Neural Data Shaping and Evaluation via Mutual Information Estimation

by

William Wu

B.S. Computer Science, MIT (2021)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2022

Author . . . .
Department of Electrical Engineering and Computer Science
September 5, 2022

Certified by . . . .
Muriel Médard
NEC Professor of Software Science & Engineering, MIT
Thesis Supervisor

Certified by . . . .
Homa Esfahanizadeh
Postdoctoral Researcher
Thesis Supervisor

Accepted by . . . .
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Neural Data Shaping and Evaluation via Mutual Information Estimation

by

William Wu

Submitted to the Department of Electrical Engineering and Computer Science
on September 5, 2022, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Machine learning in sensitive domains like healthcare currently faces a major bottleneck due to the scarcity of data that is publicly available. Privacy protection regulations such as HIPAA and GDPR and recent progress in information estimation literature motivate us to investigate the issue from an information theoretic perspective. In this thesis, we propose *InfoShape*, an encoder training scheme that aims to maintain privacy while also preserving utility for downstream prediction tasks. We achieve this by utilizing mutual information neural estimation (MINE) [2] to estimate two quantities, privacy leakage: the mutual information between the original inputs and the encoded representations, and utility score: the mutual information between the encoded representations and the intended labeling information for classification. We train a neural network as our encoder by using our privacy and utility measures in a Lagrangian optimization. We show empirically on Gaussian generated data that *InfoShape* is capable of altering encoded sample outputs such that the privacy leakage is reduced and the utility score increases. Moreover, we observe that the classification accuracy of downstream models has a meaningful connection with the utility score, which improves after we train an encoder compared to the untrained encoder. This work has profound implications for privacy-preserving machine learning and could serve as a pivotal tool in the future for revolutionizing AI in areas like healthcare.

Thesis Supervisor: Muriel Médard
Title: NEC Professor of Software Science & Engineering, MIT

Thesis Supervisor: Homa Esfahanizadeh
Title: Postdoctoral Researcher

Acknowledgments

I'd like to take this opportunity to thank all the gracious people who have helped me in my journey throughout MIT. The last 5 years, including my MEng year, have been a tremendous growth and learning opportunity for me. I have never felt remotely as challenged, stimulated, excited, frustrated, or achieved as I have during my time here. This has truly been a life-changing experience, and it really couldn't have happened without those around me.

First, I would like to thank Professor Muriel Médard, who made this whole opportunity possible. She coordinated everything that was behind the scenes and steered the project to the most promising direction when we were the most at sea, which was absolutely essential for the project reaching its current state. Besides also being a compassionate supervisor and mentor, she was inspirational in how she seemingly handled twenty things at once, yet still planned with crystal clear vision. Once she even carved out time while taking care of her children at home to hop on a meeting and discuss our project progress. I also took Muriel's class, a seminar on Topics in Information Theory, which exposed me to an entire field that I had barely touched before the MEng. The discussions that Muriel facilitated for the various papers we read were insightful and stimulating, especially since I was among fellow classmates and paper authors who were at the cutting edge of the field. I'll forever be grateful to Muriel for developing the information theoretic flair that will be a part of my analytic thinking in all my future endeavors.

I would also like to thank Homa Esfahanizadeh, my advisor and close confident during my MEng project. From the first day we talked about our project ideas until the very last (even as I write this section of the thesis 2 days before it is due), she has always been there as an anchor, a sail, and a compass, giving me the strength and direction to achieve what I had thought was merely an interesting yet formidable undertaking at first. I have bothered her with countless questions during our meetings, and we have revised our path many times to the promising direction that it is now. With her intelligent guidance, I've finally arrived at the other end of the journey, surprised by how far this project has come, yet also humbled by how much further we can advance.

I also want to thank my friends, who were all there to help during difficult times, whether that be with technical problems like the absolutely nasty Pytorch bug that I had been stuck on for a long time (shoutout Wonjune Kang and Salil Desai), or with helping me even get food to my apartment after I had surgery for my torn achilles (shoutout Justin Restivo, Anisha Agarwal, Eli Kramer, Dan Sun, and my roommates Liam Conboy, Rinik Kumar, Paolo Adajar, and Matt Feng, among many others!).

Finally, words cannot describe how grateful I am to my parents, who have always been there to have my back, as well as giving me the guidance and motivation that I needed when things were tough. My dad even flew to Boston and lived in my apartment for a few weeks while my post-surgery pain and inconveniences were at their peak. I am sincerely grateful for everything that they've done for me and for raising me to be equipped to tackle the challenges of life head on.

This thesis is a culmination of all the work that I've done in the past year with the immense help from those around me. I hope you enjoy reading the thesis and are also equally inspired by its promising results and the encouraging future directions that it reveals.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contemporary machine learning efforts in various sensitive domains like healthcare face a major bottleneck due to the shortage of publicly available training data [17]. Release and acquisition of sensitive medical data containing identifiable information is currently federally prohibited by laws such as HIPAA [6] and GDPR [9]. However, without the ability to train on large amounts of data, predictive models are not able to confidently capture enough labeling information to accurately perform intended classification tasks. This motivates us to approach the issue from an information theoretic perspective. In this thesis, we investigate an encoding scheme that allows data-owners to publicly release their encoded data with labels such that a model can be trained for downstream predictive tasks. This could lead to the widespread release of larger and more diverse medical datasets that simultaneously limit the use of data for undesired applications.

1.1 Current Techniques

The state-of-the-art solutions to tackle this privacy-utility tradeoff primarily involve allowing data-owners to share their encrypted data [4][7][16] or their noisy data [32][19]. However, contemporary encryption practices are computationally inefficient and ensure that an adversary learns nothing about the original data with very high probability. This is not necessary for our purposes. One can see how in a chest x-ray

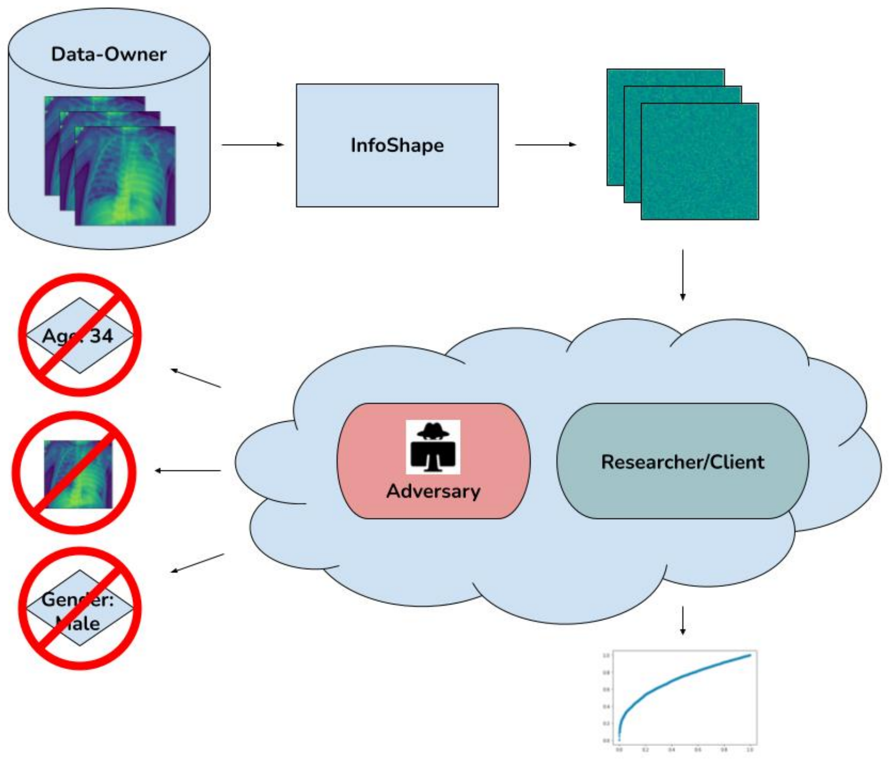


Figure 1-1: The data-owner tries to share its sensitive data that is encoded using *InfoShape* with the researcher/client. The encoded data must reveal as little information as possible to the adversary besides what is needed for the researcher/client's ML task.

dataset, there is no need to hide the fact that the images are actually chest x-rays, or that a chest x-ray image has 24 ribs in it (most humans have this many). In addition, only limited operations can be performed on encrypted data. This implies that downstream models must modify their architectures in order to train on encrypted data, which could negatively affect accuracy. On the other hand, using the approach of training on noisy data with methods like differential privacy lead to notable utility costs.

1.2 Problem Statement

Our problem setting is as follows: we would like a data-owner of sensitive information to be able to encode their data such that any malicious adversary would learn as little information as possible beyond what is necessary to conduct proper inference in downstream tasks. An example would be a hospital that would like to release encoded patient chest x-rays to AI researchers, who wish to train models for pneumonia detection. From the perspective of the data-owner, once it sends its encoded data, any adversary (including potentially the researcher/client *themselves*!) has access, and, combined with other publicly available data, must not be able to learn anything beyond what is necessary for the downstream classification task. Potentially sensitive attributes that must remain hidden include age, gender, or race. We can see a diagram of this example in Figure 1-1.

In other words, we would like an approach that aims to achieve both privacy guarantees as well as the preservation of utility for downstream machine learning tasks. For privacy, our goal is to limit the amount of information that is revealed about each sample beyond its labels. For utility, there could be different goals depending on the task at hand. We focus on the goal of being able to train a classifier on the transformed data with the original labels, such that it performs close to the baseline model that was trained on the original, un-encoded data. In practice, these two design goals are competing. Therefore, providing a solution that offers a meaningful and tune-able trade-off between these two goals has significant practical importance.

1.3 Our Contribution: InfoShape

We propose a novel dual optimization mechanism, dubbed *InfoShape*, to simultaneously preserve privacy while also maintaining utility on downstream classification tasks. We choose the name of *InfoShape* since our scheme trains a neural network encoder to act as a task-specific lossy compressor, by keeping as much relevant information as possible for our intended downstream task while "shaping" the data to achieve a private representation. This is similar in concept to the idea of network functional compression with distortion in [14], where they discuss how to compress input data such that a function on the compressed data can be computed at the receiving end.

Mutual information is chosen as our primary metric to quantify the privacy and utility performance. Treating the combined privacy and utility measures as an objective function allows us to train our encoder through traditional machine learning methods. The importance of mutual information on bounding the privacy leakage has already been investigated in previous work [20]. However, due to the computational complexity of estimating mutual information for a high-dimensional dataset in the image domain, using this critical metric has seen limited use. To circumvent this challenge, recent works have used neural networks to estimate variational bounds on mutual information [2][25][30].

