# ALGORITHMS IN PRIVACY & SECURITY

# ALGORITHMS IN PRIVACY & SECURITY FOR DATA ANALYTICS AND MACHINE LEARNING

BY
YUTING LIANG, BMath, M.Sc.

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTING & SOFTWARE
AND THE SCHOOL OF GRADUATE STUDIES
OF MCMASTER UNIVERSITY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Master of Science (2020)
(Computing & Software)

McMaster University
Hamilton, Ontario, Canada

| | |
|---|---|
| TITLE: | Algorithms in Privacy & Security for Data Analytics and Machine Learning |
| AUTHOR: | Yuting Liang<br>M.Sc. (Financial Math), McMaster University<br><br>BMath, University of Waterloo |
| SUPERVISOR: | Dr. Reza Samavi |
| NUMBER OF PAGES: | x, 83 |

# Lay Abstract

Applications employing very large datasets are increasingly common in this age of Big Data. While these applications provide great benefits in various domains, their usage can be hampered by real-world privacy and security risks. In this work we propose algorithms which aim to provide privacy and security protection in different aspects of these applications. We address the problem of data privacy; when the datasets used contain personal information, they must be properly anonymized in order to protect the privacy of the subjects to which the records pertain. We propose two practical algorithms for anonymization which are also utility-centric. We address the problem of application security, specifically for Deep Learning applications where adversaries can use minimally perturbed inputs to cause a neural network to produce incorrect outputs. We propose an approach which protects against these attacks. We provide experimental results to demonstrate the effectiveness of our algorithms for both problems.

# Abstract

Applications employing very large datasets are increasingly common in this age of Big Data. While these applications provide great benefits in various domains, their usage can be hampered by real-world privacy and security risks. In this work we propose algorithms which aim to provide privacy and security protection in different aspects of these applications. First, we address the problem of data privacy. When the datasets used contain personal information, they must be properly anonymized in order to protect the privacy of the subjects to which the records pertain. A popular privacy preservation technique is the *$k$-anonymity* model which guarantees that any record in the dataset must be indistinguishable from at least $k-1$ other records in terms of quasi-identifiers (i.e. the subset of attributes that can be used to deduce the identity of an individual). Achieving $k$-anonymity while considering the competing goal of data utility can be a challenge, especially for datasets containing large numbers of records. We formulate $k$-anonymization as an optimization problem with the objective to maximize data utility, and propose two practical algorithms for solving this problem. Secondly, we address the problem of application security; specifically, for predictive models using Deep Learning, where adversaries can use minimally perturbed inputs (a.k.a. *adversarial examples*) to cause a neural network to produce incorrect outputs. We propose an approach which protects against adversarial examples in image classification-type networks. The approach relies on two mechanisms: 1) a mechanism that increases robustness at the expense of accuracy; and, 2) a mechanism that improves accuracy. We show that an approach combining the two mechanisms can provide protection against adversarial examples while retaining accuracy. We provide experimental results to demonstrate the effectiveness of our algorithms for both problems.

*For my parents.*

# Acknowledgements

I would like to thank my supervisor, Dr. Reza Samavi, for his support, guidance and patience throughout my graduate studies.

I would also like to thank Dr. Frantisek (Franya) Franek and Dr. Antoine Deza for their helpful comments and feedback.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Increasing amounts of information are being collected, digitized and made available for various uses in this age of Big Data. Most applications employing immense datasets can be broadly categorized as belonging either to Data Mining or Machine Learning. While both types of applications involve detecting and identifying patterns and links within very large datasets, in Data Mining one is more focused on the *extraction* of knowledge, which could be further analyzed and used for different purposes (Han *et al.*, 2011); whereas in Machine Learning, the goal is to enable future predictions and decisions via building a mathematical model, which could implicitly capture patterns that are too large and too complex for humans to understand (Shalev-Shwartz and Ben-David, 2014), whereby tasks are performed often without explicitly revealing the patterns. While these applications provide great benefits in various domains, their usage can be hampered by real-world privacy & security risks. In this work, we address two problems concerning such risks: 1) preserving data privacy for Data Analytics; and, 2) improving application security for Machine Learning algorithms. In Sections 1.1 and 1.2 of this chapter, we introduce and describe the two problems in detail. We then give an overview for the contributions of this work and outline the structure for the remaining of this thesis in Section 1.3.

## 1.1 Preserving Data Privacy with $k$-Anonymity

Data mining tasks often involve privacy sensitive information, such as medical records, financial transactions and location footprints. These tasks require special treatment of the data in order to minimize privacy risks ideally without sacrificing utility of the underlying tasks. A common practice to address privacy concerns is the $k$-anonymity model which guarantees that each individual record in the dataset is indistinguishable from at least $k-1$ other records, so the probability of re-identification is always less than or equal to

$1/k$ (Samarati and Sweeney, 1998; Sweeney, 2002). While other disclosure limitation techniques such as adding noise (Chawla *et al.*, 2005), statistical obfuscation (Burridge, 2003; Ardagna *et al.*, 2011), data perturbation (Liu *et al.*, 2006) and more recently differential privacy (Dwork, 2011) have been developed to decrease privacy risks, the $k$-anonymity model is relevant and desirable when individual records must remain truthful to their origins after the privacy preservation technique is applied. The truthfulness characteristic and the fact that the model was recommended in different privacy legislation and guidelines (such as HIPAA (U.S. Department of Health & Human Services, 2015) and FIPPA (Information and Privacy Commissioner of Ontario, 2016)) contributed to the wide adoption of $k$-anonymity and multiple algorithms (e.g., (Bayardo and Agrawal, 2005; Byun *et al.*, 2007; El Emam and Dankar, 2008; Zhang *et al.*, 2017; Doka *et al.*, 2015; Lee *et al.*, 2017)) have been devised for application of the technique to privacy-sensitive dataset prior to release.

In this research we adopt the $k$-anonymity model for privacy preservation. We aim to provide practical algorithms for $k$-anonymization while striving to maximize data utility.

## 1.2 Improving Application Security in Machine Learning Algorithms

Machine Learning algorithms are useful in many applications where there is not enough expert knowledge, or there is simply too much data for manual construction of a model. Deep neural networks (DNN) are being increasingly adopted to perform a wide range of tasks from navigation and personal recommendation systems for consumer use to larger scale decision making systems such as speech recognition and computer vision. However, application of DNN in safety critical systems is hampered by its vulnerability to *adversarial examples*, where an adversary uses carefully crafted small amounts of perturbations to force the DNN to make erroneous classifications. Image classification-type networks are especially susceptible to adversarial examples, where multiple algorithms have succeeded in crafting these by adding only small amounts of perturbations to the input image (Szegedy *et al.*, 2014; Goodfellow *et al.*, 2015; Kurakin *et al.*, 2017; Papernot *et al.*, 2016b; Carlini and Wagner, 2017). We give examples of such attacks in Fig. 1.1a and Fig. 1.1b, where original images are displayed on the left column, the adversarial examples crafted for these images are on the right column and the pixels of perturbations are in the middle column, for the MNIST (LeCun *et al.*, 1998) and CIFAR10 (Krizhevsky, 2009) datasets. Note that for CIFAR10, the perturbations have been scaled up 15× for better visualization, the actual perturbations are not noticeable at all.

In this research, we aim to provide an approach for image classification-type networks that protects against adversarial examples.

(a) MNIST: target = 0 (b) CIFAR10: target = airplane

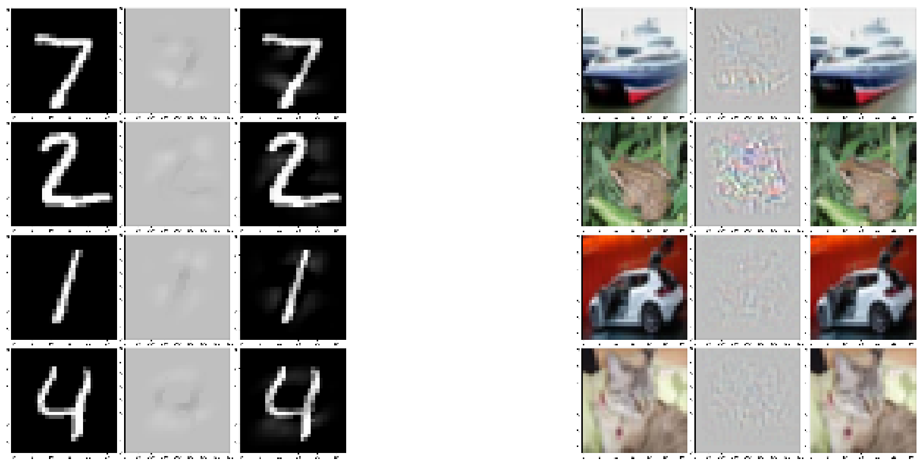


Figure 1.1: Adversarial Examples using Carlini-Wagner attack

## 1.3 Thesis Outline and Major Contributions

The main content of this thesis is structured as follows: in Chapter 2 we provide a literature review on the two problems we are setting out to address, and identify the specific gaps with brief high-level descriptions of our work. In Chapter 3 we present our algorithms for utility-centric privacy preservation using the $k$-anonymity model. The key contributions of Chapter 3 are:

1. We formulate utility-centric $k$-anonymization as an MILP (Mixed Integer Linear Program) with a weighted objective function to minimize information loss, subject to the constraint of satisfying $k$-anonymity. The weights can be customized to suit different research uses.

2. We devise two practical algorithms for solving the MILP, with experimental results to demonstrate their scalability to large datasets.

3. We present benchmarking results with other algorithms and provide a discussion on the results, with highlights on the differentiating aspects of our algorithms that contributed to better performance or utility.

In Chapter 4 we present our approach for improving robustness in neural networks. The key contributions of Chapter 4 are:

1. We develop an approach for protecting against adversarial examples, by combining two mechanisms, where one mechanism increases robustness at the expense of accuracy while the other mechanism improves accuracy.
2. We formulate potential attacks on our combined approach and provide experimental results to demonstrate the effectiveness of our approach.
3. We show that one of the formulated attacks increases transferability rate across MNIST networks (without protection mechanism) and can serve as an attack for ensemble networks with a reasonable success rate.

The work of Chapter 3 was published in Computers & Security[1], and the work of Chapter 4 was submitted to ICML 2020[2]. The author of this thesis is the first author for both works, which were co-authored by her supervisor. The content of Chapter 3 appears mostly unchanged from that in (Liang and Samavi, 2020), except for minor edits and restructuring. We improved the presentation of the experimental evaluations for Chapter 4 from that in the submission, including adding another section to further demonstrate the workings for one of the mechanisms. We also rewrote literature review and future work for this research.

We conclude this thesis and provide directions for future research and studies in Chapter 5.

[1]https://www.journals.elsevier.com/computers-and-security
[2]https://icml.cc/

# Chapter 2

# Literature Review

In this chapter, we discuss the related work of the two problems to identify gaps and motivate our research.

## 2.1 On $k$-Anonymization

The model of $k$-anonymity for privacy protection from its introduction by Samarati and Sweeny (Samarati and Sweeney, 1998; Sweeney, 2002) has received considerable attention from the research community and over years different algorithms have been proposed for applying generalization and suppression techniques in order to achieve $k$-anonymity. The algorithms differ depending on the domain of application (e.g., data publishing, data mining and statistical disclosure control) (Ayala-Rivera *et al.*, 2014) or the techniques used to provide $k$-anonymity guarantee while preserving data utility.

Among these algorithms many seek to achieve $k$-anonymity using search strategies or optimization objectives (Bayardo and Agrawal, 2005; Xiao and Tao, 2006b,a; Li *et al.*, 2007). Multiple research studies (e.g., (Aggarwal *et al.*, 2005; Meyerson and Williams, 2004; Xu *et al.*, 2006)) have shown that optimally solving a $k$-anonymization problem is an NP-hard problem. Thus, a number of efficient algorithms have been developed to enable the practicability of anonymization on large datasets. For the purpose of studying our related work, we group the algorithms by the primary techniques or problem representations they use, as partitioning-based, clustering-based and heuristic-based.

The algorithms that employ a partitioning-based technique take a geometric view (exhibiting notions of hyper-cubes) on the problem. They can be seen as taking a *top-down* approach, where the dataset is repeatedly partitioned into smaller subsets according to some criteria, usually defined by thresholds, where eventually partitions of size at least $k$ are formed (LeFevre *et al.*, 2006). Partitioning-based algorithms usually have the advantage of efficient performance, as equivalence classes are created in a *one-pass* manner without repeated visits to the constructed equivalence classes.

The clustering-based algorithms can be seen as taking a spatial view (but not necessarily geometric due to the non-definitive shapes of the clusters) on the problem, where clusters are formed based on the similarities of the records represented as points in the space. These algorithms usually take a *bottom-up* approach, in that clusters are formed in a successive manner until the entire dataset is exhausted (Byun *et al.*, 2007; Goldberger and Tassa, 2009; Gionis *et al.*, 2008; Aghdam and Sonehara, 2016). Clustering-based algorithms appear naturally suited for $k$-anonymization, where information loss can be appropriately encoded as a distance metric, and equivalence classes are clusters of records of size at least $k$.

We consider as heuristic-based algorithms those that employ a specific problem representation; in particular those algorithms which represent the dataset as a graph, where the records are viewed as vertices, and equivalence is defined via matchings (Tassa *et al.*, 2012; Doka *et al.*, 2015). The benefit of sound problem representation is evident in the case of the Doka Hungarian algorithm (Doka *et al.*, 2015), where an existing algorithm (Hungarian for finding perfect matchings) can be readily used for solving the problem.

In this work, we aim to provide a perspective from an optimization stand point. The problem of maximizing utility while satisfying $k$-anonymity constraint is clearly an optimization problem. A mathematical formulation of $k$-anonymity as an optimization problem allows us to gauge how much utility is lost in any practical algorithms we devise compared to a theoretical optimum. A similar approach has been taken by Zhang et al. (Zhang *et al.*, 2017) where a formal optimization problem is defined and *$k$-anonymity by containment* is considered the constraint. They defined containment as a subset $S$ of the feature space that satisfies $k$-anonymity. The focus of this proposal is on categorical data (specifically binary data) only, with the objective to maximize the size of subset $S$. In our work, we formulate the general optimal $k$-anonymization problem as an MILP. While the general model has exponential complexity, we use two properties to devise a practical optimization-based algorithm (Split & Carry): 1) records that are far apart are unlikely to end up in the same equivalence class in an optimal solution; 2) equivalence in $k$-anonymity is transitive. We remark that a global optimal solution is sometimes achieved with our algorithm, as will be demonstrated in Section 3.4.2 of our experimental evaluation. However, we recognize the cases where there are many attributes or a larger $k$ is desired, Split & Carry will have inefficient performance and thus provide an alternative (Greedy Search) that provides less utility but efficient performance. Our Greedy Search, inspired by the Doka SortGreedy algorithm (Doka *et al.*, 2015), employs a search strategy and shares similar aspects with the K-Member Clustering algorithm proposed by (Byun *et al.*, 2007), these will be discussed in Section 3.4.3.

## 2.2 On Robustness of Neural Networks

The task of finding an adversarial example involves adding perturbations to an image to cause a network to misclassify, usually to some targeted label. Ideally the amount of perturbations added should be small for the resulting image to appear genuine. Since we can represent images as vectors of pixel intensity values, the problem of finding adversarial examples can be naturally represented as an optimization problem, where the objective is to minimize the size of the perturbation vector, which can be captured by some norm function, and the constraint is that the network should classify the perturbed image (original input vector plus perturbation vector) as the target label. After formulating this task as an optimization problem, Szegedy et al. (Szegedy *et al.*, 2014) transformed the constrained problem into a non-constrained problem (up to box constraints) using a penalty method, with the loss function applied to the perturbed image and target label as the penalty function. They then solved the transformed problem using the L-BFGS[1] algorithm. Surprisingly, it turned out the amount of perturbations needed is usually very small.

Following the invention of the first adversarial example, Goodfellow et al. provided a fast way for generating adversarial examples which uses only the sign of the gradient of the loss function with respect to the input pixels, which is inexpensive to compute (Goodfellow *et al.*, 2015). Their method involves perturbing all pixels of the input simultaneously by a small amount in the direction of the sign of the gradient, hence the name *fast gradient sign method*. When each pixel is perturbed by a very small amount (especially if it's less than the amount of precision in the features), the effect on the resulting image will be imperceptible. However, they argued that the change in activation due to the small perturbation will grow linearly in the size of the weight vector (since weights are applied at each layer to the previous output layer), eventually adding up to a large change in the final output.

The fast gradient sign method uses a pre-selected magnitude of perturbation. It was later extended by Kurakin et al. into an iterative version that applies the base method multiple times, with a smaller step size, as to reduce the magnitude of the perturbation applied(Kurakin *et al.*, 2017).

Papernot et al provided a method that takes on a different approach than searching along the direction of the gradient of the loss function (Papernot *et al.*, 2016b). Their method is an iterative scheme that loops over each pixel to determine whether it should be increased, until the network classifies the perturbed image as the target or if the magnitude of the perturbation has reached a pre-defined threshold. The main idea is to increase a pixel if doing so increases the probability of the resulting image being classified as the target. First, the total forward derivative $\nabla F$ of the network $F$ with respect to each

[1]Optimization algorithm, to be discussed in 4.1.1.

pixel is computed. Then for a specific target $t$, the component derivative $\nabla F_t$ corresponding to $t$ is examined. The adversary increases the value of a pixel $x_i$ if $\partial F_t/\partial x_i$ is positive. i.e., increasing the value of $x_i$ increases the probability corresponding to $t$ in the output vector.

Shortly after the inventions of the above methods for crafting adversarial examples, a relatively recent defence system known as Defensive Distillation (Papernot *et al.*, 2016a) was devised, which at the time was shown a great success as it resisted all adversarial examples described above. Unfortunately, Defensive Distillation was soon defeated by the Carlini-Wagner attack (Carlini and Wagner, 2017). We further discuss these two works in Chapter 4 where we develop the ideas for our approach. In this research, we first dissect the underlying vulnerabilities of prior attacks that led to the temporary success of Defensive Distillation, and discuss how the Carlini-Wagner attack was able to escape those vulnerabilities. Then, we build our defence system using some aspects of Defensive Distillation, plus an obscuring mechanism for the iterative scheme used by Carlini & Wagner. Although the mechanism was developed with Carlini-Wagner in mind, theoretically it should work in general on any iterative scheme that relies on outputs from some layers of the network.

Our intuition for creating such a defence also relies on the idea of training an ensemble of networks and using the aggregate outputs of the networks to decide on a final output. A similar voting mechanism proved successful for protecting data privacy via differential privacy in PATE (Papernot *et al.*, 2017), where the training set containing sensitive data is partitioned into disjoint sets, each is used for training a *teacher* network. The mechanism of PATE relies on partitioning and keeping the sensitive data secret to provide data privacy. Then PATE uses a voting mechanism which serves two purposes: 1) compiling the outputs from the partitioned sensitive data to ensure correct final output; and 2) carefully adding random noise to the aggregate votes to ensure differential privacy. In our case, the intent of using an ensemble is to minimize the probability of a successful attack by increasing the amount of effort, and potentially increasing the amount of perturbation to an image so the attack becomes noticeable (i.e. fails).

# Chapter 3

# Optimization-Based $k$-Anonymity Algorithms

In this chapter, we present our practical algorithms for achieving utility-centric $k$-anonymity. The structure of this chapter is as follows. In Section 3.2 we introduce a mathematical formulation for the $k$-anonymization process, followed by discussions on several aspects of the model including complexity, weighting attributes, treatment of categorical data and the utility metric. We then present two practical algorithms for $k$-anonymity in Section 3.3, with testing results in Section 3.4 to demonstrate their optimality and performance. In Section 3.4 we also present some benchmarking results with similar algorithms and scalability measure for our algorithms to show they can be used for anonymizing large datasets. We briefly discuss alternate forms of $k$-anonymity in Section 3.5 and conclude the chapter in Section 3.6.

## 3.1 Preliminary

The process of $k$-anonymization involves applying two major operations: generalization and suppression. In generalization the value of an individual attribute is replaced with a broader category (e.g., Age: 54 will be replaced by Age: 50-60). When generalization is not applicable or will not achieve $k$-anonymity, record-level or attribute-level suppression will be applied, where the entire record or the cell value will be deleted, respectively. The operation of generalization will impact the utility of the anonymized dataset variably depending on the degree of generalization (and in a worst case suppression). For example, Age: 54 can be generalized to Age: 50-60 or Age: 50-55, where more information loss is incurred in the former than in the latter. Thus, it is desirable that while $k$-anonymity is guaranteed, the objective of maximal utility be built into the anonymization process. The objective of this work is to formulate the anonymization process as a mathematical optimization problem, where we seek to maximize data utility subject to $k$-anonymity constraints. Although in general the $k$-anonymization process is NP-hard (Meyerson and Williams, 2004), an optimal formulation is valuable as it can provide insights into the complexity of the problem, serve as a basis for developing heuristics and as a benchmark tool for future algorithms.

In this work, we focus on the generalization operation of $k$-anonymity and define information loss for each attribute as the ratio between the range of anonymized values and the range of the possible values of the attribute. For example, if the permissible upper and lower bounds for Age is [0, 100], then generalization of the attribute value to 50-55 leads to an information loss value of 0.05 ($= (55-50)/(100-0)$) compared to information loss value of 0.1 when the attribute value is generalized to 50-60. Thus, the former generalization is favoured by the objective function. We also introduce a weight vector $W$ for the attributes involved in the generalization operation such that the relative importance of each attribute in information loss can be customized by the user. We formulate the model as a Mixed Integer Linear Program (MILP). Given the complexity of MILP, we then propose two practical algorithms based on the intuition that rows of data that are *closer* to each other are likely to end up as equivalent rows. For example, if the dataset has just a single attribute with numerical values (e.g., Age), then neighbouring rows of the sorted column are relatively closer to each other and the operation of generalization will only involve grouping consecutive records. We capture the concept of *closeness* with a preprocessing step of sorting the dataset for numerical values. When we are facing multiple attributes, we use variance for determining the relative order of the sorting among attributes. Although we are not able to provide a rigorous proof in this work, we observe empirically that an attribute with less variance incurs more information loss when generalized compared to an attribute with larger variance. Therefore, the attribute with less variance will be sorted first.

Our first heuristic model is based on splitting and carrying over records between the split subsets (Split & Carry algorithm). The splitting allows us to solve a sequence of smaller sub-problems with MILP. By carrying over a portion of records from each sub-problem we provide a linkage between the otherwise disjoint sub-problems. Although the complexity of this algorithm is linear in the number of the records, it is very sensitive to the number of attributes or what is called *the curse of dimensionality* (Aggarwal, 2005). To address the dimensionality problem, we develop a Greedy Search algorithm inspired by the algorithm presented in (Doka *et al.*, 2015), however, with the difference that our algorithm generates anonymized datasets consistent with the general definition of $k$-anonymity (not based on the customized definition in (Doka *et al.*, 2015)).

## 3.2 General Optimal Model

In this section we first provide a preamble to the optimal model including the mathematical definitions of $k$-anonymity and the necessary mechanics for model formulation (Section 3.2.1). We then formulate $k$-anonymity as a general Mixed Integer Linear Program (MILP) and discuss its complexity in Section 3.2.2. In Section 3.2.3 we discuss an initial feasible solution which is the basis of our heuristic algorithms. Two additional aspects in our optimal model formulation, weighting the attributes and treating categorical attributes are discussed in Section 3.2.4 and Section 3.2.5, respectively.

### 3.2.1 Definitions

The following definitions can be found throughout various $k$-anonymity literature; here we state them in the context of our general optimal model.

DEFINITION 1: Quasi-identifiers are subsets of attributes of a dataset which can be used to deduce the identity of an individual.
Note: In this paper all attributes are considered as quasi-identifiers.

DEFINITION 2: $k$-anonymity is achieved in a dataset if each record of the dataset cannot be distinguished from at least $k-1$ other records by quasi-identifiers.

DEFINITION 3: $k$-anonymization is the procedure of applying generalization and suppression to the quasi-identifiers of the dataset in order to achieve $k$-anonymity.

DEFINITION 4: Let $x$ be an $n \times m$ matrix of records with each column corresponding to a quasi-identifier and each row corresponding to a record of some subject. Let $x'$ be the matrix obtained from $x$ by generalization, and $x'_{ij}$ denote the generalized value of an entry $x_{ij}$ of $x$ for attribute $j \in J := \{1, 2, ..., m\}$ on record $i \in I := \{1, 2, ..., n\}$. We can write $x'_{ij}$ as:

$$x'_{ij} = [y_{ij}, z_{ij}], \tag{3.2.1}$$

where $y_{ij}$ and $z_{ij}$ are the lower and upper bounds for the generalized values of $x_{ij}$ and $y_{ij} \le x_{ij} \le z_{ij}$. Then the degree of information loss, $D_{ij}$, for the entry $x_{ij}$, is defined to be:

$$D_{ij} = \frac{z_{ij} - y_{ij}}{U_j - L_j}, \tag{3.2.2}$$

where $U_j$ , $L_j$ are the minimum and maximum permissible values of attribute $j$. Note that:

$$0 \leq \frac{z_{ij} - y_{ij}}{U_j - L_j} \leq 1. \tag{3.2.3}$$

When $y_{ij} = x_{ij} = z_{ij}$, information is not generalized and the entry has 0 information loss. When $y_{ij} = L_j$, $z_{ij} = U_j$, all information about this entry is lost, i.e., the attribute is suppressed.

LEMMA 1: Let $v_{il}$ be a binary variable, let $y_{ij}$, $z_{ij}$ be the lower and upper bounds that represent the generalized entry $x'_{ij}$, for $1 \leq i, l \leq n$, $i \neq l$, and $1 \leq j \leq m$. The following set of constraints guarantees $k$-anonymity in the matrix $x'$:

$$\begin{bmatrix}1 & -1\end{bmatrix} \begin{bmatrix}y_{ij} \\ y_{lj}\end{bmatrix} \leq M^y(i,l,j)(1 - v_{il}) \tag{3.2.4}$$

$$\begin{bmatrix}1 & -1\end{bmatrix} \begin{bmatrix}y_{ij} \\ y_{lj}\end{bmatrix} \geq m^y(i,l,j)(1 - v_{il}) \tag{3.2.5}$$

$$\begin{bmatrix}1 & -1\end{bmatrix} \begin{bmatrix}z_{ij} \\ z_{lj}\end{bmatrix} \leq M^z(i,l,j)(1 - v_{il}) \tag{3.2.6}$$

$$\begin{bmatrix}1 & -1\end{bmatrix} \begin{bmatrix}z_{ij} \\ z_{lj}\end{bmatrix} \geq m^z(i,l,j)(1 - v_{il}) \tag{3.2.7}$$

$$\sum_{l \neq i} v_{il} = k - 1, v_{il} \in \{0, 1\} \, \forall i \neq l \tag{3.2.8}$$

where $M^y(i,l,j)$, $M^z(i,l,j)$ are upper bounds and $m^y(i,l,j)$, $m^z(i,l,j)$ are lower bounds on the pair $(y_{ij} - y_{lj}, z_{ij} - z_{lj})$.

