Utilising Autoencoder Latent Representations to Pseudonymise Data whilst Retaining Data Utility

Daniël Salomons, BSc
6508383
D.Salomons@students.uu.nl

Master Computing Science
Department of Information and Computing Sciences
Faculty of Science
Utrecht University

First supervisor:
dr. ing. Georg Krempl
Algorithmic Data Analysis
Faculty of Science
Utrecht University
G.M.Krempl@uu.nl

Second supervisor:
dr. Mihaela Mitici
Algorithmic Data Analysis
Faculty of Science
Utrecht University
M.A.Mitici@uu.nl

Daily supervisor:
Maarten Vos, MSc
Chapter Lead Data & AI
Business Unit Mobility & Public
Info Support
Maarten.Vos@infosupport.com

Master Thesis

August 16, 2023

Abstract

Autoencoders are small encoder-decoder pair networks that learn to compress data into a latent representation of smaller dimension. This thesis aims to outline the benefits and drawbacks of using latent representations as a utility-preserving data pseudonymisation method for machine learning. We consult existing anonymisation literature and EU legislature, followed by experiments on latent representation decoding, data utility and other latent representation properties. We found that without a leak of the original data along with its latent representation, it is difficult for an adversary to generate a well-performing reconstruction of the encoded dataset. This method is more effective if the latent representation is randomly permuted. This permutation is not easily reversed by a clustering algorithm. A latent representation preserves its data utility well for classification algorithms, even when permuted. Our experiments indicate that a dataset can be represented by multiple, well-performing latent representations, making it difficult for an adversary to discern which dataset was originally encoded. Autoencoders are quick to train, making it a quick method to pseudonymise data whilst retaining data utility for classification algorithms. As a pseudonymisation method, it is possible for the data holder to obtain a reconstruction of the data. However, latent representations would likely not be considered anonymised data by GDPR. Furthermore, regression algorithms perform worse than classification algorithms on latent representations. Finally, despite the popularity of mean squared error, we find that this loss function does not maximise data utility in latent representations.

Keywords: Data, anonymisation, pseudonymisation, GDPR, privacy-utility trade-off, autoencoder, hidden representation, latent representation

Contents

Table 1: List of abbreviations

Abbreviation	**Definition**	
MSE	Mean Squared Error	
SGD	Stochastic Gradient Descent	
$\mathbf{x}$	Input vector of original dataset	
$\mathbf{y}$	Class label vector	
$\mathbf{z}$	Latent representation of original dataset	
$\mathbf{x}'$	Reconstruction of original dataset	
$f(\cdot)$	Encoder network	
$f'(\cdot)$	Decoder network	
$g(\cdot)$	Adversarial decoder network	
TL	True Loss	$MSE(g(\mathbf{z}), \mathbf{x})$
AL	Approximate Loss	$MSE(g(\mathbf{z}), \mathbf{x}')$
HM	Heterogeneity Measure	
WP	Article 29 Working Party	

Chapter 1

Problem introduction

1.1 Introduction

Modern research and business are increasingly driven by data analysis and automated decision-making (Micheli et al., 2020; Yen, 2021). The cloud-based AI market was valued at 44 billion dollars in 2022 and this market is expected to grow exponentially (Grand View Research, 2023). Data analysis is an important tool that also comes with its own risks. Negligent handling of big data leading to data leaks occurs worryingly frequently (Drapkin, 2023) and can have incredibly damaging consequences for affected individuals, such as identity theft, fraud, or damage to one's social standing. To ensure that companies and researchers handle this powerful approach with appropriate care, the European Commission introduced the General Data Protection Regulation (GDPR) in 2016 which came into effect in 2018 (European Commission, 2016c). Almost immediately after coming into effect in 2018, the first multimillion fines were imposed on various companies in Europe for breaching data protection agreements (Panda Security, 2019), highlighting the need for secure data storage and handling.

In cases where the minimisation of data collection is impossible, it would at least be ideal for sensitive data to be anonymised, or at least pseudonymised. This way, even if the data was compromised, the persons to whom the data belong would not be affected. The trivial solution is to remove direct identifiers, such as names, identification numbers, phone numbers, or any other characteristics that are unique to a natural person. The literature refers to this type of data as Personally Identifiable Information (PII). However, a small amount of indirectly identifying data has been found to be used to identify natural persons in anonymised datasets (Sweeney, 2000). This is done through a process called *record linkage*. Record linkage is the process of identifying the same person in two separate anonymised datasets. Whilst each individual dataset on its own is properly anonymised prior to being released or sold, when combining the two

datasets, a person may be identified by the matching remaining (indirect) identifiers that were left in the dataset. A prominent example of this is in the work of Sweeney, 2002 in which she identified the medical records of William Weld, governor of Massachusetts, matching his ZIP code, birth date, and sex between two datasets. She achieved this by linking an anonymised voter list, which was publicly available for twenty dollars, and an anonymised medical dataset which was distributed to researchers. Governor Weld previously assured the public of the anonymity of the same medical dataset. This method of record linkage is direct; more statistical approaches, such as using string similarity metrics to attempt to deanonymise names that are partially changed or omitted, are also used. Removing even more data to prevent indirect record linkage has a significant impact on the efficacy of data analysis, leading to the need for alternative methods that strike a balance between privacy and data utility.

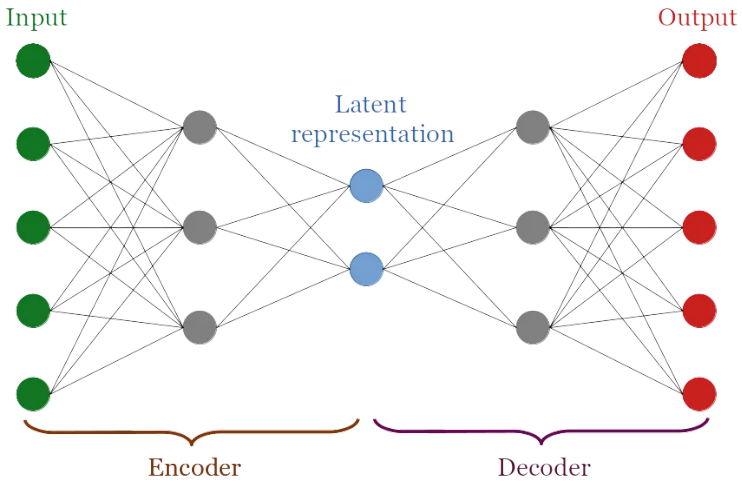

Figure 1.1: A simple example autoencoder network structure (Yang, 2020).

1.1.1 Introducing autoencoders

Anonymisation methods that involve the usage of neural networks have recently attracted great interest. An example is *autoencoder* networks. Autoencoders are relatively small, unsupervised encoder-decoder pair networks. Analogically, the encoder and decoder networks 'cooperate' to devise a smaller, well-representative summary of the input data. Like summarising a long text document with fewer words, an encoder network learns to represent a higher-dimensional dataset in fewer dimensions. In turn, the decoder network tries to reconstruct the original data based on this summary.

An encoder network structure can be likened to a funnel. Continuing our earlier analogy, the encoder iteratively makes a smaller summary of the input until the

desired 'bottleneck' size has been reached. The decoder network mirrors this structure, progressively increasing the size until the data is returned back to the dimensionality it originally was. This allows us to compare the reconstructed result with the original dataset and train the network. Refer to Figure 1.1 for an illustration of the architecture of an autoencoder network. We refer to the output of the encoder network (the summary in our analogy) as the *latent representation* of the data. The maths behind autoencoders is explained in more detail in Section 2.3. The exact origin of autoencoders is debatable, but the ideas behind them have been discussed since 1985 (Rumelhart et al., 1985).

Autoencoder networks can be used in various ways. Autoencoders as described above can be used for a process called *feature selection*, where the encoder network is used as a feature selection method to represent higher-dimensional data in lower-dimensionality data whilst retaining the most pertinent features or combining multiple features into one. This can improve the performance of classification methods that are sensitive to high dimensionality (Balın et al., 2019; Han et al., 2018; Sun et al., 2016; Wang et al., 2017; Xu et al., 2019). We explore this property of autoencoders in an experiment described in Section 4.1.4.

Autoencoders can also be used the other way around. By providing a pre-trained decoder network with a novel summary, the decoder network generates something new based on what it has learnt (Higgins et al., 2017). Although interesting, this use of autoencoders falls outside the scope of our research.

In the field of data anonymisation, both encoders and decoders are used. First, the encoder network generates a latent representation of the data. The latent representation is transformed in some way to protect the privacy of whomever the data originally belongs to. The decoder network reconstructs this transformed latent representation into an anonymised dataset. (Hajihassani et al., 2021; Malekzadeh et al., 2019; Malekzadeh et al., 2017; Perero-Codosero et al., 2022). We discuss some of these techniques in Section 2.1.

1.1.2 Problem statement

In this research, we aim to outline the benefits and drawbacks of using an autoencoder's latent representation as a method of data pseudonymisation that retains the utility of the data for the purposes of machine learning. One factor we investigate is whether a latent representation already innately suffices as an anonymisation or pseudonymisation method as proposed by Yang, 2020. In his blog post, Yang shows that a random forest classifier trained on a latent representation scores similarly in terms of accuracy, F1 score, ROC curve, and precision compared to using the original dataset. Furthermore, the decision tree trained on the latent representation attached the same importance to the features that the decision tree trained on the original dataset deemed important, further indicating that the input data is effectively summarised by the latent representation.

In terms of privacy protection, Yang argues that the data is anonymous because it is not understandable for humans. Carrying on the analogy from above, the shorthand that the encoder-decoder pair network developed to create the summaries is incomprehensible for any outside party looking in. However, Yang's article lacks scientific evidence for the claim that this representation is anonymous. That is, Yang neither shows that the method nor the nature of latent representations satisfy established privacy metrics in the literature and in the legislation or that the encoding process is an irreversible function; in other words, whether an adversary is able to decode the shorthand back into something comprehensible. Furthermore, there is no scientific literature that proves or disproves this claim. As such, this thesis aims to investigate these claims. We start by discussing and evaluating existing definitions and metrics of anonymisation in the scientific literature and EU legislature.

Next, we experimentally evaluate anonymisation metrics and data utility performance on latent representations for classification and regression algorithms not covered by Yang's article. We assume that we work with machine learning training data, or data that otherwise has a distribution that is learnable by the autoencoder. Finally, we perform a few experiments to discover properties of latent representations, and test the average training time of autoencoders on data of various sizes.

The practical application would be that data scientists could develop machine learning algorithms or neural networks with a latent representation that has a distribution similar to the original data, whilst avoiding the possibility of leaking the original data, or exposing the data scientist to personally identifiable information (PII).

1.1.3 Outline

We will briefly describe the structure of the remainder of the report. Section 1.2 lists the research questions addressed in this report. We dedicate Chapter 2 to establishing the background knowledge for this thesis. First, we discuss the relevant work on anonymising autoencoders. Second, we list existing work on anonymisation and data privacy in both the literature and EU legislature. Third, we explain the mathematical concepts behind autoencoders in greater detail. Chapter 3 relates the privacy models and metrics to the mathematical theory we discussed to draw our theoretical conclusions of the pseudonymising performance of autoencoders. Chapter 4 evaluates the experimental pseudonymisation performance of autoencoders. Section 4.1 describes the experimental setup along with the measures, the datasets, and the experimental procedures used. Section 4.2 shows the results generated by the experiments and provides a discussion. The report draws its general conclusions in Chapter 5, followed by a discussion of the limitations of the chosen treatment, and ends with ideas for future work.

1.2 Research questions

This thesis aims to answer the following main research question:

- **RQ**: What are the advantages and disadvantages of using an autoencoder's latent representation as a utility-preserving data pseudonymisation method for machine learning?

To answer the main research question, we have devised the following subquestions (**SQ**) which, when answered, should cumulatively result in the answer of our research question:

- **SQ1**: What privacy standards, models, and metrics exist in the current literature or legislation and which of these measures apply to latent representations?
- **SQ2**: What experimental measures of pseudonymity does an autoencoder's latent representation have?
- **SQ3**: What is the utility loss incurred when using a latent representation of the data rather than the original data?

Chapter 2

Related work and background

2.1 Related work

In this chapter, we first describe existing data anonymisation methods using autoencoders. We then discuss existing privacy metrics both proposed in scientific literature and established in EU legislation with a focus on GDPR. Finally, we explain the mathematical theory of autoencoders in greater detail.

2.1.1 Latent representation manipulation

A paper by Hajihassani et al., 2020 defines the concept of a *mean latent representation*. Consider data that has one or more attributes (columns) that explicitly need to be kept private, such as which gender belongs to which sports time series data point. Hajihassani et al. call these *private variables*. The authors postulate that calculating the mean of all latent representations belonging to one specific value of the private variable must inherently capture how that value of the private variable is represented in the latent representation. For example, taking the latent representation of a record of male sports data, subtracting the mean latent representation of all male sports data from said latent representation and adding the mean latent representation of all female sports data should result in a record that originally belonged to a male sporter, but in the reconstructed data is now attributed to a female sporter. Effectively, this anonymises the distributions of male and female sports data by obfuscating one with the other and vice versa. Experimenting with this pipeline shows that the accuracy of attempting to identify gender in the sports data is significantly reduced whilst largely retaining the utility of recognising the sports activity.

2.1.2 ObscureNet

Another approach in the literature is ObscureNet (Hajihassani et al., 2021). In this paper, the authors propose a *Conditional Variable Autoencoder* (CVAE) network (Sohn et al., 2015), which is a variation on the 'standard' autoencoder network architecture. A variable autoencoder (VAE) learns a *latent distribution* of the input, rather than a direct, 1:1 latent representation of the input. Furthermore, this distribution is assumed to be Gaussian: hence, the latent distribution is denoted with a median and a standard deviation. This has the benefit of being able to be used as a generative model. If one samples this learnt latent distribution and feeds it to the trained decoder, the decoder generates new datapoints based on its training dataset (Kingma and Welling, 2022).

Expanding on this idea, a CVAE improves this VAE network in that the latent distribution is conditioned on an additional input (Kingma et al., 2014; Sohn et al., 2015). During training, the latent distribution learns various modes based on this additional input. For example, a CVAE trained on the MNIST dataset, a popular image dataset of handwritten numbers (Deng, 2012, and see also Figure 2.3), can be conditioned to learn which mode belongs to which number. This results in being able to ask this network to specifically generate the number three, rather than any of the ten numbers in its dataset.

ObscureNet reverses this result. The CVAE network is conditioned on private variables. But instead of training the network to generate new datapoints based on the mode provided, ObscureNet uses *adversarial information factorisation*. As a result, the latent distribution is trained to be statistically independent of the private variable on which it is conditioned. Therefore, sampling and decoding from this network results in new datapoints with the conditioned private variable obscured. Attempts to infer the private variable based on this transformed data are significantly less accurate, whilst preserving data utility in classification tasks revolving around non-anonymised variables. To anonymise multiple private variables, the authors propose chaining these CVAE networks, each conditioned on different private variables repeating this process. This chained design keeps the network structure small and allows its usage in computationally light Internet of Things devices.

2.1.3 Replacing autoencoder

A paper by different authors proposed a "replacing" autoencoder (Malekzadeh et al., 2017). This paper defines three disjoint sets of inferences: white-listed inferences that are desired by some third party requesting the data, grey-listed inferences that are neither requested by said third party nor considered sensitive by the users to whom the data belongs, and black-listed inferences that are considered sensitive by the users.

An autoencoder is trained to learn to replace black-listed data with grey-listed data to protect sensitive inferences. Their experiments show that their trained autoencoder was successfully able to replace even numbers of the MNIST dataset with zeroes whilst preserving odd numbers when black-listing even numbers from their output.

2.1.4 Mobile sensor data anonymisation

Malekzadeh et al. later proposed a multi-objective *loss function* to be used in deep autoencoder networks using an information-theoretic approach (Malekzadeh et al., 2019). A loss function is the metric that a neural network aims to minimise whilst training, and in doing so learns the desired function that the creator of the network intended for it to learn. Hence, a loss function shapes the entire learning process of a model and should therefore be carefully designed. The loss function Malekzadeh et al. propose has three regularisation parameters to decide a privacy-utility trade-off: a parameter to determine the importance of identity loss (or generalised: the attribute(s) to anonymise), a parameter for the preservation of activity patterns (or generalised: the attribute(s) to keep), and a parameter for the distance function that regulates the distortion between the reconstructed dataset and the original dataset. This loss function was derived from the *mutual information* between the reconstructed dataset and the information of any user in the original dataset. Mutual information is a concept defined in simple terms as the measure of how much information one distribution gives us about the nature of another distribution. Therefore, Malekzadeh et al. aim to train an autoencoder that alters the distribution of the input database in such a way that the resultant reconstructed distribution gives an adversary as little data about individual users as possible. This performs well: A network trained on the original data was unable to determine user identities in a reconstructed dataset produced by the anonymising autoencoder network.

2.1.5 X-vector anonymisation

A technique proposed by Perero-Codosero et al., 2022 uses autoencoders to anonymise speech data. First, an *x-vector* characterising the attributes of the speech is extracted from a speech sample. Then, an autoencoder network trained to generate a latent representation invariant to speaker characteristics is used to remove speaker characteristics from the x-vector, anonymising it. This latent representation is reconstructed back to an x-vector, which is then fed to a speech synthesiser to recreate the now-anonymised speech sample.

Similar works can be found encoding raw data in a latent representation, transforming the latent representation in some way, and decoding it into an anonymised reconstructed data set (Espinoza-Cuadros et al., 2020; Nousi et al., 2020; Saunders et al., 2021; Weggenmann et al., 2022).

2.2 Privacy metrics

In this section, we describe the privacy metrics used in legislation and the literature. After explaining the functionality of an autoencoder in more detail in Section 2.3, we will return to each of these metrics to assess whether the latent representation produced by an autoencoder meets each of these requirements in Chapter 3.

2.2.1 Legislation

The General Data Protection Regulation (GDPR) introduced by the European Commission in 2016 standardises the laws and regulations on data privacy for European countries. GDPR concerns itself with sensitive personal data, which it defines as

> "any information concerning an identified or identifiable living natural person" (taken directly from European Commission, 2016a).

Recital 26 specifies that sufficiently anonymised data falls outside the purview of this regulation. It defines anonymised data as

> "Information which does not relate to an identified or identifiable natural living person, or personal data rendered anonymous in such a manner that the data subject is not or no longer identifiable" (taken directly from European Commission, 2016d).

Additionally, the GDPR defines *pseudonymization* as

> "the processing of personal data in such a manner that the personal data can no longer be attributed to a specific data subject without the use of additional information, provided that such additional information is kept separately and is subject to technical and organisational measures to ensure that the personal data are not attributed to an identified or identifiable natural person" (taken directly from European Commission, 2016a).

A simple example of pseudonymisaton is to replace names of participants in a survey with names of animals, and to keep a table of names that links back 'rabbit' with 'Jane Doe' on a separate, secure location. This table linking true names with pseudonyms is considered to be the 'additional information' as per this regulation stipulation. In contrast, anonymisation would be dropping the names column in its entirety.