We utilize the potential of neural estimation of mutual information to numerically identify bounds on mutual information between original data and encoded data (as a measure of privacy) and between labels and encoded data (as a measure of utility) for a given neural network that encodes the data. We then propose to combine these two measures as competing loss values into a single loss metric for training the encoder. Once the encoder is trained, it can be utilized by individual data owners as a task-based lossy compressor to encode their data and allow the release of data for collaborative learning. Our overall scheme is illustrated in Figure 1-2.

Through experiments performed on a synthetic Gaussian dataset in Section 4.2, we show that *InfoShape* successfully trains an encoder to simultaneously optimize privacy and utility measures. Our empirical results and challenges that we faced provide a rich set of paths for future work. We are hopeful that given better mutual information estimators and careful encoder architecture construction, this work can be extended into the image domain and be potentially used in encoding real, sensitive medical images.

1.4 Thesis Overview

The rest of the thesis will be organized as follows: Chapter 2 will cover some of the background of information theory concepts and related work that contribute to the initial idea of *InfoShape*. Chapter 3 gives an overview of the landscape of mutual information estimation, with a particular focus on Mutual Information Neural

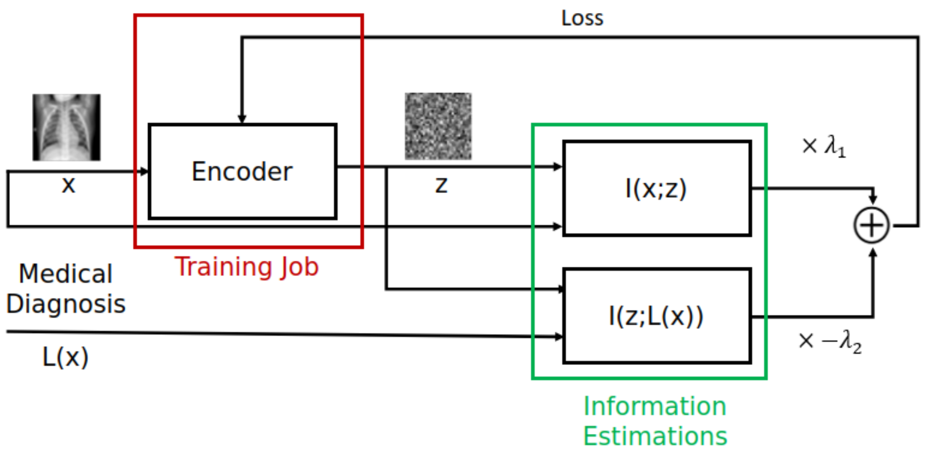


Figure 1-2: General *InfoShape* blueprint for an example application in healthcare. X represents our inputs, which could be chest x-ray images in this case. Z represents the latent representations outputted by our encoder. The encoder can be thought of as a lossy compressor that hides enough information for privacy guarantees, while still keeping enough information for the final medical diagnosis, $L(x)$. The Information Estimations calculate our measure of privacy leakage ($I(x; z)$) and utility score $I(z; L(x))$, potentially via variational estimation methods. Both measures are then combined in a Lagrangian optimization as the encoder's loss function. The Training Job modifies the encoder's weights using back propagation, and the cycle repeats.

Estimation (MINE), which was a core component of our scheme. Chapter 4 ties everything together into the main section on *InfoShape* and our promising results demonstrating *InfoShape's* ability to learn private and useful data representations. Chapter 5 concludes the thesis and suggests areas of future work.

Chapter 2

Background

In this section, we will explain some of the necessary background related to information theory concepts as well as related work. Our work centers around improving upon existing work related to data privacy and information estimation in order to obtain a complete framework for optimizing the privacy-utility tradeoff.

2.1 Information Theory

To begin, we first review the idea of **entropy**, which measures the amount of "uncertainty" or "surprise" present in a probability distribution. The idea was first introduced by Claude Shannon in his 1948 paper "A Mathematical Theory of Communication," where he showed that entropy provided a theoretical bound on how well certain data could be losslessly compressed in a perfectly noiseless communication channel [27]. Given a discrete random variable X that takes values from the alphabet $\mathcal{X}$ with probability function $P : \mathcal{X} \to [0, 1]$, entropy is given by the following formula:

$$H(X) = -\sum_{x \in \mathcal{X}} P(x) \log P(x) \tag{2.1}$$

Common units that are used for entropy include bits and nats, which indicate the usage of either $\log_2$ or ln in Equation 2.1. As a concrete example, if X has an outcome space of all the integers from 0 to 7, each occurring with uniform probability, then we can efficiently and uniquely express any outcome as a binary number with 3 bits. Hence, $H(X) = \log_2(1/8) = 3$ bits. However, if only the integers from 0 to 3 inclusive have a non-zero and equal probability of occurring, then the entropy drops down to 2 bits since the entire outcome space can be described using just those 2 bits of information.

Similarly, we can define joint entropy, the "uncertainty" for the outcomes of two random variables X and Y, by the following formula:

$$H(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P(x,y)\log P(x,y) \tag{2.2}$$

Conditional entropy, the "uncertainty" for the outcome of X *given* Y, is defined as follows:

$$H(X|Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P(x,y)\log P(x|y) \tag{2.3}$$

Armed with these measures of entropy, we can now define the quantity of **mutual information** between X and Y as follows:

$$I(X;Y) = H(X) - H(X|Y) \tag{2.4}$$

Mutual information can be interpreted as the reduction in uncertainty of X given Y. The idea for mutual information was originally derived in Shannon's seminal 1948 paper. Some basic properties:

$$\begin{aligned} I(X;Y) &= I(Y;X) \\ &= H(X) - H(X|Y) = H(Y) - H(Y|X) \\ &= H(X) + H(Y) - H(X,Y) \\ &= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P(x,y)\log\frac{P(x,y)}{P(x)P(y)} \end{aligned} \tag{2.5}$$

Another useful tool in information theory is the **Kullback-Leibler (KL) divergence**, which measures the "statistical distance" between two probability distributions. The KL-Divergence between two discrete probability distributions, P and Q, over the same probability space $\mathcal{X}$, is defined as:

$$D_{KL}(P||Q) := \sum_{x \in \mathcal{X}} P(x) \log \frac{P(x)}{Q(x)} \tag{2.6}$$

KL Divergence is also sometimes called relative entropy, which we can understand with the interpretation that $D_{KL}(P||Q)$ is the excess surprise from using Q as a model for some data when the underlying distribution is actually P [18].

This allows us to define mutual information alternatively by interpreting Equation 2.5 as the KL-divergence between the joint distribution between X and Y, P_{XY}, and the product of the marginals, $P_X \otimes P_Y$:

$$I(X;Y) := D_{KL}(P_{XY}||P_X \otimes P_Y) \tag{2.7}$$

Intuitively, the smaller the divergence between the joint and the product of the marginals, the lower dependence X and Y must have. In fact, mutual information is 0 precisely when X and Y are independent from each other. This definition is used in [2] for their neural MI estimation, which we in turn utilize extensively in *InfoShape*.

As an important aside, computing the mutual information for continuous or high-dimensional data is nearly impossible without previous information about the data. In particular, for most real world problems, the marginal, $P(X)$ from Equation 2.4 is intractable. As an example, for a machine learning problem on image classification, it is impossible to know the input distribution of all possible images, as this would either require an astronomical computation effort or would not even make sense to compute. Therefore, direct computation of MI is usually impossible.

However, even contemporary variational estimation methods [25] require large amounts of data to provide reliable estimates and several MI variational bounds require prerequisite knowledge of certain distributions. This motivates our search for an appropriate MI estimator that has the numerical properties that would allow us to use it in *InfoShape*.

2.2 Related Work

In this section we explore the literature in the areas of private data release and the estimation of information. For each approach in the private data release section, we list at least one reason why the approach would not address the goals in our problem statement in Section 1.2. This section then motivates our discussion on the estimation of information, in which we use previous works to illustrate why mutual information is the most logical statistic to use and how existing neural estimation methods can achieve this more reliably than conventional means.

2.2.1 Private Data Release

Anonymization and Perturbation

One of the most basic methods that attempts to preserve patient data privacy for public release is anonymization. Recent work by the authors of [24], however, show just how vulnerable anonymization efforts are to the power of deep learning-based re-identification algorithms. They claim to achieve 95.55% accuracy and an AUC of 0.9940 for identifying whether two frontal chest X-ray images are from the same person. Furthermore, they note that the model can even reveal the same person ten or more years after the initial scan. By cross-referencing other images or leaked sensitive information about an individual, this re-identification could prove detrimental to a patient's privacy.

Additionally, rudimentary attempts of image obfuscation like blurring or pixelating also do not protect a patient's privacy. Image de-blurring and the use of deep learning methods to recreate the original image is a well studied topic [15][23]. Clearly, we would like to approach our own goals from a direction that avoids these simple, outdated obfuscation methods.

Differential Privacy

The field of differential privacy centers around "the paradox of learning nothing about an individual while learning useful information about a population" [13] [10]. Differential privacy itself is just a definition and not any specific algorithm. In Chapter 1 of [13], we can see how differential privacy is defined through an example: a study is performed on a differentially private (DP) medical database that demonstrates that smoking causes lung cancer. Regardless of whether an individual smoker opted in or out of the medical database, their privacy would *not* be leaked since the study's conclusions would have been the same in either situation. To generalize, any series of responses to queries to the DP database would be "essentially" equally likely to occur, independent of whether an individual's data is in the database or not. The parameter ϵ controls the degree of privacy guaranteed. Smaller ϵ will lead to stronger privacy guarantees and thus more statistically noisy and less accurate query responses. One classic approach to achieve ϵ-differential privacy is to add Laplace noise to the inputs [13].

Unfortunately, differential privacy would not achieve our stated goals, because in this view, "acceptable" privacy would come at the cost of utility loss. We recognize that utility and privacy fundamentally offer a tradeoff; however, we are more keen on identifying encoded data representations that capture enough information to preserve utility while also achieving privacy guarantees. Even if we were to adjust ϵ or the differential privacy algorithm, this statistical approach would not fit into our problem statement mold.

Encryption Methods

As discussed in Section 1.1, encryption methods are designed to let an adversary learn nothing about the original unencrypted inputs. Encryption algorithms use computational assumptions, such as the integer factorization or discrete log problems, as a basis for methods that operate under various different cryptographic attack models, such as Chosen Plaintext Attack (CPA) or Chosen Ciphertext Attack (CCA) (both of which concern the abilities of a potential adversary), as a foundation to achieve privacy [5]. Modern cryptographic methods like fully homomorphic encryption or functional encryption [16][3] allow performing operations purely using encrypted data, such that the same results would be obtained if that operation were performed on the unencrypted data. These techniques only allow certain types of computations and are computationally expensive, however, which deters their use.