*Proof.* We provide a construction of the constraints. Recognizing that the process of $k$-anonymization involves selecting $k$ rows and generalizing the entries in such a way that the $k$ rows are indistinguishable from each other, we need to be able to represent equivalence between two rows. We want to show that $v_{il}$ is the binary variable that represents whether rows $i$ and $l$ are equivalent. Let $y_{ij}$, $z_{ij}$ be the lower and upper bounds that represent the generalized entry $x'_{ij}$, and $y_{lj}$, $z_{lj}$ represent those of $x'_{lj}$. We want to impose the relation that:

$$(v_{il} = 1) \rightarrow (y_{ij} = y_{lj} \,\&\, z_{ij} = z_{lj}). \tag{3.2.9}$$

Or equivalently for all $j \in J$:

$$(y_{ij} \neq y_{lj} \vee z_{ij} \neq z_{lj}) \rightarrow (v_{il} = 0). \tag{3.2.10}$$

In other words, rows $i$, $l$ are considered equivalent if each attribute of row $i$ has the same generalized lower and upper bounds as those of row $l$. We can formulate equation (3.2.9) as:

$$(v_{il} = 1) \rightarrow \left( \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} y_{ij} \\ y_{lj} \end{bmatrix} = 0 \,\&\, \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} z_{ij} \\ z_{lj} \end{bmatrix} = 0 \right). \tag{3.2.11}$$

Using the ideas from (Mosek, 2018), which is in turn based on the books (Williams, 1993; Nemhauser and Wolsey, 1988), the above relation can be further decomposed as equations (3.2.4) - (3.2.7). It is also easy to see that relation (3.2.9) is equivalent to equations (3.2.4) - (3.2.7); when $v_{il} = 0$, we have:

$$m^y \leq y_{ij} - y_{lj} \leq M^y, \tag{3.2.12}$$

$$m^z \leq z_{ij} - z_{lj} \leq M^z, \tag{3.2.13}$$

which are true by definitions of $m^y$, $m^z$, $M^y$ and $M^z$. When $v_{il} = 1$:

$$0 \leq y_{ij} - y_{lj} \leq 0, \tag{3.2.14}$$

$$0 \leq z_{ij} - z_{lj} \leq 0, \tag{3.2.15}$$

i.e. when $v_{il} = 1$, we have $y_{ij} = y_{lj}$ and $z_{ij} = z_{lj}$.
Note that the constraint for the other direction

$$\{(y_{ij} = y_{lj} \,\&\, z_{ij} = z_{lj}) \mid \forall j \in J\} \rightarrow (v_{il} = 1) \tag{3.2.16}$$

is not necessary. To see this, suppose we have two sets of equivalent rows $S^1$, $S^2$ such that:

$$v_{i_1 l_1} = 1 \, \forall i_1, l_1 \in S^1, \, v_{i_2 l_2} = 1 \, \forall i_2, l_2 \in S^2, \tag{3.2.17}$$

$$|S^1| = k - 1 = |S^2|. \tag{3.2.18}$$

Possibly there exists $l \in S^1 \cap S^2$, but $v_{i_1 i_2} \neq 1$ for $i_1 \in S^1 \setminus \{l\}$ and $i_2 \in S^2 \setminus \{l\}$, i.e., in $x'$ we actually observe a larger set of equivalent rows:

$$S^1 \cup S^2, \, |S^1 \cup S^2| > k - 1, \tag{3.2.19}$$

but in this case, the model interprets the situation as two non-disjoint equivalent sets.

We have shown that the binary variables $v_{il}$ as constructed represent equivalence between two rows. Then we can see immediately that for any row $i$ to be equivalent to $k - 1$ other rows, all such binary variables associated with row $i$ must sum to at least $k - 1$. Equation (3.2.8) provides the equality constraint, and the preceding discussion in equations (3.2.16) - (3.2.19) implies the inequality. □

DEFINITION 6: Let $S^1$, $S^2$ be two equivalence classes in $x'$ such that $S^{12} :=$

$S^1 \cap S^2 \neq \emptyset$. We say the records in $S^1 \setminus S^{12}$ and $S^2 \setminus S^{12}$ are indirectly equivalent.

DEFINITION 7: Finally, a general optimization problem can be mathematically defined as of the form:

$$\begin{aligned} &\min_{x \in X} f(x) \\ &\text{Subject to } g(x) \leq 0. \end{aligned} \tag{3.2.20}$$

In our context, $f(x)$ is the objective function (information loss) and $g(x)$ is the set of $k$-anonymity and generalization range validity constraints.

### 3.2.2 Model Formulation

Using Definition 7 and Lemma 1, we formally state the optimization problem for $k$-anonymization as follows:

$$\min_{(y,z,v)} \sum_{i \in I, j \in J} \frac{z_{ij} - y_{ij}}{U_j - L_j} \tag{3.2.21}$$

Subject to

$$\begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} y_{ij} \\ y_{lj} \end{bmatrix} \leq M^y(i,l,j)(1 - v_{il}) \tag{3.2.22}$$

$$\begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} y_{ij} \\ y_{lj} \end{bmatrix} \geq m^y(i,l,j)(1 - v_{il}) \tag{3.2.23}$$

$$\begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} z_{ij} \\ z_{lj} \end{bmatrix} \leq M^z(i,l,j)(1 - v_{il}) \tag{3.2.24}$$

$$\begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} z_{ij} \\ z_{lj} \end{bmatrix} \geq m^z(i,l,j)(1 - v_{il}) \tag{3.2.25}$$

$$\sum_{l \neq i} v_{il} \geq k - 1, v_{il} \in \{0, 1\} \, \forall i \neq l \,\&\, i, l \in I \tag{3.2.26}$$

and

$$y_{ij}, z_{ij} \in \mathbb{Z}, \ \forall j \in S^D \subseteq J \tag{3.2.27}$$

$$y_{ij}, z_{ij} \in \mathbb{R}, \ \forall j \in S^C \subseteq J. \tag{3.2.28}$$

Where:
$y_{ij}, z_{ij}$: upper and lower bounds in generalizing entry $x_{ij}$.
$v_{il}$: binary variable for whether row $i$ is equivalent to row $l$.
$U_j$, $L_j$: min and max permissible values of attribute $j$.
$S^C \subseteq J$: the set of indices for continuous attributes.

$S^D \subseteq J$: the set of indices for discrete value attributes.

We choose the bound constants for the $i^{th}$ and $l^{th}$ record of attribute $j$ as follows:

$$M^y(i,l,j) = \max(x_{ij}, x_{lj}) - L_j \geq y_{ij} - y_{lj}, \tag{3.2.29}$$

$$m^y(i,l,j) = L_j - \max(x_{ij}, x_{lj}) = -M^y(i,l,j) \leq y_{ij} - y_{lj}, \tag{3.2.30}$$

$$M^z(i,l,j) = U_j - \min(x_{ij}, x_{lj}) \geq z_{ij} - z_{lj}, \tag{3.2.31}$$

$$m^z(i,l,j) = \min(x_{ij}, x_{lj}) - U_j = -M^z(i,l,j) \leq z_{ij} - z_{lj}. \tag{3.2.32}$$

**Remark:** Note that by our choice of lower bound constants, it might seem that inequalities (3.2.5) and (3.2.7) are redundant as $m^y = -M^y$ and $m^z = -M^z$; in fact both constraints are needed in order to ensure relation (3.2.9) is satisfied. Suppose we remove inequality (3.2.5). When $v_{il} = 1$, to satisfy inequality (3.2.4) we can actually have $y_{ij} < y_{ij}$, which would not satisfy relation (3.2.9).

We formulated the model described above as a Mixed Integer Linear Program (MILP) which is NP-hard in general (Papadimitriou, 1981; von zur Gathen and Sieveking, 1978). There exist subclasses of MILPs which have better complexity, but measuring the complexity of any MILP itself is a difficult task and relies on heuristics and conditions on the constraint matrix (Genova and Guliashki, 2011). Meyerson and Williams showed that the general $k$-anonymization problem (using suppression) is hard using Graph Theory techniques (Meyerson and Williams, 2004).

MILPs are best solved using Branch-and-Cut (Branch-and-Bound with Cutting Planes) methods (Genova and Guliashki, 2011). Therefore, it is not surprising that many existing algorithms aim to solve a relaxed version of the problem using some variant of Branch-and-Bound (Bayardo and Agrawal, 2005; Lee *et al.*, 2017). Although MILPs are hard in general, there are commercial implementations of Branch-and-Cut which are quite efficient (e.g., Gurobi [1] or CPLEX [2]); there are also heuristics for selecting *better* nodes at which to branch out (e.g., (Bayardo and Agrawal, 2005)).

To confirm whether commercial solvers can provide any breakthrough to the complexity of our optimal formulation, we implemented our MILP model in Python with state-of-the-art optimization solver Gurobi, which implements the Branch-and-Cut method in parallel (Gurobi Optimization, LLC., 2018). Gurobi and CPLEX are arguably the best available MILP solvers (Mittelmann, 2018). We have also implemented the general model with CPLEX and open source solver CBC [3]; we ran a few simple tests. We take a small example

[1]http://www.gurobi.com/
[2]https://www.ibm.com/products/ilog-cplex-optimization-studio
[3]https://projects.coin-or.org/Cbc

| **Name** | **Age** | **Sex** | **Zipcode** | **Disease** |
|---|---|---|---|---|
| Mary | 37 | F | 22071 | Pneumonia |
| Alice | 35 | F | 22098 | Diabetes |
| Betsy | 36 | F | 23061 | Anemia |
| David | 61 | M | 55107 | Pneumonia |
| Tom | 63 | M | 55099 | Diabetes |
| James | 66 | M | 55324 | Diabetes |
| Eric | 63 | M | 55229 | Diabetes |

(a) Original dataset ((Lee *et al.*, 2017))

| **Age** | **Sex** | **Zipcode** |
|---|---|---|
| [35-37] | [0-0] | [22071-23061] |
| [35-37] | [0-0] | [22071-23061] |
| [35-37] | [0-0] | [22071-23061] |
| [61-66] | [1-1] | [55099-55324] |
| [61-66] | [1-1] | [55099-55324] |
| [61-66] | [1-1] | [55099-55324] |
| [61-66] | [1-1] | [55099-55324] |

(b) 3-anonymized outcome

Table 3.1: Example of anonymized Electronic Health Record

from (Lee *et al.*, 2017) in Table 3.1a to demonstrate the outcome of our model in Table 3.1b, where we anonymized three quasi-identifiers (Age, Sex, and Zipcode) using our optimal MILP model.

Although we found comparable performance between Gurobi and CPLEX which both are 30x faster than CBC, even the fastest solver struggles to solve the $k$-anonymity MILP on a dataset with the size of 100 records and 8 attributes in a reasonable amount of time. Therefore, devising practical algorithms is necessary for solving $k$-anonymity problems.

### 3.2.3 Initial Feasible Solution

In light of the complexity of the optimal model, we provide an initial feasible solution to the solver to prune some of the sub-optimal nodes early on in the search. The idea behind our initial feasible solution is that rows of records that are *closer* to each other are likely to end up as equivalent rows. For example, if we have just a single column of numbers, then neighbouring rows of the sorted column are close together. However, when we have multiple columns, we need to determine the order of the attributes with which we compare the tuples.

The individual information loss component of attribute $j$

$$D_j = \sum_{i=1}^{n} D_{ij} = \sum_{i=1}^{n} \frac{z_{ij} - y_{ij}}{U_j - L_j} \tag{3.2.33}$$

depends largely on the upper-lower bound gap $U_j - L_j$ where the information loss is inversely proportional to the gap. Thus, for columns with smaller upper-lower bound gaps, we would like their entries to appear as sorted as possible in the matrix; however, it is likely that we have multiple columns with the same gaps (e.g., when we have multiple binary attributes). For this reason, we opted to use ascending variance for determining the order. In general a larger upper-lower bound difference contributes to a larger variance in the column. Moreover, for attributes with the same upper-lower bound gaps, because the

| index | AGE | SEX | INJ_SEV | DRINKING |
|---|---|---|---|---|
| 0 | 64 | 2 | 4 | 0 |
| 1 | 29 | 1 | 0 | 0 |
| 2 | 42 | 2 | 0 | 0 |
| 3 | 41 | 2 | 4 | 1 |
| 4 | 53 | 1 | 2 | 1 |
| 5 | 59 | 1 | 4 | 0 |
| 6 | 49 | 1 | 4 | 0 |
| 7 | 59 | 1 | 2 | 0 |
| 8 | 80 | 1 | 4 | 0 |
| 9 | 50 | 1 | 4 | 0 |
| 10 | 64 | 1 | 3 | 0 |
| 11 | 55 | 1 | 0 | 0 |
| 12 | 25 | 1 | 0 | 0 |
| 13 | 42 | 1 | 4 | 0 |
| 14 | 33 | 1 | 2 | 0 |
| 15 | 31 | 2 | 2 | 0 |
| 16 | 68 | 1 | 3 | 0 |
| 17 | 20 | 2 | 4 | 0 |
| 18 | 40 | 1 | 4 | 1 |
| 19 | 18 | 1 | 4 | 0 |

(a) Original dataset

| index | AGE | SEX | INJ_SEV | DRINKING |
|---|---|---|---|---|
| 12 | 25 | 1 | 0 | 0 |
| 1 | 29 | 1 | 0 | 0 |
| 11 | 55 | 1 | 0 | 0 |
| 14 | 33 | 1 | 2 | 0 |
| 7 | 59 | 1 | 2 | 0 |
| 10 | 64 | 1 | 3 | 0 |
| 16 | 68 | 1 | 3 | 0 |
| 19 | 18 | 1 | 4 | 0 |
| 13 | 42 | 1 | 4 | 0 |
| 6 | 49 | 1 | 4 | 0 |
| 9 | 50 | 1 | 4 | 0 |
| 5 | 59 | 1 | 4 | 0 |
| 8 | 80 | 1 | 4 | 0 |
| 2 | 42 | 2 | 0 | 0 |
| 15 | 31 | 2 | 2 | 0 |
| 17 | 20 | 2 | 4 | 0 |
| 0 | 64 | 2 | 4 | 0 |
| 4 | 53 | 1 | 2 | 1 |
| 18 | 40 | 1 | 4 | 1 |
| 3 | 41 | 2 | 4 | 1 |

(b) Sorted by variance

Table 3.2: FARS dataset on 4 attributes

values of their entries lie within intervals of the same length, a larger variance implies the values are more dispersed on the interval; whereas a smaller variance implies there are more points centred around the mean, which leads to larger information loss when we try to bring such points to equivalent sets with those points near the boundaries. Therefore, attributes with smaller variances will be sorted first and attributes with larger variances last. We demonstrate the impact of sorting based on variance with the following example.

Consider an example of a dataset from the FARS (US Department of Transportation, 2016) database containing traffic accidents data. This dataset contains 20 records with 4 attributes (AGE, SEX, INJ_SEV (injury severity) and DRINKING). In Table 3.2a we have the original dataset, and in Table 3.2b we show the dataset sorted with comparison order determined by increasing variance. The actual equivalence classes determined by our optimal MILP are {0,2,15,17}, {1,11,12}, {3,4,18}, {5,6,8,9}, {7,10,16}, and {13,14,19}. Notice most of the equivalence classes appear as consecutive rows in the sorted matrix.

### 3.2.4 Weighting Attributes

Another observation made is that in our current optimal modelling of $k$-anonymity, all attributes are treated equally in their impact on information loss encoded as the objective function. However, generally speaking utility is a subjective measure and depending on the use, some researchers might prefer to have less information loss in certain attributes even if it is at the expense of other attributes. By placing weights on the attributes, the requester of data has some control over which attributes he/she would want to have tighter generalized bounds. We use $W$ to assign weight $w_j$ for all $j \in J$ (the set of indices of all attributes). $W$ is used to adjust the likelihood of an attribute being generalized/suppressed in the anonymization process, where a larger weight means the attribute is less likely to be generalized/suppressed. We provide an example in Appendix A.1 to demonstrate the effect of applying unequal weights to the attributes. Then the objective function in equation (3.2.21) is adjusted as follows to include the attribute weights:

$$\min_{(y,z,v)} \sum_{i \in I, j \in J} \left[ w_j \frac{z_{ij} - y_{ij}}{U_j - L_j} \right]. \tag{3.2.34}$$

We have to also add the following additional constraint to the model:

$$\sum_{j \in J} w_j = 1. \tag{3.2.35}$$

Applying weights to the initial feasible solution (Section 3.2.3) requires an additional step. When we have a vector of unequal weights, we need to scale the variances of the columns to reflect the weights in the objective function. Variances for attributes with smaller weights need to be scaled by a larger amount than those with larger weights. Therefore, we can adjust the variance function with

$$wVar(c_j) := \frac{Var(c_j)}{w_j^2}, \tag{3.2.36}$$

for column $c_j$ with weight $w_j, j \in J := \{1, ..., m\}$.

### 3.2.5 Categorical Data

The objective function in our formulation suffers from one drawback. The information loss calculated as in Definition 4 would only make sense if the subtraction between two permissible values is defined, i.e., a clear distance metric is defined for the attribute. This is also a problem for the initial feasible solution in Section 3.2.3 where sorting is undoubtedly based on a distance metric. For binary data, there is a natural distance metric; however, for some categorical data with more than two permissible values, the distance metric is not clear. One way to mitigate the problem is to number the permissible values of a categorical attribute in a way that implies a sense of distance. For example, if we have Countries as an attribute, we can number the countries based on their geographical distances to some reference country (e.g. Canada) or language similarity. If the attribute is important to the researcher, and a weak sense of distance is not adequate, then we can split the categorical data into binary data where each new attribute corresponds to one permissible value of the original categorical attribute. When we are converting categorical data to binary data, we recommend adjusting the weight by multiplicative factor of $1/|H|$, where $H$ is the set of permissible values of the categorical attribute. This puts the sum of utility loss of the constituent binary attributes to be within the range [0,1] and thus the original categorical attribute still satisfies equation (3.2.3) of Definition 4. This treatment of categorical data is similar to the Global Certainty Penalty (GCP) utility function in (Wong *et al.*, 2010; Doka *et al.*, 2015) and in fact there is a natural one-to-one correspondence from one to the other. Please see Appendix A.2 for detailed definition of GCP.

### 3.2.6 On Utility Metric

For anonymization on relational datasets (or any dataset where each record can be represented as a tuple of attributes), there are utility metrics which are more general and applicable to different data types, such as the Discernibility Metric (DM) (Bayardo and Agrawal, 2005) and Classification Metric (CM) (Iyengar, 2002); however, these metrics do not measure the amount of information loss contributed by the individual records which might be important to the user of the anonymized data. Information loss metrics are used in various literature (Ghinita *et al.*, 2009; Wong *et al.*, 2010; Doka *et al.*, 2015), especially when generalization is an operation used to anonymize the data. In the formulation of our model in Section 3.2.2, we have defined our utility metric to be the sum of the degree of information loss over each anonymized entry $x'_{ij}$ as we aim to minimize the total amount of information loss in our anonymization process, i.e., maximize the utility of the anonymized data. The linearity of this metric is also desirable as linear optimization problems are among the easiest subclass of optimization problems in terms of complexity and availability of tools. We understand that the usefulness of this utility function might be limited in some application domains. As described in Section 3.2.4, we introduced $W$ to help customize the utility function but this can still be an issue when a more complicated utility function beyond an optimized generalization is being sought. In theory our general optimal formulation can be adapted to different utility metrics; however, since sorting is an important part of our practical algorithms, ideally the utility metric should respect the ordering in the permissible values of each attribute, i.e., for permissible values $a_1 < a_2 < a_3$ the metric should favour anonymized outcome $[a_1, a_2]$ to $[a_1, a_3]$.

## 3.3 Practical Algorithms

We have implemented the general model with the best available MILP solver; however, as we have discussed in the previous section, the complexity of the model quickly leads to undesirable performance as we increase the number of records, making the model unsuitable for practical use despite offering optimal utility. We also provide an initial feasible solution which can help in pruning some of the sub-optimal nodes early on in the search. In general it is not easy to see how much performance improvement such initial solutions can provide; moreover as the size of the problem increases, the majority of the time for the solver is spent trying to improve the bounds between the so-far best feasible node and the relaxed optimum. Even if we feed in an initial solution that is optimal, the solver might still need to traverse many nodes and solve many relaxed problems before determining that the solution was already optimal. Therefore, in practice we cannot rely on solving the general model to anonymize any reasonably sized dataset. In this section, we use the initial feasible solution to devise two different practical algorithms with improved utility over the initial feasible solution.

### 3.3.1 Split & Carry Algorithm

The first practical algorithm based on the initial feasible solution is the Split & Carry algorithm. We use the initial solution to split the original problem into smaller sub-problems with manageable sizes, i.e. sub-problems that are solvable using the general MILP. As discussed in Section 3.2.3, consecutive rows in a sorted matrix are likely to be equivalent rows in an optimally $k$-anonymized dataset. Thus, we expect the rows that are far apart in a sorted matrix are unlikely to be equivalent, and can be placed into different sub-problems to be solved by the general model. This idea is the basis of our Split & Carry algorithm as described in Algorithm 1.

The input values to the algorithm are:

1. $arrType$ an array describing the data type of each column (from {Integer, Continuous}, used in Gurobi solver)
2. $x = [x_{ij}]_{i \in I, j \in J}$, the dataset in a matrix form
3. $U = [U_j]_{j \in J}$, $L = [L_j]_{j \in J}$, upper and lower bounds of each column

4. $W = [w_j]_{j \in J}$, weights for each column

5. $k$, minimum number of equivalent rows desired

6. $S$, minimum number of $k$-sets[4] in each sub-problem ($S \geq 2$).

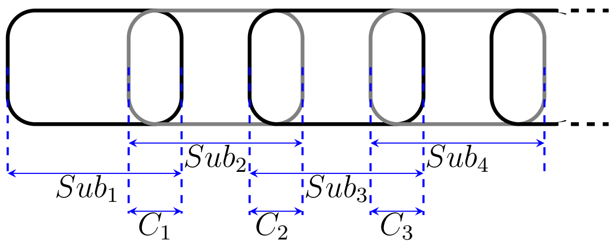
```
Data: $x$, $U$, $L$, $W$, $k$, $S$, $arrType$
Result: $\mathbf{A}$, $\mathbf{f}$
1  $VAR = [VAR]_{j \in J} \leftarrow$ compute variances of all attributes;
2  $\tilde{x} \leftarrow sort(x, VAR)$;
3  $C_0 \leftarrow \emptyset$;
4  $\mathbf{f} \leftarrow 0$;
5  for $m := 1$ to $\lceil \frac{n}{k \times S} \rceil$ do
6      $\mathbf{Sub_m} \leftarrow C_{m-1}$ + Read the next $k \times S$ rows from $\tilde{x}$;
7      $(\mathbf{A_m}, \mathbf{f_m}) \leftarrow$ Solve($\mathbf{Sub_m}$, $arrType$) optimally (Section 2.2);
8      $\mathbf{f} \leftarrow$ update($\mathbf{f}, \mathbf{f_m}$);
9      $\mathbf{A} \leftarrow$ update($\mathbf{A}, \mathbf{A_m}$);
10     $\mathbf{C_m} \leftarrow$ rows in equivalence classes of last $k$ rows in $\mathbf{A_m}$;
11     $m = m + 1$;
12 end
13 return ($\mathbf{A}$, $\mathbf{f}$)
```

**Algorithm 1:** Split & Carry Algorithm

Besides the first five input values which are common to the optimal model, we use the parameter $S$ to adjust the start size of the sub-problems; $S$ captures the minimum number of $k$-sets to be included in each sub-problem. $S$ is a user parameter with value greater than 2 because there is only one trivial $k$-set when $S = 1$. The value selected for $S$ depends on the desired efficiency and utility of the $k$-anonymization process. When $S$ is small we are limiting the potential space from which we make $k$-sets; but since $S$ determines the minimum size of the sub-problem a large value will increase the complexity of each sub-problem. We will demonstrate in our experiments that a small $S$ suffices when $k$ is small and the number of records is large. In general the running times of the sub-problems depend largely on the statistical properties of the dataset, when we have a large dataset we are likely to find many records that are similar which means each sub-problem to be solved has a small solution space; they also depend on the availability of computational resources because MILP solvers are often multi-threaded (e.g. Gurobi). In Fig. 3.5a and Fig. 3.5b we demonstrate the effect of $S$ on a reasonably-sized dataset in terms of utility and running time.

[4]$k$-set: shorthand for a set of equivalent records of size $k$.

Figure 3.1: Chain of sub-problems



In Fig. 3.1 we present an illustration of the Split & Carry algorithm. The sub-problems are indexed as $(Sub_1, Sub_2, ..., Sub_t), t = \lceil \frac{n}{k \times S} \rceil$. We have also the concept of carry-over records in this algorithm, represented as $(C_1, C_2, ...C_{t-1})$. We can visualize the algorithm as being a chain of sub-problems, where the *locks* that chain together sub-problems are the boundary carry-over records. Records in the sorted matrix that are near the boundary of two consecutive sub-problems can be similar to records in both sub-problems; therefore, after we have determined the equivalence class for such a boundary record in a sub-problem, we carry the entire equivalence class to the next sub-problem. A careful consideration is needed to ensure that the records indirectly equivalent to the boundary record can satisfy $k$-anonymity on their own, i.e. the total number of all indirectly equivalent records to this boundary record must be at least $k$; otherwise we have to also carry these records to the next sub-problem. Because the number of carry-over records from the previous sub-problem can be different, each individual sub-problem may end up with a different run-time size. In the worst case scenario, it is possible that an entire sub-problem gets carried over to the next. For example, suppose we have $k = 3$ with $S = 2$. We index the records in this sub-problem as $\{x_1, x_2, x_3, x_4, x_5, x_6\}$, one of two scenarios can happen: 1) The last $k$ records form an equivalence set within themselves. 2) Each of the first $k$ records is equivalent to at least one of the last $k$ records. In the first scenario the equivalence classes are exactly $\{x_1, x_2, x_3\}$ and $\{x_4, x_5, x_6\}$. In the second scenario we must have an equivalence class of the form $\{x_1, x_i, x_l\}$, $\{x_2, x_i, x_l\}$ or $\{x_3, x_i, x_l\}$ with $\{x_i, x_l\} \subset \{x_4, x_5, x_6\}$; in this case all of the records will be carried over to the next sub-problem. Intuitively as the size of the next sub-problem increases, it becomes unlikely that all of its records will be carried over further. Thus, we expect the sizes of the sub-problems to be bounded. We provide an upper-bound on the sizes in Lemma 2. Note that a more general implementation would be to carry over equivalent rows to the last $l$ records, where $l \neq k$ is also a customizable parameter. In the current implementation, $k$ affects the sub-problem sizes which becomes a problem as $k$ gets larger.

