontiers in Big Data* 2 (2019). ISSN: 2624-909X. DOI: `10.3389/fdata.2019.00002`. URL: `https://www.frontiersin.org/articles/10.3389/fdata.2019.00002`.

[7] Alexander Rives et al. "Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences". In: *Proceedings of the National Academy of Sciences* 118.15 (2021), e2016239118. DOI: `10.1073/pnas.2016239118`. URL: `https://www.pnas.org/doi/abs/10.1073/pnas.2016239118`.

[8] Shahan Derkarabetian et al. "A demonstration of unsupervised machine learning in species delimitation". In: *Molecular Phylogenetics and Evolution* 139 (2019), p. 106562. ISSN: 1055-7903. DOI: `https://doi.org/10.1016/j.ympev.2019.106562`. URL: `https://www.sciencedirect.com/science/article/pii/S1055790319301721`.

[9] Jens Kober, J. Bagnell, and Jan Peters. "Reinforcement Learning in Robotics: A Survey". In: *The International Journal of Robotics Research* 32 (Sept. 2013), pp. 1238–1274. DOI: `10.1177/0278364913495721`.

[10] Olivier Lebeltel et al. "Bayesian Robot Programming". In: *Autonomous Robots* 16 (Jan. 2004), pp. 49–79. DOI: `10.1023/B:AURO.0000008671.38949.43`.

[11] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning.* Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[12] Laith Alzubaidi et al. "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions". In: *Journal of Big Data* 8 (2021).

[13] Packt Ltd. *Optimization algorithms.* 2022. URL: https://subscription.packtpub.com/book/big-data-&-business-intelligence/9781788621113/9/ch09lvl1sec57/optimization-algorithms (visited on 09/30/2022).

[14] C. E. Shannon. “A Mathematical Theory of Communication”. In: *Bell System Technical Journal* 27.3 (1948), pp. 379–423. DOI: https://doi.org/10.1002/j.1538-7305.1948.tb01338.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1948.tb01338.x.

[15] Mário S. Alvim et al. *The Science of Quantitative Information Flow.* English. Information Security and Cryptography. United States: Springer, Springer Nature, 2020. ISBN: 9783319961293. DOI: 10.1007/978-3-319-96131-6.

[16] Mário S. Alvim, Andre Scedrov, and Fred B. Schneider. “When Not All Bits Are Equal: Worth-Based Information Flow”. In: *POST.* 2014.

[17] Annabelle McIver and Carroll Morgan. “Proving that Programs Are Differentially Private”. In: Nov. 2019, pp. 3–18. ISBN: 978-3-030-34174-9. DOI: 10.1007/978-3-030-34175-6_1.

[18] Cynthia Dwork. “Differential Privacy”. In: *Automata, Languages and Programming.* Ed. by Michele Bugliesi et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–12.

[19] Daniel Kifer and Ashwin Machanavajjhala. “No Free Lunch in Data Privacy”. In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data.* SIGMOD ’11. New York, NY, USA: Association for Computing Machinery, 2011, pp. 193–204. ISBN: 9781450306614. DOI: 10.1145/1989323.1989345. URL: https://doi.org/10.1145/1989323.1989345.

[20] Kostas Chatzikokolakis et al. “Broadening the Scope of Differential Privacy Using Metrics”. In: July 2013. ISBN: 978-3-642-39076-0. DOI: 10.1007/978-3-642-39077-7_5.

[21] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. “Local Privacy and Statistical Minimax Rates”. In: *2013 IEEE 54th Annual Symposium on Foundations of Computer Science.* 2013, pp. 429–438. DOI: 10.1109/FOCS.2013.53.

[22] Bargav Jayaraman and David Evans. “Evaluating Differentially Private Machine Learning in Practice”. In: *28th USENIX Security Symposium (USENIX Security 19).* Santa Clara, CA: USENIX Association, 2019, pp. 1895–1912. ISBN: 978-1-939133-06-9.

[23] Ilya Mironov. “Renyi Differential Privacy”. In: *CoRR* abs/1702.07476 (2017). arXiv: 1702.07476. URL: http://arxiv.org/abs/1702.07476.

[24] Mário S. Alvim et al. “Differential Privacy versus Quantitative Information Flow”. In: *CoRR* abs/1012.4250 (2010). arXiv: 1012.4250. URL: http://arxiv.org/abs/1012.4250.

[25] Nurislam Tursynbek, Aleksandr Petiushko, and Ivan V. Oseledets. “Robustness Threats of Differential Privacy”. In: *CoRR* abs/2012.07828 (2020). arXiv: 2012.07828. URL: https://arxiv.org/abs/2012.07828.

[26] Martin Abadi et al. “Deep Learning with Differential Privacy”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Oct. 2016. DOI: 10.1145/2976749.2978318. URL: https://doi.org/10.1145%2F2976749.2978318.

[27] Dan Hendrycks et al. “The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization”. In: *CoRR* abs/2006.16241 (2020). arXiv: 2006.16241. URL: https://arxiv.org/abs/2006.16241.