Furthermore, encryption techniques do not offer the granularity of privacy that we seek. As an example, we would not be looking to hide the fact that an image used to be chest x-ray, since we would still like to publicly release labels. These labels would already implicitly inform an adversary about the nature of the data anyway. Instead, we would like to encode the image such that certain privacy guarantees related to sensitive information would be upheld. We believe that this "tunable" and specific degree of privacy cannot be achieved with encryption alone.

2.2.2 Estimation of Information

In this section, we first discuss the Information Bottleneck, which is a highly related area of work that motivated our approach. Next, we discuss why mutual information is difficult to estimate for real-world datasets and what existing mutual information bounds are covered in the literature. Finally, we explore the work of [2], where they use neural networks to estimate mutual information. MINE serves as a core component in our final overall scheme.

Information Bottleneck

The information bottleneck (IB) problem aims to find a minimal representation of an input signal X, given a joint probability distribution between X and Y, which is another observed signal (this would be the target outputs in machine learning applications). The crux of the IB method is to find the best tradeoff between accuracy and compression. When viewed from an information theory lens, we let T be the compressed representation of X, which lends us the classic IB Lagrangian optimization:

$$\min_{p(t|x)}[I(X;T) - \beta I(T;Y)] \tag{2.8}$$

where β controls the tradeoff between compression and information preservation of X. [1]

This is quite similar to our dual optimization objective for *InfoShape* (see Equation 4.1), where we wish to find a suitable tradeoff between privacy and utility. We draw motivation from the theoretical discussions in [21], where the authors discuss the connection between the information bottleneck and private data transformation, via a Privacy Funnel optimization. Furthermore, one can view privacy from the lens of compression (i.e., with compressed sensing [11], where only a small subset of the input signal would need to be preserved for "faithful recovery"). Our work extends on these ideas by first identifying a concrete MI estimation algorithm, MINE, and then training a neural network acting as a lossy encoder that tries to find an optimal representation for our privacy and utility objectives.

Mutual Information Bounds

The authors of [25] provide a comprehensive review of various available MI estimation bounds, which are summarized in Figure 2-1. Our intended applications of real-world machine learning problems in fields like healthcare usually only have samples of x, y available, but with probability densities that are unknown (such as $p(y|x)$, which is usually what classification aims to predict). Mutual information itself is difficult to compute with finite datasets of this nature. As a result, we are limited to the tractable variational lower bounds that approximate MI: I_{NWJ}, I_{TUBA}, I_{NCE}, I_{α}, and I_{MINE}. One can examine Section 2.2 and 2.3 in [25] to see how each of these bounds was derived. The authors claim that I_{TUBA}, I_{NWJ}, and I_{MINE} all "provide tractable estimators which become tight with the optimal critic," which led us to settle on choosing MINE as an MI estimator, despite the warning that they also "exhibit high variance due to their reliance on high variance upper bounds on the log partition

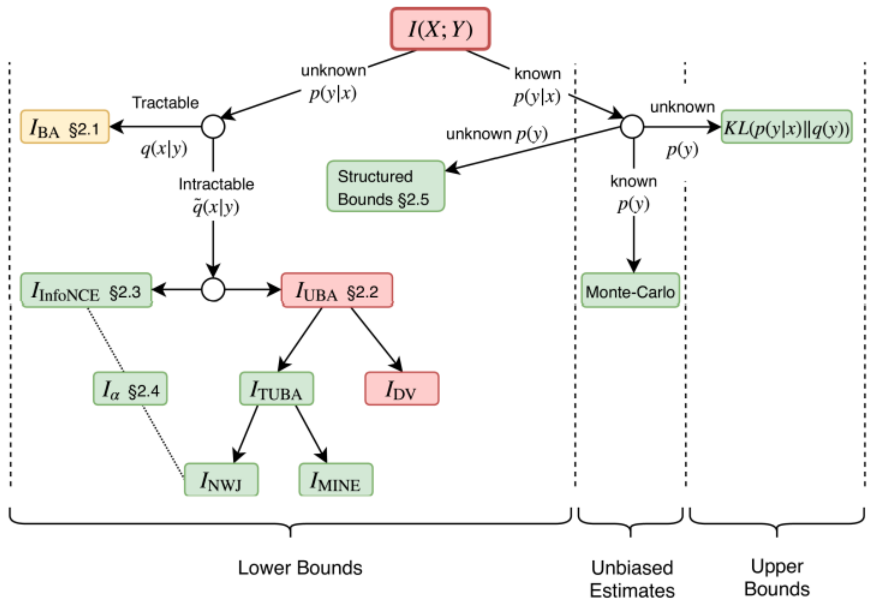


Figure 2-1: Figure 1 from [25] outlining the variational MI bounds presented in their paper. Green nodes can be used for both optimization and estimation, yellow nodes for optimization but not estimation, and red for neither. Arrows indicate approximations or assumptions building off of previous parent bounds. Since we are working with unknown $p(y|x)$ and since we would like to compute MI estimates, we mainly focus on the green nodes in the bottom left.

function.” The other bounds, I_{NCE} and I_α, had upper bounds on their MI estimates of $\log K$ and $\log \frac{K}{\alpha}$, respectively, where K is the number of samples per batch and α is a tunable parameter. We felt that using these could potentially negatively impact our estimation accuracy. The upper bound of I_α, $\log \frac{K}{\alpha}$, is tunable, though, which makes it a potential candidate for future exploration.

Mutual Information Neural Estimation (MINE)

As we have already seen in the previous section, mutual information is a difficult statistic to estimate, despite there being various bounds available if certain conditions about the data and experiment are met. Here, we briefly discuss the background for MINE, an approach that utilizes a neural network to estimate mutual information. MINE uses a deep neural network the authors call a “statistics network” to estimate MI. They use the formula from Equation 2.4 and the Donsker-Varadhan representation of KL-divergence [12] to obtain what they call the “neural information measure”:

$$I_\theta(X;Y) = D(P_{XY}||P_X \otimes P_Y) \geq \sup_{\theta \in \Theta} \mathbb{E}_{P_{XY}}[T_\theta] - \ln(\mathbb{E}_{P_X \otimes P_Y}[e^{T_\theta}]) \tag{2.9}$$

The function T_θ is parameterized by the statistics network with parameters θ, and the final neural information measure provides a lower bound on $I(X;Y)$. Using standard backpropagation and gradient ascent, this value of T_θ (in theory) converges over time to give a tight lower bound for MI [2]. More details, including the MINE algorithm, the challenges we experienced, and how we used MINE in MNIST and Gaussian datasets, are explored in Section 3.4.

2.3 Datasets Used

We initially started our work using a Kaggle dataset of chest x-ray images for pneumonia detection. Due to data imbalance and the complexity of images, we decided to switch to the classic MNIST digits dataset for our later MINE experiments. While the results from MNIST seemed promising at times, it became clear that working in the image domain led to various numerical stability issues when using MINE. Choi et al in [8] also faced similar problems and suggested improvements to MINE that would address the stability issues. However, their experiments were done on a simple, synthetic Gaussian dataset originally used by [2]. When we used their ideas on MNIST, the previous problems still persisted, albeit with milder severity. As a result, we used synthetic Gaussian data for our final experiments. Please see Section 3.1.1 for more details.

Chapter 3

Estimating Mutual Information

In this chapter, we focus on the core component of the *InfoShape* pipeline, the mutual information estimation algorithm. We first outline the setup for these experiments and discuss why we even choose mutual information as our primary statistic. Then, we cover the theory of MINE and outline the algorithm from their paper. Finally, we present the estimation results on both MNIST and Gaussian datasets.

3.1 Experiment Setup

Each of our experiments required both our dataset as well as an encoder to transform the data. We borrowed much of the MINE code from this MINE Github, in which the author replicated many of the results from the original MINE paper. This led us to stick with the author's choice of using Pytorch Lightning as the primary tool for implementation.

3.1.1 Dataset Setup

Our MINE experiments were primarily performed on the MNIST digits dataset and our synthetic Gaussian dataset, both of which were introduced in Section 2.3. For both datasets, we used 60,000 data samples. Importantly, there was no standard notion of a train and test split for the mutual information estimation experiments

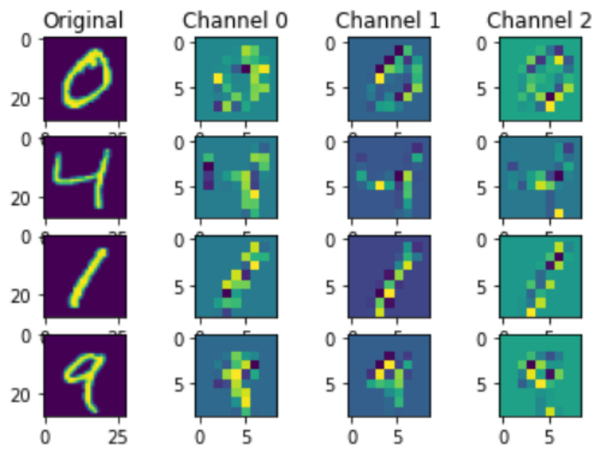


Figure 3-1: Three output channels after encoding an MNIST image using the convolutional encoder from Table 3.1 with randomly initialized weights. Most spatial information can still be observed.

since MINE is essentially an optimization problem that maximizes a lower bound of MI via random sampling. Once its lower bound is optimized, that final MI measure is our result. Furthermore, for our datasets that we passed into the MINE algorithm, we set the flag shuffle =True inside of the Pytorch DataLoader to achieve the random sampling required for the MINE algorithm (see Algorithm 1).

For the synthetic Gaussian dataset, we generated 30,000 samples each from two multivariate Gaussian distributions of dimension $D = 20$ and with two different covariance matrices. Both had variances of 1 down the main diagonal. However, for the first multivariate Gaussian, there was a constant value, $\rho_0 = -0.99$ on the two diagonals starting from $(0, \lfloor D/2 \rfloor)$ and $(\lfloor D/2 \rfloor, 0)$, with zeros everywhere else. The second was identical except it had $\rho_1 = 0.99$. Below, we illustrate what the covariance matrix would look like if the multivariate Gaussian dimensions were 4 instead of 20:

$$\begin{bmatrix} 1 & 0 & \rho & 0 \\ 0 & 1 & 0 & \rho \\ \rho & 0 & 1 & 0 \\ 0 & \rho & 0 & 1 \end{bmatrix}$$

3.1.2 Encoder Architecture

For our experiments on the MNIST data, we first tried a convolutional encoder architecture, shown in Table 3.1. Our choice of using a non-overlapping stride for the convolutional kernels was motivated by the encoder architecture from Figure 2 in [32]. After some analysis, we decided to transition to using a simple, single-layer nonlinear dense encoder instead, shown in Table 3.2. A dense encoder was able to operate on the pixel level, whereas the convolutional encoder with its non-overlapping stride would produce latent representations that still kept some spatial information (see Figure 3-1).