In Line 1 of Algorithm 1, we compute the variances of each column, where we then sort the matrix across entries with comparison order equal to the order of increasing variance in Line 2. We initialize the first set of carry-over records $C_0$ to be empty in Line 3 and initialize the objective value (i.e. total information loss) $f$ to be 0 in Line 4. We then loop over the chain of sub-problems in Line 6 - Line 11; for each sub-problem $Sub_m$ we get the set of next $k \times S$ records and add to it the carry-over records $C_{m-1}$ from the previous sub-problem in Line 6, where we then solve the resulting sub-problem by the general MILP to obtain anonymized outcome $A_m$ and objective value $f_m$ in Line 7; we then update the total information loss $f$ by adding to it the part of the objective value $f_m$ corresponding to non-carry over records (Line 8) and also store the subset of corresponding anonymized records in $A_m$ (Line 9) to $A$. In Line 10 we get the set of carry-over records $C_m$, which are records equivalent to the boundary records (i.e. the last $k$ records). We repeat Line 6 to Line 11 until we solve all sub-problems. We include a worked example in Section 3.3.3 to demonstrate the steps of this algorithm.

**Remarks:** The reason for carrying over records equivalent to the boundary records is two-fold: we have described that boundary records can be close to both sub-problems, and as equivalence is transitive this is true for their equivalent records as well. The equivalent rows also serve as initial equivalent rows to the next sub-problem, i.e. the optimal solution for the next sub-problem can in fact determine that the records carried over from a previous sub-problem are not closer to the new subset of records, and output the carry-over records as sets of equivalent rows the way they were fed in. The lower and upper bounds used in the sub-problems must be the same as those computed in the original problem, as the sub-problems should be optimized with respect to the original objective function.

LEMMA 2: The size of each sub-problem is bounded by $k \times (2k - 1 + S)$.

*Proof.* In each sub-problem, at the step where we carry over records the worst scenario happens when the last $k$ records are in distinct equivalence classes. For each boundary record, we carry over only records that are necessary, i.e., we carry over exactly $k$ records if the remaining indirectly equivalent records satisfy $k$-anonymity on their own (with size $\geq k$). Thus, the worst case happens when we have $(k - 1)$ indirectly equivalent records to each boundary record, then we must also carry over the entire set of such indirectly equivalent records. That is, we carry over $k \times (k + (k - 1))$ records to the next sub-problem, which contains $k \times S$ records before the carry-over. Thus, we have $k \times (2k - 1 + S)$ records in the resulting sub-problem in the worst possible case. □

In practice we expect that we should rarely encounter a sub-problem with size equal to its possible upper-bound; we provide the actual distributions of the sizes of the sub-problems for the tests of Fig. 3.5 in Appendix A.3, Table A.2.

THEOREM 1: An upper-bound for the complexity of the Split & Carry algorithm is:

$$O(\lceil \frac{n}{k \times S} \rceil \times 2^{p(k\times(2k-1+S)\times(\frac{k\times(2k-1+S)-1}{2}+2m))}). \tag{3.3.1}$$

*Proof.* Consider the complexity of the general MILP. Since non-integer linear programs (LP) have polynomial complexity (Dobkin and Reiss, 1980), in the worst case scenario the general MILP has to search all of the nodes and solve an LP at each node. Thus, an upper bound for the complexity of the general MILP is $O(2^{p(N)})$ where $p(\cdot)$ is a polynomial function, $N$ is the number of variables in the problem. In our formulation $N = \binom{n}{2} + 2 \times m \times n = \frac{n(n-1)}{2} + 2 \times m \times n$ for an input matrix of size $n \times m$. The Split & Carry algorithm solves a series of sub-problems each with at most $k \times (2k-1+S)$ records as shown in Lemma 2, where the complexity of each sub-problem is bounded by:

$$O(2^{p(k\times(2k-1+S)\times(\frac{k\times(2k-1+S)-1}{2}+2m))}). \tag{3.3.2}$$

Since we have to solve $\lceil \frac{n}{k\times S} \rceil$ such sub-problems, this gives expression (3.3.1). □

**Early Termination:** From expression (3.3.2) we see that the running time for solving a sub-problem can increase quickly as we increase $k, m$ or $S$. A feature of the Gurobi Solver is that it supports Early Termination, i.e., it can halt the search for feasible nodes after a pre-defined condition is reached. Thus, it is possible to set a strict time limit on the Split & Carry algorithm. For example, if one would like the algorithm to run for no more than $T$ seconds (ignoring overhead costs other than the solving of the sub-problems), one can set the "TimeLimit" parameter of the Gurobi solver to be $T/\lceil \frac{n}{k\times S} \rceil$ seconds. Inevitably, setting a time limit would reduce optimality of the solution.

**Categorical attributes encoding:** We have talked about the treatment of categorical data in our model in Section 3.2.5. If we decide to split up the categorical attribute into binary attributes we may encounter a limitation of the Split & Carry algorithm, as this will increase the number of attributes and the sizes of the sub-problems for this algorithm.

### 3.3.2 Greedy Search Algorithm

In this subsection we describe another algorithm which improves upon the initial feasible solution. As we observe in expression (3.3.1), while the complexity of Split & Carry algorithm scales linearly in the number of records as opposed to exponentially in the case of the general optimal problem, the complexity still scales exponentially in terms of the number of attributes and $k$. In this section we provide a greedy search algorithm which is not plagued with the *curse of dimensionality* (Aggarwal, 2005). This algorithm uses similar ideas as the algorithms presented by Doka et al. (Doka *et al.*, 2015), but it produces anonymized outcomes consistent with our formulation in Section 3.2.2 (i.e. containing sets of at least $k$ equivalent records) instead of outcomes appearing as in the *freeform* formulation of Doka et al. (Doka *et al.*, 2015).

In the Doka et al. algorithms, $k$ outer iterations are performed where at each iteration the original dataset structure is transformed into a complete bipartite graph[5]. In our algorithm, we perform a series of $k$ inner iterations. We again start from the initial feasible solution containing records sorted according to the order of increasing variance. Then for each indexed record $x_i$ in the sorted list, we construct the $k$-set ($Eq_i$) by iteratively choosing from the remaining records and adding to it, and subsequently removing the chosen record from the remaining set. In each iteration, we aim to generate the smallest objective value, $f_i := f(Eq_i)$, until $Eq_i$ contains $k$ records. Two advantages of our algorithm are: 1) it does not work with complete bipartite graphs and as such does not need to store weights for $n \times n$ edges; 2) it does not require a backtracking[6] process and thus does not need to store information from the previous states. Given these advantages the algorithm is more memory efficient and can be applied over datasets containing large numbers of records. However, we remark that both our Greedy Search and the Doka et al. algorithms (Doka *et al.*, 2015) have the limitation that they aim to find equivalence classes of size $k$. This is a major limitation of the greedy algorithm approach as in order to find a globally optimal solution the set should be allowed to contain many equivalence classes of different sizes greater than $k$. Our Greedy Search algorithm is described in Algorithm 2. Since this algorithm is not constrained by the number of attributes in terms of performance, we have the option of decomposing categorical attributes into vectors of binary attributes which is desirable to maintain better utility for categorical data.

The inputs to our greedy search algorithm are:

1. *arrType* an array describing the data type of each column (from {Integer, Continuous} or {Categorical, Numerical})
2. $x = [x_{ij}]_{i \in I, j \in J}$, the dataset in a matrix form
3. $U = [U_j]_{j \in J}$, $L = [L_j]_{j \in J}$, upper and lower bounds of each column

[5]A graph is complete bipartite if the set vertices can be partitioned into two disjoint sets where there is an edge from each vertex in one set to each vertex in the other.

[6]Backtracking describes a step that involves revisiting records that have been put into an equivalence class in a backward fashion as in (Doka *et al.*, 2015).

4. $W = [w_j]_{j \in J}$, weights for each column

5. $k$, minimum number of equivalent rows desired

```
Data: $x$, $U$, $L$, $W$, $k$, $arrType$
Result: $\mathbf{A}$, $\mathbf{f}$
1  $VAR = [VAR]_{j \in J} \leftarrow$ compute variances of all attributes;
2  $\tilde{\mathrm{x}} \leftarrow sort(x, VAR)$;
3  (Optional per $arrType$)Transform the columns containing categorical
     data into vectors of columns containing binary data;
4  $A \leftarrow \emptyset$; $\mathbf{f} \leftarrow 0$;
5  for $x_i \in \tilde{x}$ do
6      $Eq_i \leftarrow \{x_i\}$, initialize equivalent set $Eq_i$ for $x_i$;
7      $\mathbf{f_i} \leftarrow 0$;
8      remove $x_i$ from $\tilde{\mathrm{x}}$;
9      for $l = 1$ to $k - 1$ do
10         for $x_j \in \tilde{x}$ do
11             compute and store objective value $f_i^j := f(x_j \cup Eq_i)$ using
                 $arrType$;
12         end
13         find $x_{j'}$ that would give lowest $f_i^j$;
14         add $x_{j'}$ to $Eq_i$;
15         $\mathbf{f_i} \leftarrow \mathrm{update}(\mathbf{f_i}, f_i^{j'})$;
16         remove $x_{j'}$ from $\tilde{\mathrm{x}}$;
17     end
18     $\mathbf{f} \leftarrow \mathrm{update}(\mathbf{f}, \mathbf{f_i})$;
19     $\mathbf{A} \leftarrow \mathrm{update}(\mathbf{A}, Eq_i)$
20 end
21 return ($\mathbf{A}$, $\mathbf{f}$)
```

**Algorithm 2:** Greedy Search Algorithm

In Line 1 and Line 2 of Algorithm 2, we again compute the variances of each column, and sort the records as in the initial feasible solution and Algorithm 1. In line 3 we have an optional step to transform the columns with categorical attributes into vectors of columns of binary attributes. A discussion on how this is represented can be found in Appendix A.2. In Line 4 we initialize the total objective value $f$ to be 0, and the collection of equivalent sets $A$ to be empty. In Line 5 - Line 18 we loop over each remaining record of the sorted list; we initialize the equivalence class of record $x_i$ to be the set containing itself in Line 6, and the objective value $f_i$ for the equivalence class $Eq_i$ to be 0 in Line 7. In Line 8 we remove the record $x_i$ from the remaining records.

Then we perform $k-1$ iterations in in Line 9 - Line 17. In Line 10 - Line 17 we compute the objective value $f_i^j$ for each remaining record $x_j$, so that in Line 13 we can find the record that corresponds to incurring min loss if it is added to $Eq_i$, and it is then added to $Eq_i$ in Line 14. We then update the objective value $f_i$ to be the information loss corresponding to the new $Eq_i$ in Line 15, and remove the new member of $Eq_i$ from the remaining records in Line 16. At the end of $k-1$ iterations we obtain $Eq_i$ of size $k$, and we update the total information loss $f$ by adding $f_i$ to it in Line 18. We update the collection of equivalence classes $A$ with the new found $Eq_i$ in Line 19.

**Convergence:** Since we are always removing $k$ indices from $I := \{1, 2, ..., n\}$, when the size $n$ is a multiple of $k$, convergence is clear; if $n$ is not divisible by $k$, for each remaining index $j$ after the $\lfloor n/k \rfloor$ iterations, we distribute $x_j$ to the existing $k$-set that would incur the least utility loss if we were to add $x_j$ to it. Thus, the set of indices $I$ will be exhausted.

THEOREM 2: The complexity of the Greedy Search algorithm is:

$$O(\sum_{p=0}^{\lfloor n/k \rfloor} \{(k-1) \times (n-(p-1) \times k) + n)\}. \tag{3.3.3}$$

*Proof.* Each time we loop over an index we remove $k$ entries (including itself) from the dataset $x$. Thus, the outermost for-loop is repeated at most $n/k$ times. At the $p^{th}$ iteration of the outermost for-loop, we need to compute the min of $n-(p-1) \times k$ objective values, thus, the complexity of the middle for-loop is $O((k-1) \times (n-(p-1) \times k))$. If $n$ is not divisible by $k$, for each remaining record we distribute it to the equivalence class that would incur the least utility loss by adding this record; since there are $n/k$ equivalence classes and at most $k-1$ remaining records this last step has complexity $O(k \times \frac{n}{k})$. □

Given the above theorem, although greedy algorithm may provide relatively less favourable solutions, its complexity is much less dependent on the number of attributes and is bounded by $O(n^2)$ regardless of the choice of $k$.

### 3.3.3 Worked Examples

In this subsection we work through two examples to demonstrate the steps of the Split & Carry and Greedy Search algorithms. Our input dataset is as in Table 3.2a. We let $k = 3$ and $S = 3$.

**Example 1: Split & Carry**

The first step is to sort by variance as described in Section 3.2.3. The initial feasible solution creates 6 $k$-sets, namely $\{[12, 1, 11], [14, 7, 10], [16, 19, 13], [6, 9, 5],$

$[8, 2, 5], [17, 0, 8, 18, 3]\}$. The first sub-problem contains the first $3(S = 3)$ $k$-sets, and an initial solution with each $k$-set as equivalent rows is passed into to the Gurobi solver. The optimal solution for the first sub-problem is give in Table 3.3b. The $k$ boundary records are $\{16, 19, 13\}$, and they are equivalent to records $\{14, 7, 10\}$, thus the carry-over records from Sub-problem 1 to Sub-problem 2 are $\{14, 7, 10, 16, 19, 13\}$. The equivalence classes of the non-carried records become part of the final solution. The carry-over records are added to the remaining 3 $k$-sets to form Sub-problem 2 as in Table 3.4a. We then solve Sub-problem 2 and obtain the optimal solution in Table 3.4b. Putting the solutions together we obtain the outcome in Table A.1b. In this example, the optimal solution to Sub-problem 2 does not change the equivalence classes of the carry-over records, in general this need not be the case. We also note that the final solution provided by Split & Carry in this example is the same as the optimal solution by the general MILP.

(a) Sub-problem 1 Dataset

| index | AGE | SEX | INJ_SEV | DRINKING |
|---|---|---|---|---|
| 12 | 25 | 1 | 0 | 0 |
| 1 | 29 | 1 | 0 | 0 |
| 11 | 55 | 1 | 0 | 0 |
| 14 | 33 | 1 | 2 | 0 |
| 7 | 59 | 1 | 2 | 0 |
| 10 | 64 | 1 | 3 | 0 |
| 16 | 68 | 1 | 3 | 0 |
| 19 | 18 | 1 | 4 | 0 |
| 13 | 42 | 1 | 4 | 0 |

(b) Solution to Table 3.3a

| index | AGE | SEX | INJ_SEV | DRINKING |
|---|---|---|---|---|
| 12 | [25 - 55] | [1 - 1] | [0 - 0] | [0 - 0] |
| 1 | [25 - 55] | [1 - 1] | [0 - 0] | [0 - 0] |
| 11 | [25 - 55] | [1 - 1] | [0 - 0] | [0 - 0] |
| 14 | [18 - 42] | [1 - 1] | [2 - 4] | [0 - 0] |
| 7 | [59 - 68] | [1 - 1] | [2 - 3] | [0 - 0] |
| 10 | [59 - 68] | [1 - 1] | [2 - 3] | [0 - 0] |
| 16 | [59 - 68] | [1 - 1] | [2 - 3] | [0 - 0] |
| 19 | [18 - 42] | [1 - 1] | [2 - 4] | [0 - 0] |
| 13 | [18 - 42] | [1 - 1] | [2 - 4] | [0 - 0] |

Table 3.3: Sub-problem 1 of Worked Example 1

**Example 2: Greedy Search**

The first step is again to sort by variance as described in Section 3.2.3. Then we move down the sorted list of records. For each record we find $k-1$ records such that the least information loss is incurred when they are assigned to the $k$-set of this record. In this example, we first look at the record with index 12, and determine $\{1, 11\}$ to be in its equivalence class. We remove the records $\{12, 1, 11\}$ from the list. The next record in the remaining sorted list is 14, and we determine to $k$-set for 14 to be $[14, 7, 10]$ and remove these from the list. We continue in this manner until we are left with 2 records $\{4, 18\}$ in the list. For each remaining record, we find in our existing collection of $k$-sets the set that would incur the least information loss if the said record was added to it, and add the said record to this $k$-set. The complete set of iterations for this example is given in Table 3.5.

(a) Sub-problem 2 Dataset

| index | AGE | SEX | INJ_SEV | DRINKING |
|---|---|---|---|---|
| 14 | 33 | 1 | 2 | 0 |
| 7 | 59 | 1 | 2 | 0 |
| 10 | 64 | 1 | 3 | 0 |
| 16 | 68 | 1 | 3 | 0 |
| 19 | 18 | 1 | 4 | 0 |
| 13 | 42 | 1 | 4 | 0 |
| 6 | 49 | 1 | 4 | 0 |
| 9 | 50 | 1 | 4 | 0 |
| 5 | 59 | 1 | 4 | 0 |
| 8 | 80 | 1 | 4 | 0 |
| 2 | 42 | 2 | 0 | 0 |
| 15 | 31 | 2 | 2 | 0 |
| 17 | 20 | 2 | 4 | 0 |
| 0 | 64 | 2 | 4 | 0 |
| 4 | 53 | 1 | 2 | 1 |
| 18 | 40 | 1 | 4 | 1 |
| 3 | 41 | 2 | 4 | 1 |

(b) Solution to Table 3.4a

| index | AGE | SEX | INJ_SEV | DRINKING |
|---|---|---|---|---|
| 14 | [18 - 42] | [1 - 1] | [2 - 4] | [0 - 0] |
| 7 | [59 - 68] | [1 - 1] | [2 - 3] | [0 - 0] |
| 10 | [59 - 68] | [1 - 1] | [2 - 3] | [0 - 0] |
| 16 | [59 - 68] | [1 - 1] | [2 - 3] | [0 - 0] |
| 19 | [18 - 42] | [1 - 1] | [2 - 4] | [0 - 0] |
| 13 | [18 - 42] | [1 - 1] | [2 - 4] | [0 - 0] |
| 6 | [49 - 80] | [1 - 1] | [4 - 4] | [0 - 0] |
| 9 | [49 - 80] | [1 - 1] | [4 - 4] | [0 - 0] |
| 5 | [49 - 80] | [1 - 1] | [4 - 4] | [0 - 0] |
| 8 | [49 - 80] | [1 - 1] | [4 - 4] | [0 - 0] |
| 2 | [20 - 64] | [2 - 2] | [0 - 4] | [0 - 0] |
| 15 | [20 - 64] | [2 - 2] | [0 - 4] | [0 - 0] |
| 17 | [20 - 64] | [2 - 2] | [0 - 4] | [0 - 0] |
| 0 | [20 - 64] | [2 - 2] | [0 - 4] | [0 - 0] |
| 4 | [40 - 53] | [1 - 2] | [2 - 4] | [1 - 1] |
| 18 | [40 - 53] | [1 - 2] | [2 - 4] | [1 - 1] |
| 3 | [40 - 53] | [1 - 2] | [2 - 4] | [1 - 1] |

Table 3.4: Sub-problem 2 of Worked Example 1

| Index | Remaining Records | k-Sets |
|---|---|---|
| 12 | {14,7,10,16,19,13,6,9,5,8,2,15,17,0,4,18,3} | [12,1,11] |
| 14 | {16,19,13,6,9,5,8,2,15,17,0,4,18,3} | [12,1,11],[14,7,10] |
| 16 | {19,13,6,8,2,15,17,0,4,18,3} | [12,1,11],[14,7,10],[16,5,9] |
| 19 | {8,2,15,17,0,4,18,3} | [12,1,11],[14,7,10],[16,5,9],[19,13,6] |
| 8 | {2,15,4,18,3} | [12,1,11],[14,7,10],[16,5,9],[19,13,6],[8,0,17] |
| 2 | {4,18} | [12,1,11],[14,7,10],[16,5,9],[19,13,6],[8,0,17],[2,15,3] |
| 4 | {18} | [12,1,11],[14,7,10,4],[16,5,9],[19,13,6],[8,0,17],[2,15,3] |
| 18 | {} | [12,1,11],[14,7,10,4],[16,5,9],[19,13,6,18],[8,0,17],[2,15,3] |

Table 3.5: Steps of Greedy Search for Example 2

## 3.4 Experimental Evaluation

In this section, we present the experimental evaluation of our practical algorithms in terms of optimality, performance and scalability. In Section 3.4.1 we describe the dataset we have used for running the experiments and the experimental environment setup. In Section 3.4.2 we present the first set of experiments and provide the results from our algorithms along with results from the optimal solution (i.e. solution of the general MILP). In Section 3.4.3 we provide benchmark results of our algorithms against the other algorithms in terms of utility and running time. In Section 3.4.4, we report the results of the last set of experiments focusing on the scalability aspect of the algorithms. We selected three classes of $k$-anonymity algorithms for this study; namely heuristic-based algorithms by (Doka *et al.*, 2015), a partitioning-based algorithm by (LeFevre *et al.*, 2006) and a clustering-based algorithm by (Byun

| Attribute | Cardinality | [Min,Max] | DataType |
|---|---|---|---|
| \Gender | 2 | [1,2] | Categorical/Integer |
| \Age | 75 | [16,90] | Numerical/Integer |
| \Marital Status | 6 | [1,6] | Categorical/Integer |
| \Race | 7 | [1,7] | Categorical/Integer |
| \Birthplace | 83 | [1,426] | Categorical/Integer |
| \Education Level | 17 | [4,20] | Numerical/Integer |
| \Work Class | 7 | [13,29] | Categorical/Integer |
| \Occupation | 47 | [4,79] | Categorical/Integer |

Table 3.6: CENSUS dataset on 8 attributes

*et al.*, 2007). We provide a discussion and interpretation of the results in Section 3.4.5.

## 3.4.1 Experimental Setup

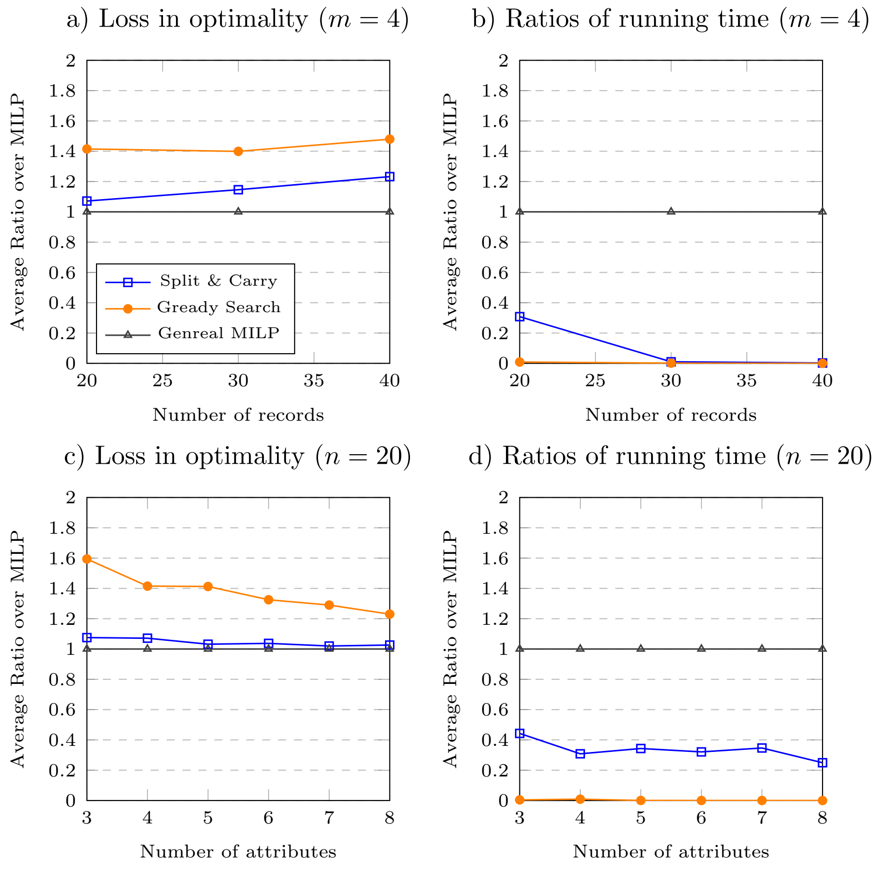
The main database for our tests is from IPUMS USA, consisting of 500,000 records of U.S. census data (University of Minnesota, 2018) (hereafter referred to as the CENSUS dataset). For each experiment we extracted datasets of various sizes from the main dataset as described in each experiment. Table 3.6 describes the characteristics of the main datasets with 8 attributes in terms of the attribute name, type (numerical or categorical), min/max values and number of distinct values available for each attribute. The characteristics of the extended datasets with up to 35 attributes are described in Table A.4 in Appendix A.4. The tests were performed on machines with 8 CPUs and 7.2 - 8 GB of RAM. The results were validated to be consistent across the machines. Moreover, tests appearing in the same figure were strictly run on the same machine.

To avoid database bias, we have reported the results of our experiments using an alternative database in Appendix A.5.

## 3.4.2 Optimality

To gauge the loss in optimality, we compare our practical algorithms (Split & Carry and Greedy Search) against the optimal formulation (general MILP with Gurobi solver) in terms of information loss, and from different dimensions. In Fig. 3.2a and Fig. 3.2c we show the averages of ratios of objective values from our algorithms over those from the MILP when the number of records grows and when the number of attributes grows, respectively. To reduce the impact of variability in the datasets the averages are computed over 20 datasets randomly drawn from the CENSUS database. The number of attributes is set to 4 for the results illustrated in Fig. 3.2a and Fig. 3.2b, and the number of records is set to 20 for the results shown in Fig. 3.2c and Fig. 3.2d. We observe that consistently our Split & Carry achieves better utility than Greedy Search. An interesting finding from this experiment is that increasing the number of attributes does not increase loss in optimality for both algorithms.