Up to 25 May 2018, the Article 29 Working Party (WP) was an independent European working party that dealt with issues related to data protection and provided advice and opinions on various topics. This working party has since been replaced by the European Data Protection Board (EDPB) when GDPR came into full effect. The work of the WP remains archived for future reference. One such advisory, 'Opinion 05/2014 on anonymisation techniques' (Art. 29

WP, 2014), provided a critique on various anonymisation methods and popular privacy models, some of which we will discuss later in this report. These anonymisation methods are evaluated with three metrics:

- Is it still possible to single out an individual in the anonymised dataset?
- Is it still possible to link records relating to an individual between the same or two different databases?
- Can the value of an attribute be inferred from the values of other attributes?

In addition, the WP is of the opinion that pseudonymisation is not a method of anonymisation. Whilst acknowledging the utility of pseudonymisation as a security measure, they claim pseudonymisation only reduces the possibility of record linkage to retrieve the true identity of the subject. Instead, they underline the importance of the irreversibility of an anonymisation method and whether the anonymised data is safe against means that can reasonably be used by the data owner or a third party to deanonymise the data. In other words: "whether identification has become 'reasonably' impossible" (Cited from Art. 29 WP, 2014).

Encryption and hashing

Most crucially for our research, this Working Party lists the acts of encryption and (keyed) hashing as pseudonymisation methods. In Section 3.1, we will use this opinion as a precedent to determine whether autoencoder-based methods would fall under anonymisation or pseudonymisation. We will now briefly explain encryption and hashing to later be able to discuss their similarities with our proposed method.

Encryption is the act of making data unintelligible without access to a private key (Kessler, 2003). The encrypted data is considered to be the pseudonym of the unencrypted data, and the private key is seen as the 'additional information' required to reverse the encrypted data. As such, any party who has access to both the key and encrypted data should be considered to have reasonable access to the full sensitive dataset (Art. 29 WP, 2014)

Hashing is a method that generates a fixed-size output from any input (which could be one or more attributes of the dataset). A stronger requirement is to use a collision-free hash function that (1) is one-way (irreversible) and (2) it is unreasonably difficult to find another input that results in the same output of the hash function (Russell, 1993). The WP opinion considers hashing a pseudonymisation method rather than an anonymisation method, as the possibility of brute-forcing these input values is still theoretically possible, regardless of how negligent the probability may be. This is further corroborated by the existence of 'rainbow table attacks', which use a look-up table of input and output values of a certain hash function (Oechslin, 2003). A 'salted' hash function uses

a (not necessarily secret) random value in addition to the input to further obscure the possible inputs used to generate the hash output (Gauravaram, 2012). A 'keyed' hash function uses a similar concept in that the function accepts an additional (secret) 'key' input that influences the hashed output (Bellare et al., 1996). This key can also be deleted afterwards. Despite these adjustments to increase the strength of hashing functions, all hashing methods are considered a pseudonymisation method by the WP advisory opinion.

Critically, pseudonymised data is not granted the same freedom as anonymised data, as it is still considered personally identifiable information which is thus subject to most GDPR restrictions:

> "Personal data which have undergone pseudonymisation, which could be attributed to a natural person by the use of additional information should be considered to be information on an identifiable natural person." (Taken directly from European Commission, 2016d).

However, article 6(4)(e) allows a data owner to take the pseudonymisation of the data into account when considering the legality of using pseudonymised data for purposes beyond what the data was originally collected for (European Commission, 2016b).

2.2.2 Scientific literature

Majeed and Lee, 2020 created a comprehensive survey on anonymisation techniques for privacy-preserving data publishing. Among other things, they mention metrics used for anonymisation methods for relational data and social network graph data. The scope of this thesis pertains to metrics for relational data anonymisation. These are the following:

- Calculating a measure of anonymous and original data set linkage via quasi-identifiers.
- Evaluation of privacy protection in combination with background knowledge of one or more relevant users.
- Privacy evaluation based on privacy-sensitive rules on one or more data columns.

Hamm, 2017 defines that the privacy of filtered (anonymised) data is measured by the expected risk of success of adversarial algorithms in specific inference tasks such as identification. Mathematically, they formulate this as the expected loss between a guess and the ground truth, similar to *mean squared error* (see Section 2.3). However, this condition, when considered in isolation, can be perfectly fulfilled by an algorithm that produces a random result regardless of input. Therefore, this measure is counterbalanced by minimising the disutility of anonymised data, formulated as the expected squared loss between the anonymised data and the original dataset. In his paper, they use

these combined measures to formulate an information-theoretic approach towards creating a minimax (minimum disutility, maximum privacy) filter for the purpose of data anonymisation.

2.2.3 k-anonymity, l-diversity and t-closeness

We will now discuss three popular privacy models in the current literature. These data anonymisation models are used to ensure k-anonymity, l-diversity, and t-closeness in queries of a relational database.

k-anonymity is the practise of ensuring that when looking up a certain value in a database, at least $k-1$ other data points could be confused for any given data point returned as a result of that query. Groups of records that share the same value for a given feature are called *equivalence classes*. An example of an equivalence class is the set of records that share the common value of 'age' within the range of $(25, 35)$. However, recent literature has shown attacks on this metric of data anonymisation. k-anonymity is weak against an attacker who has background information on a person they are querying about, or if the k-anonymous data is too homogeneous (Domingo-Ferrer and Torra, 2008; Machanavajjhala et al., 2007): If all records within the age range $(25, 35)$ also share the same medical condition, then an adversary who knows that (1) data of a friend of theirs is somewhere in this database and (2) this friend of theirs is within the age range of $(25, 35)$, they would glean that their friend suffers from that medical condition if they were to query this k-anonymous database. This weakness led to the development of l-diversity that is used in conjunction with k-anonymity.

l-diversity is a metric that ensures that there are at least l 'well-represented' possible values in the column(s) holding sensitive data when looking up any equivalence class. Machanavajjhala et al., 2007 propose three definitions of 'well-represented':

- Distinct l-diversity: There exist l distinct sensitive values per equivalence class. That is, l distinct medical conditions per age range.
- Entropy l-diversity: *Entropy* is a mathematical measure of surprise of a random value. If a random variable X generally hovers around a certain value, then a sample from X does not give us much information. It would be more surprising or informative if X had a wider range of possibilities. Mathematically, entropy is defined as $H(X) = -\sum_{i=1}^{n} p_i \log p_i$. In the case of entropy l-diversity, the random variable is the equivalence class, and a sample is one of these records belonging to the equivalence class. To continue with the example, the probability of a certain medical condition occurring within the age range equivalency class is how often that medical condition occurs within the equivalency class divided by the total amount of records of that equivalency class. The entropy of the age-range equivalency class increases if more distinct medical conditions occur. This fits perfectly in the function for entropy; thus a database is defined to be

entropy l-diverse if for every equivalence class, $H(X) \geq log(l)$. In other words, there are a certain number of distinct medical conditions with such a distribution that sampling from the equivalence class of age range is at least as surprising as $log(l)$.

- Recursive $(c - l)$-diversity. In short, ensuring recursive $(c - l)$-diversity means ensuring that the most frequently occurring medical condition appears within proportion of other medical conditions, bounded by a constant c.

However, l-diversity has been shown to have weaknesses depending on the distribution of the l-diversified data, or if the l-classes are distinct but insufficiently semantically different, or if different values of an l-diverse column are values of varying sensitivity (Domingo-Ferrer and Torra, 2008; Li et al., 2006): if the distinct l-diverse equivalence class has l distinct instances of stomach illnesses, then the adversary knows that his friend suffers from a stomach illness.

t-closeness obligates the data controller to ensure that the l well-represented sensitive values within the equivalence classes reflect the distribution of the entire data set when querying the data. However, the paper that proposes t-closeness as a measure to further improve anonymisation (Li et al., 2006) fails to provide a computational procedure to enforce t-closeness, and enforcing t-closeness strongly restricts data utility (Domingo-Ferrer and Torra, 2008).

2.3 Background of autoencoders

So far in this chapter, we have described methods that use various autoencoder networks to perform data anonymisation tasks. We also discussed privacy metrics used in the literature and in the legislation in the previous two sections. In this section, we establish the background concepts and mathematical theory behind autoencoders. In Chapter 3, we use these mathematical concepts to evaluate whether autoencoders and latent representations satisfy the privacy metrics we have discussed.

Standard autoencoders consist of two networks, an encoder and a decoder network. Both networks are simple, fully connected multilayer perceptrons. The size of the input layer of the encoder network is equal to or greater than the size of the input data, and each successive layer shrinks in dimensionality until the desired 'bottleneck' size is achieved. There is no standard rate or ratio at which encoder networks shrink in dimensionality, although a popular approach is to halve the size of each layer. There is also no standard for how many layers these networks should consist of, although deeper networks have been found to learn more complicated functions than shallower networks (Rumelhart et al., 1985). The output of the encoder bottleneck layer is called a latent representation. The bottleneck layer of the encoder network should not exceed the dimensionality of the input data. If that is the case, it is possible for the encoder to learn the identity function and achieve a loss of 0, but ultimately become useless for

dimensionality reduction, anonymisation or pseudonymisation.

To aid in our explanation we will define a few variables and functions.

- Let $\mathbf{x}$ be an input vector of features $\mathbf{x} = \{x_1, x_2, ..., x_d\} \in \mathbb{R}^d$,
- $\mathbf{y}$ an output vector of a layer $\mathbf{y} = \{y_1, y_2, ..., y_{d'}\} \in \mathbb{R}^{d'}$,
- $\theta = \{\mathbf{W}, \mathbf{b}\}$ a set of parameters consisting of a weight matrix $\mathbf{W} \in \mathbb{R}^{d \times d'}$ and bias vector $\mathbf{b} \in \mathbb{R}^d$,
- and finally ϕ an activation function.

A popular activation function is the Rectified Linear Unit (ReLU) function, mathematically defined as $ReLU(x) = max(0, x)$, or alternatively:

$$ReLU(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

ReLU is a simple-to-calculate activation function and gives the network a wide range of numbers to represent the importance of features with. This is in contrast to the functions sigmoid ($\delta(x) = \frac{1}{1+e^{-x}}$) and tanh ($\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$) that bound the output to $(0, 1)$ and $(-1, 1)$, respectively. See Figure 2.1 for a plot of each of these activation functions. Features with high importance saturate values approaching 1, and unimportant features saturate 0 or -1. Hence, the sigmoid and tanh activation functions are also referred to as saturating activation functions, whilst ReLU belongs to the class of non-saturating activation functions.

Saturating activation functions pose a problem when it comes to updating weights in deep networks during backpropagation. Backpropagation multiplies the derivative of these weights in every layer, which, for saturating activation functions, is bound between $[-2, 2] \in \mathbb{R}$. Backpropagation works its way from the output layer back toward the input layer (hence the name). Successively multiplying a small number (the result of the partial derivatives of the activation function) by a small number each layer means that there is little left to propagate by the time the process has arrived at the first (input) layer. This phenomenon is referred to as the *vanishing gradient problem* (Hochreiter, 1998), and is mitigated by using a non-saturating activation function. This, along with its computational simplicity, led ReLU to become a popular activation function.

The output of each layer i is defined as $\mathbf{y}_i = f_{\theta_i}(\mathbf{x}) = \phi(\mathbf{W}_i^T \mathbf{x}_i + \mathbf{b}_i)$. The output of the bottleneck layer is defined as the latent representation of the autoencoder, which we will name $\mathbf{z}$. Written out for an encoder network consisting of three layers, the latent representation is calculated by:

$$\mathbf{z} = f_{\theta_3}(f_{\theta_2}(f_{\theta_1}(\mathbf{x}))) = \phi(\mathbf{W}_3^T \phi(\mathbf{W}_2^T \phi(\mathbf{W}_1^T \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2) + \mathbf{b}_3)$$

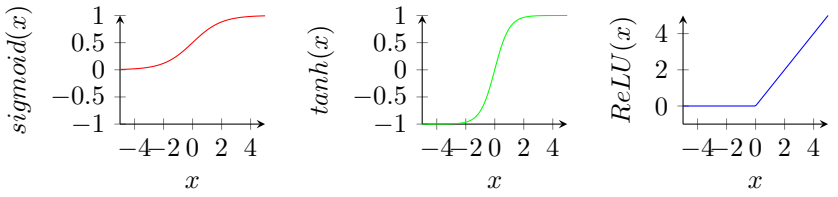

Figure 2.1: Plots of activation functions sigmoid ($\delta(x) = \frac{1}{1+e^{-x}}$) in red, tanh ($\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$) in green, and ReLU ($ReLU(x) = max(0, x)$) in blue.

Hereinafter, to be able to make a distinction between an encoder network and a decoder network of an autoencoder, we will refer to an encoder network and its corresponding parameter set as $f_\theta(\cdot)$, and a decoder network and its corresponding parameter set $f'_{\theta'}(\cdot)$.

2.3.1 Loss and mean squared error

During training, the latent representation is fed to the decoder layer as input. The decoder network expands the latent representation back to the dimensionality of the input to reconstruct the dataset. Usually, the decoder network mirrors the architecture of the encoder network. Decoding the latent representation results in the reconstruction $\mathbf{x}'$. To measure the performance of the autoencoder, a simple and popular distance metric called the *mean squared error* (MSE) is used as loss function:

$$MSE(\mathbf{x}, \mathbf{x}') = \frac{1}{n}\sum_{i=1}^{n}(x_i - x'_i)^2$$

MSE is popular for pragmatic reasons. First, calculating the deviation of one vector from another by subtraction is intuitive. Then, squaring this deviation ensures that the error is always a positive number, and punishes greater deviations more than smaller deviatons. Finally, taking the mean regulates the loss numbers to not become excessively large, and spreads the error over all tested datapoints to ensure that the network doesn't overfit on outliers.

When comparing the MSE of a batch of multiple vectors, the MSE is either summed, or summed and divided over the total amount of samples in the batch. The latter method makes comparisons between two test runs with different batch sizes possible as the loss is normalised over batch size. Notice that this function measures the loss of the performance of both the encoder and decoder network, with the encoder's responsibility to create a salient latent representation, and the decoder's responsibility to relate this representation back to the input.

2.3.2 Weight initialisation

Each neuron in a neural network has weights that assign importance to incoming signals. In a traditional supervised learning network using an activation function with output $y \in \{0, 1\}$, the function of these weights are more obvious; if the sum of the incoming signal multiplied by their respective weights exceeds a threshold, output 'yes', or 1, otherwise 'no', or 0. In an unsupervised use-case such as clustering or learning a latent representation, the simple semantic meaning of weights in a network is not as obvious, but the importance of learning correct weights remains all the same, as learning the correct weights will make our autoencoder function the way we desire it to.

The learning process of a neural network is finding a set of parameters $\theta = \{\mathbf{W}, \mathbf{b}\}$ that achieves our goal of learning to turn an input into a salient latent representation. These weights need a good starting point. The scientific literature provides us with many ways to initialise weights, with some methods working better with saturating activation functions such as the sigmoid and tanh activation functions, and other initialisation methods working better with non-saturating activation functions such as ReLU (Kumar, 2017). One such initialisation method shown to work well with ReLU is the initialisation method proposed by He et al., 2015, which is sampling weights from a zero-centred Gaussian distribution with a standard deviation engineered to prevent magnifying signals exponentially, since ReLU does not provide an upper bound for signals.

2.3.3 Stochastic gradient descent

These initialised weights can now be iteratively improved by evaluating their performance with MSE as mentioned above. During learning, we aim to minimise the loss function. The first derivative of the loss function denotes its slope. As we aim to minimise this function, we move down the slope to reduce the loss in the next time step. The gradient of MSE is as follows:

$$\nabla MSE(\mathbf{x}, \mathbf{x}') = \frac{\delta MSE}{\delta \mathbf{x}'} \frac{1}{n} \sum_{i=1}^{n} (x_i - x_i')^2 = \frac{2}{n} \sum_{i=1}^{n} (x_i - x_i')$$

Here, we observe that squaring the error serves one final function: the derivative is simple to calculate. This simplicity is beneficial when it comes to the training time of the network, given how often we have to calculate the error.

Given time step t, we can generate the parameters to be used in the next time step θ_{t+1} given learning rate η by

$$\theta_{t+1} = \theta_t - \eta \nabla MSE(\mathbf{x}, \mathbf{x}')$$

The minus in the equation is to minimise the slope of the gradient. As mentioned above, the output of a neural network is the result of the calculations

of successive layers of weights and activation functions. As such, the weights of each of these layers need to be updated accordingly. To do so, the error gradient is fed back to the network starting from the output layer. Due to the interdependent nature of these layers, the error is sent through the network on a layer-by-layer basis by a process called *backpropagation*. Backpropagation uses the chain rule of partial derivatives to resolve these dependencies: For a function p which depends on q which in turn depends on r, the derivative of p with respect to r is:

$$\frac{\delta p}{\delta r} = \frac{\delta p}{\delta q}\frac{\delta q}{\delta r}$$

Recall that the output of each layer i is defined as $\mathbf{y}_i = f_{\theta_i}(\mathbf{x}) = \phi(\mathbf{W}_i^T\mathbf{x}_i + \mathbf{b}_i)$. In a network consisting of three layers, the backpropagated error to update the weights of the initial layer (θ_1) is calculated as follows:

$$\frac{\delta MSE(\mathbf{x}, \mathbf{x}')}{\delta \theta_1} = \frac{\delta MSE(\mathbf{x}, \mathbf{x}')}{\delta \mathbf{x}'}\frac{\delta \mathbf{x}'}{\delta \mathbf{y}_3}\frac{\delta \mathbf{y}_3}{\delta \mathbf{y}_2}\frac{\delta \mathbf{y}_2}{\delta \mathbf{y}_1}\frac{\delta \mathbf{y}_1}{\delta \theta_1}$$

By approaching this calculation on a layer-by-layer basis, the answers of the partial derivatives of the previous layers can be re-used for a simpler computation.

As the volume of training data increases, the number of pointwise comparisons of the reconstructed and true dataset increases along with it. Computing the gradient descent of the full dataset is a costly operation. To combat this, the process is modified to *stochastic gradient descent* (SGD): calculating the gradient descent of one or a few datapoints at a time (Saad, 1999). This increases the amount of total time-steps needed in order to converge to a minimal loss, but each individual time-step is computationally less intense. In addition to computational benefits, this approach has another benefit: Despite the simplicity of MSE, we cannot assume that the gradient contains only one minimal point. Stochastic gradient descent optimises towards the nearest minimising slope. This nearest slope could be what is called a *local optimum*; whilst that set of parameters results in less loss than other 'topographically close' parameter sets, this local optimum may not be the lowest the loss could globally be. The lowest overall loss is called the *global optimum* (or in the context of a value to minimise, the global minimum). Stochastic gradient descent introduces a measure of randomness in the 'walk' on the gradient of the loss function. If the best-performing set of parameters is stored, then there is a much higher chance that this global optimum is achieved, as the randomness allows the SGD method to 'escape' a local optimum to explore other optima instead of remaining stuck in the first optimum it finds. Refer to Figure 2.2 for an illuminating illustration of SGD.