For the synthetic Gaussian dataset, we used a dense encoder with fewer output nodes, outlined in Table 3.3. By having fewer output nodes than total image pixels or multivariate Gaussian components, we wished to architecturally induce all of our encoders to act as lossy compressors. We paired each of these three encoders with a compatible statistics network to calculate our MINE estimates: we used Table 3.4 for the MNIST convolutional encoder, Table 3.5 for the MNIST dense encoder, and Table 3.6 for the Gaussian dense encoder.

Layer	Number of Outputs	Kernel Size	Stride	Activation Function
Input $\mathbf{x}$	28×28			
Convolution	$9 \times 9 \times 3$	3×3	3	ReLU

Table 3.1: Convolutional encoder architecture used for MNIST data. The use of non-overlapping stride was motivated by the encoder family of functions from [32].

Layer	Number of Outputs	Activation Function
Input $\mathbf{x}$	28×28	-
Flatten		
Dense	100	ReLU

Table 3.2: Dense encoder architecture used for MNIST data.

Layer	Number of Outputs	Activation Function
Input $\mathbf{x}$	20	-
Dense	10	Tanh
Dense	3	Tanh

Table 3.3: Dense encoder architecture used for synthetic Gaussian data. An extra dense layer was added for more nonlinearity in the data.

3.2 Correlation vs Mutual Information

Before diving into MINE, we first studied the effects of using a simpler statistic like correlation instead of mutual information for our *InfoShape* training pipeline. Our experiment treated correlation as a metric for privacy leakage. This circumvented the need for a complex algorithm like MINE. We trained the encoder from Table 3.2 with $-1\times$ the correlation between the original image and the encoder output as our loss for 25 epochs with learning rate $1e$-3. We used -1 to maximize the correlation. As can be seen in Figure 3-2, on Epoch 0, our encoder completely scrambled the original image with its randomly initialized weights. As the encoder trained, it approximated the behavior of an identity function.

This should not be surprising since, from a theory point of view, correlation can only measure the strength of the linear relationship between the pixels from our two images. Therefore from this experiment, we can conclude that utilizing correlation in our objective function would not induce a private encoded representation of data, as having a simple, linear metric like correlation will just induce an encoder towards the identity function. This motivates us to seek mutual information as a metric instead due to its ability to measure non-linear relationships. Furthermore, we also draw inspiration from [21], where they provide a proof for the notion that "the inference threat under any bounded cost function can be upper bounded by an explicit function of the mutual information between private data and disclosed data." Their concept of private data is precisely our original input data, and the disclosed data would be our encoder outputted latent representations. With this theoretical backing for mutual

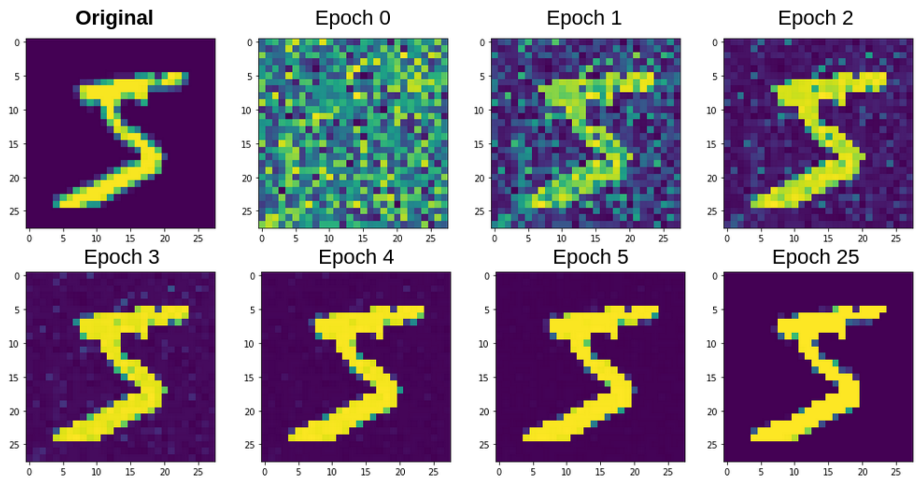


Figure 3-2: Encoder latent representations between training epochs. The top left image shows the original MNIST digit, denoted X, and the progression of encoder latent representations, denoted Z, clearly shows that the encoder learns the identity function $(T(x) = x)$. We use the Pearson product-moment correlation coefficient matrix from torch.corrcoef calculated for an input matrix with X and Z flattened and stacked on top of each other as rows (ie. a matrix of shape $2 \times$ (IMG_WIDTH $\times$ IMG_HEIGHT).) The actual loss value is $R_{0,1}$, the correlation value itself from the correlation matrix, R.

information as motivation, we now turn to the estimation of mutual information.

3.3 Mutual Information Neural Estimation (MINE) Formulation

As discussed previously in Section 2.1 and Section 2.2.2, directly computing MI is essentially impossible if we are only provided with limited data samples with no knowledge of underlying distributions. As a result, we turn to variational estimation, which will give us a bound on the true MI. To derive the estimation formula for MINE's variational method, we start with Equation Section 2.7, repeated here:

$$I(X;Y) := D_{KL}(P_{XY}||P_X \otimes P_Y)$$

The authors of MINE utilize the Donsker-Varadhan representation of KL-Divergence, which is shown below for reference:

$$D_{KL}(P||Q) = \sup_{T:\Omega\to\mathbb{R}} \mathbb{E}_P[T] - \log \mathbb{E}_Q[e^T] \tag{3.1}$$

where the supremum is taken over all functions T such that the two expectations are finite. This lends itself nicely to an optimization problem to estimate MI.

To establish MINE's objective function, the authors define $I_\Theta(X;Z)$ as the *neural information measure*, which acts as a lower bound of $I(X;Z)$.

$$I_\Theta(X;Z) := \sup_{\theta\in\Theta} \mathbb{E}_{P_{XZ}}[T_\Theta] - \log \mathbb{E}_{P_X \otimes P_Z}[e^{T_\theta}] \tag{3.2}$$

The expectations in Equation 3.2 should be estimated using i.i.d empirical samples from P_{XZ} and $P_X \otimes P_Z$. The marginal samples $\hat{x} \sim \mathbb{P}_X$ and $\hat{z} \sim \mathbb{P}_Z$ can be obtained by dropping whichever other value from the joint samples, ie $(\hat{x}, z)$ and $(x, \hat{z}) \sim \mathbb{P}_{XZ}$.

Now we are ready to present the MINE algorithm (1), as shown in Section (3.1) of the MINE paper [2].

Algorithm 1 MINE

```
$\theta \leftarrow$ initialize network parameters
**repeat**
    Draw $b$ minibatch samples from the joint distribution:
    $(x^{(1)}, z^{(1)}), \ldots, (x^{(b)}, z^{(b)}) \sim \mathbb{P}_{XZ}$
    Draw $b$ samples from the $Z$ marginal distribution:
    $\bar{z}^{(1)}, \ldots, \bar{z}^{(b)} \sim \mathbb{P}_Z$
    Evaluate the lower bound:
    $\mathcal{V}(\theta) \leftarrow \frac{1}{b}\sum_{i=1}^{b} T_\theta(x^{(i)}, z^{(i)}) - \log\left(\frac{1}{b}\sum_{i=1}^{b} e^{T_\theta(x^{(i)}, \bar{z}^{(i)})}\right)$
    Evaluate bias corrected gradients (e.g. moving average):
    $\hat{G}(\theta) \leftarrow \tilde{\nabla}_\theta \mathcal{V}_\theta$
    Update the statistic network parameters:
    $\theta \leftarrow \theta + \hat{G}(\theta)$
**until** convergence
```

The authors show that the SGD gradients for Algorithm 1 exhibit high bias, meaning that the final estimates of MI fall short of the true MI. We can see that the gradient for the samples of minibatch B, denoted $\hat{G}_B(\theta)$, is:

$$\begin{aligned}\hat{G}_B(\theta) &:= \nabla_\theta \mathcal{V}(\theta) \\ &= \mathbb{E}_B[\nabla_\theta T_\theta] - \frac{\mathbb{E}_B[\nabla_\theta T_\theta e^{T_\theta}]}{\mathbb{E}_B[e^{T_\theta}]}\end{aligned} \tag{3.3}$$

This follows from applying the chain rule and taking expectations over minibatch B. In order to correct for the bias, the authors replaced the denominator of the second term in Equation 3.3 with an exponential moving average across iterations of the MINE training loop, which improved empirical performance.

3.4 MINE Implementation and Extensions

3.4.1 MINE Architecture

We show our implementation of both the convolutional (3.4) and dense (3.5) statistics networks for MINE used to calculate the privacy leakage measure when working with the MNIST data. For the utility score, we used a separate statistics network with the same architecture except with different input dimensions to calculate $I(Z; L(X))$. Both convolutional and dense architectures drew inspiration from [2] in their "GAN + MINE: Stacked-MNIST" and "Information Bottleneck with MINE" experiments.

Our synthetic Gaussian data was much simpler than even MNIST images, since it was 20-dimensional data, with only 10 different components of the multivariate Gaussian having non-zero covariance, whereas MNIST images depicted 28×28-pixel handwritten digits of different styles. For this reason, we simplified our statistics network for MINE to the architecture in Table 3.6. The rest of the MINE implementation can be seen in Our Github, where we use Pytorch Lightning to implement the training loop of MINE.

Layer	Num. Outputs	Kernel	Stride	Padding	Activation Fn
Input $\mathbf{z}$	$9 \times 9 \times 3$	-	-	-	-
Convolution	$5 \times 5 \times 16$	5×5	2	2	ReLU
Convolution	$3 \times 3 \times 32$	5×5	2	2	ReLU
Convolution	$2 \times 2 \times 64$	5×5	2	2	ReLU
Flatten	-	-	-	-	-
Input $\mathbf{x}$	28×28	-	-	-	-
Convolution	$14 \times 14 \times 16$	5×5	2	2	ReLU
Convolution	$7 \times 7 \times 32$	5×5	2	2	ReLU
Convolution	$4 \times 4 \times 64$	5×5	2	2	ReLU
Flatten	-	-	-	-	-
Hidden Flattened ($\mathbf{x}$,$\mathbf{z}$)	1280	-	-	-	-
Dense	100	-	-	-	ReLU
Dense	100	-	-	-	ReLU
Dense	1	-	-	-	-

Table 3.4: MINE convolutional statistics network architecture for MNIST data, used for the privacy leakage measure. The output values are used to calculate an estimated lower bound for $I(X; Z)$, where X is the original data and Z is the encoded data. Inputs z and x each undergo three convolutional layers, and then are flattened and concatenated. The concatenated result then is passed through three dense layers to output a final estimate.