Figure 3.2: Optimality and performance of Split & Carry and Greedy Search ($k = 3$)



We also show the averages of ratios for the running times on the same datasets when the size of dataset increases and when the number of attributes grows in Fig. 3.2b and Fig. 3.2d, respectively. We see that there are significant performance benefits in our algorithms especially as we increase the size of the dataset. In particular we observe that the running time for Greedy Search compared to the optimal solution is negligible. Although we see in Fig. 3.2 that Greedy Search has significantly better performance than Split & Carry, it is not generally the case and we show this as we further compare the running times of these two algorithms for large datasets in the scalability section. The difficulty in measuring the complexity of the Split & Carry algorithm comes from the difficulty in providing an exact or even tighter upper bound in the complexity of any specific instance of the MILP. The complexity for Split &

Carry we've provided in Theorem 1 is merely an upper bound, and in practice as the size of our dataset increases we expect a much better performance than what we presented in Theorem 1. Nonetheless, expression (3.3.1) in Theorem 1 provides the relationship between the number of records $n$ and complexity of the Split & Carry; that is, the complexity scales linearly in the number of records, which we also demonstrate in the scalability section.

### 3.4.3 Benchmarking

We first benchmarked our algorithms with the three algorithms presented by Doka et al. (Doka *et al.*, 2015), hereafter we refer to these as: Doka Greedy, Doka SortGreedy and Doka Hungarian. We implemented all three algorithms in Python. While we have confirmations from the first author regarding specific details in the implementations, we acknowledge that there could be implementations that can give slightly better running times than what we will display here; for example we had used the Munkres module[7] for the Hungarian algorithm combined with the iteration scheme described in (Doka *et al.*, 2015) for the implementation of Doka Hungarian. Despite potential differences in the implementation details, however, two important facts provide a sound basis for any conclusions we draw from our experiments:

1. As consistent with the results in (Doka *et al.*, 2015), in general the Doka Hungarian algorithm provides the best utility among all three Doka algorithms.

2. By theoretical construction, our Greedy Search algorithm has better complexity than all three Doka algorithms.

We also benchmarked our algorithms against the Mondrian Multidimensional $k$-Anonymity algorithm (LeFevre *et al.*, 2006) (hereafter referred to as Mondrian), and we were able to download the program directly from (UT Dallas Data Security and Privacy Lab, 2012). The Mondrian algorithm is a partitioning-based algorithm in which the dataset is repeatedly partitioned into disjoint subsets according to some statistic in a tree-like manner. The Mondrian algorithm can produce anonymized outcome as generalized ranges, it can also produce outcome providing complete distribution information for categorical data; however the Mondrian does not output a utility measure. We could compute our utility measure directly on the Mondrian, but the algorithm provides open bounds for some records and in fact it does not always provide the tightest possible bounds; e.g., in the anonymized outcome by Mondrian in Fig. 3.5, for one equivalence class on the Age attribute it produced generalized range (60, 90] while the tightest bounds for this equivalence class were [61,

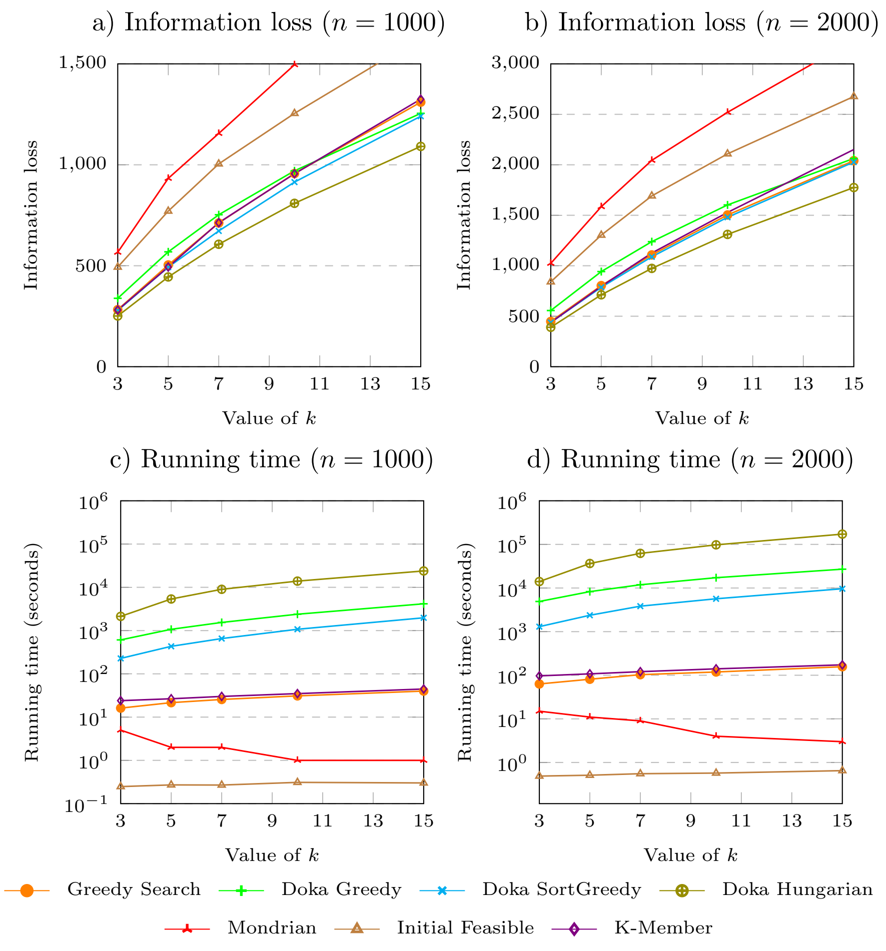
[7]munkres 1.0.12: https://pypi.org/project/munkres/

80]. We make the following notes for the benchmarking results we present on Mondrian:

1. Since it takes a simple post-processing step to find the tightest bounds for any equivalence class, utility values were computed on the equivalence classes using the tightest bounds.

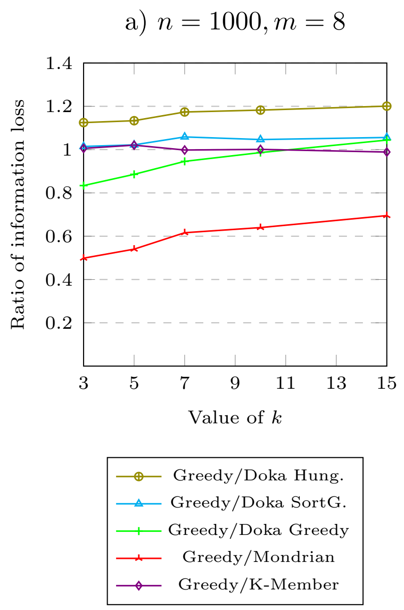
2. The running times presented for Mondrian do not include the time it took to compute utility; however the Mondrian has theoretical complexity equal to $O(n \log(n))$ where $n$ is the size of the matrix.

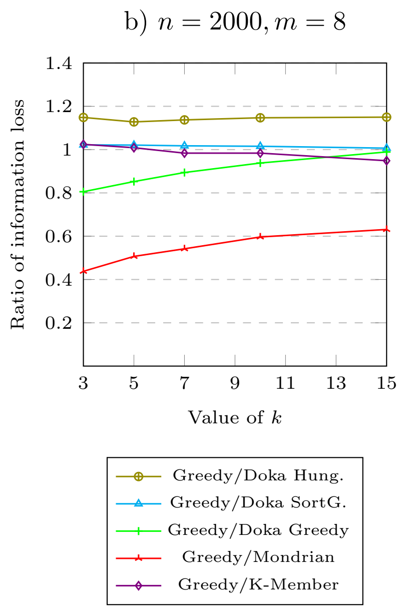
Figure 3.3: Benchmarking Greedy Search against other algorithms ($m = 8$)



Finally we benchmarked our algorithms against the K-Member Clustering algorithm by (Byun *et al.*, 2007) (hereafter referred to as K-Member). K-Member is a clustering algorithm where the algorithm forms clusters of size $k$ as equivalence classes. There are similarities between K-Member and our

Figure 3.4: Ratios of objective values of Greedy Search over other algorithms

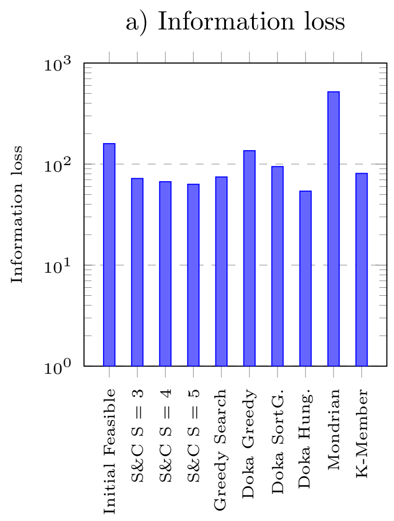


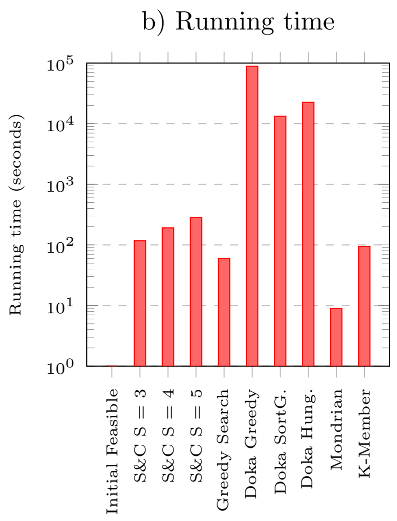


Greedy Search algorithm, in that for the record currently under consideration, both algorithms greedily finds the next record to be in its equivalence class by minimizing the additional information loss incurred. Apart from incorporating weights, the most differentiating step in Greedy Search is the order with which we move to the next record. The basis of Greedy Search is to improve upon the initial feasible solution.

We present our benchmark results against Mondrian, K-Member and Doka algorithms in Fig. 3.3 and Fig. 3.4 with various values of $k$, where we use the GCP to measure utility as the GCP had been used in other literature previously and so does not put our algorithm at an advantage (please see Appendix A.2 for GCP). This also shows that our Greedy Search algorithm can be adapted to other utility measure since its optimality and performance do not depend on the choice of utility measure. In the tests reported in Fig. 3.3a and Fig. 3.3c we used 1000 records on all 8 attributes, and in Fig. 3.3b and Fig. 3.3d we used 2000 records on the 8 attributes. We can see that Greedy Search can achieve utility that is not too far from those achieved by Doka algorithms but it has a much better running time; while the Mondrian has a very efficient running time and always finishes in a few seconds, the utility it provides is far less favourable. Our Greedy Search performs similarly to K-Member, but as the value of $k$ increases we begin to notice a slight advantage. We can also observe that as we increase the dataset size from 1000 records to 2000 records, the gap in the utility between our Greedy Search and the Doka algorithms narrowed, and in particular the utility of our Greedy Search converged to that

Figure 3.5: Benchmarking Split & Carry and Greedy Search against other algorithms ($n = 5000, m = 4$)



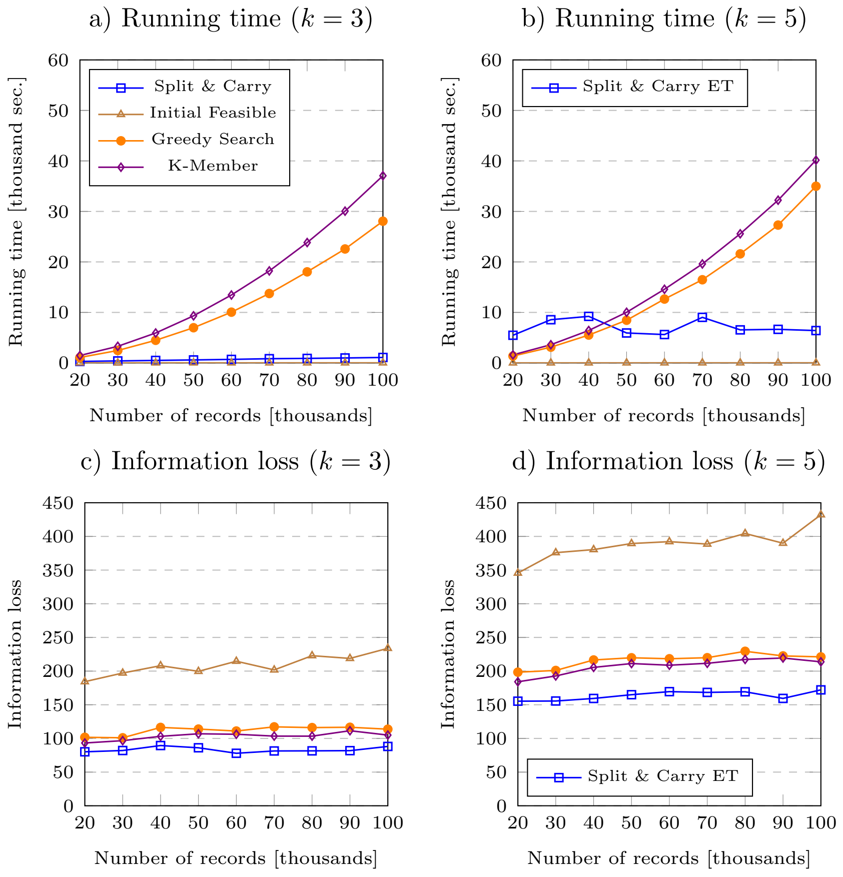


of Doka SortGreedy. We also display the ratios of objective values of our Greedy Search over Mondrian, K-Member and the Doka algorithms. We see that for small $k$, Greedy Search is performing better than Doka Greedy, but as $k$ increases Doka Greedy starts to have advantage over our Greedy Search. The general trend for the tests on 1000 and 2000 records is, when we fix the size of the dataset, as $k$ increases the utility of the Doka algorithms increases. We also see in general that for a fixed $k$ the utility gap between our Greedy search and Doka Hungarian decreases, when we increase the dataset size. That is, as we increase the size of the dataset, the probability of having records that are equivalent or very similar increases. Therefore, even though Doka algorithms produce outcomes that assume a *freeform*, which has a larger solution space than our anonymized outcomes (containing equivalence classes of size at least $k$), the utility gain from using freeform is small as the anonymized outcomes of these algorithms contain many records that differ in a small number of attributes by small amounts. Fig. 3.4a and Fig. 3.4b also show that as we increase from 1000 to 2000 records, the utility of Greedy Search increases relative to all algorithms (as the ratios of objective values over Doka, Mondrian and K-Member decrease).

In Fig. 3.5 we compare the objective values and running times of our algorithms (Split & Carry with $S = 3, 4, 5$ and Greedy Search) against the Doka, Mondrian and K-Member algorithms, on a dataset of 5000 records with 4 attributes, where the attributes are treated as numerical. We see that for a dataset with a reasonable size and small number of attributes, and for small

$k$ $(= 4)$, our algorithms have utility close to Doka Hungarian but with better performance, and our algorithms perform better against Doka Greedy and Doka SortGreedy in both utility and performance; whereas the Mondrian has much better performance but also far less utility than our algorithms. However, if the user is only interested in performance, then we recommend the initial feasible solution which provides decent utility and efficient performance and in this test it performs better than Mondrian in both aspects.

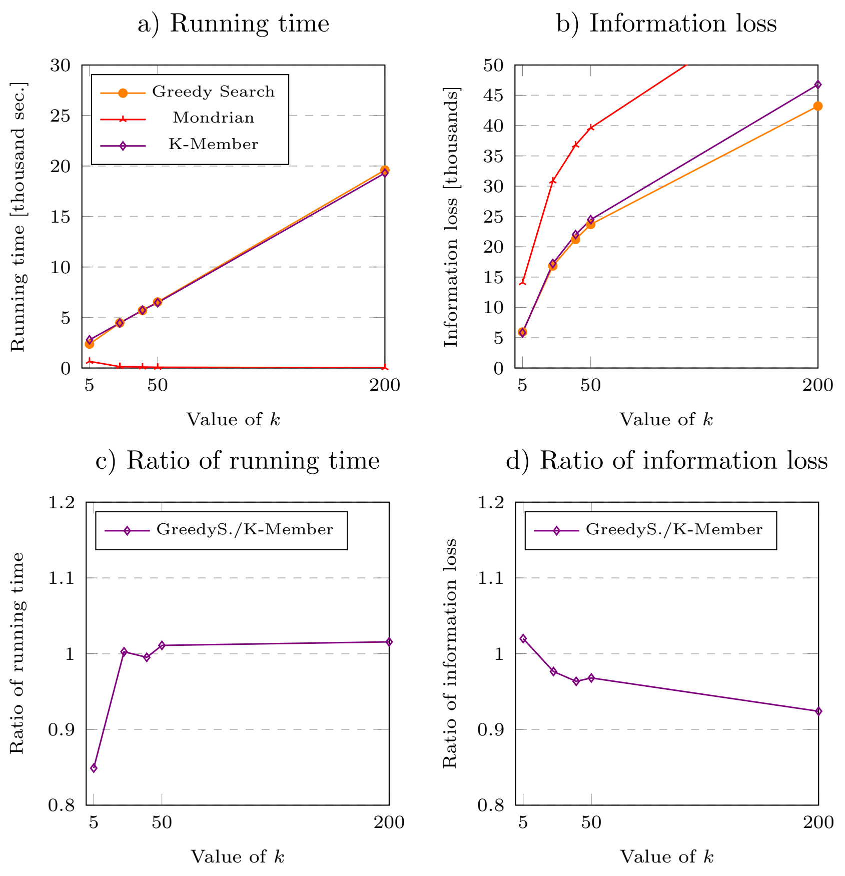
Figure 3.6: Scalability of Split & Carry and Greedy Search in $n$ for smaller $k = 3, 5$ $(m = 4)$



## 3.4.4 Scalability

In this section we examine the scalability of the algorithms. The tests in this section are based on datasets containing at least 20,000 records. We do not include the Doka algorithms in our scalability evaluation due to two reasons: 1) we observed in Section 3.4.3 that the Doka algorithms took 3 - 24 hours to anonymize 5000 records (Fig. 3.5b); 2) we tried to run the Doka algorithms

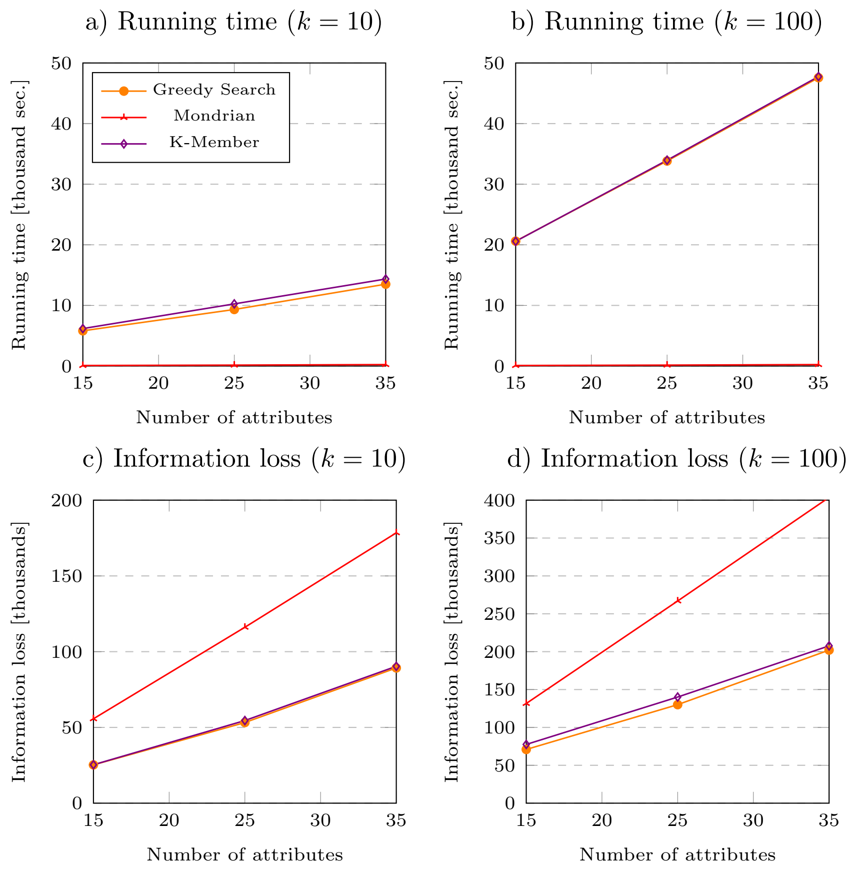
Figure 3.7: Scalability of Greedy Search in $k$ ($n = 20,000$, $m = 8$)



on a dataset with 10,000 records but we received an "OutOfMemoryError". Indeed, the Doka algorithms require storage of previous states for backtracking as well as representation of $n \times n$ edges for complete bipartite graphs, where the latter requires storing 100,000,000 numbers when $n = 10,000$. We consider scalability of the remaining algorithms in terms of the number of records as well as the number of attributes in Fig. 3.6 - Fig. 3.8, where the attributes are treated as numerical.

In Fig. 3.6 we show the running time and information loss values of the algorithms for small values of $k$ ($= 3, 5$) on several CENSUS datasets with 4 attributes. The sizes of the datasets range from 20,000 to 100,000. Note for the case of $k = 5$ in Fig. 3.6b and 3.6d we used the Early Termination (ET) feature for Split & Carry, and set "TimeLimit" to be 1000 seconds for each sub-problem, though only very few sub-problems actually triggered ET. We see that for a small number of attributes and small values of $k$, K-Member has slightly better utility than Greedy Search, but Greedy Search has better

Figure 3.8: Scalability of Greedy Search in $m$ ($n = 20,000$ and $k = 10$ or $100$)



running time. We also see that Split & Carry significantly outperforms K-Member and Greedy Search in both running time and utility. The efficiency of Split & Carry is due mainly to two aspects: 1) it solves small sub-problems and efficiency is reaped from state-of-the-art MILP solver; 2) the small sub-problems have small solution spaces since it's likely that there are identical or very similar records in a sub-problem as the dataset contains a large number of records. The results for Mondrian were omitted from these figures for better visual presentation of the other algorithms. Although Mondrian always finishes in a few minutes in all of the tests in this experiment, it has information loss ranging 9 - 56 times that of the initial feasible solution.

In Fig. 3.7 we look at how the algorithms react to increasing values of $k$. We acknowledge that Split & Carry is not suited for large values of $k$ and thus it has been excluded from all the experiments with large $k$. We use a CENSUS dataset with 20,000 records and 8 attributes. We see that while Mondrian has negligible running time compared to other algorithms, its utility is much less. Moreover, the utility gap between Mondrian and the other algorithms increases as $k$ increases. On the other hand, we see that although Greedy Search has slightly less utility than K-Member for small $k$ $(= 5)$, but as $k$ increases it starts to gain utility over K-Member and this utility gap further widens for large $k$ $(= 200)$.

In Fig. 3.8 we look at how the algorithms perform when we increase the number of attributes. We use CENSUS datasets containing 20,000 records on 15, 25 and 35 attributes. We observe that Mondrian is still taking minimal time to compute, however the utility gap between Mondrian and other algorithms increases as the number of attributes increases. At small $k$ $(= 10)$ we see that Greedy Search and K-Member have almost the same utility. At $k = 100$ we observe again that Greedy Search has slightly better utility. Interestingly, we observe consistent results across different numbers of attributes, it appears only the value of $k$ affects how these two algorithms perform compared to each other.

### 3.4.5 Discussion on Evaluation Results

In the Optimality experiments in Section 3.4.2 we see that our practical algorithms provide near-optimal utility for a small $k$ with significant performance improvement over the general MILP. In some cases, Split & Carry actually provides the same optimal objective value as the MILP. We see in the Scalability experiments that for a small number of attributes and small $k$, Split & Carry delivers excellent performance and utility. Split & Carry is distinguished from the other algorithms in that it always produces optimal solutions for small subsets of the original dataset, and the chaining of the sub-problems enables (in some cases) the possibility of a global optimal solution. In Split & Carry, many equivalence classes might have size greater than $k$ as long as doing so minimizes the information loss. On the other hand, the Greedy Search, K-Member and Doka algorithms are greedy algorithms but their common focus is on finding equivalence classes of size $k$. Other than indirectly equivalent records, only in cases where the number of records is not divisible by $k$ one would see a small number of equivalence classes with size greater than $k$ in these algorithms. In the case of the Mondrian partitioning algorithm, while it is true that records in the same partition are *close* to each other, the boundary records of two adjacent partitions are also *close*. Since the partitions are determined by median values, utility might not be preserved as such boundary records might be closer to each other than to the records in their own partitions. Thus Split & Carry features two important aspects which are crucial to solving the $k$-anonymity problem optimally in an efficient way: 1) that the construction of the equivalence classes is not constrained to finding sets of size $k$, but also any size greater than $k$; 2) when we work with smaller subsets for efficiency reasons, there is possibility to move records from one subset to another, i.e. we do not lose the link to the original dataset.

The biggest challenge for our Split & Carry algorithm is still in the size of the sub-problems. For efficiency we are limited to only small sub-problems on small numbers of attributes with small values of $k$ ($\leq 6$). In practice we often see $k = 3$ or 5 being cited as the minimum requirement (Statistics Canada, 2014; National Center for Health Statistics, 2019; Cancer Care Ontario, 2018; Canada Institute for Health Information, 2014). Therefore our Split & Carry algorithm can still be a useful tool for $k$-anonymization.

We see that in the Benchmarking experiments in Section 3.4.3, the Doka algorithms in general have better utility than our practical algorithms. We discussed in Section 3.4.3 that this utility gain in Doka algorithms, comes mainly from the fact that their $k$-anonymized outcomes take a *freeform*, which does not make $k$ equivalent records in the anonymized outcome but rather that for each anonymized record one can identify $k$ records from the original dataset that fall into the ranges of the anonymized record. Moreover, we see in our experiments that for a fixed value of $k$, as the number of records increases, the utility gap between the Doka algorithms and Greedy Search decreases; in particular Greedy Search performs similarly to Doka Greedy and Doka SortGreedy in Fig. 3.3b where $n = 2000, m = 8$, and Split & Carry is close to Doka Hungarian in Fig. 3.5a where $n = 5000, m = 4$. However, our algorithms have much better running time. In general, we expect that for datasets where the value of $k$ and $m$ are large relative to $n$ (e.g. $n < 5000, m = 8, k > 15$) the Doka algorithms will have the best utility due to the *freeform* formulation; but scalability becomes an issue for Doka algorithms for large-sized datasets due to their theoretical complexities and their need to store previous states and $n \times n$ edges. In the cases where $k$ is small relative to $n$, then the advantage of *freeform* diminishes due to high probability of having many similar records in the dataset.