[28] Dan Hendrycks and Kevin Gimpel. “A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks”. In: *CoRR* abs/1610.02136 (2016). arXiv: 1610.02136. URL: http://arxiv.org/abs/1610.02136.

[29] Aleksander Madry et al. *Towards Deep Learning Models Resistant to Adversarial Attacks*. 2017. DOI: 10.48550/ARXIV.1706.06083. URL: https://arxiv.org/abs/1706.06083.

[30] Martin A. Fischler and Robert C. Bolles. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. In: *Commun. ACM* 24.6 (1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: https://doi.org/10.1145/358669.358692.

[31] Packt Ltd. *Robust regression with random sample consensus*. 2022. URL: https://subscription.packtpub.com/book/data/9781785889622/4/ch04lvl1sec34/robust-regression-with-random-sample-consensus (visited on 09/30/2022).

[32] Jing Lin, Laurent L. Njilla, and Kaiqi Xiong. “Robust Machine Learning against Adversarial Samples at Test Time”. In: *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. 2020, pp. 1–6. DOI: 10.1109/ICC40277.2020.9149002.

[33] Nataniel Ruiz et al. “Simulated Adversarial Testing of Face Recognition Models”. In: *CoRR* abs/2106.04569 (2021). arXiv: 2106.04569. URL: https://arxiv.org/abs/2106.04569.

[34] Gagandeep Singh et al. “Fast and Effective Robustness Certification”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/f2f446980d8e971ef3da97af089481c3-Paper.pdf.

[35] Krishnamurthy Dvijotham et al. “A Dual Approach to Scalable Verification of Deep Networks”. In: *CoRR* abs/1803.06567 (2018). arXiv: 1803.06567. URL: http://arxiv.org/abs/1803.06567.

[36] George. Casella. *Statistical inference / George Casella, Roger L. Berger*. eng. 2nd ed. Australia ; Thomson Learning, 2002. ISBN: 0534243126.

[37] Yuan Gong and Christian Poellabauer. “Protecting Voice Controlled Systems Using Sound Source Identification Based on Acoustic Cues”. In: July 2018. DOI: 10.1109/ICCCN.2018.8487334.

[38] Anish Athalye et al. “Synthesizing Robust Adversarial Examples”. In: *CoRR* abs/1707.07397 (2017). arXiv: 1707.07397. URL: http://arxiv.org/abs/1707.07397.

[39] Cynthia Dwork and Jing Lei. “Differential Privacy and Robust Statistics”. In: *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing.* New York, NY, USA: Association for Computing Machinery, 2009, pp. 371–380. ISBN: 9781605585062. DOI: 10.1145/1536414.1536466. URL: https://doi.org/10.1145/1536414.1536466.

[40] F. D. McSherry. *How many secrets do you have?* 2017. URL: https://github.com/%20frankmcsherry/blog/blob/master/posts/2017-02-08.md.

[41] Natasha Fernandes et al. *Universal Optimality and Robust Utility Bounds for Metric Differential Privacy.* 2022. DOI: 10.48550/ARXIV.2205.01258. URL: https://arxiv.org/abs/2205.01258.

[42] Jeremy Gibbons et al. “Quantitative Information Flow with Monads in Haskell”. In: *Foundations of Probabilistic Programming.* Ed. by Gilles Barthe, Joost-Pieter Katoen, and AlexandraEditors Silva. Cambridge University Press, 2020, pp. 391–448. DOI: 10.1017/9781108770750.013.

Appendices

Appendix A: Properties & Definitions

A.1 Differential Privacy Closure Properties

1. Post-processing:

Let $\mathcal{M} : \mathcal{X}^n \to Y$ be a (ϵ, δ)-dp algorithm, and let $f : Y \to Z$ be an arbitrary random function. Then $f \circ A$, some arbitrary function composition, is also (ϵ, δ)-dp.

2. Basic composition:

Let $\mathcal{M}_i : \mathcal{X}^n \to Y$ be (ϵ_i, δ_i)-dp, $\forall i \in [k]$. Then the algorithm $\mathcal{M} : \mathcal{X}^n \to Y^k$ defined by $\mathcal{M}(x) = (\mathcal{M}_1(x), \dots, \mathcal{M}_k(x))$ is $(\sum_{\forall i} \epsilon_i,\ \sum_{\forall i} \delta_i)$-dp.

3. Group privacy:

Let $\mathcal{M} : \mathcal{X}^n \to Y$ be ϵ-dp. Then, for any $X, X' \in \mathcal{X}^n$ with $||X - X'||_0 \leq k$, and $S \subseteq Y$:
$Pr\{\mathcal{M}(X) \in S\} \leq e^{k\epsilon} \cdot Pr\{\mathcal{M}(X') \in S\}$.

Note, properties 1. and 2. also hold for ϵ-dp.

A.2 Rényi Divergence

Let $P(x)$, be the the probability density of P at x, similarly Q(x), and let $\mathbb{E}[x]$ be the expectation of x. Then for an order, $\alpha > 1$, the Rényi Divergence is defined to be:

$$D_\alpha(P||Q) = \frac{1}{1-\alpha} \log_e \mathbb{E}_{x \sim Q} \left\{ \frac{P(x)}{Q(x)} \right\}^\alpha$$

Note, that for $\alpha = 1$, the Rényi divergence is equivalent to the Kullback-Leibler divergence (or relative entropy).