Layer	Number of Outputs	Activation Function
Input $\mathbf{x}$, $\mathbf{z}$		
Gaussian Noise(std=0.3)	-	-
Dense Layer	512	ELU
Gaussian Noise(std=0.5)	-	-
Dense Layer	512	ELU
Gaussian Noise(std=0.5)	-	-
Dense Layer	1	None

Table 3.5: MINE dense statistics network architecture for MNIST data, used for the privacy leakage measure. The output values are used to calculate an estimated lower bound for $I(X; Z)$. This architecture is exactly the same as the architecture used for the "Information Bottleneck with MINE" experiment performed in Belghazi et al. [2].

Layer	Number of Outputs	Activation Function
Input **x, z**		
Dense Layer	100	ReLU
Dense Layer	100	ReLU
Dense Layer	1	None

Table 3.6: MINE statistics network architecture for the Gaussian data, used for the privacy leakage measure. The output values are used to calculate an estimated lower bound for $I(X;Z)$. Since the Gaussian data itself was much simpler in nature compared to the MNIST images, we chose to use a simpler statistics network.

3.4.2 Convergence, Numerical Stability, and Variance Challenges

Our initial estimation results from MINE were erratic, as can be seen in Figure 3-3. In particular, we observed three major challenges:

1. **Convergence** - MINE sometimes did not successfully converge to a non-zero value when using convolutional statistics network architectures like those from Section 8.1.3 from the appendix in [2].

2. **Numerical Stability** - We experienced exploding gradients (usually after many iterations of MINE), which led to estimation values of NaN.

3. **High Variance** - With lower mini-batch sizes, the MI graphs across epochs exhibited high variance and would appear extremely jumpy. We were only able to obtain meaningful results when applying an aggressive smoothing factor.

For MNIST data, when we switched from the convolutional MINE and statistics network architectures to their dense counterparts, we observed better convergence results, perhaps due to the encoded outputs having minimal spatial information that convolutional layers could exploit. With dense architectures, both the encoder and the MINE statistics networks could operate at the granularity of the pixel-level, which we believe was at least one factor in removing our convergence issues.

As for numerical stability, this was an issue that plagued us until we switched from MNIST to our synthetic Gaussian data. We believe that the sheer complexity of the image domain and the limitations of either design choices or estimation abilities of MINE indicate that operating strictly on Gaussian data is necessary until structural improvements can be made. We also observe moderate improvements when modifying our underlying estimation algorithm from MINE to Regularized MINE (ReMINE), which we will discuss in the next section. Please also see Section 4.2 for our results on Gaussian data.

Finally, high variance was particularly challenging to tackle, since there was inherently a trade-off between bias and variance, as discussed in Section 2.2.2. However, by providing a larger mini-batch size to to each iteration of MINE, we were able to reduce the variance dramatically. We will also see how combining this optimization with ReMINE on Gaussian data mostly resolved the difficulties that we faced.

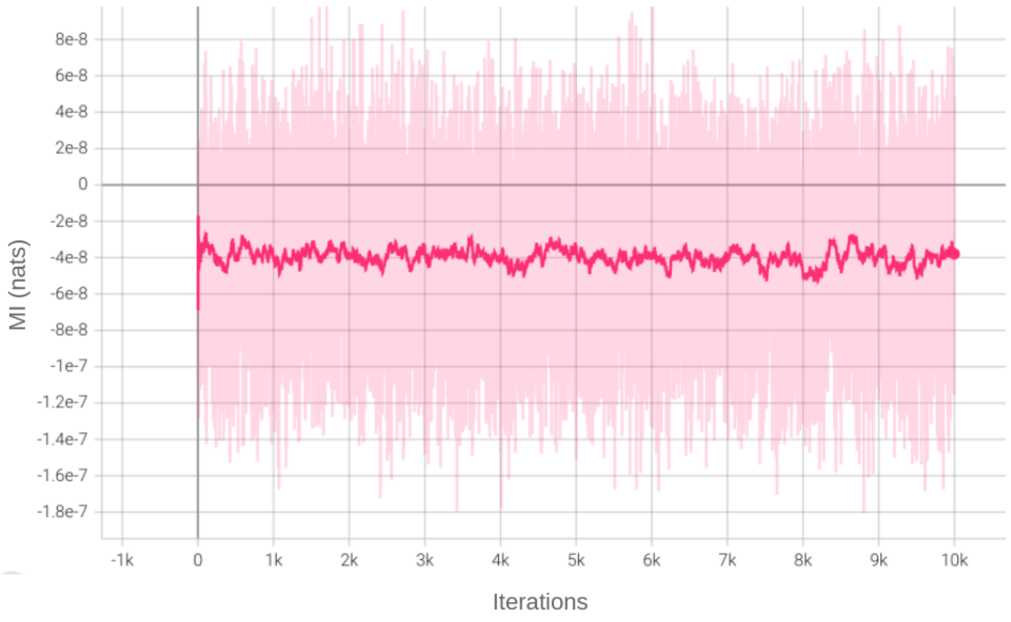
3.4.3 Regularized MINE (ReMINE) for Better Numerical Stability

Due to the various challenges that we faced in Section 3.4.2, we turned to the literature for assistance. Regularized MINE (ReMINE) [8] proved to be extremely helpful. The authors observed very similar issues with MINE as we did: in particular, they described a "drifting phenomenon" where the estimates of $\mathbb{E}_{P_{XY}}[T]$ and $\log\left(\mathbb{E}_{P_X \otimes P_Y}[e^T]\right)$ would drift in parallel, *even after* MI converged. They also observed exploding statistics network outputs with smaller batch sizes, consistent with our observations. This was backed by the literature surrounding the batch size limitation problem [22] [29], which would require for MINE "a batch size proportional to the exponential of true MI to control the variance of the estimation" [8].

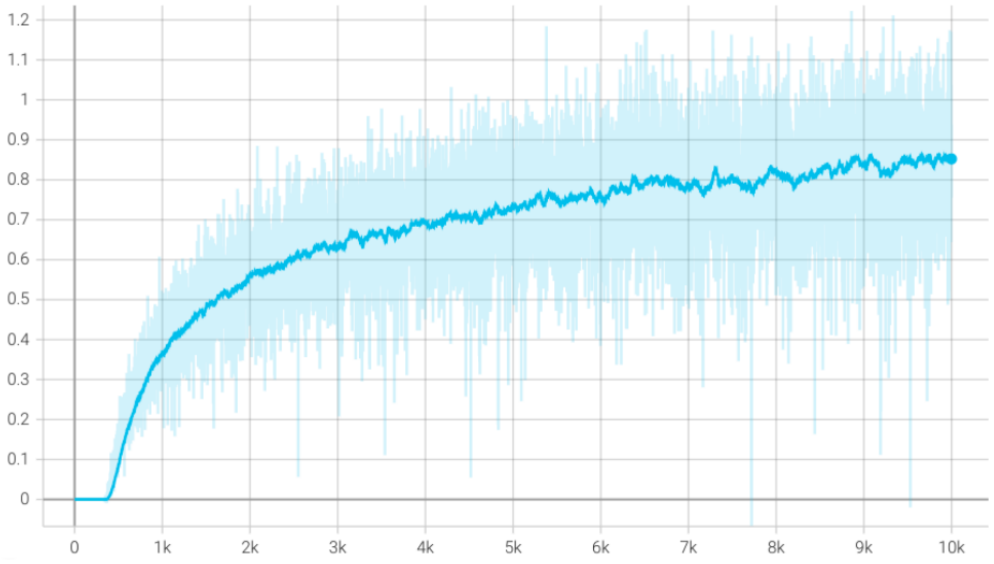
While we omit the theoretical backing and proofs from [8], we will provide the ReMINE loss function from their Theorem 5 as reference: *Let d be a distance function on $\mathbb{R}$. For $\Omega \subset \mathbb{R}^d$, any constant $C' \in \mathbb{R}$, and function $T : \Omega \to \mathbb{R}$,*

$$D_{KL}(\mathbb{P}||\mathbb{Q}) = \sup_{T:\Omega\to\mathbb{R}} \mathbb{E}_{\mathbb{P}}[T] - \log(\mathbb{E}_{\mathbb{Q}}[e^T]) - d(\log(\mathbb{E}_{\mathbb{Q}}[e^T]), C') \tag{3.4}$$

The extra regularization term at the end, $d(\log(\mathbb{E}_{\mathbb{Q}}[e^T]), C')$, prevents the drifting

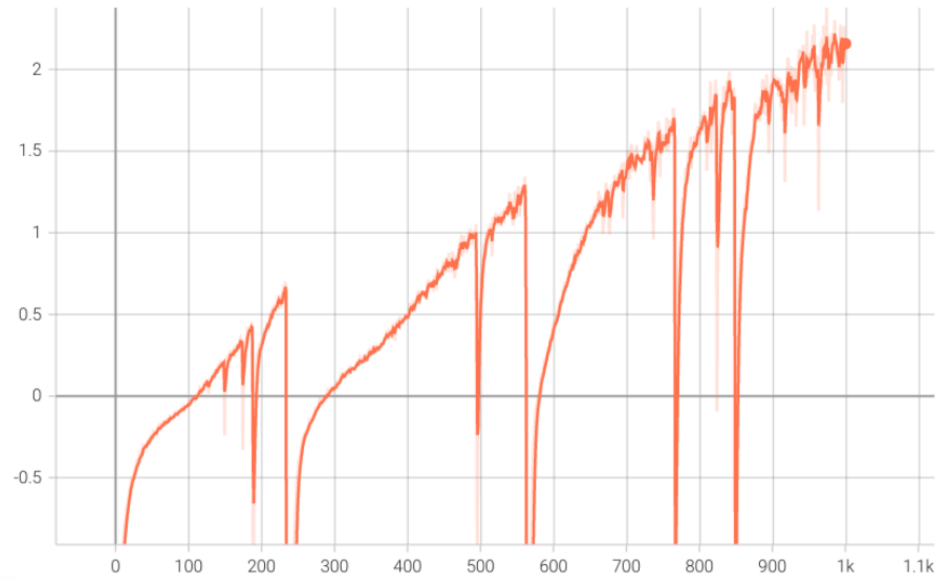


(a)