Throughout the Benchmarking and Scalability experiments, we observe that Mondrian has efficient running time but much larger information loss compared to other algorithms. Thus, it is only scalable in terms of running time; but we note that still the initial feasible solution provides better running time and even better utility than Mondrian. Moreover, the utility gap between Mondrian and Greedy Search increases with the value of $k$ and the number of attributes, as can been seen in Fig. 3.7b, Fig. 3.8c, and Fig. 3.8d.

The initial feasible solution captures an ordering that gives Greedy Search an advantage over K-Member for large values of $k$. Consider a sorted list of records, such as the one in Table 3.2b. Notice that the top record appearing in the sorted list is only close to the rest of the list in one direction, i.e., it is only close to records that are immediately below it in this list. Suppose we begin constructing the equivalence class for record $i$, where $i$ is not the top record (e.g. in this example $i \neq 12$). If we remove records that are close to 12, then 12 becomes farther away from the rest of the list, and more information loss would be incurred to bring 12 to another equivalence class. The larger the value of $k$, the more records we are likely to remove that are close to 12 before we get to it. Now if we first start with 12, then after removing the equivalent records for 12, we will have a new top record in the remaining list and the process repeats. This is the logic behind the choice of ordering to construct equivalence classes for Greedy Search. We see the effectiveness of this ordering in the Benchmarking and Scalability experiments, where in Fig. 3.3, Fig. 3.4, Fig. 3.7 and Fig. 3.8, Greedy Search has consistent better utility (though sometimes only slightly) than K-Member for larger values of $k$ (e.g. $k \geq 15$).

## 3.5 On Alternate Solution Forms

In the work by Doka et al. (Doka *et al.*, 2015), they offer a MIP (Mixed Integer Program) formulation for the *freeform* $k$-anonymity problem. The formulation by Doka et al. is freeform in the sense that a record can be assigned to many different equivalent classes and such equivalence is not transitive. This gives rise to a larger solution space and thus better utility in the optimization problem. In freeform formulation, each anonymized record matches $k - 1$ other records of the original dataset, but the number of matches is unknown when we link it to an externally available dataset. Because the data custodian cannot check every external table, Samarati (Samarati, 2001) argued that $k$-anonymity requirement should be satisfied in the anonymized data themselves.

On the same note as freeform $k$-anonymity, we also remark that there are related algorithms which seek to provide weaker forms of $k$-anonymity (Gionis *et al.*, 2008; Tassa *et al.*, 2012) that do not make $k$ identical records, but either that an original record is consistent with $k$ records in the anonymized dataset ((1,$k$)-anonymity); or that an anonymized record is consistent with $k$ records in the original dataset (($k$,1)-anonymity), or both (($k$,$k$)-anonymity). Tassa et al. (Tassa *et al.*, 2012) further introduced *$k$-concealment* that is a stronger variant of (1,$k$)-anonymity which they argued to offer the same security guarantee as $k$-anonymity *computationally*. This gives an interesting research direction for future work, as our Split & Carry can theoretically be adapted to these forms of $k$-anonymity by relaxing some constraints.

## 3.6 Concluding Remarks

In this chapter we introduced a mathematical formulation for $k$-anonymity as a Mixed Integer Linear Program with a weighted objective function, where the weights can be customized to adjust the likelihoods of generalization on the attributes to suit different research uses. Recognizing that MILPs are hard in general (Papadimitriou, 1981; von zur Gathen and Sieveking, 1978), we also introduced two practical algorithms, both of which are memory efficient and can be applied to very datasets containing large numbers of records. We evaluated our algorithms based on their optimality, performance and scalability; we also provided benchmark tests against other classes of algorithms (heuristic-based, partitioning-based and clustering-based). We demonstrated that our Greedy Search algorithm can either achieve similar utility with more efficient running time, or better utility with a less efficient yet still very good running time. We also provided tests to show a remarkable improvement in performance and utility in our Split & Carry over other algorithms when the number of records is large and the number of attributes is small, for small $k$. From our experiments and theoretical justifications, we conclude that both our algorithms are suitable tools for anonymization on large datasets. We recommend that the Split & Carry algorithm be used when the number of attributes is small and the dataset is very large, for common values of $k$ which are small. When we have many attributes or if we are interested in a larger $k$ value, the recommended choice is Greedy Search whose complexity is always bounded by $O(n^2)$ regardless of the number of attributes and choice of $k$.

The $k$-anonymity model as a privacy preservation technique has a number of limitations that impact our formulation. When an adversary is in possession of external knowledge that can be linked to the anonymized dataset, the adversary might be able to deduce additional information about an individual; some of such adversarial scenarios are described by De Capitani Di Vimercati et al.(De Capitani Di Vimercati *et al.*, 2012). In these scenarios there are attributes that contribute more to potential re-identification of an individual than others (e.g., a person with a particular birth date who lives in an area of a particular zip code). In our formulation, we can assign smaller weights to these attributes to over generalize them, thus decrease the risk of re-identification. There is, however, one concern with using weights; in a scenario that an adversary has access to multiple datasets originated from the same dataset but anonymized by different weight vectors, the adversary might be able to infer additional information on the original dataset. In particular, in each anonymized set, the attribute with the largest weight has the smallest gap in its generalized bounds. Thus by combining different datasets the adversary can arrive at a much finer dataset. This problem can be mitigated by randomizing the anonymized datasets before their release, however the adversary might still be able to detect additional patterns. A more in-depth study to address the security properties of applying weights presents an interesting future research direction.

# Chapter 4

# Towards Robustness in Neural Networks

In this chapter, we present our approach for improving robustness in neural networks. In Section 4.1 we provide some background information on adversarial examples, including a brief account from the first published attack to a recent development that aimed to defend against these attacks, and a generic formulation for adversarial examples. In Section 4.2.1 we briefly describe our intuition behind the mechanisms used in our approach, with technical descriptions of the mechanisms ensuing in Section 4.2.2 and Section 4.2.3. We present the setup and testing results of our experiments in Section 4.3, with a discussion on the results in Section 4.5. We conclude the chapter in Section 4.6

## 4.1 Preliminary

### 4.1.1 L-BFGS

The problem of creating an adversarial example is to perturb some pixels on an image such that the classifier will label the perturbed image differently from the original image's correct label. These adversarial examples are generally categorized as being *targeted*, where the adversary carefully adds perturbations to cause the classifier to output a specific incorrect label; or *untargeted*, in which the adversary's goal is to cause the classifier to output any incorrect label. Naturally, one can think of both types of adversary examples as targeted with a set of target labels. In the former case there is only one label in the target set, whereas in the latter case the target set consists of all incorrect labels. Intuitively if one keeps adding random perturbations to an image, eventually even the most sophisticated classifier (including humans) will start to produce incorrect labels. Moreover, such perturbed images likely do not appear genuine. Thus, for an adversarial example to be considered successful, the amount of perturbations to be made to the original image must be minimal. Formally, we can define the problem of finding adversarial examples as

$$\begin{aligned} &\min_{x' \in D} ||x' - x|| \\ &s.t.\ F(x') = t \end{aligned} \tag{4.1.1}$$

for some classifier $F(\cdot)$, where $t$ is the target label, $x$ is the original input image. $D$ is the domain of all images which are usually represented as a set of multi-dimensional vectors, where the permissible values fall within a bounded range, with some distance metric $|| \cdot ||$. This definition is used in various literature up to notational differences.

The first published attack (Szegedy *et al.*, 2014) on neural networks arose by solving the above problem using a box-constrained L-BFGS (limited-memory BFGS, (Liu and Nocedal, 1989)) method, where $D = [0, 1]^m \subseteq \mathbb{R}$. In the L-BFGS example, the above problem is first transformed from a constrained optimization problem to an unconstrained optimization problem (up to the box constraints) using a penalty method, where the penalty function $L_F(\cdot)$ is the loss function applied to $x'$ and the target $t$. In other words, for constant $c > 0$, the above problem becomes:

$$\min_{x' \in D} c \cdot ||x' - x|| + L_F(x', t). \tag{4.1.2}$$

The loss function is appealing as the penalty function because it's non-negative and captures the distance between $F(x')$ and $t$ with the same metric used in the training of the network. Moreover, the loss function is zero exactly when the original constraint $F(x') = t$ is satisfied. Thus an optimal solution to the original problem will also be an optimal solution to the transformed problem.

L-BFGS belongs to the class of line search methods which are commonly used for solving optimization problems over a compact domain. A line search method is an iterative scheme, and typically involves repeatedly identifying a search direction and moving the feasible point along the search direction. The scheme will converge to a global optimum if the formulation satisfies certain conditions. In the problem above, if the loss function is sufficiently differentiable, then a global optimum always exists. In other words, an adversarial example is guaranteed to exist; however the perturbation $||x' - x||$ is not necessarily small.

### 4.1.2 Defensive Distillation

Following the invention of the L-BFGS adversarial examples, other methods such as (Goodfellow *et al.*, 2015; Kurakin *et al.*, 2017; Papernot *et al.*, 2016b) soon emerged which further accentuated the presence of significant vulnerabilities in classification-type networks. Shortly after these inventions, a technique for training classification-type networks, known as Defensive Distillation (Papernot *et al.*, 2016a), was devised. Defensive Distillation employs the idea of using a *teacher* network to train a second network (*student*), in which the soft labels (each containing a vector of probabilities rather that a single final label) of the teacher network are used for training the student network. Moreover, a different temperature constant at the last layer is used in training the student than it was used in training the teacher. The authors in (Papernot *et al.*, 2016a) suspected the success of Defensive Distillation could have been due to additional hidden knowledge learned by the student from the soft labels, as well as the higher temperature constant used in the student training, as it softens the probabilities in the output layer and thus decreases *over-fitting*.

### 4.1.3 Carlini-Wagner Attack

Defensive Distillation was soon defeated by the Carlini-Wagner attack (Carlini and Wagner, 2017). The success of Carlini & Wagner demonstrated the important fact that the prior success of Defensive Distillation was indeed due to the different temperature constants used between the *teacher* and *student* network, as well as the assumption that attacks would be crafted based on the original temperature (same as used for the teacher). In particular, Carlini & Wagner highlighted a common characteristic in the prior attacks that Defensive Distillation successfully defended against: that the attacks depend on the the gradient of the network, either used as a multiplicative term in the search direction or to determine the amount of perturbations to add to specific pixels.

The softmax function is usually used for classification-type networks at the second last layer to normalize outputs into discrete probabilities, i.e., for network $F$, we can write $F := \sigma \circ G$ for softmax function $\sigma$, where $G$ is the composition of all previous layers. As such, the gradient of the network contains the derivative of the softmax function as a multiplicative term:

$$\frac{\partial F}{\partial x_k} = \frac{\partial F}{\partial z^j}(\frac{\partial z^j}{\partial x}) = \frac{\partial \sigma(z^j)}{\partial z^j}(\frac{\partial z^j}{\partial x}) = \frac{1}{T} z^j (\frac{\partial z^j}{\partial x_k}), \tag{4.1.3}$$

for input pixel $x_k$, where the softmax function $\sigma(\cdot)$ at the second last layer for the $j^{th}$ classification is

$$\sigma(z^j) = \frac{e^{z^j/T}}{\sum_i e^{z^i/T}}, \tag{4.1.4}$$

and $z^j := [G(x_k)]_j$ is the $j^{th}$ component of the output from the previous layers. In Defensive Distillation, a large temperature $T$ is used for the student network, but the attacks are assumed to be crafted with $T = 1$ in the softmax function.

The larger $T$ reduces the gradient by $T$, essentially disabling the search to advance toward an optimal solution. Thus, it becomes clear that a *tampered* gradient can impair the advance of any adversarial attacks crafted using the gradient of the original network function. Noting this observation, in order to remove the dependency on the gradient of the network, Carlini & Wagner re-formulated the problem in (4.1.2). In particular, they introduced different penalty functions which do not depend on the original network output; instead they advertised the use of penalty functions that depend on the output at the second last layer (i.e. the *logits*). Thus, a more general formulation for finding adversarial examples is, for penalty function $L(\cdot)$ not necessarily equal to the original loss function:

$$\min_{x' \in D} c \cdot ||x' - x|| + L(x', t). \tag{4.1.5}$$

Since the softmax function applied at the last layer is monotonic, the final output is already decided at the second last layer, thus equivalent penalty functions to that in (4.1.2) can be created which do not depend directly on the original output. Moreover, the logits do not depend on the temperature constant and thus are not impacted by larger temperature constant used in Defensive Distillation.

## 4.2 The Model

### 4.2.1 Motivation

In the previous section, we discussed that the success of attacks prior to Defensive Distillation relied on the accessibility and integrity of the gradient of the network. From Carlini & Wagner we learned that if we remove the dependency of the attack formulation on the gradient, such attack will defeat Defensive Distillation. Carlini & Wagner advertised the use of logits to design a penalty function that achieves an equivalent effect. However, such formulation then relies on the accessibility and integrity of the logits, thus a defensive mechanism that impairs integrity of the logits can potentially resist a Carlini-Wagner attack. In the next section, we propose a simple mechanism due to this motivation, where random noise is added at query time to obscure the solution of the optimization problem for creating adversarial examples.

We are also motivated by the success of PATE (Papernot *et al.*, 2017) to devise another mechanism, where although our specific concern is not in data privacy, the idea of having an ensemble of networks and voting allows the correct decision on output to be made in situations when some of the networks are under attack.

In a way, voting can be seen as adding back information that could have been lost in partitioning the dataset. Each network trained on a sub-dataset might contain information not present in another network trained on a different sub-dataset. When a large subset of networks agree on the output of a single query, intuitively one can think that a property of the original parent dataset has been captured by these sub-datasets. Thus, an output is more likely to be correct if it was obtained by voting among an ensemble of networks, than if it were the output of a single network. In other words, voting improves accuracy.

### 4.2.2 Noisy Logit

Since Carlini-Wagner attacks require access to the logits in the solution to (4.1.5), we can obscure the search for solution by adding random noise to the logits. Note that if we add random noise directly to the original logits, an adversary might recover the original logits by making multiple queries and averaging the resulting *noisy* logits. Instead, we apply random noise at query time to the input, then respond to the query with the logit of the perturbed input. Also, since the softmax function is monotonic, we must ensure the final output is a result of the *noisy* logit, otherwise the genuine logit can be recovered by applying the inverse of the softmax function to the result at the output layer. Let $z_0$ be an input with $F(z_0)$ its output from the network $F$. Moreover, suppose $F$ has $n$ layers besides the input layer and for $1 \leq i \leq n$

$$z_i := F_i \circ F_{i-1} \circ \cdots \circ F_1(z_0), \tag{4.2.1}$$

where $\circ$ denotes composition and

$$F(z_0) := z_n = F_n \circ F_{n-1} \circ \cdots \circ F_1(z_0). \tag{4.2.2}$$

Then at the output layer $i$, the Noisy Logit mechanism will produce output

$$F_i(z'_{i-1}) = F_i \circ F_{i-1} \circ \cdots \circ F_1(z'_0), \tag{4.2.3}$$

where

$$z'_0 = z_0 + g(\vec{a}) \tag{4.2.4}$$

and $g(\vec{a})$ is a random noise function with parameter(s) $\vec{a}$. Note that by this procedure, naturally we can respond to queries at any layer with a noisy output, thus preventing an adversary from trying to reconstruct a genuine output at any layer (including the logits) by making queries to the network.

### 4.2.3 Ensemble Voting

The purpose of having an ensemble of networks is two-fold: 1) to provide resilience in the *combined* network when some of the networks are under attack; 2) to improve accuracy in the *combined* network over the individual networks. If we require that each individual network must successfully classify the input image, then assuming independence of success probabilities across the networks, the probability of simultaneous success across the networks is the product of the success probability of each network, which might be less than a desirable level of accuracy if the total number of networks is large since we are multiplying a series of numbers less than 1. However, if we only require success in the largest subset of the networks, then since there are many possible permutation of subsets when the number of networks is large, the success probability of the combined network as an aggregate might be much better than that of each individual network. Let $S := \{F^1, F^2, ..., F^m\}$ be a collection of $m$ networks, let $\delta(S)$ be the set of all partitions of $S$. For each partition $h \in \delta(S)$, let $L_h$ denote the largest subset in $h$. Then, the probability of success by voting is:

$$\sum_{h \in \delta(S)} P(L_h), \tag{4.2.5}$$

where $P(L_h)$ is the probability of simultaneous success in $L_h$. Note that when $m$ is large, the number of possible partitions is large which means the success probability by voting can be high.

We note that although Carlini-Wager attacks are able to defeat a network trained with any temperature constant, an attack crafted for a network trained with one temperature constant might not work on another trained with a different temperature, due to differences in the trained parameters. Thus we propose a mechanism where we train an ensemble of networks, each trained with a different temperature constant, where we respond to queries using the aggregate outputs of the ensemble of networks.

### 4.2.4 Threat Model

**Noisy Logit**

We discussed that Carlini-Wagner attacks rely on accessibility and integrity of the logits, and we proposed a mechanism that adds perturbations to the original logits (or any other layer). Thus, this defensive mechanism works under the assumptions that: 1) an adversary can query the outputs at any layer of the network; 2) but the adversary does not have complete knowledge and resources to construct the original network offline. If the adversary has an exact copy of the original network, then he/she could craft attacks based on the original network with genuine logits, and would defeat this mechanism.

**Ensemble Voting**
The Ensemble Voting mechanism works by providing resilience in situations where some networks might be under attack, thus it assumes the adversary can defeat a subset of the networks (which implies the adversary could have full knowledge of these networks). Therefore, complete white-box attacks are assumed in this model.

**Bounded Perturbations**
We discussed in the Introduction that for sufficiently differentiable loss functions, a solution to (4.1.5) is guaranteed to exist. However, if the perturbations are so large that even humans will mis-classify, then any mis-classification cannot be considered fault of the network. Thus, we assume the goal of adversary is craft adversarial examples with perturbations that are (reasonably) unnoticeable.

## 4.3 Experimental Evaluation

In this section we assess the effectiveness of our approach by formulating two types of potential attacks on our model. In the first type we consider an attack crafted for a randomly chosen network in the ensemble. Since the model architectures are so similar across the networks, transferability is possible where networks other than the chosen one might still incorrectly classify. In the second type of attacks we consider superimpositions of adversarial examples to examine whether this could further increase transferability. For a chosen subset of the networks, each with a corresponding adversarial example, one could reasonably suspect that other networks beyond the chosen subset could incorrect classify as the superimposition could have captured perturbations that are commonly effective on many other networks.

### 4.3.1 Test Setup

We conducted experiments on two datasets, MNIST (LeCun *et al.*, 1998) and CIFAR10 (Krizhevsky, 2009). For the following tests, the architecture of each network is the same as the one in Carlini & Wagner (Carlini and Wagner, 2017), which we provide in Table 4.1. We first trained an ensemble of networks $F^l$, each with temperature $T_l, l = 1, 2, ..., m$. We used $\{T_l | l = 1, 2, ..., 50\} = \{10, 20, ..., 500\}$. In all the experiments, the $L^2$ norm is used in all places where a norm is needed, including the $L^2$ version of the Carlini-Wagner attack. For the models with Noisy Logit, we employ a Laplace noise function $g(\mu, b)$ with location $\mu = 0$, scale $b = 0.5$ for MNIST and $b = 0.03$ for CIFAR10. We experimented with different values for the scale parameter, and found these values provide sufficient noise without losing too much accuracy.

| **Layer** | **MNIST** | **CIFAR10** |
|---|---|---|
| Convolution + ReLU | $3 \times 3 \times 32$ | $3 \times 3 \times 64$ |
| Convolution + ReLU | $3 \times 3 \times 32$ | $3 \times 3 \times 64$ |
| Max Pooling | $2 \times 2$ | $2 \times 2$ |
| Convolution + ReLU | $3 \times 3 \times 32$ | $3 \times 3 \times 64$ |
| Convolution + ReLU | $3 \times 3 \times 32$ | $3 \times 3 \times 64$ |
| Max Pooling | $2 \times 2$ | $2 \times 2$ |
| Fully Connected + ReLU | 200 | 256 |
| Fully Connected + ReLU | 200 | 256 |
| Softmax | 10 | 10 |

Table 4.1: Model architectures used in the ensemble of networks

| **Parameter** | **MNIST** | **CIFAR10** |
|---|---|---|
| Learning Rate | 0.01 | 0.01 |
| Decay | $1.00e-06$ | $1.00e-06$ |
| Momentum | 0.9 | 0.9 |
| Dropout | 0.5 | 0.5 |
| Batch Size | 128 | 128 |
| Partitioned Training Set | Yes | No |
| Training Set Size (Per Network) | 1100 | 45000 |
| Validation Set Size | 5000 | 5000 |
| Epochs | 3000 | 150 |
| Laplace Noise Scale | 0.5 | 0.03 |

Table 4.2: Parameters used in the ensemble of networks

**MNIST**
We first partitioned the original dataset into 50 training subsets (1100 samples each) and one validation set (5000 samples). We trained 50 teachers individually on the partitioned subsets, each was trained with a different temperature constant for 3000 epochs. The average validation accuracy among the ensemble of networks was 96.80%.

**CIFAR10**
In the setup for CIFAR10 we used a single training set (45000 samples) and one validation set (5000 samples). We do not use the networks trained on partitioned datasets for testing on CIFAR10, because we observed that a small training dataset resulted in low accuracy for CIFAR10. We trained 50 teachers individually on the same training set, each was trained with a different temperature constant for 150 epochs. The average validation accuracy among the ensemble of networks was 79.53%.

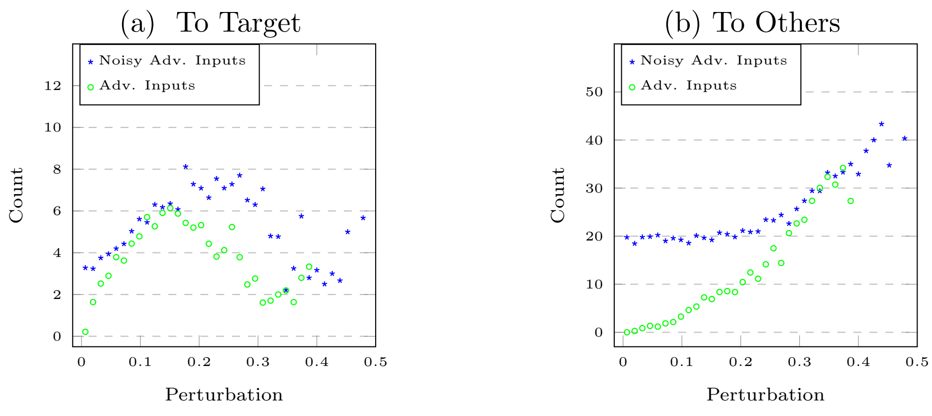


Figure 4.1: MNIST: Counts of Networks Changed

### 4.3.2 Random Single Network Attack

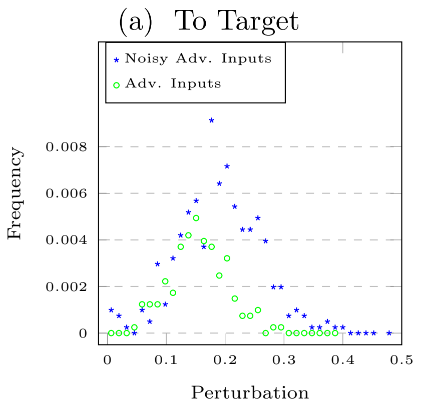
In this section, we look at the possible outcomes of an adversarial example crafted to defeat a single network, to see how it can potentially transfer across the ensemble. Since an adversarial example crafted for one network can potentially fool a different network (Papernot *et al.*, 2016c), we expect transferability in our ensemble of networks especially given that they have very similar model architectures. Given some sample input $s$ and target label $t$, for each $F^l$ we craft an adversarial example $A(F^l, s, t)$ using the Carlini-Wagner attack, and look at: 1) how each $F^{l'}$ classifies $A(F^l, s, t)$; and 2) how the ensemble classifies the example through voting with and without Noisy Logit.

We generate a set of 9 input samples, $s_k, k = 1, ..., 9$. For each $s_k$ we craft an adversarial example $A(F^l, s_k, t_j)$ on network $F^l, l = 1, ..., 50$, for targets $t_j, j = 1, ..., 9$, for a total of $9 \times 50 \times 9 = 4050$ adversarial examples. We define the perturbation of an adversarial example as its normed difference with the original input over the norm of the original input, as follows:

$$p(a; s) = \frac{||a - s||}{||s||}, \tag{4.3.1}$$

where $a$ is an adversarial example on the input $s$. We bucket the range of perturbations into 40 equally spaced bins, $x_b, b = 1, ..., 40$. In the plots in this section, we aggregate by the perturbation bins and represent the bins by their mid-points on the x-axis.

In Fig. 4.1a we have for the MNIST dataset, the average counts of networks whose classifications change to the target of the adversarial example, from



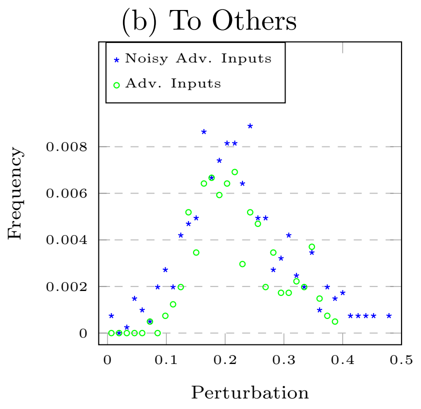


Figure 4.2: MNIST Ensemble: Aggregate Outputs Changed

some other original classification, i.e., for bucket $x_b$, the values $y_b$ on the y-axis are:

$$y_b = \frac{1}{|x_b|} \sum_{x \in x_b} |\{F^l : F^l(a) \neq F^l(s), F^l(a) = t, p(s,a) = x, l = 1, ..., 50\}|. \tag{4.3.2}$$

In Fig. 4.1b the counts are on the classifications that change to something other than the target, or:

$$y_b = \frac{1}{|x_b|} \sum_{x \in x_b} |\{F^l : F^l(a) \neq F^l(s), F^l(a) \neq t, p(s,a) = x, l = 1, ..., 50\}|. \tag{4.3.3}$$

These counts are averaged by perturbation bin. The green and blue curves represent the results corresponding to adversarial examples crafted on networks without and with Noisy Logit applied, respectively.