The last variable used in the formula of SGD is η, or *learning rate*. This is the rate at which the weights adjust on the basis of the calculated gradient. If SGD approaches an optimum, an overly large learning rate might 'overstep' and miss

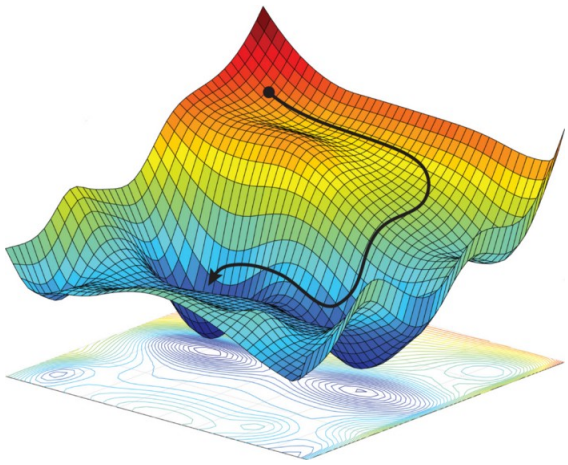

Figure 2.2: Illustration of stochastic gradient descent (SGD) with multiple local optima and one global optimum (Amini et al., 2019).

this optimum. Therefore, it is favourable to dynamically adjust this learning rate to the proximity of an optimum.

One popular method to adjust learning rates is called Adam, derived from adaptive moment estimation (Kingma and Ba, 2017). Briefly summarised, rather than backpropagating the loss gradient directly, it estimates (and over time, decays) the first ($E[g]$) and second moment ($E[g^2]$) of the gradient $\nabla MSE(\mathbf{x}, \mathbf{x}')$. This experimentally leads to a faster convergence rate than standard SGD whilst remaining computationally light. For further details, we refer the reader to its seminal paper, but for this thesis, it suffices to know that it is a more performant variant of SGD to update the parameters θ of the model.

In summary, an autoencoder is an encoder-decoder pair network consisting of layers of perceptrons. Each of these perceptrons has corresponding weights for each input and a bias accompanying each layer. These weights and biases are the parameters $\theta = \{\mathbf{W}, \mathbf{b}\}$ that the network trains to optimise over time. It does so by minimising the loss between the reconstructed dataset $\mathbf{x}'$ and its ground truth $\mathbf{x}$ with the mean squared error as the loss function $MSE(\mathbf{x}, \mathbf{x}')$. To optimise the parameters of the network θ, it can use a stochastic gradient descent variant called Adam. When training is complete, the result is an encoder network that can transform an input dataset $\mathbf{x}$ into a latent representation $\mathbf{z}$ and a decoder network that can reconstruct a latent representation $\mathbf{z}$ into a reconstruction $\mathbf{x}'$.

2.3.4 Latent representations

In this section, we will discuss some properties of, and assumptions about latent representations and provide an example. As mentioned above, the latent

representation is the output of the bottleneck layer of an autoencoder network. Usually, this bottleneck layer is of a smaller dimensionality than the input data to force the network to learn which features are most important. Unlike most neural network problems, an autoencoder network trained for our purposes does not need to generalise on unseen data: it can be trained on the full input dataset to create the most informed latent representation possible. This is under the assumption that we are dealing with an offline data pseudonymisation problem, where the whole dataset is available to us beforehand and that no new datapoints need to be pseudonymised later.

For any latent representation generated by a reasonably well-trained[1] encoder network, there exists a corresponding decoder network that creates the best possible approximation of the original dataset $f'_{\theta'}(\mathbf{z}) = \mathbf{x}'$. This is the decoder network that is trained in conjunction with the encoder network. For the purposes of data pseudonymisation, this decoder network has the greatest chance of re-identifying records in the latent representation and therefore forms the baseline worst-case scenario for the data controller, or the best-case scenario for an adversarial party.

In addition, there exists a range of parameter sets that are not the best decoder network, but still decently reconstruct the original dataset. The measure of which these parameter sets diverge from the original data set $\mathbf{x}$ or the best reconstruction $\mathbf{x}'$ of the dataset can also be measured by comparing their resultant reconstructions with mean squared error or other similar loss functions.

We also assume that due to the highly stochastic nature of weight initialisation and gradient descent, there exist multiple well-trained latent representations for any given dataset, each with its own corresponding best-performing decoder. Similarly, due to the nature of representing higher-dimensional data in a lower dimensionality, we postulate that it is possible for a latent representation to represent multiple datasets at the same time. An encoder is a function $f : \mathbb{R}^d \to \mathbb{R}^b_+$ where, in general, $b < d$. Similarly, a decoder is a function $f' : \mathbb{R}^b_+ \to \mathbb{R}^d$. As an encoder is a function that maps a larger infinite space to a smaller infinite space and multiple encoders can exist as per our earlier assumption, we conclude that it is possible for any given latent representation to represent multiple datasets at once. The chances that a collision occurs in the latent space are extremely low, but theoretically possible. Regardless, the consequence of these two assumptions is that the search space for an adversary for any given latent representation is very large. Without a priori knowledge of what kind of data is encoded into a latent representation, we assume that the search for a decent reconstruction is a difficult task.

We experimentally evaluate these assumptions and claims in Chapter 4. Next, we discuss an example of a latent representation.

[1] Poorly trained autoencoders trivially produce poorly representing latent representations or poorly performing decoder networks and thus need not be considered. We can safely assume that a party interested in pseudonymising data with an autoencoder would have a vested interest in training a reasonably performing autoencoder for the task.

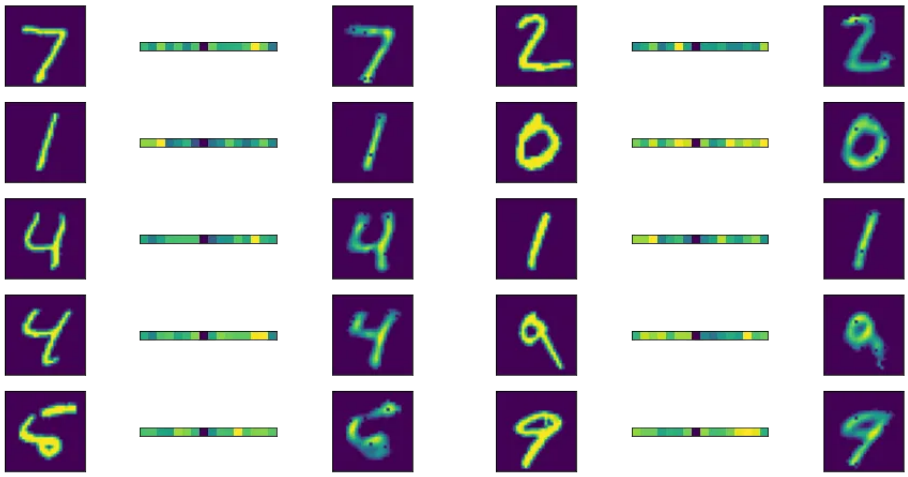

Figure 2.3: A visualisation of MNIST digits, their corresponding latent representations and their reconstructions (Yang, 2020).

Figure 2.3 shows an input data point $\mathbf{x}$, the latent representation $\mathbf{z}$, and the corresponding decoded reconstruction of a handwritten number $\mathbf{x}'$ of an autoencoder trained on the MNIST handwritten number dataset (Deng, 2012). The MNIST dataset represents 28x28-pixel images of handwritten digits as $28 * 28 = 784$ features of greyscale values ranging between $(0, 1) \in \mathbb{R}$. In his blog post, Yang, 2020 transforms the input data into a 16-dimensional latent representation and reconstructs this latent representation back to the original dimensionality. As visible in the figure, the latent representation and the corresponding trained decoder manage to capture the features of the digits efficiently. Following his argument on latent representation anonymity, any human exclusively reading the latent representation out of context would not be able to discern which values correspond to the greyscale value of which pixel. However, the latent representations of numbers written in the same way share a lot of similarities in their latent representations, therefore, these relations evidently still exist for the decoder network which was trained along with the encoder network.

Chapter 3

Theoretical investigation

3.1 Theoretical investigation

In this chapter, we discuss which privacy metrics in the literature and the legislature listed in the previous chapter are relevant, allowing us to answer **SQ1**: What privacy standards, models, and metrics exist in the current literature or legislation and which of these measures apply to latent representations?

3.1.1 Legislation

First, we discuss existing legislation on data anonymisation.

As mentioned in Section 2.2, hashing and encryption-based data transformation methods are considered pseudonymization methods rather than anonymisation methods. We will use this ruling to determine what the legislation may reasonably assert our autoencoder-based method may fall under. A characteristic of pseudonymisation methods is that they retain a method by which the pseudonymised data can be returned to the original format. True anonymisation methods must be irreversible. As we have described in Section 2.3, a decoder network is trained together with an encoder network to evaluate the performance of both the encoder and decoder networks and to propagate a loss. Similarly to the theoretical possibility of reversing an encryption or hash, regardless of the negligible probability of doing so, it is theoretically possible to determine this decoder network by brute force to at least be able to retrieve the best possible reconstruction of the input $\mathbf{x}'$. As such, this decoder network could be considered the 'additional information' in the same sense that the key of an encryption method or input of a hash function are considered the 'additional information', making our autoencoder-based method most likely to be considered pseudonymisation within the purview of GDPR. As such, we refer to this autoencoder-based method as a pseudonymisation method in this

work. Note that this conclusion is merely an extrapolation based on existing regulatory considerations and, as such, is not intended to be actual legal advice.

To evaluate privacy metrics in both the European legislature and scientific literature, we define a *semantic deanonymisation method*. By a 'semantic' method, we refer to the act of comparing known values or their semantic meaning to anonymised or pseudonymised data. In our case, the pseudonymised data is the latent representation of the original dataset. Crucially, in a latent representation, the semantic meaning of any given value is obscured or lost entirely. To be able to discern what the original values were, one would have to either decode the latent representation to a reconstruction or input known values into the encoder to check for a match in the latent representation. Without access to the original autoencoder network or an adversarial approximation thereof, these semantic deanonymisation methods fail on latent representations.

The Working Party uses three data anonymisation metrics. The first is whether it is possible to single out an individual in the anonymised dataset. The second is whether it is still possible to employ record linkage between the same or two different datasets. The third is whether it is possible to infer values of attributes on the basis of other attributes still present in the data. All of these metrics are what we would consider semantic deanonymisation methods. For the first method, as the semantic meaning of pseudonymised values is obscured, it is unlikely that even a well-informed third party would be able to single out any individual in the latent representation. For the second method, assuming that the two data releases are encoded by two different encoders, it is similarly unlikely that any record linkage could successfully occur (see our results of latent representation similarity in Subsection 4.2.5). For the third and final metric, we have seen that the latent representation saliently encodes the input data in such a manner that it preserves data utility for logistic regression and decision trees (Yang, 2020) to the extent that autoencoders can be used as a feature selection method. Therefore, it is possible to infer the value of one of the attributes of a latent representation on the basis of the values of other attributes of the latent representation. However, much like the argument for the first two metrics, the semantic meaning of the inferred value is obscured.

Instead, we argue that for latent representations, the pseudonymisation guarantee depends on whether it is possible for an adversarial party to train an adversarial approximation of the encoder or decoder networks used to generate the latent representation in question. As such, we extensively research this in our experiments, specifically focussing on whether it is possible to train an adversarial decoder network to reconstruct a latent representation into a semantically meaningful format.

Finally, even if our autoencoder-based method were to be vulnerable to one of these metrics, the opinion expects the data controller to be cognisant of the risks and trade-offs of their chosen methods and to use multiple anonymisation methods if possible to ensure a higher standard of anonymisation.

3.1.2 Scientific literature

Next, we will discuss privacy metrics found in the literature on privacy-preserving data publishing and data anonymisation algorithms. Majeed and Lee, 2020 evaluate non-graph-based anonymisation techniques by three metrics. Their first metric is to calculate a measure of anonymous and original data set linkage on the basis of the quasi-identifiers that remain in the anonymised the dataset. This is yet another semantic approach towards deanonymisation. Once again, the efficacy reduces to the adversary's ability to decode the latent representation, which is experimentally tested in Chapter 4.

The second metric of Majeed et al., evaluating privacy protection in combination with perfect background knowledge on one or more relevant users in the database, similarly fails due to the aforementioned reasons of features being encoded. The same applies to the evaluation based on privacy-sensitive rules on one or more data columns. This is the process of analysing an anonymised dataset with rules such as 'What are the medical conditions of all patients within the age range $(25, 35)$?' similar to what we discussed in the section on k-anonymity. Effectively, latent representations are inherently resistant against privacy attacks based on semantic approaches, and their usage reduces to the measure of how well an adversary can find a well-performing set of decoder network parameters. The privacy models k-anonymity, l-diversity and t-closeness similarly do not apply to latent representations.

Hamm, 2017 proposes a minimax filter to minimise data disutility whilst maximising data anonymity. In his approach, they formulate it as minimising the expected loss between the anonymised data and ground truth, whilst maximising the expected loss between an adversary's guess what the original dataset may be based on the anonymised data and the ground truth. This measure is difficult to incorporate in our application. We minimise the disutility of the data by minimising the mean squared error of the decoder reproduction $\mathbf{x}'$ compared to the original dataset $\mathbf{x}$: $\min MSE(\mathbf{x}, \mathbf{x}')$. We define our pseudonymised dataset as our latent representation. This means that to maximise the privacy of the data, we would have to maximise the loss between the performance of a decoder reconstruction and the ground truth: $\max MSE(\mathbf{x}, \mathbf{x}')$. To apply this measure of privacy, we would have to minimise the performance of adversarial decoders whilst preserving the performance of the decoder that is trained together with the encoder. However, we conclude that the originally trained decoder and adversarial decoder are too similar to fruitfully utilise this concept.

In Section 2.1 we discussed various related works using autoencoders to anonymise data. These works discussed various implementations of encoding input data into a latent representation, transforming the latent representation in some way, and reconstructing the latent representation to the original dimensionality. These articles measure the success of their anonymisation methods by the difference in the accuracy of the inference of the private value in anonymised data by a network trained to learn the private value on raw data, whilst maintaining

the performance of an inference model of a supplementary task to illuminate the retention of the utility of anonymised data (Hajihassani et al., 2021; Malekzadeh et al., 2019; Malekzadeh et al., 2017; Perero-Codosero et al., 2022). Our method does not reconstruct the latent representation back to its original dimensionality. Although it is possible to learn to infer a value of the latent representation, this inferred value has lost its semantic meaning. Therefore, whilst this metric is technically possible to measure, it is not meaningful.

3.1.3 Conclusion

Section 2.2 and the latter half of this chapter was dedicated to answering our first sub-question **SQ1**: What privacy standards, models, and metrics exist in the current literature or legislation and which of these measures apply to latent representations? Accordingly, we discussed various standards used in legislation and in the literature of privacy-preserving data publishing, algorithmic and machine learning-based data anonymisation methods.

Most importantly, based on the precedent that encryption and hashing methods are considered pseudonymisation methods, our autoencoder-based method may legally be classified as a pseudonymisation method under the purview of GDPR, rather than an anonymisation method. As such, we refer to this autoencoder-based method as a pseudonymisation method in this work. However, article 6(4)(e) allows pseudonymised data more freedoms than nonpseudonymised data for purposes beyond what the data was originally collected for (European Commission, 2016b). Please be advised that this conclusion is merely an extrapolation based on existing regulatory considerations and, as such, is not intended to be actual legal advice.

We conclude that most of these measures fail to apply to the latent representation of our autoencoder-based method due to the violation of the assumption that the pseudonymised data is stored in a format that preserves the semantic meaning of the data. Moreover, the pseudonymisation guarantee of our method depends on the ability of an adversary to successfully find a decoder network to return a latent representation into a semantically meaningful reconstruction of the input data. As such, we will experimentally evaluate the possibility in Chapter 4.

Chapter 4

Experimental investigation

4.1 Experimental setup

In Chapter 3, we concluded that the privacy guarantees provided by using an autoencoder-based method depend on the ability of an adversary to train a decoder to reconstruct any given latent representation into the original dimensionality of the dataset. Most currently existing privacy metrics use semantic approaches to evaluate privacy, which do not apply to our latent representations. In this chapter, we evaluate how sensitive latent representations are to decoding by experimentally varying parameters surrounding the dataset, encoder, and adversarial decoder network. In addition, we measure the extent of utility loss as a result of using latent representations as a training and test dataset, rather than using the original dataset. In doing so, our aim is to answer our two remaining sub-questions **SQ2**: What experimental measures of pseudonymity does an autoencoder's latent representation have? And **SQ3**: What is the utility loss incurred when using a latent representation of the data rather than the original data?

4.1.1 Setup

Primarily, our experiments revolve around varying properties of our data and hyperparameters of our encoder and adversarial decoder network. To do so in a structured manner, we use the MLflow Python library to collect and track hyperparameters and metrics of our experiments. The plots are generated using MatPlotLib (Hunter, 2007).

To test our experiments, our networks are trained on an Intel i7-12800H CPU with 32GB of RAM running Windows 11 with Python 3.10.10.

In our experiments, we train a relatively simple 5-layer autoencoder network, or 3-layer encoder and 3-layer decoder network, unless mentioned otherwise.

This corresponds with the diagram as seen in Figure 1.1. The dimensionality of the input layer (and output layer) is equal to the total number of features in the dataset. The dimensionality of the hidden layer is 75% of the number of input features. The dimensionality of the bottleneck layer is 50% of the number of input features. Hence, these dimensionalities and corresponding network architectures are automatically adjusted on the basis of the input dataset.

4.1.2 Measures

Data protection measures

To be able to quantify the level of protection our autoencoder-based method grants us, we measure the ability of an adversarial decoder to approximate our original dataset. Recall that $f(\cdot)$ denotes the encoder network and $f'(\cdot)$ the decoder network that is trained in conjunction with the encoder network. Let $\mathbf{x}$ be our input data set and $f(\mathbf{x}) = \mathbf{z}$ our latent representation, and let $f'(\mathbf{z}) = \mathbf{x}'$ be the best possible reconstruction in the original dimensionality. We measure the Baseline Loss (BL) for a frame of reference as:

$$BL = MSE(\mathbf{x}', \mathbf{x})$$

Now, let $g(\cdot)$ be the adversarial decoder that attempts to reconstruct a latent representation. We measure the mean squared error between the adversarial approximation and the true dataset as True Loss (TL):

$$TL = MSE(g(\mathbf{z}), \mathbf{x})$$

In addition, we measure the mean squared error between the adversarial approximation and the best possible reconstruction made by the decoder created during training as Approximate Loss (AL):

$$AL = MSE(g(\mathbf{z}), \mathbf{x}')$$

Naturally, approximating the true dataset $\mathbf{x}$ is the objective of the adversary. Realistically, the best possible reconstruction $\mathbf{x}'$ is also the best case the adversary could expect, and thus exists as the baseline, 'worst-case' scenario of our latent representation reidentification. For the TL and AL measures, a higher loss corresponds to a worse reconstruction $g(\mathbf{z})$ generated by the adversary and thus a higher level of protection granted by our pseudonymisation method.

Finally, we define a heterogeneity measure (HM) of a reconstructed dataset $g(\mathbf{z})$. This is to measure how well an adversarial decoder has achieved the ability to distinguish between encoded datapoints. This is calculated as the mean of the mean squared errors between each distinct datapoint in the reconstructed dataset $g(\mathbf{z})$. This measure is denoted mathematically as:

$$HM = \frac{1}{|g(\mathbf{z})|} \sum_{i,j=1,\ i \neq j}^{|g(\mathbf{z})|} MSE(g(\mathbf{z})_i, g(\mathbf{z})_j)$$

Each of these values only has a relative meaning. The absolute value of mean squared error loss and measurement of heterogeneity depend greatly on the mean, standard deviation, and distribution of data. However, they are meaningful in context: A decoder would have to achieve both a high (or equal to the original dataset's HM) measure of heterogeneity and a low amount of loss in order to be considered well-performing. It is possible to achieve a high measure of heterogeneity by randomly generating large numbers, but the loss would be very large because these numbers would not at all correspond to the original dataset. Similarly, loss could be minimised by always guessing a median datapoint, but this would substantially lower the heterogeneity measure, as none of the datapoints would be very distinct from one another.