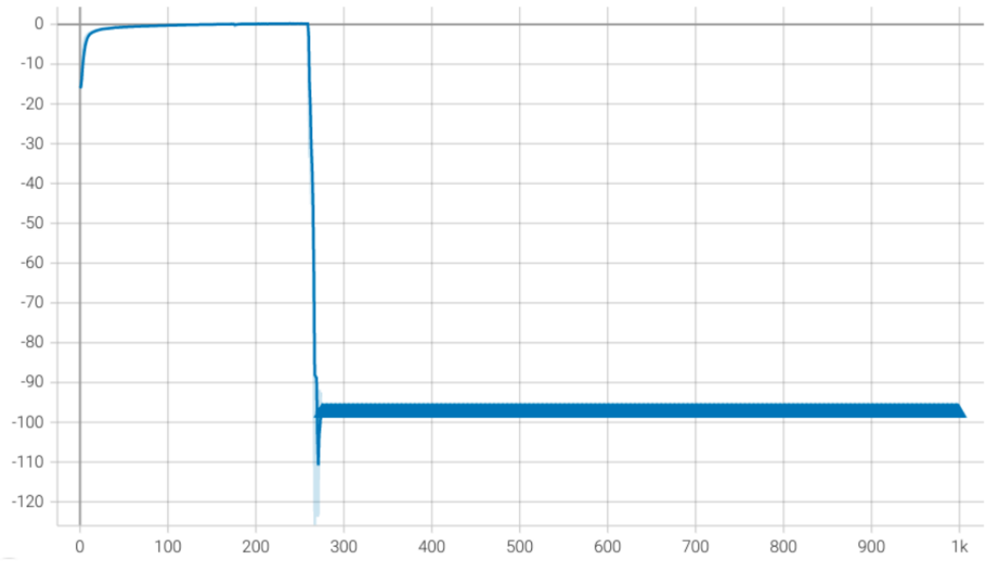


(b)

Figure 3-3: Various numerical challenges that we faced when using the MNIST dataset. For all graphs, light colors represent true MI estimation values across iterations, and the dark lines track the EMA with a smoothing factor of 0.6. The top is a graph of the MI estimate across MINE iterations for the convolutional encoder and MINE architectures from Table 3.1 and Table 3.4, respectively. We can see that the MI lower bound estimate fails to converge to a non-zero value. The top shows the same experimental setup with a different seed for the convolutional encoder's random initialization of weights. This one converges; however, the variance is still high relative to the estimated MI value itself.



(c)



(d)

Figure 3-3: Both graphs are the MI estimation results from using a dense encoder with a dense MINE architecture from Table 3.2 and Table 3.5, respectively. We observed gradient explosions with large spikes in both MI graphs. In the case of the bottom graph, the MI value converged to NaN, or $-\infty$ (there are many bold triangles bunched together in a line, indicating a NaN value for each epoch past the explosion point).

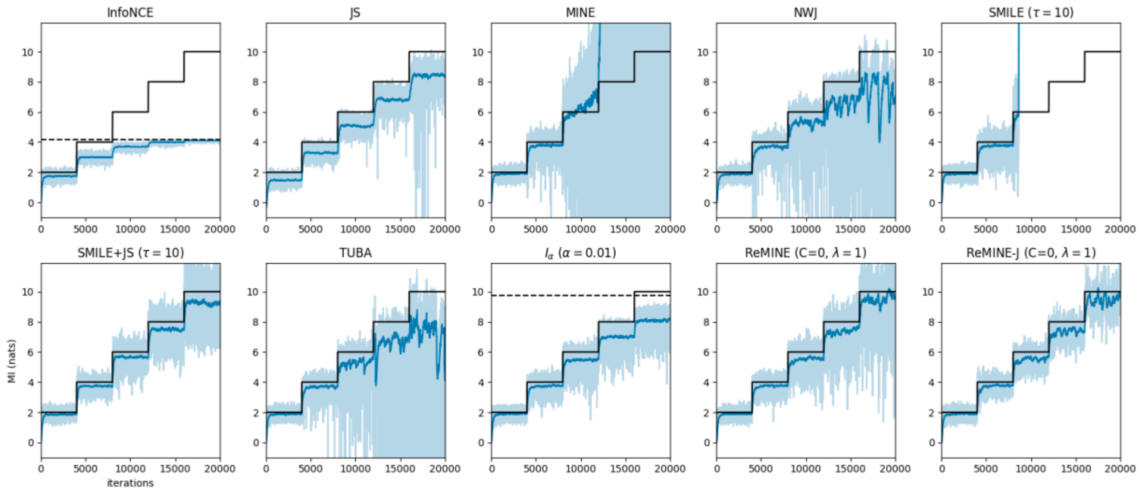


Figure 3-4: Figure 6 from [8] comparing the different performance of multiple different mutual information estimators on a 20-dimensional Gaussian dataset. Every 4000 iterations of the underlying estimation algorithm, the value of ρ in the covariance matrix was increased (the same Gaussian distribution construction as described in Section 3.1.1. “The estimated MI (light) and smoothed estimation with exponential moving average (dark) are plotted for each method,” and the dotted lines represent theoretical bounds on MI estimation (as described in Sections 2.3 and 2.4 in [25].

Algorithm 2 ReMINE

```
$\theta \leftarrow$ Initialize network parameters, $K \leftarrow$ Moving average window size, $i \leftarrow 0$
repeat
    Draw $J$ samples from the joint distribution:
    $(x_i^{(1)}, y_i^{(1)}), \ldots, (x_i^{(J)}, y_i^{(J)}) \sim \mathbb{P}_{XY}$
    Draw $M$ samples from the marginal distribution:
    $(\bar{x}_i^{(1)}, \bar{y}_i^{(1)}), \ldots, (\bar{x}_i^{(M)}, \bar{y}_i^{(M)}) \sim \mathbb{P}_X \otimes \mathbb{P}_Y$
    Evaluate the lower bound:
    $\hat{\mathbb{E}}_{\mathbb{P}_{XY}} \leftarrow \frac{1}{J} \sum_{j=1}^{J} T_\theta(x_i^{(j)}, y_i^{(j)})$
    $\hat{\mathbb{E}}_{\mathbb{P}_X \otimes \mathbb{P}_Y} \leftarrow \log(\frac{1}{M} \sum_{m=1}^{M} e^{T_\theta(\bar{x}_i^{(m)}, \bar{y}_i^{(m)})})$
    $\mathcal{V}(\theta) \leftarrow \hat{\mathbb{E}}_{\mathbb{P}_{XY}} - \hat{\mathbb{E}}_{\mathbb{P}_X \otimes \mathbb{P}_Y} - d(\hat{\mathbb{E}}_{\mathbb{P}_X \otimes \mathbb{P}_Y}, C)$
    Update the statistics network parameters:
    $\theta \leftarrow \theta + \nabla_\theta \mathcal{V}_\theta$
    Estimate MI based on the last window $W = [\max(0, i - K + 1), \min(K, i)]$ of
size $K$:
    $\hat{I}(X;Y) = \frac{1}{J \cdot |W|} \sum_{w \in W} \sum_{j=1}^{J} T_\theta(x_w^{(j)}, y_w^{(j)}) - \log(\frac{1}{J \cdot |W|} \sum_{w \in W} \sum_{m=1}^{M} e^{T_\theta(\bar{x}_w^{(m)}, \bar{y}_w^{(m)})})$
    Next iteration: $i \leftarrow i + 1$
until convergence
```

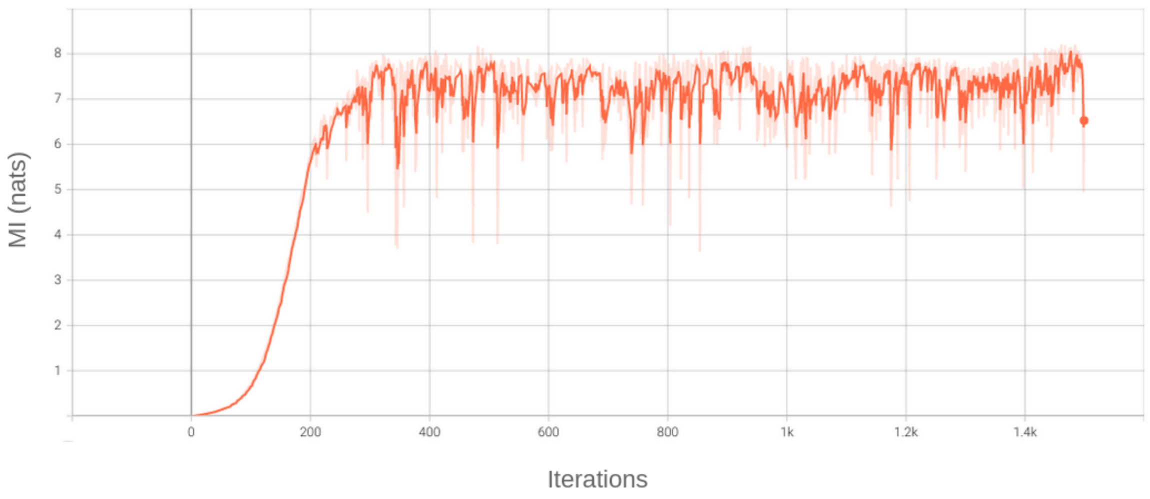
phenomenon by warping the loss surface so that there is only one solution, as opposed to in MINE where there are many possible values for the first and second terms of the Equation 3.2 that lead to an optimal lower bound. Furthermore, the authors propose a micro-averaging strategy in the ReMINE Algorithm 2 to estimate MI based on a sliding window of ReMINE results across previous iterations. Empirically, this was shown to mitigate the exploding gradients issue. One can also see in Figure 3-4 that ReMINE outperforms many other state of the art MI estimators (including MINE itself) with respect to numerical stability, low bias, and variance reduction. We observed similar outcomes in our own findings (see Section 3.5) when using ReMINE.

For the implementation, we followed the authors' recommendation of using L2 regularization as the distance metric with default values of $C = 0$ and $\lambda = 0.1$, and we used a simple average with a sliding window of size $K = 10$ for the final MI estimate.