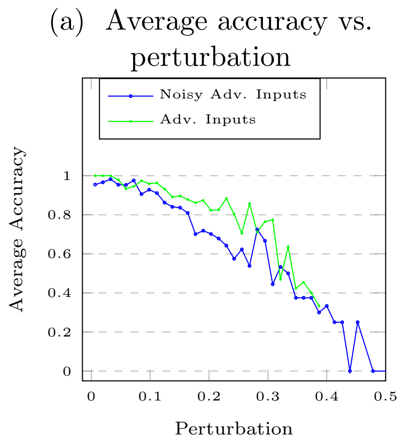
In Fig. 4.2a we show the frequencies of aggregate outputs of the ensemble which changed to the target of the adversarial example, i.e.:

$$y_b = \frac{1}{4050} \sum_{x \in x_b} |\{F^*(a) : F^*(a) \neq F^*(s), F^*(a) = t, p(s,a) = x\}|, \tag{4.3.4}$$

where $F^*(\cdot)$ represents the aggregate output by voting among the ensemble. Similarly, we have in Fig. 4.2b the frequencies corresponding to some label other than the target:

$$y_b = \frac{1}{4050} \sum_{x \in x_b} |\{F^*(a) : F^*(a) \neq F^*(s), F^*(a) \neq t, p(s,a) = x\}|. \tag{4.3.5}$$

The frequencies are obtained by normalizing the total changed outputs by the



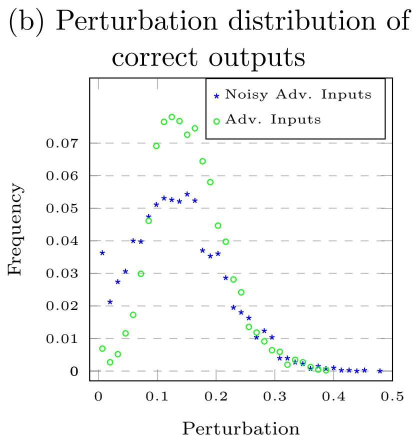
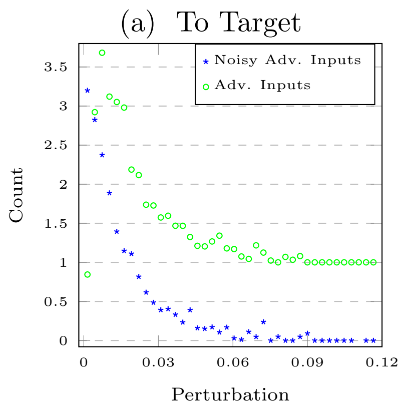


Figure 4.3: MNIST Ensemble: Perturbation & Accuracy



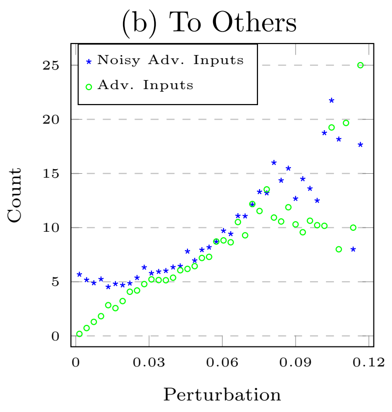


Figure 4.4: CIFAR10: Counts of Networks Changed

total number of adversarial examples, which is 4050 in this case.

In Fig. 4.3a we show the average accuracy of the ensemble by perturbation bin. We see that when the amount of perturbation is between 10% - 30%, applying Noisy Logit causes the ensemble to lose accuracy since additional noise is added to the adversarial examples. This, together with Fig. 4.1a - 4.2b seem to suggest that applying Noisy Logit tends to decrease accuracy in the ensemble for perturbation bins in this range. However, note that by applying Noisy Logit the distribution of the perturbations is significantly changed, where we see increased frequency in smaller perturbations, and also occurrences of large perturbations in the range of 40%-50% which do not occur without applying Noisy Logit. In particular, MNIST Ensemble with Noisy Logit is able to correctly classify more adversarial examples with small perturbations, as well as adversarial examples with larger perturbations, as shown in Fig. 4.3b.

In Fig. 4.4a and 4.4b we have the average counts of networks whose classifications change due to the adversarial examples, as in equations 4.3.2 and 4.3.3, for the CIFAR10 dataset. In Fig. 4.4a we notice a decrease in the number of networks in the ensemble whose classifications change to target when Noisy Logit is applied, and consistently so across different perturbation bins. This suggests that applying Noisy Logit reduces transferability rate for CIFAR10 networks. In Fig. 4.4b we see an increase in the counts of networks that change classifications to others for the smallest perturbation bins, which could be due to the general (slightly) lowered average accuracy when Noisy Logit is applied. Beyond the smallest perturbation bins, there are no noticeable differences in the counts whether Noisy Logit is applied or not.

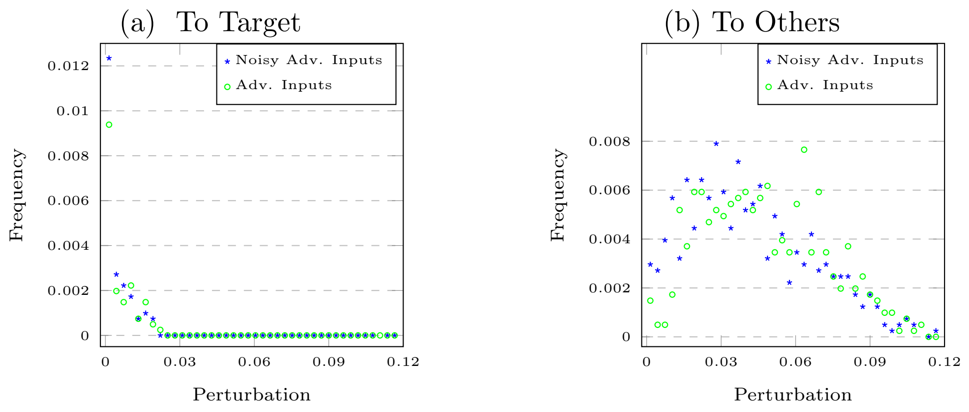


Figure 4.5: CIFAR10 Ensemble: Aggregate Outputs Changed

In Fig. 4.5a and 4.5b we show the frequencies of aggregate outputs of the ensemble with changed classifications for the CIFAR10 dataset, as in equations 4.3.4 and 4.3.5. In Fig. 4.5a there appears to be no meaningful difference in the transferability rate with respect to the aggregate output of the ensemble. Note this is not a contradiction to Fig. 4.4a above, because the counts in Fig. 4.4a are very small (contrary to the case of MNIST). Since the ensemble consists of 50 networks, an extra count or two that classify the adversarial example as the target label will not change the aggregate output (by voting) from the ensemble. In Fig. 4.5b there appears to be a shift in the perturbation distribution for adversarial examples that cause the ensemble to change aggregate output to other labels, where there are more adversarial examples with smaller perturbations when Noisy Logit is applied.

We show the average accuracy of the ensemble of networks for CIFAR10 in Fig. 4.6a. There appears to be no significant difference in the average accuracy of the ensemble whether or not Noisy Logit is applied. We show the distribution of the perturbations for the correctly classified adversarial examples in Fig. 4.6b. Observe the range and frequencies of the perturbation

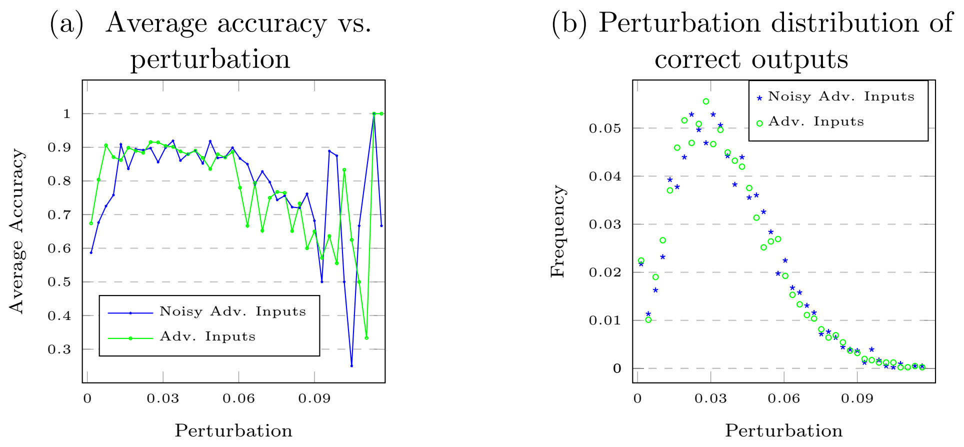


Figure 4.6: CIFAR10 Ensemble: Perturbation & Accuracy

distributions are very similar whether or not Noisy Logit is applied.

We summarize the results in Table 4.3 and Table 4.4. Observe that for CIFAR10 Ensemble with Noisy Logit, nearly identical accuracy to the original clean accuracy is achieved for adversarial examples that target any single network in the ensemble. We calculate accuracy as the ratio of the number of correct aggregate outputs over the total number (4050) of test attacks. Note that since for each sample we generate $50 \times 9$ adversarial examples, the clean accuracy is effectively over 9 samples only.

| **Model** | **Clean Accuracy** | **SN Attack Accuracy** |
|---|---|---|
| MNIST Ensemble | 100.000% | 88.370% |
| MNIST Ensemble with Noisy Logit | 98.667% | 79.975% |
| CIFAR10 Ensemble | 88.889% | 85.185% |
| CIFAR10 Ensemble with Noisy Logit | 84.321% | 84.691% |

Table 4.3: Accuracy on clean inputs vs. on Single Network (SN) adversarial inputs

### 4.3.3 Superimposition Attacks

In this section we consider superimposition attacks consisting of adversarial examples targeting two or three of the networks in the ensemble. Due to the large number of possible subsets of length two or three, we do not consider every such combination; instead, since the objective of crafting an adversarial example is to minize the perturbations while causing a network to incorrectly

| **Model** | **Correct** | **Target** | **Other** |
|---|---|---|---|
| MNIST Ensemble | 88.37% | 3.85% | 7.78% |
| MNIST Ensemble with Noisy Logit | 79.98% | 8.35% | 11.68% |
| CIFAR10 Ensemble | 85.19% | 1.80% | 13.01% |
| CIFAR10 Ensemble with Noisy Logit | 84.69% | 2.15% | 13.06% |

(a) Classifications

| **Model** | **Correct** | **Target** | **Other** |
|---|---|---|---|
| MNIST Ensemble | 15.34% | 15.52% | 22.16% |
| MNIST Ensemble with Noisy Logit | 13.58% | 19.06% | 22.42% |
| CIFAR10 Ensemble | 3.59% | 0.51% | 4.77% |
| CIFAR10 Ensemble with Noisy Logit | 3.67% | 0.42% | 4.13% |

(b) Average perturbations

Table 4.4: Distributions for Single Network adversarial inputs

classify, we consider a greedy type superimposition where adversarial examples of minimal perturbations are used. We do not look at superimpositions of more than three adversarial examples, as we will see that with three adversarial examples, total perturbation can already be as high as $> 50\%$ in an attack, as shown in Table 4.8b.

**Superimposition of Two Adversarial Examples**

We generate a set of 30 input samples, for each sample and each target we need to first craft an adversarial example for each network $F^l$. Then we pick the two adversarial examples with the smallest perturbations and superimpose them to arrive at an attack for one test. This gives a total of $30 \times 9 = 270$ tests; however, the total number of individual adversarial examples crafted is $270 \times 50 = 13500$.

| **Single Network** | **Clean Accuracy** | **SI2 Attack Accuracy** |
|---|---|---|
| MNIST Network | 96.8000% | 55.2963% |
| MNIST Network with Noisy Logit | 79.6296% | 69.1333% |
| CIFAR10 Network | 79.5333% | 71.4000% |
| CIFAR10 Network with Noisy Logit | 74.8889% | 71.3556% |

Table 4.5: Average single network accuracy on clean inputs vs. on Superimposition (2×) of adversarial inputs

In Table 4.5 we have the average accuracy over all networks in the ensemble, where in the left column the accuracy is on the original images, in the right column the accuracy is on the adversarial examples. We see that with a superimposition of only two images, the average accuracy of an MNIST network reduced to 55.30% from 96.80%; whereas for a CIFAR10 network the average accuracy is reduced from 79.53% to 71.40%, a much smaller reduction. This again suggests that CIFAR10 networks are more robust to transferability. Observe that with Noisy Logit, the accuracy reduction is less prominent.

| **Model** | **Correct** | **Target** | **Other** |
|---|---|---|---|
| MNIST Ensemble | 66.2963% | 13.3333% | 20.3704% |
| MNIST Ensemble with Noisy Logit | 96.6667% | 0.0000% | 3.3333% |
| CIFAR10 Ensemble | 87.7778% | 1.1111% | 11.1111% |
| CIFAR10 Ensemble with Noisy Logit | 84.0741% | 1.4815% | 14.4444% |

(a) Classifications

| **Model** | **Correct** | **Target** | **Other** |
|---|---|---|---|
| MNIST Ensemble | 14.6285% | 25.7126% | 24.3226% |
| MNIST Ensemble with Noisy Logit | 6.4906% | 0.0000% | 24.3824% |
| CIFAR10 Ensemble | 2.7488% | 0.0002% | 3.9650% |
| CIFAR10 Ensemble with Noisy Logit | 2.6369% | 0.0002% | 2.7721% |

(b) Average perturbations

Table 4.6: Distributions for Superimposition (2×) of adversarial inputs

In Table 4.6a, we have the accuracies of applying Ensemble Voting with and without Noisy Logit. In MNIST Ensemble without Noisy Logit, we obtained a 66.3% accuracy for correctly classifying the adversarial images, 13.3% success rate in the targeted attacks, and in the remaining 20.37% of the tests the combined network classified the adversarial images as something other than the correct or targeted labels. In MNIST Ensemble with Noisy Logit, accuracy is vastly improved to 96.67%, nearly identical to the average single network accuracy on clean inputs. Moreover, observe that the average accuracy for single MNIST networks with Noisy Logit is only 69.13% when applied on superimposition adversarial inputs, but as an ensemble of networks we achieve a much better accuracy rate. We make similar observations for CIFAR10, except that it's less prone to transferability, and suffers smaller accuracy reduction with or without Noisy Logit when applied on superimposition adversarial inputs. Also, the CIFAR10 ensemble achieves better accuracy on adversarial inputs than the average single CIFAR10 network accuracy on clean inputs.

We provide the distribution of perturbations corresponding to Table 4.6a in Table 4.6b. The average perturbation for MNIST Ensemble with Noisy Logit for correctly classified adversarial examples is smaller than that without Noisy

Logit, observe this is consistent with what we saw in Fig. 4.3b, where applying noise shifts the distribution of perturbations in the adversarial examples.

**Superimposition of Three Adversarial Examples**
We generate a set of 10 input samples, then we craft an adversarial example similar to what was done for superimposition of two adversarial examples, except we use three in this case. We have a total of $10 \times 9 = 90$ tests, where the total number of individual adversarial examples crafted is $90 \times 50 = 4500$.

| **Single Network** | **Clean Accuracy** | **SI3 Attack Accuracy** |
|---|---|---|
| MNIST Network | 95.6000% | 24.6889% |
| MNIST Network with Noisy Logit | 74.6222% | 55.9333% |
| CIFAR10 Network | 83.0000% | 68.6222% |
| CIFAR10 Network with Noisy Logit | 78.6222% | 73.0667% |

Table 4.7: Average single network accuracy on clean inputs vs. on Superimposition ($3\times$) of adversarial inputs

In Table 4.7 we have the average accuracy over all networks in the ensemble. Observe with superimposition of 3 adversarial examples, the average single MNIST network accuracy is reduced drastically from 95.6% to 24.69%. When the individual networks have such low accuracy, the benefit of having an ensemble is minimal as we see in the first row in Table 4.8a. However, with Noisy Logit, MNIST Ensemble is able to achieve a very large improvement in accuracy, from the 26.67% without Noisy Logit to 90.00% with Noisy Logit. Moreover, in the 2.22% of successful attacks on MNIST Ensemble with Noisy Logit, the average perturbation was $> 50\%$. We remark that the models on CIFAR10 are again impacted to a much smaller degree in terms of accuracy loss.

## 4.4 A Second Look at Noisy Logit

The results for the Random Single Network Attack in Section 4.3.2 provided some insights into how Noisy Logit works to reduce transferability across different neural networks. However, in the case of the MNIST dataset, we observed that although Noisy Logit changes the distribution of perturbations in the adversarial examples, it appears there is no benefit to using Noisy Logit as Ensemble Voting alone provides better accuracy rates. In this section, we look at the output of each individual network in isolation when it's being targeted, to see whether applying Noisy Logit improves robustness in a single network.

| **Model** | **Correct** | **Target** | **Other** |
|---|---|---|---|
| MNIST Ensemble | 26.6667% | 57.7778% | 15.5556% |
| MNIST Ensemble with Noisy Logit | 90.0000% | 2.2222% | 7.7778% |
| CIFAR10 Ensemble | 86.6667% | 1.1111% | 12.2222% |
| CIFAR10 Ensemble with Noisy Logit | 87.7778% | 2.2222% | 10.0000% |

(a) Classifications

| **Model** | **Correct** | **Target** | **Other** |
|---|---|---|---|
| MNIST Ensemble | 13.0187% | 29.0136% | 34.7022% |
| MNIST Ensemble with Noisy Logit | 8.3600% | 50.5385% | 26.6385% |
| CIFAR10 Ensemble | 3.5315% | 0.0003% | 5.1646% |
| CIFAR10 Ensemble with Noisy Logit | 3.4029% | 0.0003% | 3.9467% |

(b) Average perturbations

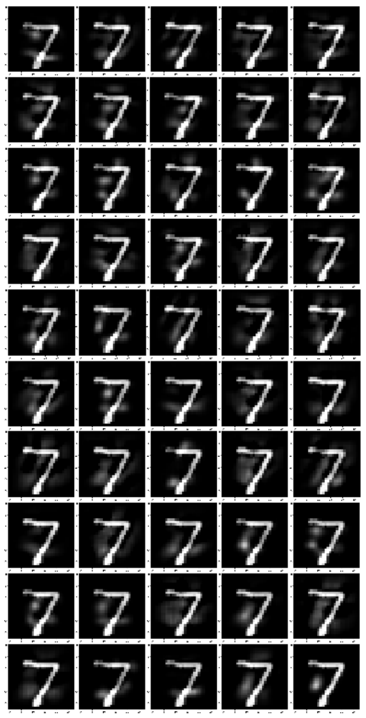
Table 4.8: Distributions for Superimposition (3×) of adversarial inputs

In Fig. 4.7 we craft adversarial examples corresponding to a single sample and single target, on each of the 50 networks in the ensemble.

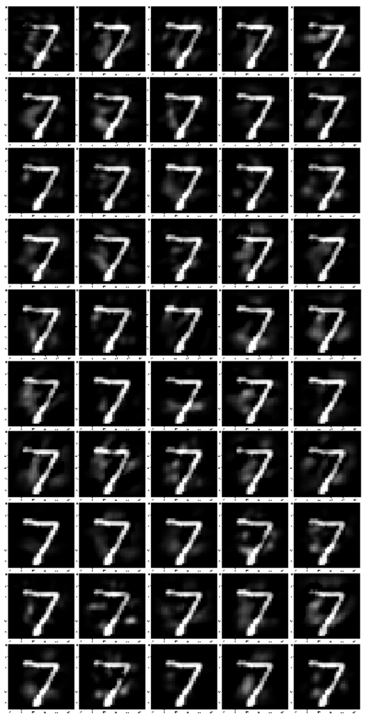
In Fig. 4.7a and Fig. 4.7b, the sample input is the digit 7 and target is the digit 0. Observe that the distribution of perturbations is changed if we apply Noisy Logit, where in Fig. 4.7b we see more occurrences in the tails (i.e. very small or very large perturbations). In Fig. 4.7a each targeted network misclassifies its corresponding adversarial example as 0 (corresponding to 100% success rate of Carlini-Wagner on a single network); whereas in Fig. 4.7b, only 8 of the networks misclassify as 0, and 29 of the networks still correctly classify as 7. Therefore, for an individual MNIST network with Noisy Logit applied, the success rate of a targeted Carlni-Wagner attack is low. Thus, a single MNIST network is more robust to adversarial examples if Noisy Logit is applied, however the accuracy rate suffers since extra noise is added.

In Fig. 4.7c and Fig. 4.7d, the sample input is the object ship and the target is the object airplane. There is no noticeable difference in the distribution of perturbations whether or not Noisy Logit is applied. In Fig. 4.7c, again each targeted network misclassifies its corresponding adversarial example as the target (airplane); whereas in Fig. 4.7d, only 1 of the networks misclassifies as airplane, and 38 of the networks still correctly classify as ship. Therefore, the success rate of a targeted Carlni-Wagner attack on a single CIFAR10 network is very low, i.e., robustness of a single CIFAR10 network is increased in the presence of Noisy Logit. Note also the accuracy rate only slightly suffers from extra noise added.

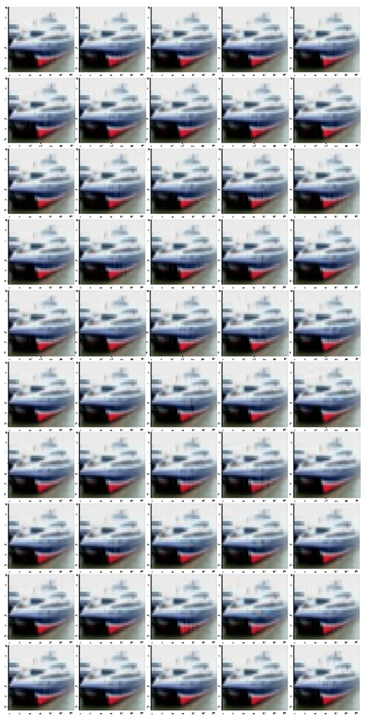
(a) MNIST No Noisy Logit



(b) MNIST With Noisy Logit



(c) CIFAR10 No Noisy Logit



(d) CIFAR10 With Noisy Logit

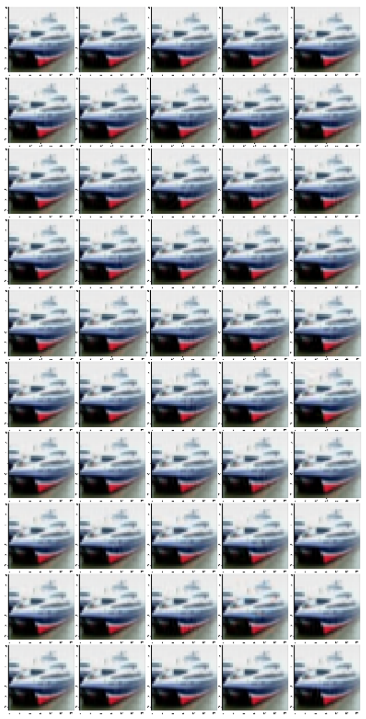


Figure 4.7: Adversarial examples without and with Noisy Logit applied

## 4.5 Discussion

The goals of this chapter might appear somewhat contradictory. On one-hand, we are providing a mechanism that protects against adversarial examples, and it relies on the ground that transferability properties of adversarial examples between networks are not *completely* known. On the other hand, we've shown how easy it is to craft an adversarial example using superimposition, that would result in an example that can fool not only the original networks from which the example was crafted, but can potentially fool other networks as well (that is, in the absence of a protection mechanism). We believe the key to finding an adversarial example that will truly resist all protection mechanisms, including our work here, is to understand the transferability properties of such examples across different networks. Based on our testing results, it appears transferability likely depends on the complexity of the classification task. The images in the CIFAR10 dataset have many more features than those in the MNIST dataset. While the CIFAR10 and MNIST networks were trained with similar model architectures, the MNIST networks are much more prone to transferability than the CIFAR10 networks. Also, we note that it takes a much smaller amount of perturbations to craft an adversarial example on a single CIFAR10 network than it does for an MNIST network, which could mean that on a CIFAR10 image there is a small subset of pixels which are most important to the model. When this small subset of pixels varies across different models, then it becomes very unlikely that an adversarial example crafted with one model would succeed on a different model; unless there is some model-independent small subset of pixels which are universally important to every model.

**Threat Analysis**
We note that most adversarial examples failed because they relied on integrity of the outputs at a layer (in this paper we focused on the logits due to nature of the Carlini-Wagner attacks). Therefore, in the presence of the noisy logit mechanism, an adversary might try to circumvent this obstacle by trying to infer the genuine outputs from the noisy outputs. Note that since the neural network is not necessarily a continuous function, an average of the noisy outputs, which are function values of a random distribution, is not necessarily equal to the genuine output of the average of perturbed inputs. However, we remark that it is valuable to perform detailed testing on how well an average of the outputs can function as approximation to the genuine output, though we also remark that attacks of this sort would require much more effort as the adversary might need to collect a large number of outputs based on the same input for each iteration required in crafting an attack.

We note that transferability of neural networks was studied by Papernot et al. (Papernot *et al.*, 2016c), where it was shown in the experiments that transferability from one neural network to another was as high as 38% using the Fast Gradient Sign method (Goodfellow *et al.*, 2015). In (Papernot *et al.*, 2016c), they did not consider an explicit algorithm for attacking an ensemble model. In our testing results in Tables 4.5 and 4.6a, we showed that using a simple superimposition of two adversarial examples, the average accuracy on the MNIST networks was reduced from 96.80% to 55.30%, with ensemble accuracy of 66.30%. When three adversarial examples were used, the average accuracy was reduced from 95.60% to 24.69%, with ensemble accuracy of 26.67%, as shown in Tables 4.7 and 4.8a. Note that transferability rate is also dependent on the classification task, where we saw in the same tables that the same experiments performed on the CIFAR10 dataset showed a much lower transferability rate on similar superimposition attacks.