A.3 Estimators of Central Tendency

The central tendency is the central or most typical value of a distribution. There are many different estimators to measure the central tendency. Two are defined below for a sample size of n and values $x_1 \leq x_i \leq x_n$.

The mean:

$$\bar{X} = \frac{1}{n} \sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

The median:

$$\tilde{X} = \begin{cases} x_{(n+1)/2} & \text{if } n \text{ is odd} \\ \frac{x_{n/2}+x_{((n/2)+1)}}{2} & \text{if } n \text{ is even} \end{cases}$$

The robustness of estimators can be measured by the following properties: their rejection point, gross-error sensitivity and local-shift sensitivity.

Appendix B: Further Examples & Explanations

B.1 An example Loss Function

Given the scenario where an adversary is trying to guess the exact value of a password, where the only information leaked by the program is whether the password is correct or not. Now, given, $x{:}\,\mathcal{X}$, $w{:}\,\mathcal{W}$, $\mathcal{W} = \mathcal{X}$ and $\ell{:}\,\mathcal{W} \times \mathcal{X} \to \mathbb{R}$, the associated loss function would be:

$$\ell(w, x) = 0 \text{ if } w = x \text{ else } 1.$$

Here, the actions the attacker can make are in the same type as the secret, where the loss function either returns maximal loss for an incorrect guess or zero loss for a correct guess. In practice, more sophisticated loss functions exist which are far more informative to the attacker.

B.2 The Laplacian Mechanism

The Laplacian distribution with scale b and location $\mu = 0$, denoted Laplace(b), has probability density function: $Laplace(b) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right)$. The Laplacian distribution is one of the most common sources of adding noise (so-called Laplacian noise) to a function to ensure differential privacy of the overall mechanism. Together with some query of interest, f, the Laplacian distribution is used to ensure the ϵ-dp of estimators (or query functions). This is done using a Laplace Mechanism.

Let $\mathcal{D}$ denote the universe of databases. For $f : \mathcal{D} \to \mathbb{R}^k$, the mechanism $\mathcal{M}_f$ that on database input, d, computes $f(d)$ and then adds independently generated noise with distribution $Laplace(\Delta f/\epsilon)$ to each of the k output terms and outputs these k sums, satisfies ϵ-differential privacy. The Laplace mechanism is defined as:

$$\mathcal{M}_f(X) = f(X) + (Y_1, \ldots, Y_k),$$

where the Y_i are independent $Laplace(\Delta f/\epsilon)$ random variables and Δf is the sensitivity of a function f, defined in section 1.3.2.

For a given query on a database, e.g. the mean, using a Laplace mechanism over the mean, guarantees that the mean is an ϵ-dp estimator. If all such query functions $f : \mathcal{F}$ are handled similarly, then the database (or dataset) is said to be ϵ-dp.

B.3 PixelDP and Robustness through Differential Privacy

The three main components of PixelDP that provide robustness to adversarial attack.

B.3.1 Bounding Adversarial Examples

Define $B_p(r) = \{\alpha \in \mathbb{R}^n : ||\alpha||_p \leq r\}$ to be a p-norm ball with radius r. The volume of these n-dimensional balls represents the maximal noise of an adversarial attack; i.e. how distorted adversarial examples can be from the original examples. Now, given f, a classification model, and with fixed input, $x \in \mathbb{R}$' an attacker can design adversarial examples of size L for a given p-norm if they can find some $\alpha \in B_p(L)$ such that the model outputs the incorrect label $(f(x+\alpha) \neq f(x))$.

B.3.2 Defining Robustness as Insensitivity to Attack

A model f is insensitive to attacks of p-norm L (i.e. robust) for input x, if

$$f(x) = f(x+\alpha),\, \forall \alpha \in B_p(L)$$

Or in the case of classification with more than two labels:

$$\forall \alpha \in B_p(L),\, y_{f(x)}(x+\alpha) > \max_{\forall i \neq f(x)} y_i(x+\alpha)$$

where $y(x)$ is a vector, $y(x) = (y_1(x), \ldots y_K(x))$, that corresponds to the probabilities given by f that input x should be labelled 1 through K.

B.3.3 Define a Condition of Robustness on the Model

Suppose a randomised function, A, is (ϵ, δ)-dp wrt. a p-norm metric. For any input x, and some label, $k \in \mathcal{K}$, where $\mathcal{K}$ is the set of possible labels. Then if:

$$\mathbb{E}(A_k(x)) > e^{2\epsilon} \cdot \max_{\forall i \neq k} \mathbb{E}(A_i(x)) + (1+e^{\epsilon})\delta$$

the model is robust to attacks, α, of size $L = 1 \geq ||\alpha||_p$. $\mathbb{E}(A(x))$ is the expected model output of A and the final label selected is: $f(x) = argmax_{k \in \mathcal{K}} \mathbb{E}(A_k(x))$.

(Note, above the model, A, is a multi-class classification model).