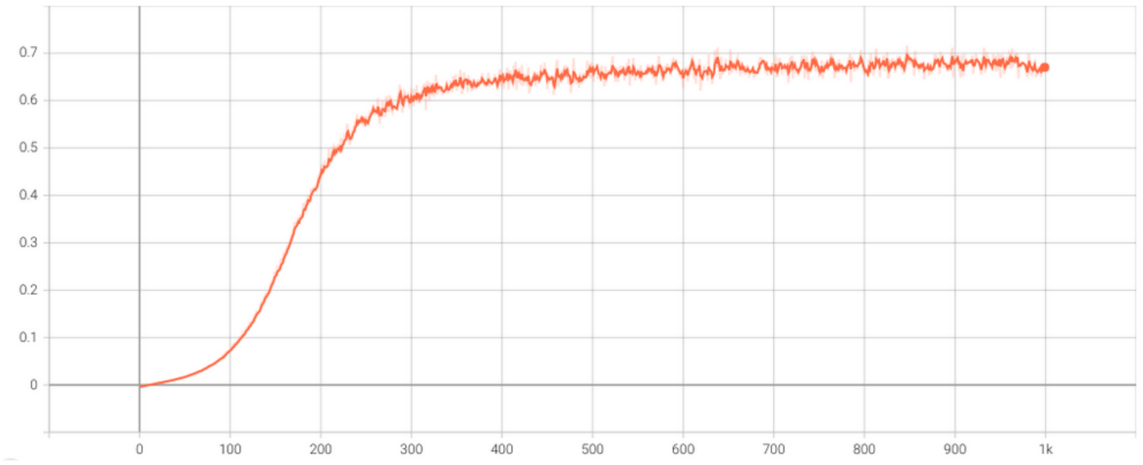
3.5 Analysis on Gaussian Dataset and MNIST

In this section, we present MI estimation results from utilizing ReMINE on Gaussian data. We omit the results for MNIST data since the challenges illustrated in Section 3.4.2 persisted despite our switch to ReMINE. This was to be expected though, as the authors of ReMINE described in Section 5.3 of [8] that even ReMINE (as well as the other DV-based estimators like MINE and SMILE) failed to produce estimates close to the ideal values in the image domain with the CIFAR-10 dataset. All ReMINE charts in this chapter and the next show the convergence of MI estimation, with each iteration's estimation value in a lighter color and the smoothed result from applying EMA with $\alpha = 0.6$ on the darker line. For each result in the ReMINE estimation tables, we take the average of the last 10 values from the ReMINE charts.

Our initial ReMINE estimation results are shown in Figure 3-5 and Figure 3-6. We can observe that the final estimation values in Figure 3-6a are upper bounded by those in Figure 3-5a, ie: $I(X;Z) <= I(X;X) = H(X)$. This inequality should directly follow from the MI formula, expressed in terms of entropy values in Equation

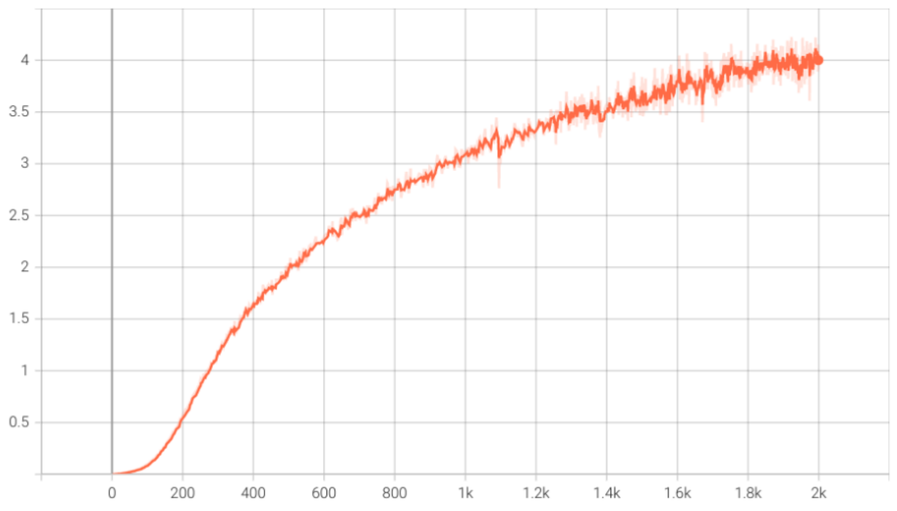


(a)

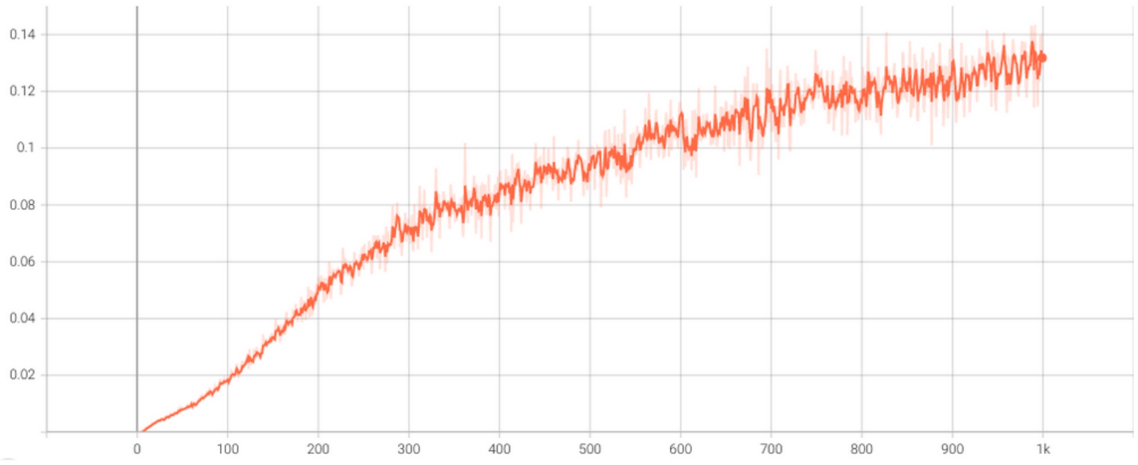


(b)

Figure 3-5: ReMINE estimation results. (3-5a) $\hat{I}(X;X)$, which is a lower bound on $I(X;X) = H(X)$. The true entropy of our synthetic Gaussian distribution is 8.79. (3-5b) $\hat{I}(X;L(X))$, where $L(X)$ is the binary label of whether an input belongs to the first or second Gaussian distribution (see our experimental setup in Section 3.1.1). Note that $H(L(X)) = 0.69$.



(a)



(b)

Figure 3-6: ReMINE estimation results with an untrained encoder generating Z. (3-6a) $\hat{I}(X;Z)$, and (3-6b) $\hat{I}(Z;L(X))$. Notice that both estimated values are lower than their theoretical bounds (8.79 and 0.69, respectively).

2.5: $I(X;Z) = H(X) - H(X|Z) \leq H(X)$. We can also directly calculate the entropy of our Gaussian dataset to verify the estimation result for $I(X;X)$.

$$\begin{aligned} H(X) &= \frac{H(X|\rho_0)}{2} + \frac{H(X|\rho_1)}{2} \\ &= H(X|\rho_0) \\ &= \frac{\ln|\Sigma|}{2} + \frac{D}{2}(1 + \ln(2\pi)) \\ &= 8.79 \end{aligned} \tag{3.5}$$

The first line follows from the nature of how the Gaussian dataset was constructed, with 30,000 samples each from 2 different Gaussian distributions. The second line is due to the covariance matrices for both distributions having the same structure and $\rho_0 = -1 \times \rho_1$, which ensures that the determinant of their covariance matrices are the same. This would cause $H(X|\rho_0) = H(X|\rho_1)$. The third line is the definition of the entropy for a multivariate Gaussian dataset of $\mathcal{N}(\mu, \Sigma)$ that is D-dimensional. We notice that the estimation result for $I(X;X)$ is around 7.59 (final 10 values averaged), which is an appropriate lower bound on the actual value, $H(X) = 8.79$.

Furthermore, since Z was obtained via an encoder transformation that reduced the amount of output nodes, we can compare Figure 3-6b and Figure 3-5b to verify that $I(Z;L(X)) < I(X;L(X)) <= H(L(X))$. We can also directly compute the entropy of the labelling function, since our experimental data was set up with binary labels for the two groups of 30,000 samples from separate Gaussians: $H(L(X)) = -1 \times (0.5 \times \ln 0.5 + 0.5 \times \ln 0.5) = 0.6931$. This corroborates the ReMINE results for Figure 3-5b, where $I(X;L(X))$ converges to a value around 0.69.

Now that there is a numerically stable MI estimation procedure for our Gaussian data, we are ready to apply it for the *InfoShape* training pipeline, detailed in the next section.

Chapter 4

Encoder Training Pipeline

In this chapter we first describe the system model for *InfoShape* in Section 4.1. Then, we discuss the empirical results of various experiments with encoder training in Section 4.2. These results pose many research questions, which lead nicely into the future works section in 5.1.

4.1 System Model

Our *InfoShape* schematic, which utilized ReMINE for the synthetic Gaussian dataset, is outlined in Figure 4-1. In our implementation, we used ReMINE to train the underlying statistics network until convergence for both privacy leakage and utility score measures, and then, as the authors recommended, we took a simple average of an extra 150 iterations of the algorithm as our final MI estimate. We further normalized the privacy leakage and utility score against our empirical estimate of $H(X)$ from earlier in Figure 3-5a and $H(L(X))$ in Figure 3-5b. These were combined, weighted, and applied as the encoder's loss function, which is shown here:

$$\mathcal{L}(\theta, X, Z, L(X)) = \lambda_1 \frac{\hat{I}(X;Z)}{H(X)} - \lambda_2 \frac{\hat{I}(Z;L(X))}{H(L(X))} \tag{4.1}$$

Since there is no guarantee that we can directly calculate the entropy of the input data, in our experiments we replace it with $\hat{I}(X;X)$, which gives a lower bound

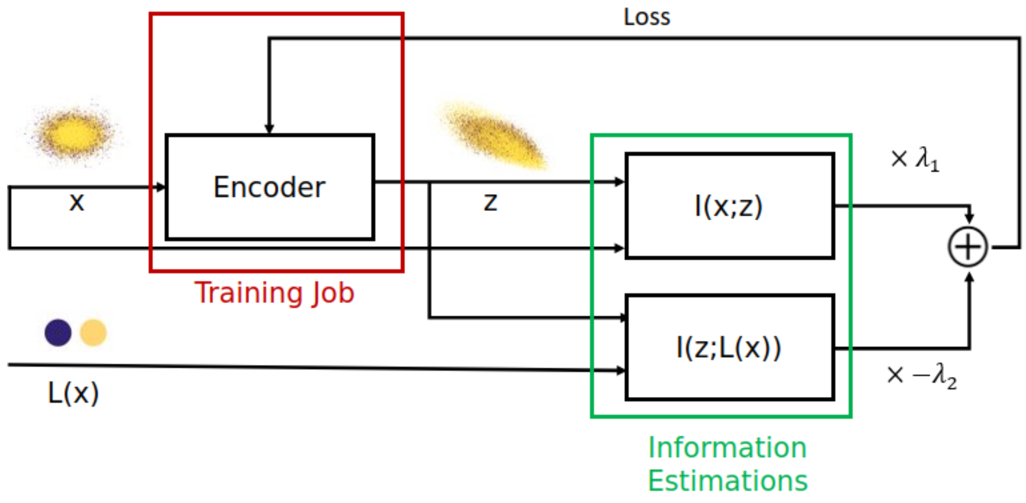


Figure 4-1: A specific adaptation of Figure 1-2 for ReMINE experiments on the synthetic Gaussian dataset. The inputs, X, are samples from two possible 20-dimensional multivariate Gaussians: $\mathcal{N}(0, \Sigma_0)$ or $\mathcal{N}(0, \Sigma_1)$, with each covariance matrix differing in just two diagonals (see Section 3.1.1 for details). Z are the 3-dimensional encoded outputs. The Information Estimations step utilizes ReMINE to minimize the MI lower bounds on $I(X;Z)$ for the privacy leakage and maximize $I(Z;L(X))$ for utility score, where L is the labeling function. The final MI values are combined in a Lagrangian optimization as the encoder's loss function. The Training Job modifies the encoder's weights using classic back propagation, and the cycle repeats.

estimate on $H(X)$.