Utility loss measure

We utilise two sets of measures to evaluate the utility difference between using the original dataset and a latent representation of the dataset. Classification models are evaluated by their balanced accuracy, weighted F1 scores, the Area Under the Receiver Operating Characteristic Curve (ROC AUC or AUROC), and their average precision. We measure the fit of the regression models by their r^2 scores and the MSE between the predicted outcome and the ground truth. For multiclass classification problems, we calculate the unweighted arithmetic mean over each class, also known as macro averaging. To contextualise these results, all classification metrics have values ranging from $[0, 1] \in \mathbb{R}$. For regression models, r^2 is generally a value within that range as well, but can fall below 0 if the fit of the model is worse than always predicting the median value of the result.

4.1.3 Data

As mentioned in the introduction, we assume that we work with numerical machine learning training data or data that otherwise has a distribution that our autoencoder can learn. This is due to our assumption that the data with which we work is intended to be pseudonymously analysed by classification or regression algorithms and to simplify the use of autoencoders. Experimenting with unstructured data, such as textual data, falls outside the scope of this thesis and is suggested as future work.

We conduct our experiments on both synthetic data generated by the Python package Sckikit-Learn and public domain datasets sourced from Kaggle. The synthetic dataset is generated by the `make_classification` function featured in Scikit-Learn (Pedregosa et al., 2011; Scikit-learn, 2011). By default, the generated synthetic dataset consists of 1000 generated records of 21 attributes, where each attribute has an average of around 0 and a standard deviation of around 1. The `scale` parameter of this function allows us to manipulate these standard deviations by multiplying all values by a scalar. We explore the effect of this in our first experiment.

The public domain datasets on which we experiment are the MNIST handwritten digit dataset (Deng, 2012), the Pima Indian Diabetes dataset (Smith et al., 1988) and the California Housing Prices dataset (Pace and Barry, 1997). The exact contents and relations within these datasets are largely irrelevant; we are primarily interested in whether our proposed methods behave consistently over vastly different datasets. Regardless, to contextualise our results, we will briefly describe the datasets.

MNIST

As mentioned in Section 2.3 and seen in Figure 2.3, the MNIST dataset consists of 70000 records of 784 greyscale pixel features ranging from $[0, 1] \in \mathbb{R}$ depicting handwritten digits. This dataset is commonly used to train a network to classify which digit is written on the basis of which pixels have ink and which do not. In our data pseudonymising context, our aim would be to make it impossible for a network to discern which latent representation record corresponds to which original digit.

Pima Indians Diabetes

The Pima Indians Diabetes dataset consists of 768 records of nine various medical characteristics of Pima Indian women. Traditionally, this dataset is used as a benchmarking dataset for supervised classification algorithms to predict which medical characteristics correlate with diabetes. Our goal is to pseudonymise which latent record corresponds to which original record. Unlike the MNIST data and the synthetic datasets, each attribute has a larger difference in terms of average and standard deviation, which is more realistic for real-world datasets.

California Housing Prices

The California Housing Prices dataset consists of 20433 records of nine numerical attributes and one categorical attribute describing various characteristics of houses in California gathered from a 1990 census. Commonly, the objective is to predict the price of a given house based on its features. Our objective is to securely pseudonymise which encoded record maps to which house in the original dataset.

For the sake of our experiments, we dropped the categorical attribute and discarded records containing `NaN` in any attribute. Yang, 2020 shows that categorical attributes can be encoded by means of one-hot vector encoding should it prove necessary. However, we decided to not carry that out in our experiments for the sake of simplicity. Our use of this dataset is to observe how well our autoencoder method fares with vastly varying averages and standard deviations between each column, including the average of the longitude column being a negative number. This is to further explore how robust our proposed methods are against various different datasets.

Table 4.1: Two tables describing our experiments in short.

Nr.	Experiment	Goal	Adversary has access to
1a	Data Leakage	Decode remainder of non-leaked data	Full latent representation, p% true dataset
1b	Data Leakage (permuted)	Decode remainder of non-leaked data	Permuted latent representation, p% true dataset
1c	Data Leakage (reconstructed)	Decode remainder of non-leaked data	Full latent representation, p% reconstructed dataset
2	Cluster Matching	Undo latent representation permutation	Permuted latent representation, p% true dataset

Nr.	Experiment	Goal
3	Data Utility	Measure utility loss of latent representations
4	Misalignment of MSE	Investigate misalignment of using MSE as autoencoder loss function for the preservation of data utility
5	Measuring latent rep. similarity	Analyse if different autoencoders produce similar latent representations for the same dataset
6	Runtime Experiments	Evaluate the influence of the data shape on autoencoder training time

4.1.4 Experimental procedure

This section describes the theory, intent, and setup of a variety of experiments to verify properties of latent representations in the context of both data pseudonymisation and utility for machine learning applications. These experiments are individually elaborated in further detail below, but first, we give a global summary.

Primarily, we are interested in the pseudonymisation capabilities of a latent representation. We concluded in Chapter 3 that existing anonymisation metrics in the literature do not apply to latent representations. To enable us to evaluate the pseudonymity granted by any given latent representation, we defined measures of reidentification in Subsection 4.1.2 achieved by an adversarial party. Experiments 1(a-c) and 2 listed in Table 4.1.4 cover this goal and are relevant to our second research sub-question, **SQ2**: What experimental measures of pseudonymity does an autoencoder's latent representation have?

Secondarily, we are interested in measuring the data utility penalty incurred when using a latent representation instead of the original data to train a supervised machine learning algorithm. We test the performance of the original data of a variety of classification and regression datasets as a baseline. Afterwards, we compare the performance of models trained on latent representations. Experiments 3 and 4 in the experiment table investigate this goal are relevant to our third research sub-question, **SQ3**: What is the utility loss incurred when using a latent representation of the data rather than the original data?

Finally, we conduct smaller experiments (number 5 and 6) to discover various properties of latent representations to provide a well-rounded answer for our research question **RQ**: What are the advantages and disadvantages of using an autoencoder's latent representation as a utility-preserving data pseudonymisation method for machine learning?

1. Varying levels of data leakage

Idea. In the literature, the authors of ObscureNet (Hajihassani et al., 2021) described how their deterministic anonymisation method is rendered ineffective if 20% of the original dataset leaks along with the anonymised dataset. As ObscureNet is also an autoencoder-based method, we explore the same weakness in our approach.

In this experiment, not only is the latent representation $\mathbf{z}$ available to the adversary, but a percentage $p\%$ of distinct records belonging to the true dataset $\mathbf{x}$ is leaked, denoted $\mathbf{x}_{p\%}$. Naturally, since the adversary knows that these two datasets are related, the adversary trains a decoder on the subset $\mathbf{x}_{p\%}$ to attempt to retrieve the remaining $(100 - p)\%$ encoded rows from the latent representation. The aim of the experiment is to measure how well protected a latent representation is, should a latent representation and a part of the original dataset be leaked to the public.

A variation of this idea is the situation where instead a subset of the best possible reconstruction $\mathbf{x}'_{p\%}$ is leaked, to test whether that helps an adversary more than leaking the original. Another variation is if the adversary does not know which records in the latent representation $\mathbf{z}$ correspond to which records in the leaked original dataset $\mathbf{x}$ or not.

Bear in mind that this experiment models an already quite pessimistic situation when it comes to data anonymisation or pseudonymisation, especially with high ($> 50\%$) levels of original dataset leakage.

Setup. Execution of this experiment is divided into runs and subruns. Each run generates a synthetic dataset using `make_classification` as provided by Scikit Learn. Ten autoencoders are trained, where the best performing network is used to generate a latent representation $\mathbf{z}$ and a measure of the baseline decoder loss (BL). A subrun leaks a certain percentage $p\%$ of data ranging from 5% to 100% with increasing steps of 5%. Three different situations are investigated: (1) The first $p\%$ rows of $\mathbf{x}$ are leaked in the same order as the latent representation, allowing the adversary the advantage of a one-to-one matching of records belonging to the latent representation and records belonging to the true dataset. (2) The first $p\%$ rows of the original dataset are leaked, but the order of the latent representation records is randomly permuted. (3) The first $p\%$ rows of the best possible reconstruction $\mathbf{x}'$ generated by the original decoder are leaked. For each of these situations, ten adversarial decoders are trained on the leaked subset of data, where the adversarial decoder that performs the best is selected

to generate a reconstruction of the latent representation $\mathbf{z}$. The training of ten encoders and decoders is to compensate for the stochasticity of initialisation and updating of network parameters as described in Section 2.3. Finally, the true loss (TL) and approximate loss (AL) between the reconstruction of the best adversarial decoder and the respective comparison dataset are calculated as described above and stored in MLflow. In the case that the experiment varies the standard deviation of the synthetic dataset, the standard deviation increases each run, to experiment whether the standard deviation of a dataset has any effect on its deanonymisation weakness.

2. Clustering permuted datasets

Idea. In the previous experiment we discussed the efficacy of permuting the order of records as an additional measure of protection. To verify this, we attempt to reverse this permutation. The naïve approach to do this would be to brute-force the ordering. However, this approach quickly becomes infeasible as the brute-force runtime increases factorially. Furthermore, for each of these permutations, a reasonably-performing decoder network has to be trained to evaluate the performance of a particular ordering. Instead, we approach this by approximation. The goal of an adversarial decoder is to learn the pattern by which the original encoder encoded the latent representation to reconstruct the original dataset. Hence, similar original data is encoded similarly in the latent representation. This is clearly visible in Figure 2.3, where, for example, the handwritten digit one seems to frequently feature a high value in the third dimension (column) of the latent representation, and the number four seems to characteristically feature a high value in the fourteenth dimension.

Setup. To find and match such characteristics in an unsupervised manner, we use a K-means clustering algorithm on both the latent representation and the leaked original data. Next, multiple adversarial decoders are trained for each possible cluster permutation. The best-performing decoder may have benefitted from learning the relations that made the clustering algorithm put these records together and as such is used to decode the full latent representation. This approximation aims to reduce the computational complexity from brute-forcing all possible permutations of records, a complexity of $O(p!)$ with p denoting the size of the dataset, to all possible permutations of clusters, which is still a computational complexity of $O(n!)$ where n is the number of clusters used.

3. Exploration of data utility loss

Idea. To be able to answer **SQ3**: What is the utility loss incurred when using a latent representation of the data rather than the original data? We compare the performance between classification models trained on baseline data and latent representations of the data to determine the extent of utility loss.

Setup. We test Scikit-Learn's logistic regression algorithm and the XGBoost classifier for classification problems, and Scikit-Learn's linear regression and the XGBoost regressor for regression problems. We chose these algorithms due to the popularity of Scikit-Learn and XGBoost, and to cover different algorithms from the ones Yang tested in his blog post. Three different data conditions are tested, where for each condition, the data is tested in its original form as a baseline, its latent representation, and the permuted latent representation to measure if permuting has an effect on data utility. When testing the permuted latent representation, the class label vector is permuted the same way as the latent representation as to not lose those relations for the data controller. The first condition is the baseline performance of the unscaled dataset as retrieved from their respective data source. The second condition scales each feature down to $[0, 1]$ on a per-feature basis based on their respective minimal and maximal values, a process known as min-max scaling. The third condition min-max scales the latent representation of the unscaled data set to $[0, 1]$. As mentioned in Section 2.3, the latent representation of the unscaled dataset consists of values of $\mathbf{z} \in \mathbb{R}_+^b$ with bottleneck dimensionality b. This step scales this latent representation back to $[0, 1]$ to test whether scaling latent representations can result in an increase in performance. Finally, the results are averaged from a ten-fold cross-validation process. In these experiments, no extensive model selection or other careful data preparation beyond min-max scaling is performed. This means that the performance of these models could be potentially improved; however, doing so is beyond the scope of these experiments.

We test on two additional datasets for the regression algorithms for this experiment. These datasets are similar to the California House Price Dataset in that they also feature various housing attributes and their corresponding house prices for homes in Paris, France and King County, United States. Both datasets are public domain datasets sourced from Kaggle (Kaggle, 2016, 2021).

4. Misalignment of MSE

Idea. We investigate the misalignment of using MSE as a loss function to train autoencoders for the purpose of retaining data utility in their latent representations. As mentioned in Section 2.3, MSE penalises larger differences in the reconstruction $\mathbf{x}'$ more than smaller differences. Features with larger mean and standard deviation contribute more to the loss function than small features, such as binary labels. We hypothesise that autoencoders exhibit the following two behaviours: First, these larger values are over-represented not only in the reconstruction $\mathbf{x}'$ created by the autoencoder, but also in the latent representation $\mathbf{z}$. Second, as a result, if the over-represented values are highly informative (or in other words, correlate greatly with) the label vector, the data utility remains high when training on a latent representation $\mathbf{z}$ compared to the original dataset $\mathbf{x}$. Conversely, if only the smaller and thus under-represented values are informative, the data utility will be significantly worse as these highly informative variables are neglected by MSE.

Setup. First, we use Scikit-Learn's `make_classification` function to generate the informative features and corresponding y labels. These informative features are either 'large', or multiplied by a scalar of 100 by means of the `scale` parameter, or 'small', and kept between $[0, 1] \in \mathbb{R}$. These informative features are supplemented by ten similarly small or large noisy features generated by NumPy's `random.normal` function, where these noisy values are scaled similarly to the informative features. (Harris et al., 2020). The large informative features are paired with ten small noisy features to ensure that the informative features are over-represented in the resulting latent representation $\mathbf{z}$. Likewise, small informative features are paired with large noisy features to attempt to drown these out as hypothesised. The results of this experiment are shown in Subsection 4.2.4.

5. Exploring similarities between latent representations

Idea. As we postulate in Subsection 2.3.4, owing to the highly stochastic nature of weight initialisation and gradient descent, there exist multiple well-trained latent representations for any given dataset, each with its own corresponding best-performing decoder that an adversary would have to find.

Additionally, by representing higher-dimensional data in a lower dimensionality, we surmise that certain latent representations might represent multiple datasets at once. On top of that, that the odds of a collision occurring in the latent space are extremely low, but theoretically possible. Regardless, the consequence of these two assumptions is that the search space for an adversary for any given latent representation is large. Without a priori knowledge of what kind of data is encoded into a latent representation, we gather that the search for a decent reconstruction is unreasonably difficult.

Assuming that a collision in latent space is hypothetically possible but extremely rare, we do not attempt to force a collision by brute force. Instead, we generate a multitude of latent representations of the same dataset to explore the validity of our conjecture. We acknowledge that exploring this conjecture in this manner will never truly provide a definite answer, short of providing a definite proof. Nonetheless, it should give us a clearer image on the nature of latent representations for the purposes of data pseudonymisation.

Setup. We train ten autoencoders on the Pima Indians Diabetes dataset and ten autoencoders on the default synthetic dataset as provided by Scikit-Learn's `make_classification` function, and choose the autoencoder that performs best. These networks are used to generate latent representations of their respective dataset. The weights of the bottleneck layer of these autoencoder networks are also stored. This process is repeated ten times. Once completed, these resulting latent representations and bottleneck weight vectors are compared using MSE.

6. Runtime Experiments

Idea. We run an experiment to show the relatively short training time of a simple autoencoder as defined in Subsection 4.1.1 on various data shapes, even when trained on a CPU.

Setup. Once again we use Scikit-Learn's `make_classification`. We vary the number of features, the number of records, and both simultaneously. Whilst increasing the number of features, we generate datasets of 10000 records. When the number of records is increased, the number of features is set to 700. Specifically in this experiment, we do not use our standard method of training ten autoencoders and choosing the one that performs best. Rather, we are solely interested in the training time of a single autoencoder regardless of performance. Autoencoders stop training when the delta of validation loss between epochs falls below 0.0001.

4.2 Results

4.2.1 Varying levels of data leakage

Synthetic data

Figure 4.1 shows six scatterplots depicting the loss achieved by adversarial decoders trained on a data leak of synthetic datasets. The loss is the result of the attempt to decode the full latent representation $\mathbf{z}$ into a reconstruction $\mathbf{x}'$. The severity of the data leak that determines the adversarial decoder training set is varied on the x-axis. Furthermore, the figure compares three situations of data leakage. The first row of graphs shows the situation in which the first $p\%$ records of the original dataset $\mathbf{x}$ are leaked. The second row of graphs shows the situation in which the randomly permuted $p\%$ records of the original data set $\mathbf{x}$ are leaked. The third row shows the situation in which the first $p\%$ records of the best possible reconstruction $\mathbf{x}'$ created by the initial decoder are leaked. In each of these plots, the visualised data consist of ten runs that all run this experiment using the same generated synthetic dataset.

First, we discuss our observations between the true loss (TL) and the approximate loss (AL). The approximate loss is lower than the true loss. Adversarial decoders are trained to approximate the best possible decoder; it intuitively follows that they create a similar result. However, it is interesting that the approximate loss never reaches zero: in other words, adversarial decoders fail to perfectly replicate the best possible decoder, even when the full dataset is leaked. In the third row, adversarial decoders are trained using the reconstruction of the latent representation $\mathbf{x}'$ as the training set. These adversarial decoders achieve less approximate loss compared to the other situations. However, this does not seem to affect the true loss compared to leaking the true dataset.

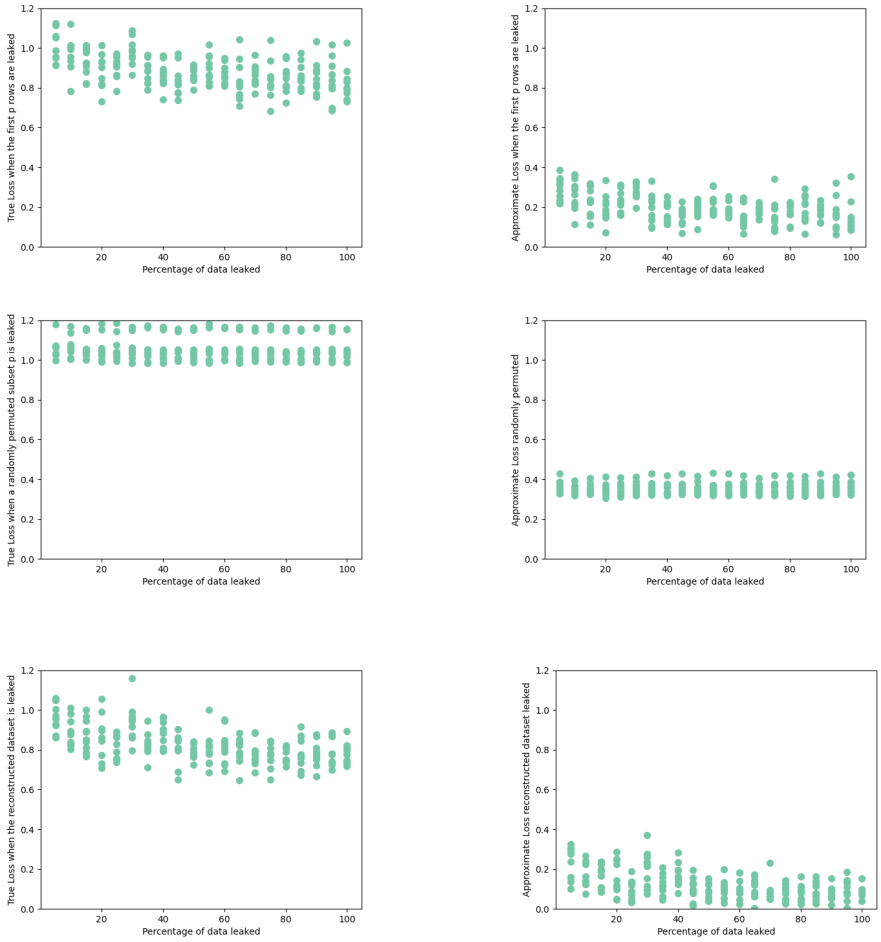

Figure 4.1: Scatterplots between the percentage $p\%$ of data leaked and the resultant loss of an adversarial decoder on 10 distinct synthetic datasets. Left column: True Loss (TL). Right column: Approximate Loss (AL). First row: The first $p\%$ of $\mathbf{x}$ has been leaked. Second row: A randomly permuted subset $p\%$ of $\mathbf{x}$ has been leaked. Third row: The first $p\%$ of $\mathbf{x}'$ has been leaked.