## 4.6 Concluding Remarks

In this chapter we introduced an approach for protecting image classification networks from adversarial examples. The approach is composed of two mechanisms - Noisy Logit and Ensemble Voting, which were evaluated in Section 4.3. We saw that Ensemble Voting improves accuracy over the base model, while Noisy Logit reduces transferability across different networks in classifying adversarial examples. Moreover, the approach combining the two mechanisms was shown to have comparable accuracy in classifying adversarial examples as in classifying genuine inputs, for both MNIST and CIFAR10 if superimposition attacks were considered, and for CIFAR10 if random single network attacks were considered. Using Noisy Logit impedes the adversary's ability to accurately solve the optimization problem for crafting adversarial examples, as solving the problem requires access to outputs at some layers which have been tampered by the Noisy Logit mechanism. Ensemble Voting works on white-box attacks and is a mechanism that provides resilience as well as improves accuracy. Since using Noisy Logit reduces accuracy in general, the addition of Ensemble Voting complements the approach by improving the reduced accuracy.

# Chapter 5

# Conclusion

In this chapter, we conclude this thesis with a summary of our contributions to the two problems we set out to address, as well as research directions and ideas for future work.

## 5.1 Thesis Summary & Contributions

In this work we aimed to address two problems concerning privacy & security risks in Data Analytics and Machine Learning algorithms. We described the specific problems in the introduction, and motivated our research with some discussions of related work in Chapter 2.

In Chapter 3 we provided a thorough exposition into our algorithms for anonymization. The main contributions in Chapter 3 were: 1) introduced a formulation of $k$-anonymity as an MILP with the objective to maximize data utility; 2) devised two practical algorithms with experimental results to demonstrate their scalability to large datasets; 3) presented benchmark results of our algorithms against existing algorithms from different classes; and 4) provided a discussion on the similarities and differences between existing algorithms and the algorithms we introduced, highlighting aspects of our algorithms that contributed to better performance and utility.

In Chapter 4 we gave a brief account of works that motivated the creation of the state-of-the-art algorithm for crafting adversarial examples - the Carlini-Wagner attack. We discussed the vulnerabilities of prior attacks that were defeated by Defensive Distillation, and how the Carlini-Wagner attack escaped those vulnerabilities. The main contributions in Chapter 4 were: 1) developed an approach combining two mechanisms, Noisy Logit and Ensemble Voting. Noisy Logit works to increase robustness at the expense of accuracy, and the lost accuracy is recouped by the Ensemble Voting mechanism; 2) formulated potential attacks on our combined approach with experimental results to demonstrate its effectiveness; 3) showed that without any protection mechanisms, a superimposition attack consisting of three adversarial examples with minimal perturbations could cause $> 75\%$ of the individual MNIST networks to misclassify, and could cause the MNIST ensemble to misclassify $> 73\%$ of the time.

## 5.2 Future Work

There are several directions for future work. We describe some ideas that might potentially improve the efficiency of our $k$-anonymization algorithms below. We also list some interesting research studies to extend the work on our approach for improving robustness of neural networks.

**Improving Optimization-Based $k$-Anonymization**
We have implemented the Split & Carry algorithm with the number of carry-over records equal to $k$ plus the number of distinct equivalent rows to the last $k$ records. As discussed this can lead to performance issues as $k$ becomes larger. As a future research direction, we can instead use a customizable parameter $l$ in place of $k$ to mitigate the problem. It would be interesting to experiment with this parameter to see whether there would be recommendable choices for $l$.

We can also modify the existing algorithm in the set of records we carry. The current algorithm carries over the $k$-set of each boundary record if there are either zero or at least $k$ remaining indirectly equivalent records, otherwise we carry over the $k$-set plus all indirectly equivalent records. We could instead carry over a subset of the $k$-set, in particular for the $m^{th}$ sub-problem, we can determine the subset as follows: Let $x_i$ be the boundary record under consideration, denote its equivalence class by $Eq_i := \{x_i, x_{i_2}, ..., x_{i_k}\}$. For $l = i_2, .., i_k$, compute the objective value $f_l := f(Eq_i \setminus x_l)$. Then we define the set of carry-over records for $x_i$ to be the $k-(k-r)$ records with the largest $f_l$'s, where $r$ is the size of the set of indirectly equivalent records, $r < k$. In other words, we carry those records that contribute the smallest information loss in $Eq_i$, which means these records can be potentially grouped into an equivalence class with less information loss than $Eq_i$ in the next sub-problem.

Another way to remedy the situation is, instead of specifying $S$ - the minimum number of the $k$-sets to include in a sub-problem, we can use a parameter $P$ to specify the exact size of each sub-problem, and a parameter $A$ which describes how many $k$-sets we accept into our final solution. For example, for $k = 3$ we can have $P = 12$, and $A = 2$, i.e. we accept the two $k$-sets with minimum utility loss into our final solution, and carry the remaining records into the next sub-problem of exact size $P = 12$, creating a chain of sub-problems by *bubble carry*. Inevitably in this modified algorithm we will have to solve more sub-problems than our current Split & Carry (leading to a linear increase in complexity), however, in this way we are controlling the size of each sub-problem which we know to have good complexity when the number of attributes is small. This simple modification might have better running time than our current algorithm where in the latter there is some variation in the sizes of the sub-problems.

Both our algorithms use the initial feasible solution as the starting point, and we have seen in the experiments that the initial feasible solution is very efficient and already has utility advantage over the Mondrian algorithm. Therefore, determining an optimal sorting order is also valuable as future work; it can potentially improve utility to an extent that the initial feasible solution by itself, can also be merited as an efficient solution for anonymization.

**Further Studies on Robustness Approach**
There are a number of future directions to extend the current work. In the Noisy Logit mechanism we determined the amount of noise to be added by experimenting with different values. One can also study the relationship between the number of networks to be used with the amount of noise to be added to the inputs, or consider potentially different distributions used for generating the noise. The amount of noise is an important parameter, as we saw that adding noise indeed obscured the search for successful Carlini-Wagner adversarial examples, however it also lowered the accuracy rate in the standalone networks.

We can consider Ensemble Voting using a collection of networks with very different architectures (rather than the ones considered here which are the same up to the temperature constant). We saw in Chapter 4 that transferability between MNIST networks (without protection mechanism) was high, it'd be an interesting study to investigate whether using different architectures would reduce transferability in MNIST networks, although transferability was low in CIFAR10 networks while they also used the same architectures.

Given that in our experiments, MNIST and CIFAR10 networks exhibited different transferability behaviour, it'd be an interesting research study to perform similar experiments on other datasets as well, to better understand which aspects of the datasets led to the different properties. We suspect that the complexity of the underlying task (i.e. the number of features in the inputs) could be a key factor, where more complex tasks should exhibit lower transferability rates.

# Appendices

# Appendix A

## A.1 Weighting Attributes

We present an example of applying weights to the attributes using a dataset from the database of FARS (US Department of Transportation, 2016). In Table A.1, we have the original dataset in (a) on 4 attributes: AGE, SEX, INJ_SEV (injury severity) and DRINKING; we also have the outcome of the dataset anonymized with equal weights in (b), and the outcome anonymized with weight vector (0.8, 0.05, 0.1, 0.05) in (c). We see that the upper-lower bound gaps in the anonymized outcome for the AGE attribute narrowed when we applied a large weight to this attribute, i.e., we achieve better utility for AGE; however this is achieved at the expense of the SEX and DRINKING attributes.

| index | AGE | SEX | INJ_SEV | DRINKING |
|---|---|---|---|---|
| 0 | 64 | 2 | 4 | 0 |
| 1 | 29 | 1 | 0 | 0 |
| 2 | 42 | 2 | 0 | 0 |
| 3 | 41 | 2 | 4 | 1 |
| 4 | 53 | 1 | 2 | 1 |
| 5 | 59 | 1 | 4 | 0 |
| 6 | 49 | 1 | 4 | 0 |
| 7 | 59 | 1 | 2 | 0 |
| 8 | 80 | 1 | 4 | 0 |
| 9 | 50 | 1 | 4 | 0 |
| 10 | 64 | 1 | 3 | 0 |
| 11 | 55 | 1 | 0 | 0 |
| 12 | 25 | 1 | 0 | 0 |
| 13 | 42 | 1 | 4 | 0 |
| 14 | 33 | 1 | 2 | 0 |
| 15 | 31 | 2 | 2 | 0 |
| 16 | 68 | 1 | 3 | 0 |
| 17 | 20 | 2 | 4 | 0 |
| 18 | 40 | 1 | 4 | 1 |
| 19 | 18 | 1 | 4 | 0 |

(a) Original dataset

| index | AGE | SEX | INJ_SEV | DRINKING |
|---|---|---|---|---|
| 0 | [20 - 64] | [2 - 2] | [0 - 4] | [0 - 0] |
| 1 | [25 - 55] | [1 - 1] | [0 - 0] | [0 - 0] |
| 2 | [20 - 64] | [2 - 2] | [0 - 4] | [0 - 0] |
| 3 | [40 - 53] | [1 - 2] | [2 - 4] | [1 - 1] |
| 4 | [40 - 53] | [1 - 2] | [2 - 4] | [1 - 1] |
| 5 | [49 - 80] | [1 - 1] | [4 - 4] | [0 - 0] |
| 6 | [49 - 80] | [1 - 1] | [4 - 4] | [0 - 0] |
| 7 | [59 - 68] | [1 - 1] | [2 - 3] | [0 - 0] |
| 8 | [49 - 80] | [1 - 1] | [4 - 4] | [0 - 0] |
| 9 | [49 - 80] | [1 - 1] | [4 - 4] | [0 - 0] |
| 10 | [59 - 68] | [1 - 1] | [2 - 3] | [0 - 0] |
| 11 | [25 - 55] | [1 - 1] | [0 - 0] | [0 - 0] |
| 12 | [25 - 55] | [1 - 1] | [0 - 0] | [0 - 0] |
| 13 | [18 - 42] | [1 - 1] | [2 - 4] | [0 - 0] |
| 14 | [18 - 42] | [1 - 1] | [2 - 4] | [0 - 0] |
| 15 | [20 - 64] | [2 - 2] | [0 - 4] | [0 - 0] |
| 16 | [59 - 68] | [1 - 1] | [2 - 3] | [0 - 0] |
| 17 | [20 - 64] | [2 - 2] | [0 - 4] | [0 - 0] |
| 18 | [40 - 53] | [1 - 2] | [2 - 4] | [1 - 1] |
| 19 | [18 - 42] | [1 - 1] | [2 - 4] | [0 - 0] |

(b) Equal weights

| index | AGE | SEX | INJ_SEV | DRINKING |
|---|---|---|---|---|
| 0 | [64 - 80] | [1 - 2] | [3 - 4] | [0 - 0] |
| 1 | [29 - 33] | [1 - 2] | [0 - 2] | [0 - 0] |
| 2 | [40 - 42] | [1 - 2] | [0 - 4] | [0 - 1] |
| 3 | [40 - 42] | [1 - 2] | [0 - 4] | [0 - 1] |
| 4 | [49 - 53] | [1 - 1] | [2 - 4] | [0 - 1] |
| 5 | [55 - 59] | [1 - 1] | [0 - 4] | [0 - 0] |
| 6 | [49 - 53] | [1 - 1] | [2 - 4] | [0 - 1] |
| 7 | [55 - 59] | [1 - 1] | [0 - 4] | [0 - 0] |
| 8 | [64 - 80] | [1 - 2] | [3 - 4] | [0 - 0] |
| 9 | [49 - 53] | [1 - 1] | [2 - 4] | [0 - 1] |
| 10 | [64 - 80] | [1 - 2] | [3 - 4] | [0 - 0] |
| 11 | [55 - 59] | [1 - 1] | [0 - 4] | [0 - 0] |
| 12 | [18 - 25] | [1 - 2] | [0 - 4] | [0 - 0] |
| 13 | [40 - 42] | [1 - 2] | [0 - 4] | [0 - 1] |
| 14 | [29 - 33] | [1 - 2] | [0 - 2] | [0 - 0] |
| 15 | [29 - 33] | [1 - 2] | [0 - 2] | [0 - 0] |
| 16 | [64 - 80] | [1 - 2] | [3 - 4] | [0 - 0] |
| 17 | [18 - 25] | [1 - 2] | [0 - 4] | [0 - 0] |
| 18 | [40 - 42] | [1 - 2] | [0 - 4] | [0 - 1] |
| 19 | [18 - 25] | [1 - 2] | [0 - 4] | [0 - 0] |

(c) Weights = (.8,.05,.1,.05)

Table A.1: Sample dataset from FARS anonymized by equal and unequal weights

## A.2 Categorical Data

When we convert a categorical attribute into a vector of binary attributes, we use 1 to indicate presence of the property. For example, in the dataset used for the results shown in Figs. 3.3a - 3.3d, "Marital Status" attribute is represented as $(u_1, u_2, u_3, u_4, u_5, u_6)$ with each binary attribute $u_i$ representing a permissible value in Marital Status. In record 628 the entry for this attribute is 1 (representing status "Married; spouse present"), and thus represented as $(1, 0, 0, 0, 0, 0)$. Its equivalence class by Greedy Search are the records $\{628, 648, 819\}$, and the entries for records 648, 819 under Marital Status are 1 and 6 (representing status "Never married/single"), respectively. The anonymized outcome is represented as "1 | 1 | 6", which means the entries in this class can be translated back into high-level attributes as {"Married; spouse present", "Never married/single"}.

We give a definition of the GCP here that is similar to the definitions in (Wong *et al.*, 2010; Doka *et al.*, 2015).

DEFINITION 8: The Normalized Certainty Penalty (NCP) information loss metric for the anonymized outcome $x'_{ij}$ of an attribute $a_j$ is defined by:

$$NCP(x'_{ij}) = \begin{cases} \frac{count(x'_{ij})-1}{|a_j|} : a_j \texttt{ is categorical} \\ D_{ij} : a_j \texttt{ is numerical} \end{cases}$$

where $D_{ij}$ is as defined in Definition 4.

DEFINITION 9: We also define the Global Certainty Penalty (GCP) information loss metric for anonymized record $x'_i := [x'_{ij}]_{j \in J}$ to be:

$$GCP(x'_i) = \sum\nolimits_{j \in J} NCP(x'_{ij}) \tag{A.2.1}$$

## A.3 On the parameter S

The upper bounds on the sizes of the sub-problems in Fig. 3.5a - 3.5b are provided by LEMMA 2, and they are: 24, 27 and 30, for $S = 3$, $S = 4$ and $S = 5$, respectively. We give the actual distributions on the sizes of the sub-problems in Table A.2a - A.2c and see that the sizes rarely reach their upper bounds.

| Size | Count |
|---|---|
| 9 | 1 |
| 11 | 1 |
| 12 | 432 |
| 13 | 33 |
| 14 | 30 |
| 15 | 21 |
| 16 | 22 |
| 17 | 16 |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

(a) $S = 3$, $k = 3$

| Size | Count |
|---|---|
| 12 | 1 |
| 14 | 1 |
| 15 | 292 |
| 16 | 27 |
| 17 | 29 |
| 18 | 16 |
| 19 | 16 |
| 20 | 11 |
| 21 | 8 |
| 22 | 5 |
| 23 | 5 |
| 24 | 3 |
| 26 | 1 |
| 27 | 2 |
| | |

(b) $S = 4$, $k = 3$

| Size | Count |
|---|---|
| 15 | 1 |
| 18 | 200 |
| 25 | 15 |
| 22 | 16 |
| 23 | 12 |
| 26 | 19 |
| 20 | 17 |
| 28 | 9 |
| 19 | 23 |
| 21 | 12 |
| 27 | 3 |
| 24 | 4 |
| 29 | 1 |
| 30 | 1 |
| 13 | 1 |

(c) $S = 5$, $k = 3$

Table A.2: Distributions of sub-problem sizes for Fig. 3.5a - Fig. 3.5b

## A.4 Statistics of the Datasets

In this subsection we provide the statistics on the datasets used in our tests. In Table A.3 we display the variances of the CENSUS dataset (on 8 attributes) for different numbers of records, the min/max values were given in Table 3.6. We also display the statistics for the extended datasets of 15-35 attributes in Table A.4.

Note that the variances for each attribute in Table A.3 are similar across different sizes, thus we expect the information loss incurred to be similar across these datasets when the same $k$ is used. Indeed this is what we observed in Section 3.4.4 Fig. 3.6c and Fig. 3.6d.

| Number of Records | Gender | Age | Marital Status | Race | Birthplace | Education Level | Work Class | Occupation |
|---|---|---|---|---|---|---|---|---|
| 20,000 | 0.25 | 224.47 | 4.83 | 0.57 | 3340.06 | 7.33 | 11.27 | 78.00 |
| 30,000 | 0.25 | 225.03 | 4.82 | 0.56 | 3402.68 | 7.30 | 11.22 | 78.53 |
| 40,000 | 0.25 | 225.31 | 4.82 | 0.57 | 3414.58 | 7.33 | 11.20 | 78.72 |
| 50,000 | 0.25 | 225.64 | 4.81 | 0.58 | 3417.06 | 7.38 | 11.31 | 78.70 |
| 60,000 | 0.25 | 226.29 | 4.81 | 0.57 | 3372.39 | 7.36 | 11.34 | 78.66 |
| 70,000 | 0.25 | 226.37 | 4.83 | 0.56 | 3410.34 | 7.36 | 11.28 | 78.51 |
| 80,000 | 0.25 | 226.32 | 4.83 | 0.56 | 3412.74 | 7.38 | 11.28 | 78.53 |
| 90,000 | 0.25 | 226.96 | 4.83 | 0.56 | 3424.04 | 7.38 | 11.28 | 78.69 |
| 100,000 | 0.25 | 227.14 | 4.83 | 0.57 | 3412.27 | 7.39 | 11.31 | 78.50 |

Table A.3: Variances (rounded to 2 decimals) by attribute for the CENSUS datasets with 8 attributes

| Attribute | Variance | [min, max] |
|---|---|---|
| Family Unit | 0.004533004 | [1,3] |
| Family Size | 1.442669131 | [2,16] |
| Age of Oldest Child | 58.72416437 | [0,90] |
| Age of Youngest Child | 56.41184578 | [0,90] |
| Relationship to Household Head | 0.824784829 | [1,12] |
| Gender | 0.241520454 | [1,2] |
| Age | 100.447806 | [16,90] |
| Birth Quarter | 1.252327406 | [1,4] |
| Marital Status | 1.167666581 | [1,6] |
| Birth Year | 100.3407262 | [1889,1963] |
| Times Married | 0.148862341 | [0,2] |
| Age at First Marriage | 20.55224999 | [0,90] |
| Children Ever Born | 4.679614458 | [0,13] |
| Race | 0.393475514 | [1,7] |
| Birthplace | 4471.679274 | [1,465] |
| Years in the United States | 0.63840967 | [0,5] |
| Language Spoken | 30.45895319 | [1,96] |
| Highest Grade of Schooling | 8.217952835 | [1,23] |
| Class of Worker | 0.068740375 | [1,2] |
| Class of Worker - Detailed | 11.5164085 | [13,29] |
| Weeks Worked Last Year | 168.0059844 | [0,52] |
| Hours Worked Last Week | 140.8818287 | [1,99] |
| Usual Hours of Work | 156.7482934 | [0,99] |
| Weeks Unemployed Last Year | 26.31284308 | [0,52] |
| Income From Wages | 98538554.92 | [0,75000] |
| Other Income | 2669021.007 | [0,75000] |
| Occupation | 81.01775489 | [4,79] |
| Migration Status, 5 Years | 0.459946975 | [1,4] |
| Place of Work, State | 94.98776039 | [1,80] |
| Means of Transportation to Work | 39.7001238 | [11,70] |
| Vehicle Occupancy | 0.708769336 | [0,7] |
| Time in Transit | 239.9953666 | [0,99] |
| Citizenship | 0.13065803 | [0,3] |
| Year of Immigration | 0.68959079 | [0,6] |
| Work Disability Status | 0.033916173 | [1,2] |

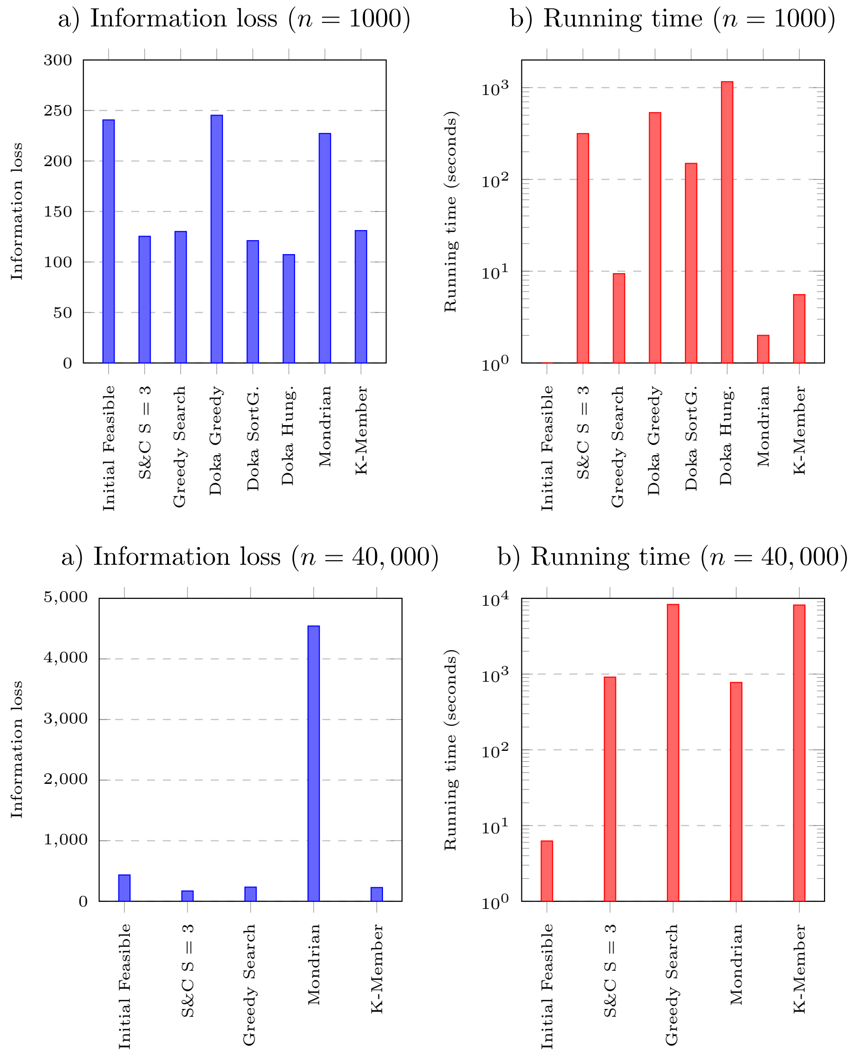
Table A.4: Variances, min/max values of the CENSUS datasets with 15-35 attributes

## A.5 Alternate Database

In this subsection we present some test results using alternative datasets from the FARS (US Department of Transportation, 2016) database containing traffic accidents data. We have datasets on 4 attributes: MONTH (of occurrence), AGE, SEX, INJ_SEV. All 4 attributes can be viewed as numeric (with SEX being binary).

In Fig. A.1a - A.1b, we see that our algorithms still achieve similar utility to the Doka Hungarian algorithm while having better performance. Our Greedy Search appears to have better performance than our Split & Carry in Fig. A.1b when we have only 1000 records; however, we see in Fig. A.1b that when we increase the number of records to 40000, Split & Carry has both better performance and utility. These results are consistent with our earlier observations.

Figure A.1: Comparison using FARS database, $m = 4, k = 3$



## A.6 Hardware Information

In this subsection we list the typical hardware specifications for running the experiments, in Table A.5a and Table A.5b. We used Google Cloud Computing Services (https://cloud.google.com/) with machine type n1-highcpu-8. In Table A.5b we list the specifications for one of the processors, the remaining seven are identical except for the processor id, core id, apicid and initial apicid.

| | |
|---|---|
| MemTotal | 7352436 kB |
| MemFree | 6587692 kB |
| MemAvailable | 6966836 kB |
| Buffers | 50336 kB |
| Cached | 542508 kB |
| SwapCached | 0 kB |
| Active | 409856 kB |
| Inactive | 233532 kB |
| Active(anon) | 50700 kB |
| Inactive(anon) | 10120 kB |
| Active(file) | 359156 kB |
| Inactive(file) | 223412 kB |
| Unevictable | 0 kB |
| Mlocked | 0 kB |
| SwapTotal | 0 kB |
| SwapFree | 0 kB |
| Dirty | 0 kB |
| Writeback | 0 kB |
| AnonPages | 50540 kB |
| Mapped | 23064 kB |
| Shmem | 10280 kB |
| Slab | 43912 kB |
| SReclaimable | 29668 kB |
| SUnreclaim | 14244 kB |
| KernelStack | 2160 kB |
| PageTables | 2780 kB |
| NFS_Unstable | 0 kB |
| Bounce | 0 kB |
| WritebackTmp | 0 kB |
| CommitLimit | 3676216 kB |
| Committed_AS | 95452 kB |
| VmallocTotal | 34359738367 kB |
| VmallocUsed | 0 kB |
| VmallocChunk | 0 kB |
| HardwareCorrupted | 0 kB |
| AnonHugePages | 0 kB |
| ShmemHugePages | 0 kB |
| ShmemPmdMapped | 0 kB |
| HugePages_Total | 0 |
| HugePages_Free | 0 |
| HugePages_Rsvd | 0 |
| HugePages_Surp | 0 |
| Hugepagesize | 2048 kB |
| DirectMap4k | 99316 kB |
| DirectMap2M | 3256320 kB |
| DirectMap1G | 5242880 kB |

(a) Memory Specifications

| | |
|---|---|
| processor | 0 |
| vendor_id | GenuineIntel |
| cpu family | 6 |
| model | 79 |
| model name | Intel(R) Xeon(R) CPU @ 2.20GHz |
| stepping | 0 |
| microcode | 0x1 |
| cpu MHz | 2200 |
| cache size | 56320 KB |
| physical id | 0 |
| siblings | 8 |
| core id | 0 |
| cpu cores | 4 |
| apicid | 0 |
| initial apicid | 0 |
| fpu | yes |
| fpu_exception | yes |
| cpuid level | 13 |
| wp | yes |
| bugs | cpu meltdown spectre v1 spectre v2 spec store bypass l1tf mds swapgs |
| bogomips | 4400 |
| clflush size | 64 |
| cache_alignment | 64 |
| address sizes | 46 bits physical, 48 bits virtual |
| power management | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

(b) CPU Specifications

Table A.5: Hardware Specifications

# Appendix B

## B.1 Sample Results for Superimposition Attacks

In Fig. B.1 - Fig. B.4 samples of resulting images with adversarial perturbations are shown. The leftmost column displays the original images, the middle columns display the adversarial examples with the two or three smallest perturbations, the last column shows the superimposition of the two or three adversarial examples. The rows correspond to different targets being applied in the adversarial examples. Classifications of these images are provided in Tables B.1 - B.2.