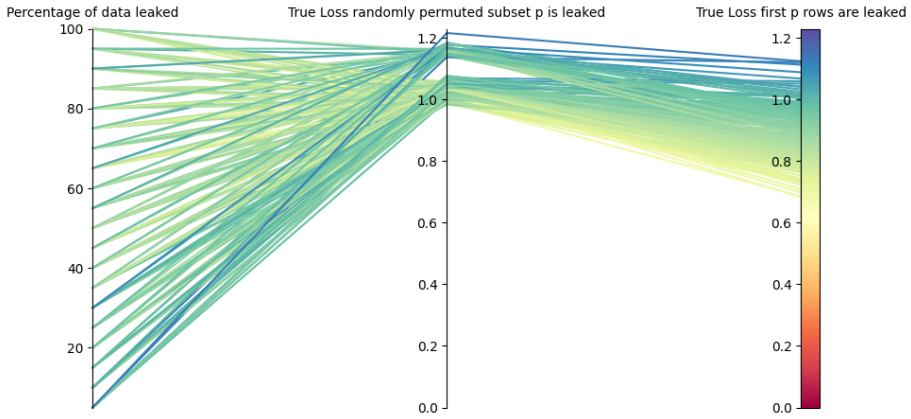

Figure 4.2: Parallel coordinates plots showing how the percentages of data leakage of synthetic data $p\%$ relate to resultant loss of an adversarial decoder. Middle: Loss when a randomly permuted subset $p\%$ of $\mathbf{x}$ has been leaked. Right: Loss when the first $p\%$ of $\mathbf{x}$ has been leaked. Note how the loss of the permuted data is strictly higher compared to the loss of leaking the first $p\%$ of $\mathbf{x}$.

Second, in all these situations, the performance of the adversarial decoder only slightly improves when trained on larger data leaks as opposed to smaller data leaks. To confirm this, we calculated the Pearson correlation coefficients (also known as Pearson's R, Freedman et al., 2007) between the percentage of data leakage p and the true loss (TL). This resulted in a correlation coefficient of $R_{p,TL} = -0.53$, which is a weak negative correlation between loss and data leakage. The implication of this result is that a data leak of any size has an effect on decoding; however, this effect does not worsen should a data leak become larger.

Third, the simple act of permuting the order of records causes a large reduction adversarial decoder performance. The true loss achieved by these adversarial decoders trained on a permuted data set is strictly greater than the loss achieved by adversarial decoders that benefit from being able to match latent representation records with (reconstructed) records of $\mathbf{x}$. Furthermore, the effect of decreased loss when the severity of data leakage increases, is even less than in the aforementioned situation. This permutation results in a correlation coefficient of $R_{p,TL} = -0.41$. Table 4.2 shows the heterogeneity measure (HM) of the permuted dataset is much lower than the other two datasets, implying that adversarial decoders trained on the permuted datasets lose a lot of valuable information to be able to learn why one encoded datapoint is different from another. This implies that a simple permutation of the latent representation is an effective way to further obscure which dataset is encoded in the latent representation. The loss achieved by even a very severe data leak hovers around or exceeds one standard deviation of the dataset.

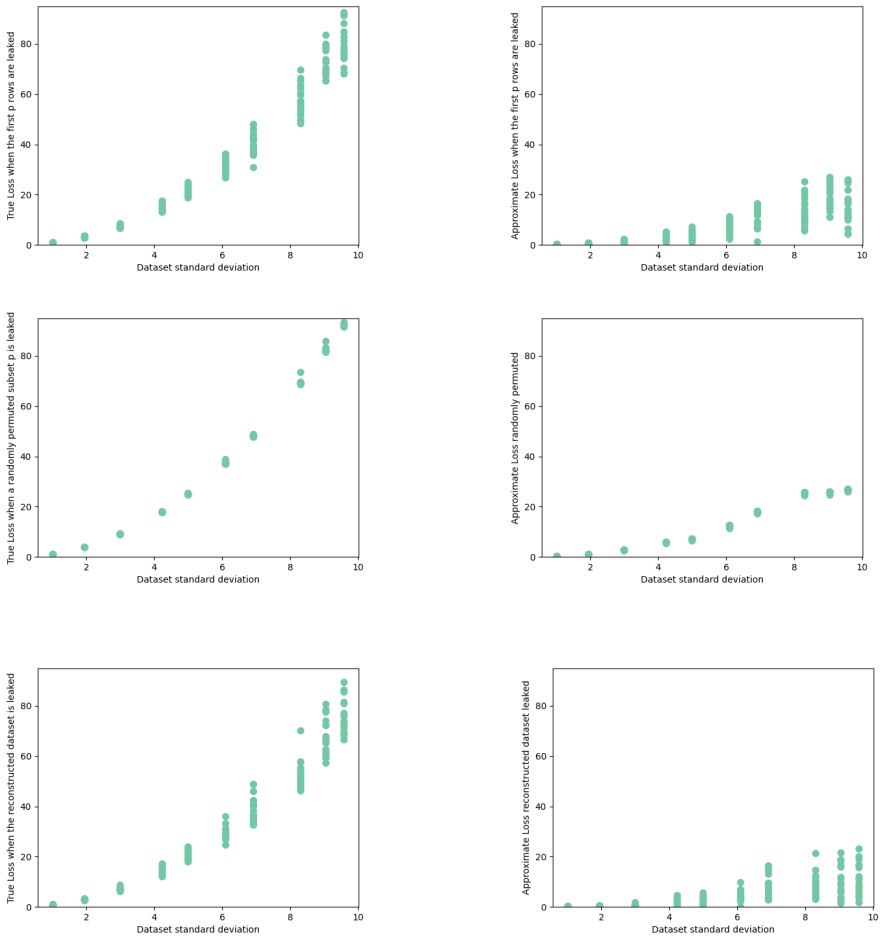

Figure 4.3: Scatterplots between the standard deviation of the generated synthetic dataset and the resultant loss of an adversarial decoder. Left column: True Loss (TL). Right column: Approximate Loss (AL). First row: The first $p\%$ of $\mathbf{x}$ has been leaked. Second row: A randomly permuted subset $p\%$ of $\mathbf{x}$ has been leaked. Third row: The first $p\%$ of $\mathbf{x}'$ has been leaked.

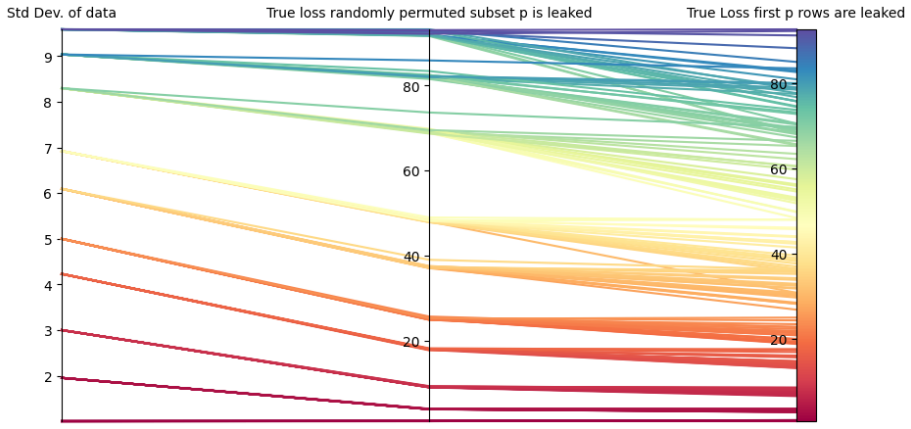

Figure 4.4: Parallel coordinates plot showing how the standard deviation of the dataset relate to resultant loss of an adversarial decoder. Middle: Loss when a randomly permuted subset $p\%$ of $\mathbf{x}$ has been leaked. Right: Loss when the first $p\%$ of $\mathbf{x}$ has been leaked.

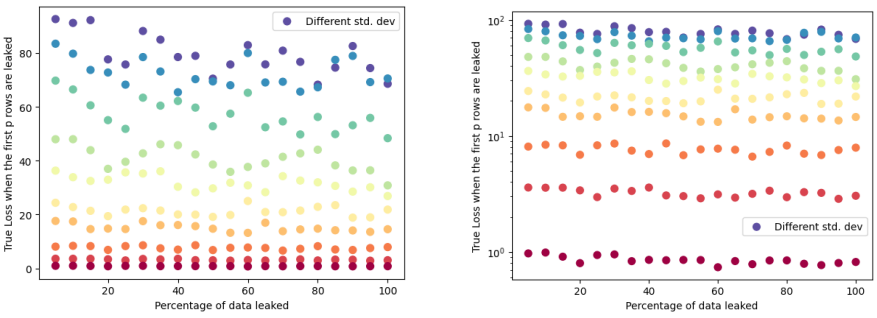

Figure 4.5: Scatterplots between the percentage of data leaked and the resulting loss of an adversarial decoder. Dots of the same colour belong to the experiment with the same dataset and its respective standard deviation. Left: linear true loss y-axis. Right: logarithmic true loss y-axis.

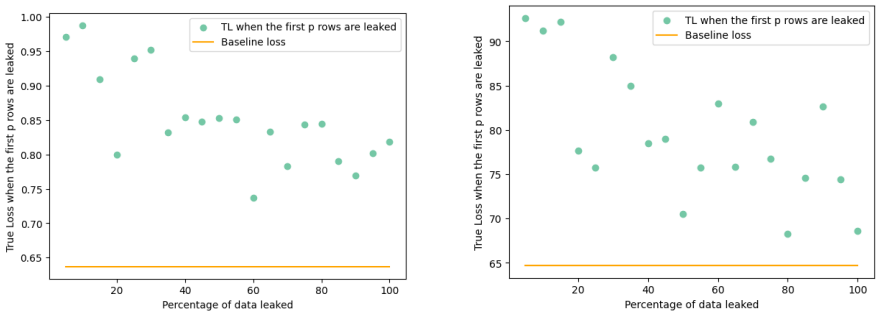

Figure 4.6: Scatterplots between when the first $p\%$ of $\mathbf{x}$ is leaked and the resultant loss of an adversarial decoder. Additionally, a horizontal line is shown showing the baseline loss of the best possible decoder. Left: A synthetic dataset with a standard deviation of 1. Right: A synthetic dataset with a standard deviation of 9.

Any information an adversary gleans from the decoder is erroneous at best, and they are better off looking at the leaked data directly. An adversary would have to brute force both the original permutation of the records and the original set of decoder parameters θ. Permuting does not have an effect on data utility (see Subsection 4.2.3).

The fourth phenomenon we observed is that the loss achieved by the adversarial decoder in these experiments is around one standard deviation of the dataset. To experiment with this, we generated datasets with standard deviations ranging from 1 to 10 using the `scale` parameter of the Sckikit-Learn `make_classification` function. Figure 4.3 shows six scatterplots depicting the relation between the standard deviation of the dataset and the resulting losses of adversarial autoencoders trained and tested in the same way as the previous experiments. The true loss shows a nearly quadratic relation between the standard deviation and the loss of the dataset. At low data leakage percentages, the loss of an adversarial decoder is around a squared standard deviation. We hypothesize that this is a result of the decoder network creating a lossy reconstruction $\mathbf{x}'$ where the values differ approximately one standard deviation from the original, resulting in a measured True Loss of one squared standard deviation due to the squaring operation in the MSE calculation. As the severity of the data leak increases, the loss of the adversarial decoder naturally sinks below a squared standard deviation. Similarly to the experiments discussed above, the approximate loss is strictly lower than the true loss and does not grow quadratically with the standard deviation. Thus, a permuted latent representation results in a strictly higher loss than if the latent representation were not permuted, as seen in Figure 4.4.

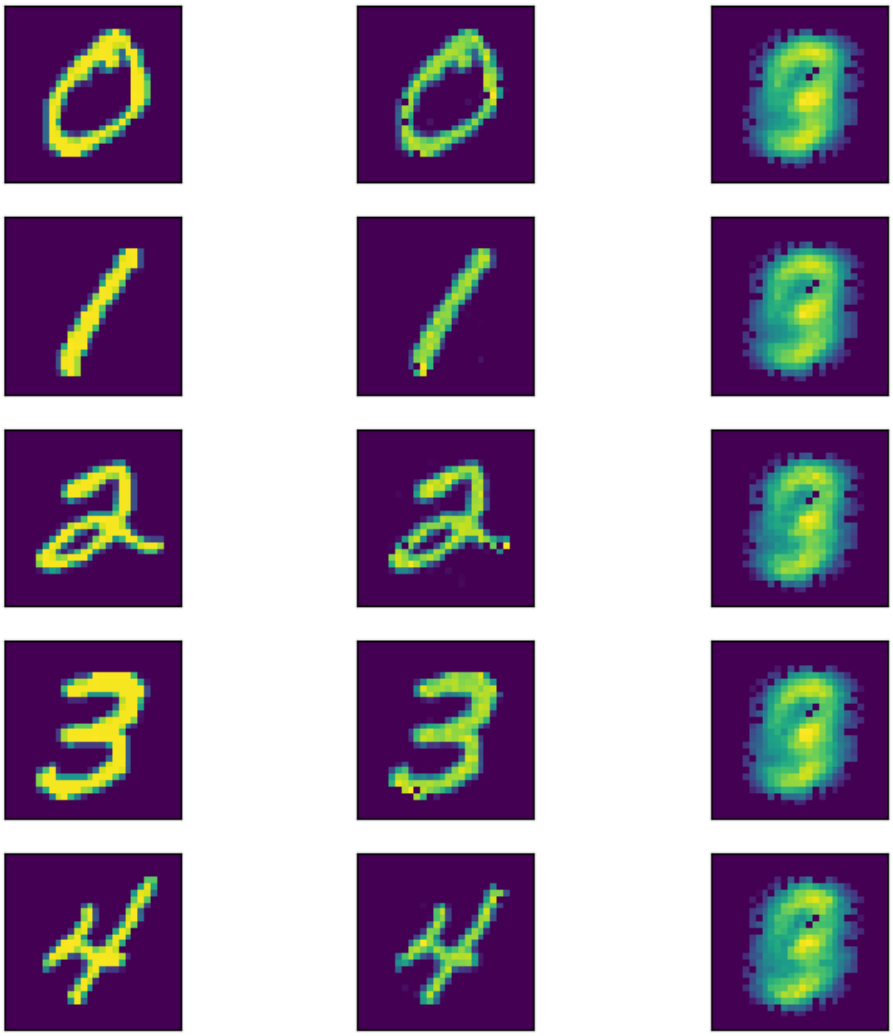

Figure 4.7: Comparing the results of adversarial decoders. Left: Original dataset $\mathbf{x}$. Middle: Reconstruction of an adversarial decoder trained on a leak of 60% of the data. Right: Reconstruction of an adversarial decoder trained on a leak of 60% of permuted data.

Table 4.2: Results of experiments with $p = 60\%$ data leakage. In the columns with three values per row, each row corresponds with leaking the first $p\%$ of data, permuted data and reconstructed data, from top to bottom, respectively.

Dataset	**BL**	**TL**	**AL**	$\mathbf{R_{p,TL}}$	**HM**($\mathbf{x}$)	**HM**($\mathbf{x'}$)
		0.848	0.166	-0.53		0.208
Synthetic	0.723	1.052	0.343	-0.41	2.392	0.039
		0.836	0.102	-0.58		0.311
		66.45	13.76	-0.34		3090.593
Pima	57.14	1250.7	1193.5	-0.30	3365.489	132.677
		76.65	27.31	-0.29		2964.417
		0.006	0.005	-0.75		0.1139
MNIST	0.003	0.068	0.065	-0.84	0.1317	0.0003
		0.009	0.005	-0.91		0.1178
		85033.8	2815.6	-0.59		2.89e9
Housing	83657.6	3.31e8	3.31e8	-0.55	2.91e9	3.03e5
		83650.2	4.076	-0.70		2.86e9

Figure 4.5 depicts the results of ten experiments that compare the percentage of data leaked to the loss of an adversarial decoder. Each experiment used a different synthetic dataset generated with a different standard deviation. Generating this scatterplot with a linear y-axis makes it seem that data sets with higher standard deviations appear more sensitive to a more severe data leak; however, plotting these same data against a logarithmic y-axis shows that this was likely a result of the squaring in the loss function. Calculating Pearson's correlation coefficient for the leakage of the dataset and permuted dataset is $R_{p,TL} = -0.54$ and $R_{p,TL} = -0.46$, respectively, achieving a very similar performance as described earlier. This confirms that pseudonymising a dataset with a greater standard deviation is not weaker to reidentification in this manner than a dataset with a smaller standard deviation.

Finally, Figure 4.6 shows the result of two singular experiments, each with its own data set and respective standard deviation. In these graphs, the baseline loss of the best possible decoder obtained during training is depicted alongside the true loss of the adversarial decoders. At higher levels of data leakage, adversarial decoders achieve a lower loss but never match the performance of the best possible decoder. Once again, the adversary is better off looking at the leaked data directly.

Pima Indians Diabetes data

To find out whether these same observations hold for different datasets, we repeated this experiment with the Pima Indians Diabetes dataset as described in Subsection 4.1.3.

Figure 4.8 shows six scatterplots showing the same setup as before: True and approximate loss of directly leaked data, permuted leaked data, and leaking the best reconstruction of the original data.

The observation of the true loss starting at one standard deviation squared does not hold for this dataset. We postulate that this is due to the fact that each attribute in the Pima Indians Diabetes dataset does not share roughly the same average and standard deviation. On account of how the networks are optimised using MSE as a loss function, decoders prioritise more accurately reconstructing attributes that contribute more towards the loss function as opposed to ones that impact the loss function less. That is, correctly reconstructing values that hover around one hundred reduces loss more than values that are binary. In fact, in our observations of decoder reconstructions $\mathbf{x}'$ of this dataset it is not uncommon to find that such attributes with low averages and standard deviations tend to be zeroed out completely, losing all information in those attributes even in the best possible reconstruction as generated during training. The implications of this are two-fold: Reconstructing data with significant differences in averages and standard deviation is likely to lose a lot of data represented as small values, such as binary labels. Similarly, the original autoencoder also uses MSE as the loss function. It might therefore be possible that these small values also neglect

to be encoded in the latent representation as well. How this affects data utility is explored in Subsection 4.2.4.

This loss as a result of a discrepancy between attributes can be partially alleviated by normalising all values beforehand (however, see the effect of doing so on data utility in Subsection 4.2.3 and Subsection 4.2.4), but this in itself also suffers from information loss should decoding a latent representation be necessary. Experiments testing the difference in performance between scaled and nonscaled datasets are discussed in Subsection 4.2.3.

The second observation is that the effect of permuting the latent representation on the loss and heterogeneity measure is larger than this effect observed on synthetic data.

The loss approaches the baseline loss at extremely high levels of data leakage as seen in Figure 4.9. This implies that not every dataset is as resistant to data leakage as synthetic data sets as seen in Figure 4.6. Furthermore, we measured Pearson's R correlation coefficient between decoder loss and percentage of data leakage of these results. Without permuting the latent representation, Pearson's $R_{p,TL} = -0.34$. This means that the correlation between data leakage and improved decoder performance is even weaker; however, this is because the baseline performance of these decoders is already much better compared to the decoders that worked with the synthetic dataset, meaning that improvements are harder to achieve. Permuting the latent representation results in $R_{p,TL} = -0.30$, which is similarly a weak correlation. Nevertheless, this result is much more optimistic: Although there is much potential to improve performance, the networks do not achieve this potential.

MNIST handwritten digits data

Figure 4.7 shows three columns. The left column shows the handwritten digits in the original dataset. The middle column shows the digits as reconstructed by an adversarial decoder trained on 60% leaked data. The right column shows the digits reconstructed by an adversarial decoder trained on 60% leaked, permuted data. There are a few conclusions to be drawn based on this visualisation.

Whilst by our measures an adversarial decoder may achieve a higher MSE loss than a baseline decoder, one should carefully consider when the reconstructed data is considered depseudonymised. The reconstructed data is always lossy. This means that when it is of critical importance that the exact value of the data is to be hidden and that a lossy reconstruction is too imprecise to be critical, then this method of pseudonymisation can suffice. But in the case of the MINST dataset as illustrated by Figure 4.7, if a mere approximation of the original already suffices as depseudonymisation, then this method proves insufficient. This visualisation shows the nuance of using MSE in our True Loss and Approximate Loss metrics as our metrics, as the digits are clearly reconstructed and distinguishable from one another.

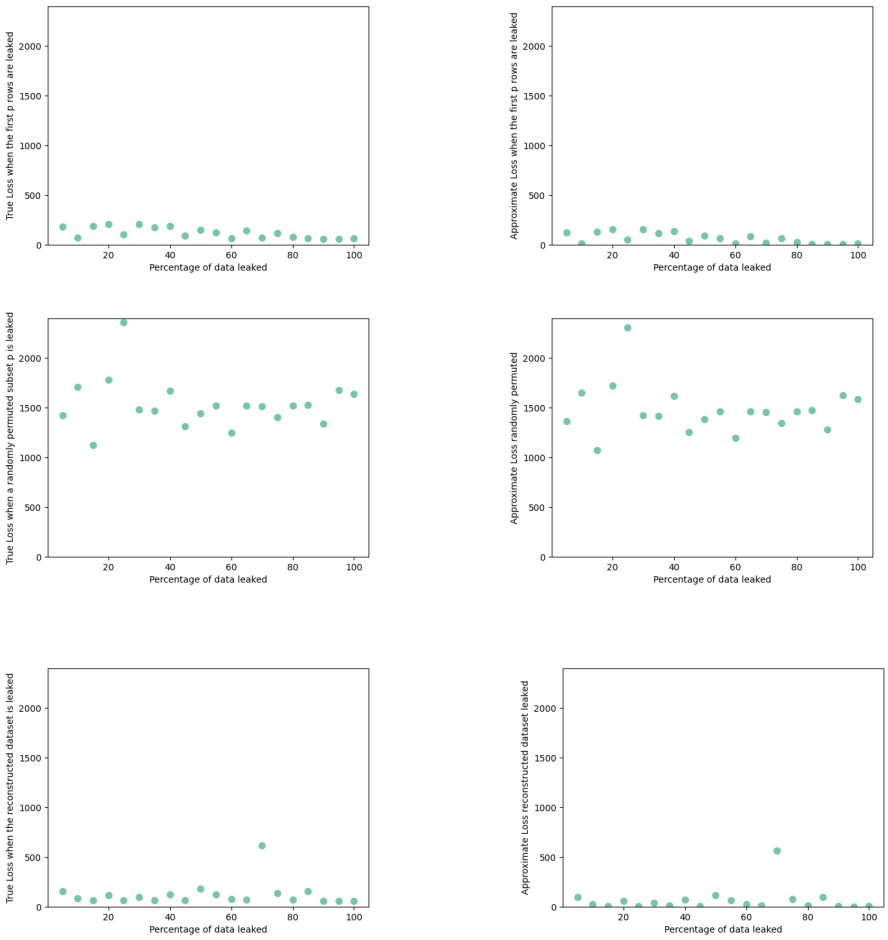

Figure 4.8: Scatterplots between the percentage $p\%$ of data leaked and the resultant loss of an adversarial decoder on the Pima Indians Diabetes dataset. Left column: True Loss (TL). Right column: Approximate Loss (AL). First row: The first $p\%$ of $\mathbf{x}$ has leaked. Second row: A randomly permuted subset $p\%$ of $\mathbf{x}$ has been leaked. Third row: The first $p\%$ of $\mathbf{x}'$ has leaked.

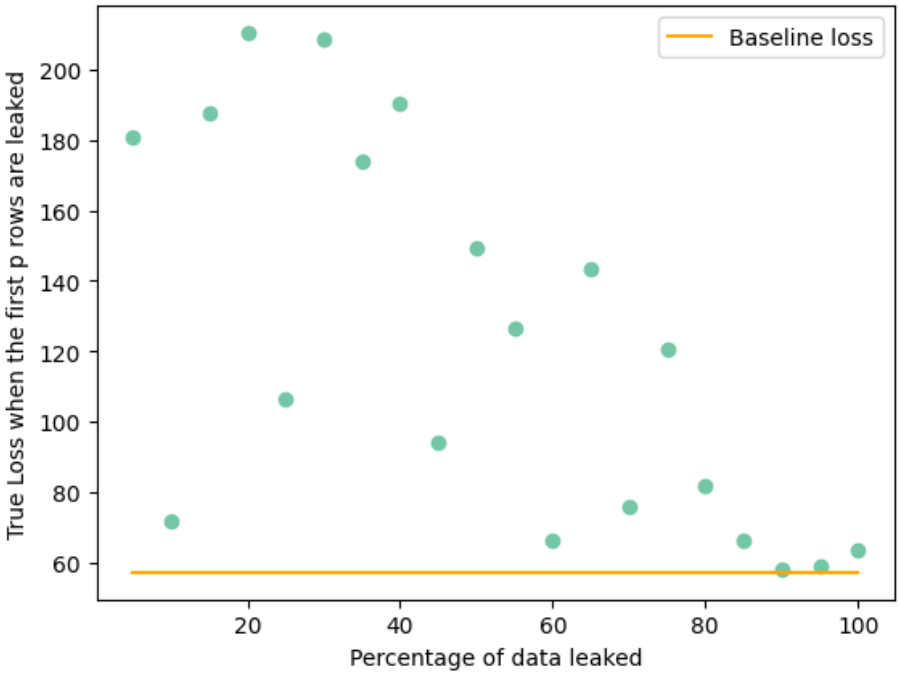

Figure 4.9: Scatterplot between when the first $p\%$ of $\mathbf{x}$ is leaked and the resultant loss of an adversarial decoder on the Pima Indians Diabetes dataset. Additionally, a horizontal line is shown showing the baseline loss of the best possible decoder.

However, the figure also shows the right column, which is a clear visualisation of the implications of the heterogeneity measure. This is the work of an adversarial decoder that is trained on permuted data. This visualisation shows why permutation is such an effective way to hinder the performance of adversarial decoders. The best the adversarial decoder could do in an effort to minimise the mean squared error is to always predict a vague combination of three, eight, and nine, where any given result is indistinguishable from one another. This is because any relation between the latent representation and the original digit is lost for the adversarial decoder.

Conclusion

In conclusion, Table 4.2 summarises our numerical results of these experiments with a $p = 60\%$ level of data leakage. It consistently appears that, for the (reconstructed) data leakage experiments, adversarial decoders are able to reasonably approximate the baseline levels of loss and heterogeneity. The severity of this result depends on the sensitivity of the data. For instance, when pseudonymising salaries, the exact salary figure of a given person will not be perfectly reconstructed. However, a decently performing adversarial decoder would be able to read a salary range of each person in the dataset. Therefore, whether inaccurate reconstructed figures are considered a breach of sensitive data depends entirely on the nature of the dataset in question. Adversarial decoders trained on an unpermuted latent representation together with a data leak are able to recreate these figures with decent accuracy, despite misrepresentations as a result of using MSE (see Subsection 4.2.4). Having said that, permuted latent representations show much more promising results, with very low heterogeneity measures (where higher is a better reconstruction) and high loss measures (where lower is better is a better reconstruction). In other words, these decoders are only able to generate a roughly average reconstruction for each of the encoded records, where each individual record is meaningless and indistinguishable from the other. This implies that latent representations are not secure from reidentification unless it is possible to permute the latent representations, so adversaries lose the knowledge of which latent representation record corresponds with which leaked data record. Both these conclusions are best visualised in Figure 4.7.

4.2.2 Clustering permuted datasets

In the results of our previous experiment, we repeatedly and consistently showed how randomly permuting a dataset significantly hinders the performance of adversarial decoders, both in terms of our defined metrics True Loss and Approximate Loss, and visually in our MNIST illustration.

Figure 4.10 depicts three scatterplots showing the results of our clustering approach. The resulting loss of using four clusters to attempt to match permuted data tends to be equal to or greater than the loss of adversarial decoders directly trained on the permuted data.

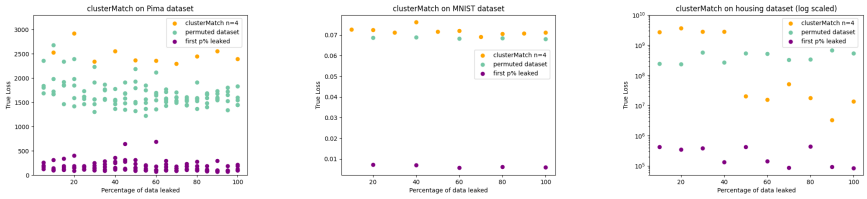

Figure 4.10: Scatterplot comparing the performance between an adversarial decoder and a decoder aided with cluster matching to attempt to reverse the randomisation. This is done on the Pima Indians Diabetes dataset (left) and the MNIST handwritten digits dataset (middle) and the California Housing Prices dataset (right, Pace and Barry, 1997). There are less cluster matching datapoints due to the increase in runtime.

This shows that the clustering approach is insufficient when using a small number of clusters. We hypothesise that increasing the number of clusters would grant the algorithm more granularity to group together more similar representations. Doing so also makes the algorithm prohibitively expensive. For example, the best-case scenario for the MNIST dataset is when the clustering algorithm manages to perfectly cluster the datapoints in ten disjoint sets for both the latent representation and the original dataset. Still, the adversary has to find the only permutation of 10! that adheres to the original data set.

This results in training an adversarial decoder on this (representative sample of) data for each of these 10! permutations. This even happens under the assumption that these clusters are perfect; any imperfections add noise to this process. In a realistic scenario, most datasets are not as heterogeneous as MNIST and the optimal number of clusters is greater than 10.

We propose researching more efficient solutions to finding the optimal permutation of clusters in future work in Section 5.3. It might be possible to devise a method that iteratively works towards the optimal permutation, or testing different clustering algorithms other than K-means. Doing so falls outside the scope of this thesis. We conclude that a naive clustering algorithm does not easily reverse the permutation of a latent representation.

4.2.3 Data utility

Table 4.3 up to and including Table 4.16 show the results of data utility experiments. First, we will consider the results of the classification algorithms, XGBoost, and logistic regression. In general, the following effects are observed: In all cases, there exists a trade-off for training on a latent representation rather than the original data, the severity of which differs per dataset. This could be a natural effect of data loss as a result of dimensionality reduction.

The utility delta for using the latent representation is the greatest in the exper-

iments with the synthetic dataset using the XGBoost algorithm. The MNIST handwritten digits and Pima Indians Diabetes datasets suffer a smaller reduction in performance. The MNIST dataset performs nearly identical when using logistic regression and only slightly worse when using XGBoost. The Pima dataset performs slightly worse with both algorithms.

The permuted latent representation always performs similarly to the unpermuted latent representation. This pattern presents itself consistently with each data-scaling condition and classification algorithm. This implies that if the utility loss of using the latent representation is already within acceptable range to trade off for the benefits of pseudonymisation, the latent representation can be permuted as a further measure of security without suffering any additional penalty to data utility.

Next, we discuss the regression algorithms. Using latent representations appears to substantially negatively affect regression algorithms. This is especially noticeable in Table 4.9 where the r^2 score drops from 0.545 on the original dataset to -0.12 when using the latent representation, implying that a model trained on the latent representation performs worse than always predicting the median price over all houses.

To further investigate this effect on regression algorithms, we test three additional datasets. In these experiments, we observe that latent representations consistently negatively affect regression performance worse than classification performance.

At the other end of the spectrum, the Paris Housing Prices dataset manages to achieve a r^2 score of 1, except when trained on the latent representation of the original scaled data set, falling to 0.44. Despite that very good r^2 score, the mean squared error score does not approach zero. Overall, the XGBoost regressor appears to perform better than linear regression but is also negatively affected by using the latent representation. In general, these results imply that latent representations are better suited to pseudonymise data sets used in classification problems than datasets used for regression problems. We further explore this in our next experiment in Subsection 4.2.4.

In terms of data-scaling conditions, we observe no large differences between scaling original data or latent representation for most datasets. The exception to this is the Paris Housing Prices regression, where the latent representation of the original scaled data performed substantially worse than the other conditions. However, this behaviour appears to be an outlier, and therefore no conclusions could be made on these results alone.

In these experiments, no extensive model selection or other careful data preparation beyond min-max scaling is performed. This means that the performance of these models could be potentially improved, however, doing so falls beyond the scope of these experiments.

Table 4.3: Results of data utility on a generated synthetic dataset predicting a binary class label using logistic regression.

Synthetic, LogR	Unscaled			Scaled			Latent Scaled	
Metric	Original	Latent	Perm	Original	Latent	Perm	Latent	Perm
B. Accuracy	0.945	0.939	0.944	0.944	0.932	0.935	0.889	0.889
Weighted F1	0.945	0.939	0.944	0.944	0.932	0.935	0.889	0.889
ROC AUC	0.974	0.974	0.974	0.975	0.972	0.972	0.959	0.962
Avg. Precision	0.946	0.940	0.945	0.946	0.936	0.939	0.891	0.893

Table 4.4: Results of data utility on the Pima Indians Diabetes dataset predicting a binary class label using logistic regression.

Pima, LogR	Unscaled			Scaled			Latent Scaled	
Metric	Original	Latent	Perm	Original	Latent	Perm	Latent	Perm
B. Accuracy	0.724	0.690	0.686	0.704	0.681	0.685	0.672	0.675
Weighted F1	0.734	0.699	0.695	0.715	0.690	0.694	0.680	0.683
ROC AUC	0.830	0.794	0.793	0.828	0.793	0.798	0.782	0.787
Avg. Precision	0.763	0.737	0.727	0.759	0.735	0.745	0.737	0.737

Table 4.5: Results of data utility on the MNIST handwritten digits dataset predicting a multiclass label using logistic regression.

MNIST, LogR	Unscaled			Scaled			Latent Scaled	
Metric	Original	Latent	Perm	Original	Latent	Perm	Latent	Perm
B. Accuracy	0.901	0.909	0.912	0.901	0.905	0.909	0.906	0.909
Weighted F1	0.900	0.908	0.912	0.900	0.905	0.909	0.905	0.909
ROC AUC	0.987	0.988	0.990	0.987	0.987	0.990	0.988	0.989
Avg. Precision	0.902	0.910	0.913	0.902	0.906	0.910	0.907	0.910

Table 4.6: Results of data utility on a generated synthetic dataset predicting a binary class label using XGBoost.

Synthetic, XGB	Unscaled			Scaled			Latent Scaled	
Metric	Original	Latent	Perm	Original	Latent	Perm	Latent	Perm
B. Accuracy	0.950	0.869	0.864	0.950	0.897	0.892	0.869	0.866
Weighted F1	0.950	0.869	0.864	0.950	0.897	0.892	0.869	0.866
ROC AUC	0.968	0.922	0.916	0.968	0.943	0.933	0.922	0.912
Avg. Precision	0.953	0.871	0.865	0.953	0.901	0.895	0.871	0.867

Table 4.7: Results of data utility on the Pima Indians Diabetes dataset predicting a binary class label using XGBoost.

Pima, XGB	Unscaled			Scaled			Latent Scaled	
Metric	Original	Latent	Perm	Original	Latent	Perm	Latent	Perm
B. Accuracy	0.702	0.654	0.653	0.702	0.648	0.645	0.654	0.649
Weighted F1	0.703	0.655	0.653	0.703	0.650	0.647	0.655	0.650
ROC AUC	0.796	0.730	0.725	0.795	0.729	0.741	0.730	0.726
Avg. Precision	0.712	0.661	0.663	0.712	0.666	0.660	0.661	0.658

Table 4.8: Results of data utility on the MNIST handwritten digits dataset predicting a multiclass label using XGBoost.

MNIST, XGB	Unscaled			Scaled			Latent Scaled	
Metric	Original	Latent	Perm	Original	Latent	Perm	Latent	Perm
B. Accuracy	0.978	0.957	0.958	0.978	0.953	0.956	0.957	0.958
Weighted F1	0.978	0.957	0.958	0.978	0.953	0.955	0.957	0.958
ROC AUC	1.000	0.999	0.999	1.000	0.998	0.999	0.999	0.999
Avg. Precision	0.978	0.957	0.958	0.978	0.953	0.955	0.957	0.958

Table 4.9: Results of data utility on the California Housing Prices dataset predicting the median house price using linear regression.

C. Housing, LinR	Unscaled			Scaled			Latent Scaled	
Metric	Original	Latent	Perm	Original	Latent	Perm	Latent	Perm
MSE	5.09e+09	1.19e+10	1.13e+10	5.09e+09	1.19e+10	1.09e+10	1.19e+10	1.13e+10
r^2	0.545	-0.12	0.148	0.545	-0.086	0.174	-0.12	0.151

Table 4.10: Results of data utility on the King County, USA Housing Prices dataset predicting the median house price using linear regression.

KC. Housing, LinR	Unscaled			Scaled			Latent Scaled	
Metric	Original	Latent	Perm	Original	Latent	Perm	Latent	Perm
MSE	4.10e+10	6.49e+10	6.47e+10	4.10e+10	9.30e+10	9.30e+10	6.49e+10	6.47e+10
r^2	0.695	0.516	0.519	0.695	0.308	0.308	0.516	0.519

Table 4.11: Results of data utility on the Paris Housing Prices dataset predicting the median house price using linear regression.