(a) Sample 1, corresponds to Fig. B.1a - Fig. B.1d.

| Target | SI (2×) | SI-NL (2×) | SI (3×) | SI-NL (3×) |
|---|---|---|---|---|
| 0 | 7 | 7 | 0 | 7 |
| 1 | 2 | 7 | 2 | 7 |
| 2 | 7 | 7 | 2 | 7 |
| 3 | 7 | 7 | 3 | 7 |
| 4 | 4 | 7 | 4 | 7 |
| 5 | 7 | 7 | 5 | 7 |
| 6 | 6 | 7 | 6 | 7 |
| 8 | 7 | 7 | 8 | 7 |
| 9 | 9 | 7 | 9 | 7 |

(b) Sample 2, corresponds to Fig. B.2a - Fig. B.2d.

| Target | SI (2×) | SI-NL (2×) | SI (3×) | SI-NL (3×) |
|---|---|---|---|---|
| 0 | 2 | 2 | 2 | 2 |
| 1 | 2 | 2 | 2 | 2 |
| 3 | 2 | 2 | 3 | 2 |
| 4 | 8 | 2 | 4 | 4 |
| 5 | 2 | 2 | 5 | 2 |
| 6 | 2 | 2 | 6 | 2 |
| 7 | 7 | 8 | 7 | 3 |
| 8 | 2 | 2 | 8 | 2 |
| 9 | 8 | 2 | 9 | 9 |

Table B.1: Classifications of MNIST Samples

(a) Sample 1, corresponds to Fig. B.3a - Fig. B.3d.

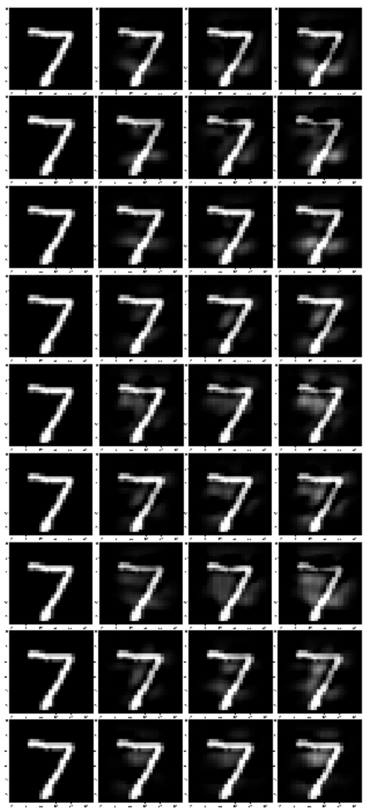
| Target | SI (2×) | SI-NL (2×) | SI (3×) | SI-NL (3×) |
|---|---|---|---|---|
| 0 | 3 | 3 | 3 | 3 |
| 1 | 3 | 3 | 3 | 3 |
| 2 | 3 | 3 | 3 | 3 |
| 4 | 3 | 3 | 3 | 3 |
| 5 | 3 | 3 | 3 | 3 |
| 6 | 3 | 3 | 3 | 3 |
| 7 | 3 | 3 | 3 | 3 |
| 8 | 3 | 3 | 3 | 3 |
| 9 | 3 | 3 | 3 | 3 |

(b) Sample 2, corresponds to Fig. B.4a - Fig. B.4d.

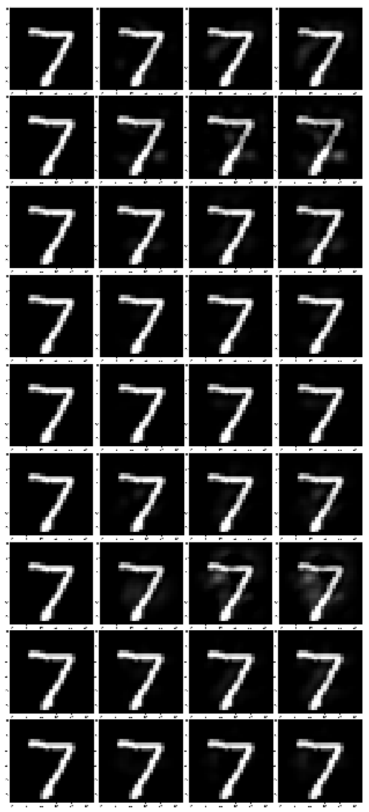
| Target | SI (2×) | SI-NL (2×) | SI (3×) | SI-NL (3×) |
|---|---|---|---|---|
| 0 | 8 | 8 | 8 | 8 |
| 1 | 8 | 8 | 8 | 8 |
| 2 | 8 | 8 | 8 | 8 |
| 3 | 8 | 8 | 8 | 8 |
| 4 | 8 | 8 | 8 | 8 |
| 5 | 8 | 8 | 8 | 1 |
| 6 | 8 | 8 | 8 | 8 |
| 7 | 8 | 8 | 0 | 8 |
| 9 | 1 | 8 | 1 | 8 |

Table B.2: Classifications of CIFAR10 Samples

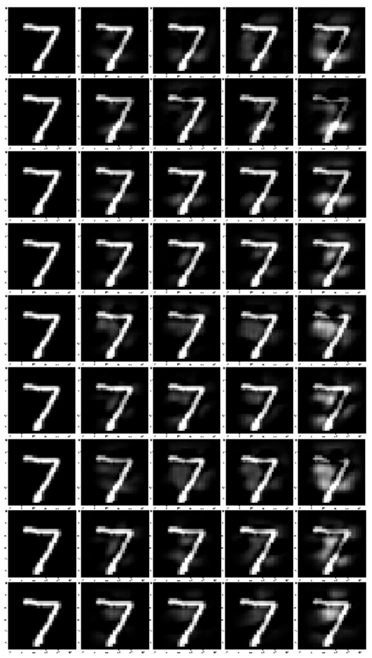
(a) SI (2×)



(b) SI-NL (2×)



(c) SI (3×)



(d) SI-NL (3×)

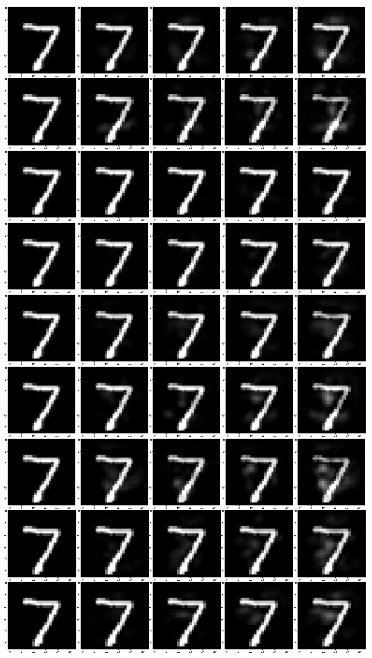
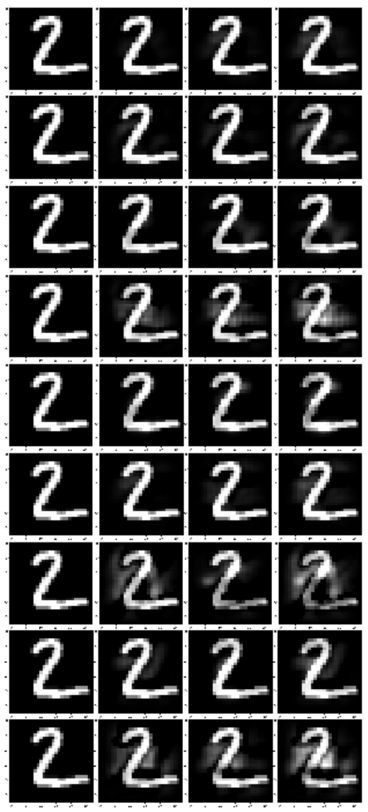
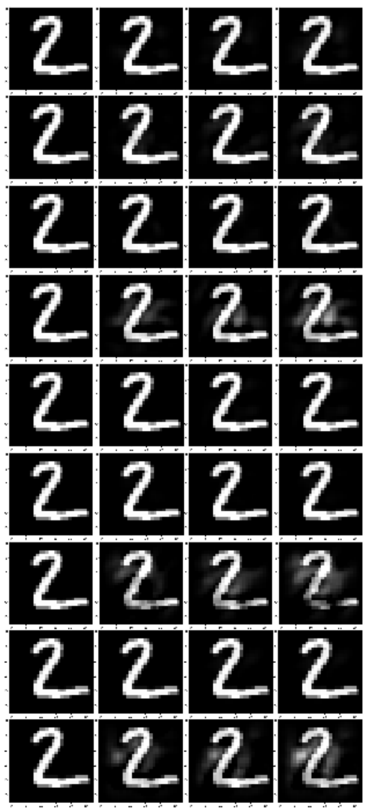


Figure B.1: Adversarial images using Superimposition, MNIST Sample 1

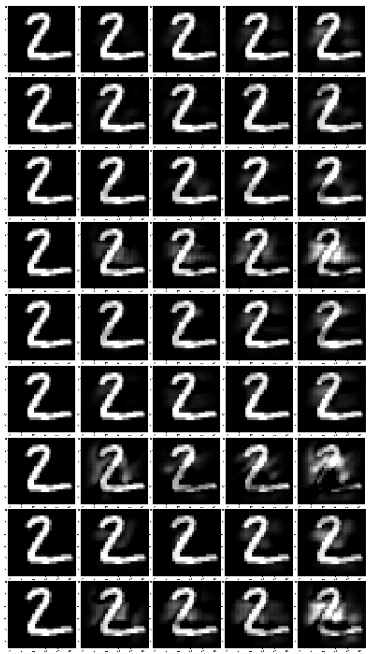
(a) SI (2×)



(b) SI-NL (2×)



(c) SI (3×)



(d) SI-NL (3×)

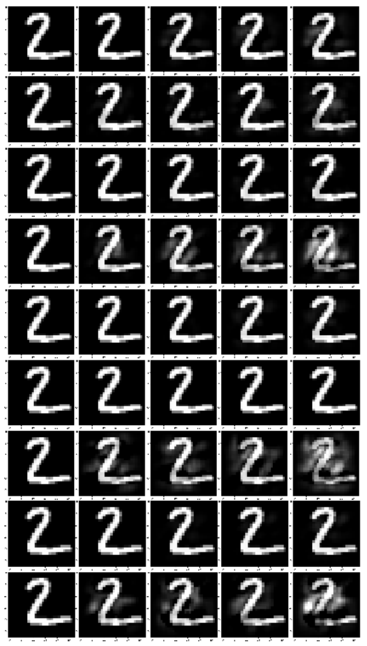
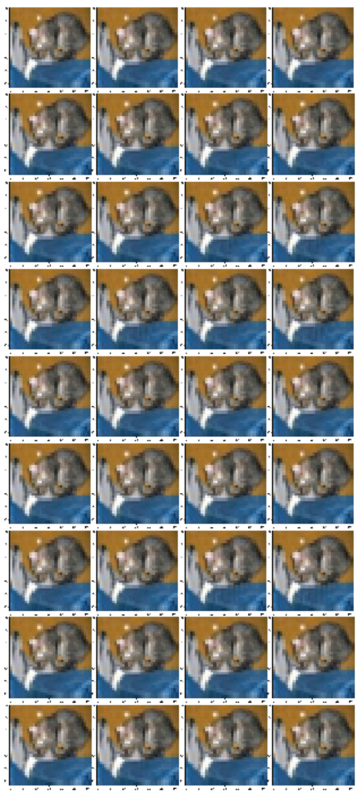
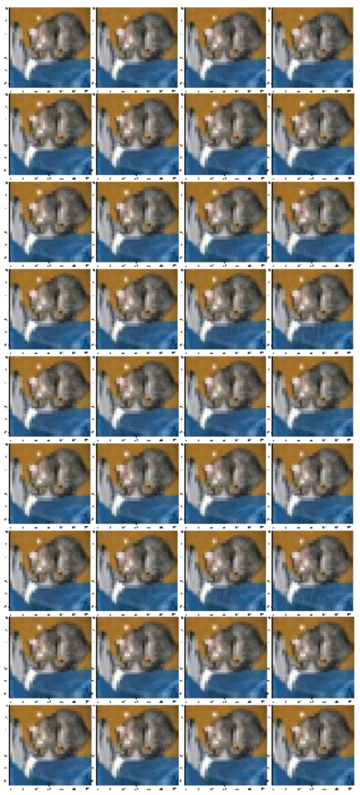


Figure B.2: Adversarial images using Superimposition, MNIST Sample 2

(a) SI (2×)



(b) SI-NL (2×)



(c) SI (3×)

(d) SI-NL (3×)

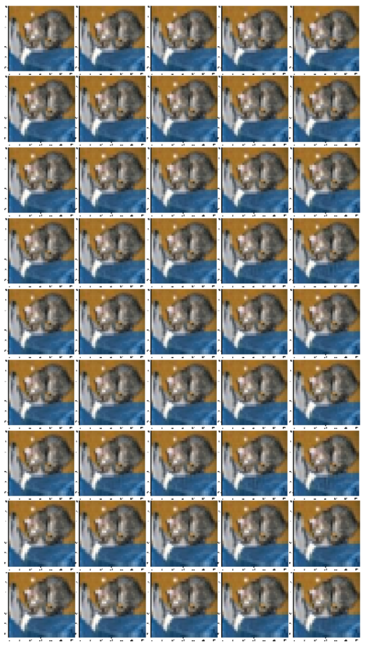
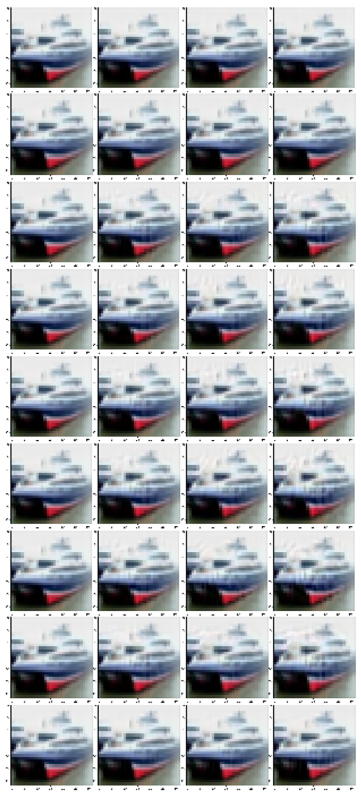
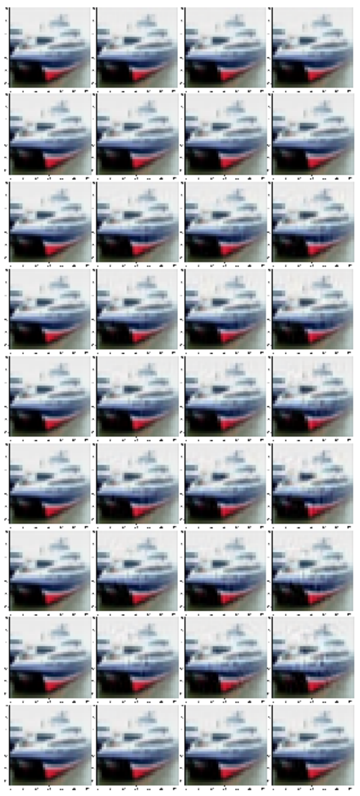


Figure B.3: Adversarial images using Superimposition, CIFAR10 Sample 1

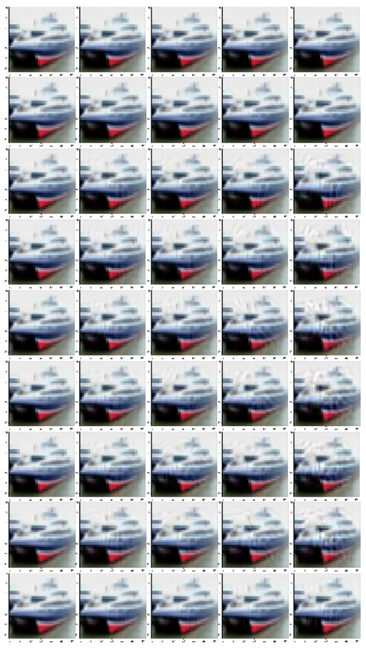
(a) SI (2×)



(b) SI-NL (2×)



(c) SI (3×)



(d) SI-NL (3×)

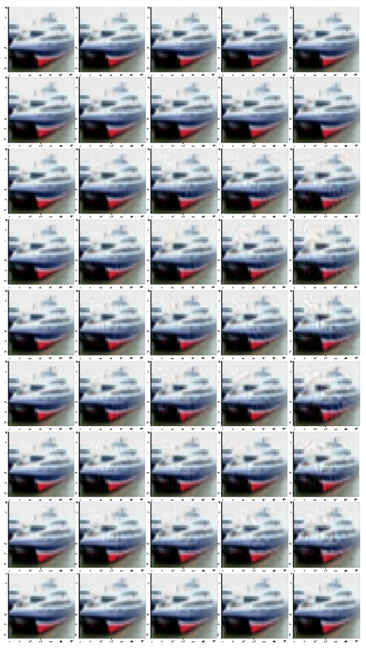


Figure B.4: Adversarial images using Superimposition, CIFAR10 Sample 2

# Bibliography

Aggarwal, C. C. (2005). On k-anonymity and the curse of dimensionality. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 901–909. VLDB Endowment.

Aggarwal, G., Feder, T., Kenthapadi, K., Motwani, R., Panigrahy, R., Thomas, D., and Zhu, A. (2005). Approximation algorithms for k-anonymity. In *Proceedings of the International Conference on Database Theory (ICDT 2005)*.

Aghdam, M. R. S. and Sonehara, N. (2016). Achieving high data utility k-anonymization using similarity-based clustering model. *IEICE Transactions on Information and Systems*, **99**(8), 2069–2078.

Ardagna, C. A., Cremonini, M., di Vimercati, S. D. C., and Samarati, P. (2011). An obfuscation-based approach for protecting location privacy. *IEEE Transactions on Dependable and Secure Computing*, **8**(1), 13–27.

Ayala-Rivera, V., McDonagh, P., Cerqueus, T., Murphy, L., *et al.* (2014). A systematic comparison and evaluation of k-anonymization algorithms for practitioners. *Transactions on Data Privacy*, **7**(3), 337–370.

Bayardo, R. J. and Agrawal, R. (2005). Data privacy through optimal k-anonymization. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pages 217–228. IEEE.

Burridge, J. (2003). Information preserving statistical obfuscation. *Statistics and Computing*, **13**(4), 321–327.

Byun, J.-W., Kamra, A., Bertino, E., and Li, N. (2007). Efficient k-anonymization using clustering techniques. In *International Conference on Database Systems for Advanced Applications*, pages 188–200. Springer.

Canada Institute for Health Information (2014). CIHI portal - privacy impact assessment. Available online at: `https://www.cihi.ca/en/portal_addendum_1407_pdf_en.pdf`. Last accessed: Aug. 2018.

Cancer Care Ontario (2018). Data use & disclosure policy. Available online at: `https://www.ccohealth.ca/sites/CCOHealth/files/assets/DataUseAndDisclosurePolicy.pdf`. Last accessed: Sep. 2019.

Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE.

Chawla, S., Dwork, C., McSherry, F., Smith, A., and Wee, H. (2005). Toward privacy in public databases. In *Theory of Cryptography Conference*, pages 363–385. Springer.

De Capitani Di Vimercati, S., Foresti, S., Livraga, G., and Samarati, P. (2012). Data privacy: definitions and techniques. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, **20**(06), 793–817.

Dobkin, D. P. and Reiss, S. P. (1980). The complexity of linear programming. *Theoretical Computer Science*, **11**(1), 1–18.

Doka, K., Xue, M., Tsoumakos, D., and Karras, P. (2015). k-anonymization by freeform generalization. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, pages 519–530. ACM.

Dwork, C. (2011). A firm foundation for private data analysis. *Communications of the ACM*, **54**(1), 86–95.

El Emam, K. and Dankar, F. K. (2008). Protecting privacy using k-anonymity. *Journal of the American Medical Informatics Association*, **15**(5), 627–637.

Genova, K. and Guliashki, V. (2011). Linear integer programming methods and approaches–a survey. *Journal of Cybernetics and Information Technologies*, **11**(1).

Ghinita, G., Karras, P., Kalnis, P., and Mamoulis, N. (2009). A framework for efficient data anonymization under privacy and accuracy constraints. *ACM Transactions on Database Systems*, **34**(2), 9.

Gionis, A., Mazza, A., and Tassa, T. (2008). k-anonymization revisited. In *2008 IEEE 24th International Conference on Data Engineering*, pages 744–753. IEEE.

Goldberger, J. and Tassa, T. (2009). Efficient anonymizations with enhanced utility. In *2009 IEEE International Conference on Data Mining Workshops*, pages 106–113. IEEE.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Gurobi Optimization, LLC. (2018). Mixed-integer programming (mip)-a primer on the basics. Available online at: `http://www.gurobi.com/resources/getting-started/mip-basics`. Last accessed: Mar. 2018.

Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

Information and Privacy Commissioner of Ontario (2016). De-identification guidelines for structured data. Available online at: `https://www.ipc.on.ca/privacy/de-identification-centre/`. Last accessed: Aug. 2018.

Iyengar, V. S. (2002). Transforming data to satisfy privacy constraints. In *Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 279–288. ACM.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Available online at: `https://www.cs.toronto.edu/~kriz/cifar.html`. Last accessed: Mar. 2019.

Kurakin, A., Goodfellow, I. J., and Bengio, S. (2017). Adversarial examples in the physical world. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net.

LeCun, Y., Cortes, C., and Burges, C. J. (1998). The mnist database of handwritten digits. Available online at: `http://yann.lecun.com/exdb/mnist/`. Last accessed: Mar. 2019.

Lee, H., Kim, S., Kim, J. W., and Chung, Y. D. (2017). Utility-preserving anonymization for health data publishing. *BMC medical informatics and decision making*, **17**(1), 104.

LeFevre, K., DeWitt, D. J., and Ramakrishnan, R. (2006). Mondrian multidimensional k-anonymity. In *Data Engineering, 2006. ICDE 2006. Proceedings of the 22nd International Conference on*, pages 25–25. IEEE.

Li, N., Li, T., and Venkatasubramanian, S. (2007). t-closeness: Privacy beyond k-anonymity and l-diversity. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 106–115. IEEE.

Liang, Y. and Samavi, R. (2020). Optimization-based anonymity algorithms. *Computers & Security*, **93**, 101753.

Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical programming*, **45**(1-3), 503–528.

Liu, K., Kargupta, H., and Ryan, J. (2006). Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Transactions on Knowledge and Data Engineering*, **18**(1), 92–106.

Meyerson, A. and Williams, R. (2004). On the complexity of optimal k-anonymity. In *Proceedings of the 23rd ACM SIGMOD Symposium on Principles of Database Systems*, pages 223–228. ACM.

Mittelmann, H. (2018). Mixed integer linear programming benchmark. Available online at: `http://plato.asu.edu/ftp/milpc.html`. Last accessed: Jun. 2018.

Mosek (2018). Modelling cookbook 2.3. Available online at: `https://docs.mosek.com/modeling-cookbook/mio.html`. Last accessed: Mar. 2018.

National Center for Health Statistics (2019). Preventing disclosures: Rules for researchers. Available online at: `https://www.cdc.gov/rdc/data/b4/disclosuremanual.pdf`. Last accessed: Sep. 2019.

Nemhauser, G. and Wolsey, L. (1988). *Integer and combinatorial optimization*. Wiley-Interscience series in discrete mathematics and optimization. Wiley.

Papadimitriou, C. H. (1981). On the complexity of integer programming. *Journal of the ACM (JACM)*, **28**(4), 765–768.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. (2016a). Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. (2016b). The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387. IEEE.

Papernot, N., McDaniel, P., and Goodfellow, I. (2016c). Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*.

Papernot, N., Abadi, M., Erlingsson, Ú., Goodfellow, I. J., and Talwar, K. (2017). Semi-supervised knowledge transfer for deep learning from private training data. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Samarati, P. (2001). Protecting respondents identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, **13**(6), 1010–1027.

Samarati, P. and Sweeney, L. (1998). Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, SRI International.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Statistics Canada (2014). Statistics canada quality guidelines - disclosure control. Available online at: `https://www150.statcan.gc.ca/n1/pub/12-539-x/steps-etapes/4058325-eng.htm`. Last accessed: Sep. 2019.

Sweeney, L. (2002). k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, **10**(05), 557–570.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. (2014). Intriguing properties of neural networks. In Y. Bengio and Y. LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Tassa, T., Mazza, A., and Gionis, A. (2012). k-concealment: An alternative model of k-type anonymity. *Transactions on Data Privacy*, **5**(1), 189–222.

University of Minnesota (2018). Ipums-usa. Available online at: `https://www.ipums.org/`. Last accessed: Sep. 2019.

U.S. Department of Health & Human Services (2015). Guidance regarding methods for de-identification of protected health information in accordance with the health insurance portability and accountability act (HIPAA) privacy rule. Available online at: `https://www.hhs.gov/hipaa/for-professionals/privacy/special-topics/de-identification/index.html`. Last acessed: Aug. 2018.

US Department of Transportation (2016). Fatal analysis reporting system. Available online at: `ftp://ftp.nhtsa.dot.gov/FARS/2016`. Last accessed: Aug. 2018.

UT Dallas Data Security and Privacy Lab (2012). UTD anonymization toolbox. Available online at: `http://www.cs.utdallas.edu/dspl/cgi-bin/toolbox/index.php`. Last accessed: Aug. 2018.

von zur Gathen, J. and Sieveking, M. (1978). A bound on solutions of linear integer equalities and inequalities. *Proceedings of the American Mathematical Society*, **72**(1), 155–158.

Williams, H. (1993). *Model Building in Mathematical Programming*. Wiley.

Wong, W. K., Mamoulis, N., and Cheung, D. W. L. (2010). Non-homogeneous generalization in privacy preserving data publishing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, pages 747–758. ACM.

Xiao, X. and Tao, Y. (2006a). Anatomy: Simple and effective privacy preservation. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 139–150. VLDB Endowment.

Xiao, X. and Tao, Y. (2006b). Personalized privacy preservation. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pages 229–240. ACM.

Xu, J., Wang, W., Pei, J., Wang, X., Shi, B., and Fu, A. W.-C. (2006). Utility-based anonymization using local recoding. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–790. ACM.

Zhang, B., Mohammed, N., Dave, V. S., and Hasan, M. A. (2017). Feature selection for classification under anonymity constraint. *Transactions on Data Privacy*, **10**(1), 1–25.