P. Housing, LinR	Unscaled			Scaled			Latent Scaled	
Metric	Original	Latent	Perm	Original	Latent	Perm	Latent	Perm
MSE	3.61e6	6.75e+07	7.16e+07	3.61e6	4.631933e+12	4.630512e+12	6.747248e+07	6.947615e+07
r^2	1.00	1.00	1.00	1.00	0.44	0.44	1.00	1.00

Table 4.12: Results of data utility on the MNIST handwritten digits dataset predicting the handwritten digit using linear regression.

MNIST, LinR	Unscaled			Scaled			Latent Scaled	
Metric	Original	Latent	Perm	Original	Latent	Perm	Latent	Perm
MSE	9.332e+18	2.888	2.881	4.925+20	2.484	2.476	2.888	2.882
r^2	-0.000	0.654	0.655	-0.000	0.703	0.704	0.654	0.655

Table 4.13: Results of data utility on the California Housing Prices dataset predicting the median house price using the XGBoost Regressor.

C. Housing, LinR	Unscaled			Scaled			Latent Scaled	
Metric	Original	Latent	Perm	Original	Latent	Perm	Latent	Perm
MSE	3.49e+09	8.94e+09	8.27e+09	3.49e+09	5.51e+09	4.39e+09	8.94e+09	8.34e+09
r^2	0.547	-0.230	0.123	0.547	0.264	0.535	-0.230	0.115

Table 4.14: Results of data utility on the King County, USA Housing Prices dataset predicting the median house price using the XGBoost Regressor.

KC. Housing, LinR	Unscaled			Scaled			Latent Scaled	
Metric	Original	Latent	Perm	Original	Latent	Perm	Latent	Perm
MSE	1.56e+10	6.40e+10	6.21e+10	1.56e+10	3.45e+10	3.38e+10	6.37e+10	6.21e+10
r^2	0.885	0.522	0.534	0.885	0.743	0.747	0.523	0.533

Table 4.15: Results of data utility on the Paris Housing Prices dataset predicting the median house price using the XGBoost Regressor.

P. Housing, LinR	Unscaled			Scaled			Latent Scaled	
Metric	Original	Latent	Perm	Original	Latent	Perm	Latent	Perm
MSE	9.12e+07	1.83e+09	1.70e+09	9.12e+07	5.41e+12	5.34e+12	1.83e+09	1.81e+09
r^2	1.000	1.000	1.000	1.000	0.346	0.354	1.000	1.000

Table 4.16: Results of data utility on the MNIST handwritten digits dataset predicting the handwritten digit using the XGBoost Regressor.

MNIST, XGBR	Unscaled			Scaled			Latent Scaled	
Metric	Original	Latent	Perm	Original	Latent	Perm	Latent	Perm
MSE	1.000	1.480	1.466	1.000	1.539	1.523	1.469	1.461
r^2	0.880	0.823	0.824	0.880	0.816	0.818	0.824	0.825

4.2.4 Misalignment of MSE

Table 4.17 up to and including Table 4.20 show the results of our MSE misalignment experiments. We observe the following phenomena:

Table 4.17 shows the latent representation outperforming the original unscaled version of the synthetic dataset. This is likely due to the over-representation of the large, informative features in the latent representation, as we had previously hypothesised. Min-max scaling this data substantially reduces the performance of both the original scaled data and the latent representation of the scaled data. Likewise, scaling the latent representation of the unscaled data also greatly reduces classification performance.

Table 4.18 shows the latent representation severely underperforming compared to the original version which corroborates our hypothesis. As the autoencoders are heavily incentivised by MSE as a loss function to underprioritise small losses, which in this case are the most important features for data utility, the performance of the classifier is substantially reduced. Interestingly, min-max scaling helps mitigate this loss in performance much more compared to the previous experiment conditions. Scaling the latent representation created from the unscaled data does not, however, implying that when this situation occurs, it is better to min-max scale the data before generating a latent representation.

Table 4.19 shows a very poor performance of the XGBoost regression algorithm on the dataset that combines large informative features with small noisy features. Similar to the classification performance on this same dataset, the latent representation of the unscaled data manages to outperform the original unscaled data. This indicates that large, informative features are over-represented in the latent representation. Regression performance on the scaled data, latent representation of the scaled data and scaled latent representation of the unscaled data is exceptionally poor, reaching negative levels of r^2 fit, indicating that always predicting the mean outcome performs vastly better than models trained on these scaled versions of the data.

Table 4.20 shows that the regression algorithm trained on the data containing small informative features exhibits a behaviour similar to the classification algorithm, in that the original regression performance is reasonable, but a regression model fit on the latent representation performs exceptionally poorly. This implies that small features are under-represented in latent representations. Scaling the data before generating a latent representation helps the performance somewhat, but the resulting model remains functionally useless.

We conclude that the feature selection by autoencoders depends a lot on the chosen loss function. MSE, as a loss function, promotes a different goal for the autoencoder than maximal data utility preservation for latent representations. As such, we suggest investigating better alternatives as future work in Section 5.3.

Table 4.17: Results of data utility on a synthetic dataset with two informative large features and ten small noise features classifying a binary label using the XGBoost classifier.

Large, Classificaton	**Unscaled**		**Scaled**		**Latent Scaled**
Metric	Original	Latent	Original	Latent	Latent
B. Accuracy	0.840	0.910	0.487	0.420	0.514
Weighted F1	0.838	0.908	0.482	0.413	0.495
ROC AUC	0.924	0.968	0.546	0.356	0.482
Avg. Precision	0.852	0.930	0.489	0.413	0.527

Table 4.18: Results of data utility on a synthetic dataset with two informative small features and ten large noise features classifying a binary label using the XGBoost classifier.

Small, Classificaton	**Unscaled**		**Scaled**		**Latent Scaled**
Metric	Original	Latent	Original	Latent	Latent
B. Accuracy	0.950	0.436	0.950	0.740	0.436
Weighted F1	0.949	0.422	0.949	0.738	0.422
ROC AUC	0.984	0.459	0.984	0.820	0.459
Avg. Precision	0.958	0.427	0.958	0.751	0.427

Table 4.19: Results of data utility on a synthetic dataset with two informative large features and ten small noise features classifying a binary goal variable using the XGBoost regressor.

Large, Regression	**Unscaled**		**Scaled**		**Latent Scaled**
Metric	Original	Latent	Original	Latent	Latent
MSE	0.145	0.097	0.334	0.394	0.361
r^2	0.287	0.532	-0.545	-0.837	-0.707

Table 4.20: Results of data utility on a synthetic dataset with two informative small features and ten large noise features classifying a binary goal variable using the XGBoost regressor.

Small, Regression	**Unscaled**		**Scaled**		**Latent Scaled**
Metric	Original	Latent	Original	Latent	Latent
MSE	0.036	0.327	0.036	0.260	0.327
r^2	0.844	-0.525	0.844	-0.040	-0.525

Table 4.21: A table comparing the MSE between ten different latent representations of the Pima Indians Diabetes dataset

MSE	1	2	3	4	5	6	7	8	9	10
1	0.000	20515.418	50251.199	136974.219	53319.523	37205.801	21240.037	29652.420	3823.734	8773.599
2		0.000	32153.834	84254.234	35446.617	26697.912	16774.590	22739.154	19929.461	6408.562
3			0.000	169337.812	91566.344	54978.133	74332.828	85604.352	46060.285	44344.320
4				0.000	59030.020	95154.031	107060.258	53434.461	139123.438	105024.164
5					0.000	75874.859	52387.055	42426.770	64108.891	29840.402
6						0.000	56974.543	29336.275	24094.768	35533.031
7							0.000	25068.242	27637.840	12405.395
8								0.000	27775.572	24372.621
9									0.000	11718.934
10										0.000

Table 4.22: A table comparing the MSE between ten different autoencoder network bottleneck layer weights of the Pima Indians Diabetes dataset

MSE	1	2	3	4	5	6	7	8	9	10
1	0.000	0.745	0.657	0.542	0.856	0.371	0.605	0.503	0.619	0.591
2		0.000	0.541	0.527	0.647	0.718	0.671	0.702	0.744	0.467
3			0.000	0.947	0.685	0.846	0.602	0.657	0.896	0.429
4				0.000	0.511	0.392	0.466	0.515	0.617	0.493
5					0.000	0.712	0.440	0.604	0.752	0.448
6						0.000	0.635	0.460	0.949	0.559
7							0.000	0.538	0.575	0.277
8								0.000	0.613	0.548
9									0.000	0.798
10										0.000

Table 4.23: A table comparing the MSE between ten different latent representations of the same synthetic dataset

MSE	0	1	2	3	4	5	6	7	8	9
0	0.000	2.691	2.469	4.709	2.855	2.676	2.321	6.030	2.876	5.254
1		0.000	2.449	4.359	2.786	2.450	2.423	4.417	2.663	7.582
2			0.000	3.814	2.821	2.060	2.238	4.180	3.083	5.955
3				0.000	4.242	5.114	4.720	7.461	3.386	7.280
4					0.000	2.910	2.638	4.250	3.386	5.839
5						0.000	1.748	4.699	3.629	6.173
6							0.000	5.285	3.716	5.133
7								0.000	5.742	9.663
8									0.000	7.310
9										0.000

Table 4.24: A table comparing the MSE between ten different autoencoder network bottleneck layer weights of a synthetic dataset

MSE	0	1	2	3	4	5	6	7	8	9
0	0.000	0.202	0.197	0.223	0.237	0.230	0.207	0.236	0.244	0.226
1		0.000	0.200	0.192	0.205	0.248	0.210	0.220	0.216	0.173
2			0.000	0.181	0.210	0.230	0.244	0.217	0.234	0.233
3				0.000	0.183	0.224	0.185	0.218	0.226	0.216
4					0.000	0.233	0.225	0.211	0.256	0.220
5						0.000	0.283	0.243	0.274	0.232
6							0.000	0.218	0.227	0.222
7								0.000	0.229	0.193
8									0.000	0.242
9										0.000

4.2.5 Latent representation similarity

Table 4.21 and Table 4.23 show the MSE similarity between ten different generated latent representations of the Pima Indians Diabetes dataset and the same synthetic dataset, respectively. Table 4.22 and Table 4.24 show the MSE similarity between the weight vectors of the bottleneck layers of the networks that produced these latent representations. Redundant table cells are grayed out for readability purposes.

Although the absolute values of these numbers are meaningless, the most important observation is that they generally do not approach zero. Instead, there even exists a variance in how much each pair of latent representation or weight vector differs from any other given pair. All these autoencoder networks follow our standard procedure of choosing the best of ten, meaning that all these networks have reasonable performance, as well. This confirms our hypothesis in Section 2.3; there exist multiple well-performing latent representations of any given dataset. The implication of this is that it is difficult for an adversary to discern what exact encoded data they might have encountered without any further knowledge, such as a data leak, as we had assumed in our data leakage experiments. Similarly, the weight vectors of the bottleneck layers differ greatly between networks. Note that the weight vectors of the other layers can also be very different, cumulatively ending up with very different parameter sets of the networks for the same dataset.

4.2.6 Runtime

Figure 4.11 and Figure 4.12 show how long it takes to train one single autoencoder as defined in Subsection 4.1.1 on datasets of various sizes. The plots on the left of Figure 4.11 show the impact of increasing the number of features. The plots on the right show the effect of increasing the number of records in the dataset. Finally, Figure 4.12 shows the effect of increasing both the number of records and features on runtime in a 3D triangular surface plot.

We observe a roughly linear increase in runtime as both variables increase. When the dataset is at its largest in this experiment (1300 features, 19800 records), the runtime averages around 35 seconds to train a single autoencoder. Intriguingly, the combination of a low number of features and a large number of records causes a much longer training time than when both properties increase proportionally, as seen in the 3D plot Figure 4.12. This behaviour is consistent over multiple runs of this experiment. Even then, the longest observed training time was only 100 seconds.

Recall that neural network weight initialisation and iterative improvement are highly stochastic processes, as described in Section 2.3. This stochasticity also influences the performance of the autoencoder. Hence, the reason why in other experiments, we train ten autoencoders and choose the network that performs best. This experiment shows the runtime of training a singular autoencoder network. Consequently, training ten networks should not take much more than

ten times the reported value on average. Runtime may increase for networks that are more complex than the networks we experimented with here, the properties of which are described in Subsection 4.1.1.

We conclude that the runtime of training one or ten autoencoder networks should not be the limiting factor for considering this method of data pseudonymisation. It is easy to train a new network for each dataset, or to train another network should the dataset gain additional records, retaining the property that these networks are free to overfit on the entire dataset.

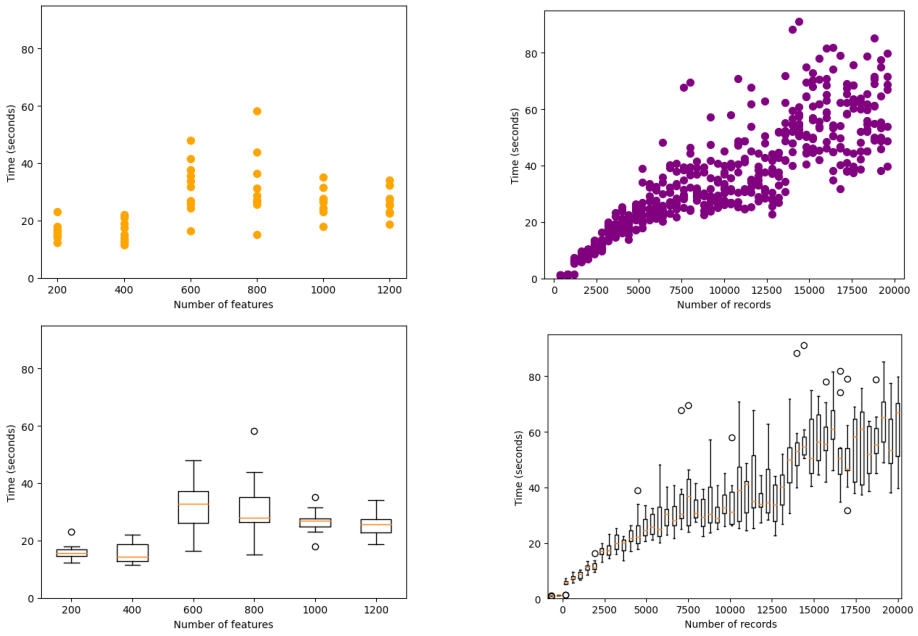

Figure 4.11: Scatter- and boxplots showing the effect of the number of features (left) or the number of records (right) on autoencoder training time. First row: All datapoints displayed in the scatterplot. Second row: Boxplots of the runtime per feature or record to display outliers.

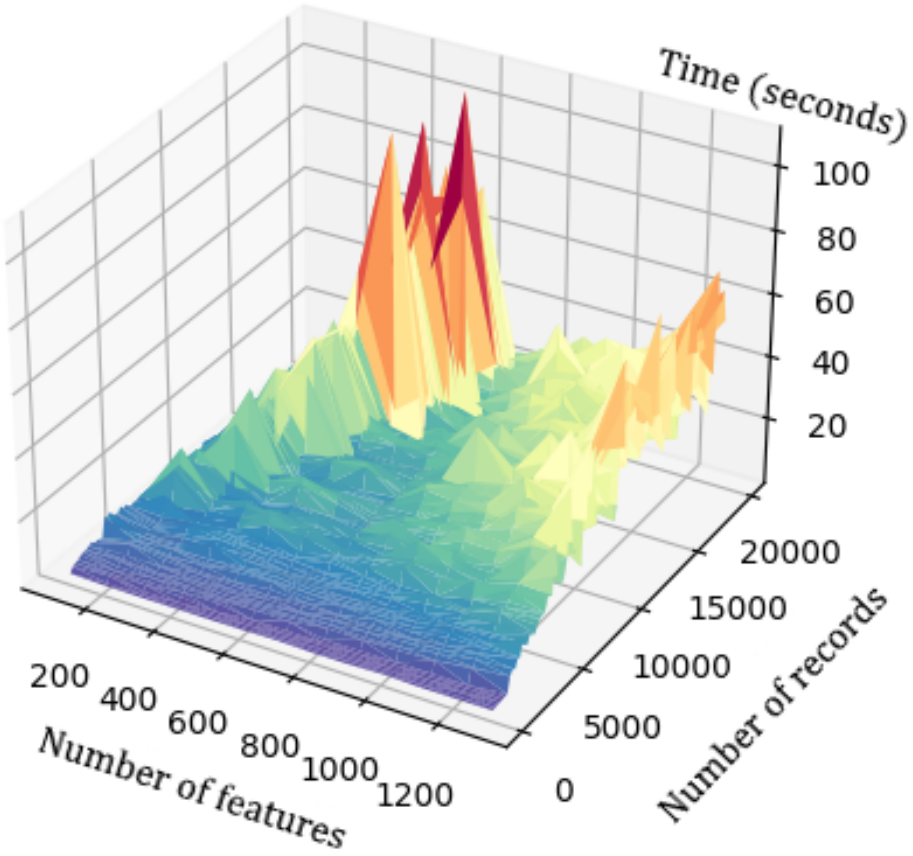

Figure 4.12: Effect of both features and records combined on autoencoder runtime displayed in a 3D triangular surface plot.

4.2.7 Discussion

In this section we globally summarise the most important results from the experiments and list our conclusions, as we have already discussed and drawn conclusions for each experiment individually in their respective results sections.

Data Leakage

Without permutation, adversarial decoders can achieve relatively low levels of data reconstruction loss if large ($> 50\%$) amounts of data of the original dataset is leaked. However, randomly permuting the dataset provides a great way to counteract this attack, provided that the adversary does not know this same permutation. As expected, if an adversary trains their adversarial decoder on a reconstruction of the dataset rather than the original, the Approximate Loss (AL) is lower, but this had no effect on the True Loss (TL).

How much TL/AL translates to (de)pseudonymisation depends entirely on the sensitivity of the data in question. If the reconstruction of the data is not considered 'leaked' whilst the reconstruction is flawed or lossy, then this method works as a pseudonymisation method. However, if even an approximation of the data is close enough to leak sensitive information, then usage of this method is not recommended, unless data permutation is possible. For example, although the exact figure of a person's salary may not be revealed by the reconstruction $\mathbf{x}'$, this would still leak the salary range of a given person. When permuted, the salary ranges may become wildly inaccurate as visualised by Figure 4.7.

Cluster Matching

In order to verify whether permuting latent representations is as secure as our data leakage experiments suggested, we experimented with a naïve attempt to cluster representations and the corresponding data points together. As visualised in Figure 2.3, latent representations for similar datapoints are encoded in the same way. If a clustering algorithm can detect these similarities and subsequently feed them to the adversarial decoder network, the permutations could be broken. Our results show that the loss of these clustered datapoints is often equal to, or even greater than, the loss of the permuted datasets, showing that permuting latent representations is resistant to a naïve attack of clustering.

Data Utility

In terms of data utility, there is a natural trade-off to using a latent representation to train classification or regression networks. However, for classification algorithms, this trade-off is not large. Regression algorithms suffer a greater loss of data utility when trained on a latent representation, implying that this method may not be suitable for regression problems. Min-max scaling of the dataset rarely has a substantial effect on data utility.

Combining the results of the first two experiments with this data utility experiment implies that permuting latent representations is a viable way of pseudonymising data without a significant impact on data utility.

Misalignment of MSE

In this thesis we trained autoencoders with MSE as their loss function. However, we observed that MSE leads to the over-representation of larger values in the reconstructed dataset $\mathbf{x}'$ over smaller values due to the squaring of the error, punishing larger absolute value differences more than smaller differences, although smaller values could be more valuable in terms of data utility. Our experiments indicate that this over-representation is also present in the latent representation $\mathbf{z}$, implying that MSE is misaligned for the goal of data utility retention in the latent representation. We suggest finding a better-aligned loss function as future work in section Section 5.3.

Measuring LR Similarity

To gain further insight in the behaviour of autoencoders, we measured how similar the latent representations and weight vectors of the bottleneck layers are of well-performing networks. Our results show that there exist multiple well-performing latent representations for any given dataset, making it more difficult for an adversary to know which exact dataset they might be dealing with when attempting to decode.

Runtime

Finally, we conclude that training autoencoders to generate well-performing latent representations does not take too much time, with the longest training time of a single autoencoder in our experiments being 100 seconds. That said, keep in mind that training larger networks than the ones we discuss may significantly impact training time.

Recommended practical application

We distill our experimentations into the following recommendation for a practical application of the findings this thesis. We suggest: Training ten autoencoders on the dataset $\mathbf{x}$ and choosing the one that performs best; using the best trained encoder network to generate a latent representation $f(\mathbf{x}) = \mathbf{z}$; randomly permuting the latent representation z along with the class label vector y to further protect the encoded data's privacy, whilst suffering no loss to data utility. This pseudonymised latent representation can then be used to train a classification algorithm or network of choice to learn $P(\mathbf{y}|\mathbf{z})$. It is up to the data controller to decide whether to discard the encoder, decoder and original dataset as a precaution; or to keep them in a secure, separate location if they are needed in the future. Refer to Figure 4.13 for an illustration of this recommendation.

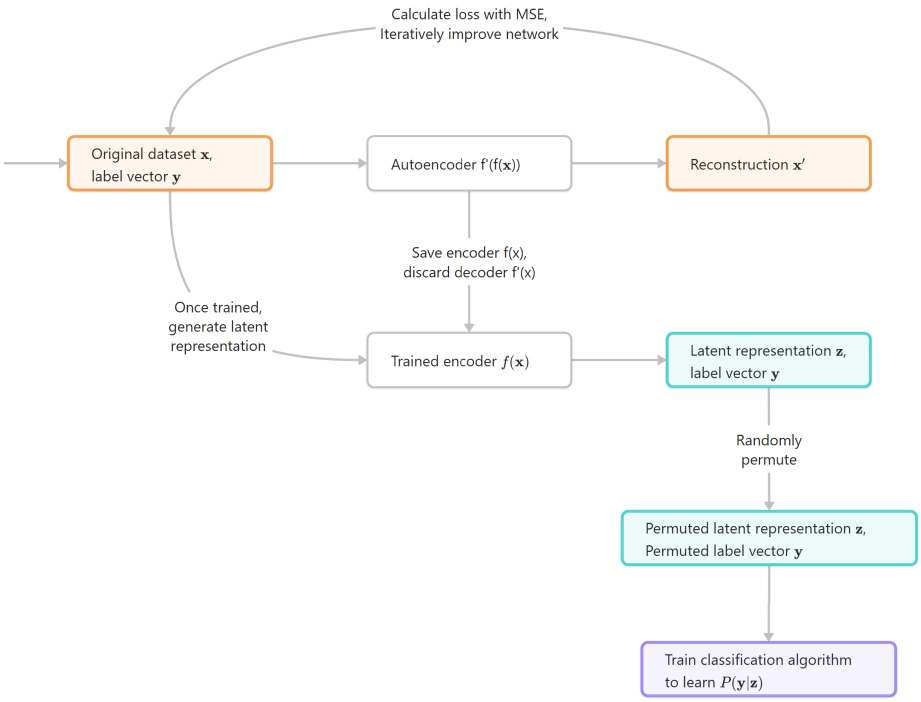

Figure 4.13: Recommended practical application of our findings.

Chapter 5

Conclusion

5.1 Main conclusions

This work sought to outline the advantages and disadvantages of using an autoencoder's latent representation as a data pseudonymisation method.

The main advantages are that without a data leak of the original data along with its latent representation, it is difficult for an adversary to generate a well-performing reconstruction of the encoded dataset. This method is more effective if the latent representation is randomly permuted. This permutation is not easily reversed using a clustering algorithm. A latent representation preserves its data utility well for classification algorithms, even when permuted. Our experiments indicate that a dataset can be represented by multiple, well-performing latent representations, making it difficult for an adversary to discern which original dataset was encoded in the latent representation. Finally, autoencoders do not take a long time to train, making it a relatively cheap method to pseudonymise data whilst retaining data utility for classification algorithms.

The main disadvantages are that by extrapolating existing GDPR regulation considerations, it is very likely that this method would be considered a data pseudonymisation method rather than an anonymisation method. Furthermore, regression algorithms suffer a larger penalty to data utility than classification algorithms when trained on a latent representation. Finally, our experiments indicated that using the mean squared error as the loss function for the autoencoders does not maximise the data utility in latent representations.

The secure handling of data becomes only more important in an increasingly data-driven society. Ever since GDPR came into effect in 2018, many companies have been fined for irresponsibly handling sensitive customer data. If sensitive data could be pseudonymised into a latent representation that would remain safe, even if leaked, whilst retaining its powerful data analysis potency, it might

be a more preferable method of handling data in the future.

5.2 Limitations

The process of training and using an autoencoder to pseudonymise sensitive data is still considered processing sensitive data according to GDPR. Our conclusions are merely an extrapolation based on existing regulatory considerations that are not intended to be actual legal advice.

Our proposed pseudonymisation method comes with a few limitations. Trivially, if the exact value is of great importance to preserve, then neither the latent representation $\mathbf{z}$ nor the reconstructed dataset $\mathbf{x}'$ will suffice. Similarly, there is no meaningful data visualisation for latent representations without contextualising it with the original dataset. It is only when the distribution of the data is important, whilst keeping the individual datapoints pseudonymous, that this pseudonymisation method is applicable.

Similarly, if an approximate reconstruction of the values in the dataset is also considered sensitive, then this method is not applicable. For example, reconstructing a latent representation of a dataset that contained exact salary figures might not reconstruct these exact numbers but may still provide the adversary with a salary range. Permuting the latent representation may somewhat mitigate this issue, as illustrated by Figure 4.7.

The most effective data pseudonymisation method whilst preserving data utility that we found, namely permuting latent representations, trivially does not apply to data where the order of records matters, such as time series data.

Finally, our data utility experiments show that regression algorithms suffer a greater loss of data utility than classification algorithms.

5.3 Future work

In this work, we have assumed to be working with machine learning data, or data that otherwise has a distribution. This does not cover the full range of data that could be anonymised or pseudonymised. Further research could be done if it is possible to pseudonymise unstructured data like textual data, by involving research on how textual data can be encoded numerically like in research of natural language processing, for example.

Our experiments indicate that using the mean squared error as the loss function for the autoencoders does not maximise the data utility in latent representations. We believe that data utility could be improved a lot more if a better-suited loss function is chosen, whilst it would retain most, if not all, of the data pseudonymisation properties discussed in this work.

The research of conditional variational autoencoders (CVAE, Kingma et al., 2014; Sohn et al., 2015) led to the idea of researching a CVAE conditioned on a large (prime) number similar to some encryption methods. This (prime) number could function as a private key such that the correct distribution is only sampled if this trained network is supplied with the corresponding key. The resultant sampled datapoints would be entirely synthetic, but the distribution of the learnt data points should remain intact, leading to a possible method to encrypt a data distribution. The practical applications of this idea remain nebulous, and exploring this is vastly out of scope for this research; therefore, we offer a potential future work in the field of data anonymisation or pseudonymisation using machine learning applications.

Our cluster matching attack could be expanded on by making it more efficient, or by testing other clustering algorithms than K-means clustering, or by a priori deducing an ideal number of clusters, or finding some kind of iterative improvement method akin to stochastic gradient descent.

There might be other, novel ways to permute the latent representation to hinder adversarial decoding whilst preserving data utility that may be interesting to experiment with. One such way is to permute the attributes rather than rows. Another experiment is to add noise to the latent representation, be it in the form of adding meaningless features or fake records. However, this noise could adversely impact data utility.

In this work, we have chosen to encode the entire dataset. Should only a subset of the features of the dataset be deemed sensitive, then further research could be done how to safely encode only some private variables whilst retaining the remainder in their original form, potentially improving data utility.

Another possible attack vector for this method of data pseudonymisation is the case where both the latent representation and encoder leak. In this situation, the adversary has access to a pair of original data and an encoding. The research question would be whether that is enough for an adversary to train a decoder to decode any given latent representation encoded by that encoder, even if it is not the same data that the decoder was trained on.

The pseudonymising autoencoder could be varied in architecture, such as adding more layers or varying the bottleneck size. More extremely, the network could consist of an ensemble of autoencoders learning to represent subsets of features of the entire dataset rather than using only one for the entirety of the dataset, to investigate if these have significant impact on both data privacy and utility.

Finally, it might be interesting to investigate whether there are possibilities to improve the data utility for regression algorithms.

5.4 Declaration of conflict of interest

This work is the result of a collaboration between Utrecht University and Info Support. Daniël Salomons received hardware and funding from Info Support. Maarten Vos is an employee of Info Support. Dr. Ing. Krempl and Dr. Mitici declare no conflict of interest.

Bibliography

Amini, A., Soleimany, A., Karaman, S., & Rus, D. (2019). Spatial Uncertainty Sampling for End-to-End Control [arXiv:1805.04829 [cs]]. https://doi.org/10.48550/arXiv.1805.04829

Art. 29 WP. (2014). Opinion 05/2014 on "anonymisation techniques". Retrieved March 21, 2023, from https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/index_en.htm

Balın, M. F., Abid, A., & Zou, J. (2019). Concrete autoencoders: Differentiable feature selection and reconstruction. *International conference on machine learning*, 444–453.

Bellare, M., Canetti, R., & Krawczyk, H. (1996). Keying hash functions for message authentication. *Advances in Cryptology—CRYPTO'96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings 16*, 1–15.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, *29*(6), 141–142.

Domingo-Ferrer, J., & Torra, V. (2008). A Critique of k-Anonymity and Some of Its Enhancements. *2008 Third International Conference on Availability, Reliability and Security*, 990–993. https://doi.org/10.1109/ARES.2008.97

Drapkin, A. (2023). Data Breaches That Have Happened in 2023 So Far - Updated List. Retrieved February 22, 2023, from https://tech.co/news/data-breaches-updated-list

Espinoza-Cuadros, F. M., Perero-Codosero, J. M., Antón-Martín, J., & Hernández-Gómez, L. A. (2020). Speaker de-identification system using autoencoders and adversarial training. *arXiv preprint arXiv:2011.04696*.

European Commission. (2016a). Art. 4 GDPR – Definitions. Retrieved March 22, 2023, from https://gdpr.eu/article-4-definitions/

European Commission. (2016b). Art. 6 GDPR – Lawfulness of processing. Retrieved March 22, 2023, from https://gdpr.eu/article-6-how-to-process-personal-data-legally/

European Commission. (2016c). On the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection

regulation). Retrieved February 22, 2023, from https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679

European Commission. (2016d). Recital 26 - Not Applicable to Anonymous Data. Retrieved March 17, 2023, from https://gdpr.eu/recital-26-not-applicable-to-anonymous-data/

Freedman, D., Pisani, R., & Purves, R. (2007). Statistics (international student edition). *Pisani, R. Purves, 4th edn. WW Norton & Company, New York*.

Gauravaram, P. (2012). Security Analysis of salt——password Hashes. *2012 International Conference on Advanced Computer Science Applications and Technologies (ACSAT)*, 25–30.

Grand View Research. (2023). Cloud Artificial Intelligence Market — Cloud AI Report 2023. Retrieved April 28, 2023, from https://www.grandviewresearch.com/industry-analysis/cloud-ai-market-report

Hajihassani, O., Ardakanian, O., & Khazaei, H. (2020). Latent representation learning and manipulation for privacy-preserving sensor data analytics. *2020 IEEE Second Workshop on Machine Learning on Edge in Sensor Systems (SenSys-ML)*, 7–12.

Hajihassani, O., Ardakanian, O., & Khazaei, H. (2021). ObscureNet: Learning attribute-invariant latent representation for anonymizing sensor data. *Proceedings of the international conference on internet-of-things design and implementation*, 40–52.

Hamm, J. (2017). Minimax filter: Learning to preserve privacy from inference attacks [Publisher: JMLR. org]. *The Journal of Machine Learning Research*, *18*(1), 4704–4734.

Han, K., Wang, Y., Zhang, C., Li, C., & Xu, C. (2018). Autoencoder inspired unsupervised feature selection. *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 2941–2945.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification [arXiv:1502.01852 [cs] version: 1]. https://doi.org/10.48550/arXiv.1502.01852

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., & Lerchner, A. (2017). Beta-vae: Learning basic visual concepts with a constrained variational framework. *International conference on learning representations*.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions [Publisher: World Scientific]. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *6*(02), 107–116.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Kaggle. (2016). House sales in king county, usa — kaggle. Retrieved July 10, 2023, from https://www.kaggle.com/datasets/harlfoxem/housesalesprediction

Kaggle. (2021). Paris housing price prediction — kaggle. Retrieved July 10, 2023, from https://www.kaggle.com/datasets/mssmartypants/paris-housing-price-prediction

Kessler, G. C. (2003). An overview of cryptography.

Kingma, D. P., & Ba, J. (2017). Adam: A Method for Stochastic Optimization [arXiv:1412.6980 [cs]]. https://doi.org/10.48550/arXiv.1412.6980

Kingma, D. P., & Welling, M. (2022). Auto-Encoding Variational Bayes [arXiv:1312.6114 [cs, stat]]. https://doi.org/10.48550/arXiv.1312.6114

Kingma, D. P., Mohamed, S., Jimenez Rezende, D., & Welling, M. (2014). Semi-supervised learning with deep generative models. *Advances in neural information processing systems*, *27*.

Kumar, S. K. (2017). On weight initialization in deep neural networks. *arXiv preprint arXiv:1704.08863*.

Li, N., Li, T., & Venkatasubramanian, S. (2006). T-closeness: Privacy beyond k-anonymity and l-diversity. *2007 IEEE 23rd international conference on data engineering*, 106–115.

Machanavajjhala, A., Kifer, D., Gehrke, J., & Venkitasubramaniam, M. (2007). L-diversity: Privacy beyond k-anonymity [Publisher: ACM New York, NY, USA]. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *1*(1), 3–es.

Majeed, A., & Lee, S. (2020). Anonymization techniques for privacy preserving data publishing: A comprehensive survey [Publisher: IEEE]. *IEEE access*, *9*, 8512–8545.

Malekzadeh, M., Clegg, R. G., Cavallaro, A., & Haddadi, H. (2019). Mobile sensor data anonymization. *Proceedings of the international conference on internet of things design and implementation*, 49–58.

Malekzadeh, M., Clegg, R. G., & Haddadi, H. (2017). Replacement autoencoder: A privacy-preserving algorithm for sensory data analysis. *arXiv preprint arXiv:1710.06564*.

Micheli, M., Ponti, M., Craglia, M., & Berti Suman, A. (2020). Emerging models of data governance in the age of datafication. *Big Data & Society*, *7*(2), 205395172094808. https://doi.org/10.1177/2053951720948087

Nousi, P., Papadopoulos, S., Tefas, A., & Pitas, I. (2020). Deep autoencoders for attribute preserving face de-identification [Publisher: Elsevier]. *Signal Processing: Image Communication*, *81*, 115699.

Oechslin, P. (2003). Making a faster cryptanalytic time-memory trade-off. *Advances in Cryptology-CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings 23*, 617–630.

Pace, R. K., & Barry, R. (1997). Sparse spatial autoregressions. *Statistics & Probability Letters*, *33*(3), 291–297.

Panda Security. (2019). The GDPR in 2019: The year of the million euro fine. Retrieved March 17, 2023, from https://www.pandasecurity.com/en/mediacenter/security/gdpr-fines-summary/

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Perero-Codosero, J. M., Espinoza-Cuadros, F. M., & Hernández-Gómez, L. A. (2022). X-vector anonymization using autoencoders and adversarial training for preserving speech privacy. *Computer Speech & Language*, *74*, 101351. https://doi.org/10.1016/j.csl.2022.101351

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). *Learning internal representations by error propagation* (tech. rep.). California Univ San Diego La Jolla Inst for Cognitive Science.

Russell, A. (1993). Necessary and sufficient conditions for collision-free hashing. *Advances in Cryptology—CRYPTO'92: 12th Annual International Cryptology Conference Santa Barbara, California, USA August 16–20, 1992 Proceedings 12*, 433–441.

Saad, D. (1999). *On-line learning in neural networks*. Cambridge University Press.

Saunders, B., Camgoz, N. C., & Bowden, R. (2021). Anonysign: Novel human appearance synthesis for sign language video anonymisation. *2021 16th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2021)*, 1–8.

Scikit-learn. (2011). SKLearn make_classification documentation. Retrieved July 10, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html

Smith, J. W., Everhart, J. E., Dickson, W., Knowler, W. C., & Johannes, R. S. (1988). Using the adap learning algorithm to forecast the onset of diabetes mellitus. *Proceedings of the annual symposium on computer application in medical care*, 261.

Sohn, K., Lee, H., & Yan, X. (2015). Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, *28*.

Sun, Y., Li, J., Wang, W., Plaza, A., & Chen, Z. (2016). Active learning based autoencoder for hyperspectral imagery classification. *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 469–472.

Sweeney, L. (2000). Simple demographics often identify people uniquely. *Health (San Francisco)*, *671*(2000), 1–34.

Sweeney, L. (2002). K-anonymity: A model for protecting privacy [Publisher: World Scientific Publishing Co.]. *Int. J. Unc. Fuzz. Knowl. Based Syst.*, *10*(05), 557–570. https://doi.org/10.1142/S0218488502001648

Wang, S., Ding, Z., & Fu, Y. (2017). Feature selection guided auto-encoder [Issue: 1]. *Proceedings of the AAAI Conference on Artificial Intelligence*, *31*.

Weggenmann, B., Rublack, V., Andrejczuk, M., Mattern, J., & Kerschbaum, F. (2022). Dp-vae: Human-readable text anonymization for online reviews with differentially private variational autoencoders. *Proceedings of the ACM Web Conference 2022*, 721–731.

Xu, X., Gu, H., Wang, Y., Wang, J., & Qin, P. (2019). Autoencoder based feature selection method for classification of anticancer drug response [Publisher: Frontiers Media SA]. *Frontiers in genetics*, *10*, 233.

Yang, S. (2020). Data Anonymization with Autoencoders. Retrieved February 13, 2023, from https://towardsdatascience.com/data-anonymization-with-autoencoders-75d076bcbea6

Yen, L. (2021). Current Trends & Future Scope of Data Mining. Retrieved March 17, 2023, from https://www.datamation.com/big-data/data-mining-trends/