Based on the lessons learned from Section 3.4.2, we used a large mini-batch size of 4000 for the ReMINE algorithm to fight variance in our estimates. This was critical since our encoder would not have a productive training process if its loss values had higher fluctuations from variance compared to any changes as a result of encoder back-propagation. Furthermore, to mitigate any potential stability issues, we batched the gradients for ReMINE in batches of 10, to discourage a single MINE back-propagation step from blowing up the gradients.

Our core MI estimation algorithm uses the ReMINE implementation discussed in Section 3.4.3. Due to the Pytorch lightning implementation choice, our ReMINE estimator would consider a full pass through every data sample in the dataset as one "epoch," which is standard, but goes against the MINE algorithm's random sampling of one mini-batch at a time being considered as a single iteration. As a result, the x-

Layer	Number of Outputs	Activation Function
Input $\mathbf{x}$	20	-
Dense	100	ReLU
Dense	1	Sigmoid

Table 4.1: Dense classifier architecture for classification on Gaussian data. Note that the outputs of the classifier are probability scores from 0 to 1 of the input data being label 1 (being sampled from $\mathcal{N}(0, \Sigma_1)$ with $\rho_1 = 0.99$).

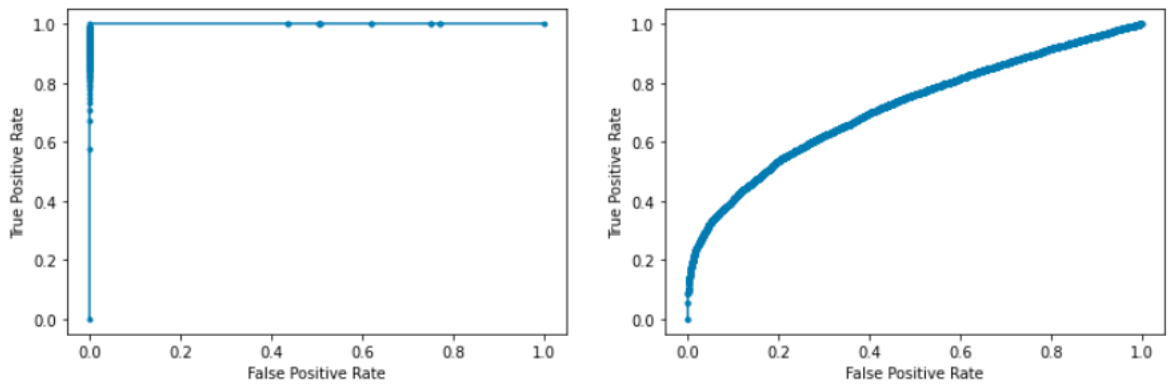


Figure 4-2: The left graph shows the ROC Curve for the classifier from Table 4.1 trained and tested on the original Gaussian data, achieving an AUC of 1.00. The right graph shows the ROC Curve for the same classifier trained and tested on randomly encoded data (classifier's first layer modified to have 3 input nodes for compatibility). The classifier achieved a noticeably worse AUC of 0.72.

axis number of iterations in all of our ReMINE charts are downsampled to only include the MI estimate after a full pass of the dataset through ReMINE's main loop. Since we used a minibatch size of 4000, our actual number of "ReMINE iterations" is $N/BS \times$ epochs. By default, we chose to use 1000 epochs for utility score estimation and 1500 epochs for privacy leakage. This amounts to 15000 and 22500 ReMINE iterations, respectively. We chose these values since running the ReMINE algorithm for this many iterations across multiple training epochs was computationally expensive and time-consuming. In later experiments we increased the number of epochs to achieve better convergence.

4.2 Results

In the left figure in Figure 4-2, we observe that on the original, unencoded Gaussian data, a simple classifier is able to achieve an AUC of 1.00 for predicting which Gaussian distribution each sample belongs to. This is not surprising, as the covariances along the diagonal with ρ have opposite values (see dataset setup in Section 3.1.1), leading to a trivial decision boundary. However, after undergoing the transform from an untrained encoder with randomly initialized weights, classification AUC suffers dramatically, dropping to 0.72, as shown in the right figure in Figure 4-2. The drop in classification performance is in line with our intentions of using the encoder as a lossy compressor, as well as the observations from Section 3.5 (reference Figure 3-6), where we showed that $\hat{I}(X; L(X)) = 0.69$ and $\hat{I}(Z; L(X)) = 0.13$.

Now, we quantify the information gain from encoder training in Figure 4-5, which shows the ReMINE estimation results for just the utility score across 10 encoder training epochs (training job 2 in Figure 4-1). On epoch 0 with the untrained encoder, $\hat{I}(Z; L(X)) \simeq 0.13$. On epoch 5, the value increased to 0.24, and on epoch 10, we obtained 0.30. Our utility score more than doubled after just 10 epochs, with a smooth increase in utility score after each epoch. This by itself demonstrates the promising ability of *InfoShape* to utilize an estimate of mutual information as a part of the loss to improve performance of a certain metric.

In Figure 4-3, we show the classifier's AUC improvement to 0.94 after training and testing on the trained encoder outputs, which verifies the observed increase in utility score. We have included the encoder outputs visualized in 3 charts as well in Figure 4-4.

While our utility score was able to increase as a result of encoder training, privacy leakage largely stayed the same, with small fluctuations that could be due to variance. The privacy leakage plots are shown in Figure 4-6. To address the stubbornness of the privacy leakage, we conducted a series of follow-up experiments where we increased the value of λ_2 to 10 in Figure 4-7, then 100 in Figure 4-8, and we also used $I(X; Z)$ as the only term in the loss function in Figure 4-9. For the last experiment,

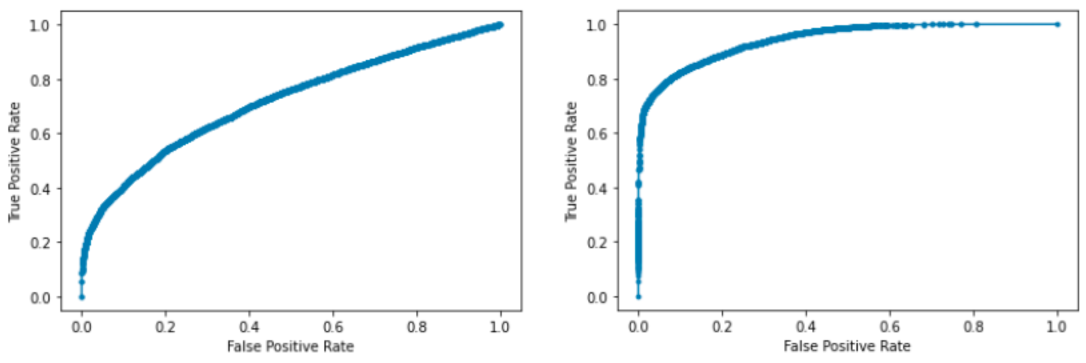


Figure 4-3: The improvement of classifier AUC after encoder training. The left graph shows the previous AUC from the classifier training and testing on the untrained encoder's outputs. The right graph shows the AUC of the same classifier after the encoder went through 10 training epochs of *InfoShape*. The AUC improved from 0.72 to 0.94, nearly at the same AUC as with the original data.

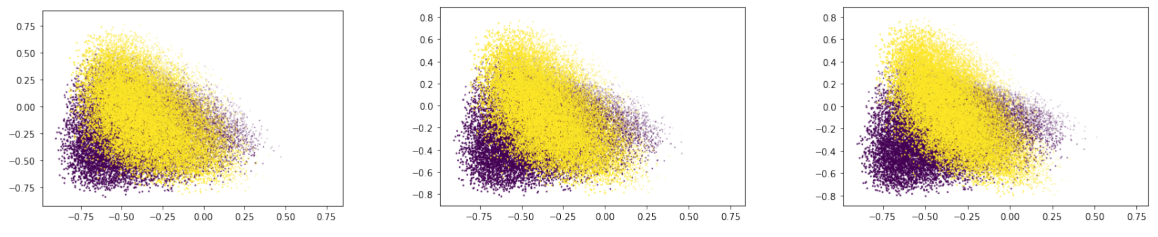
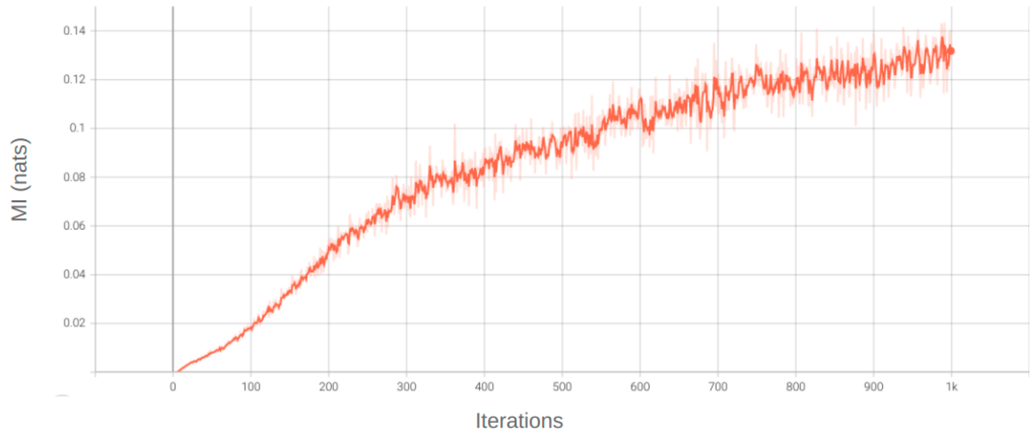
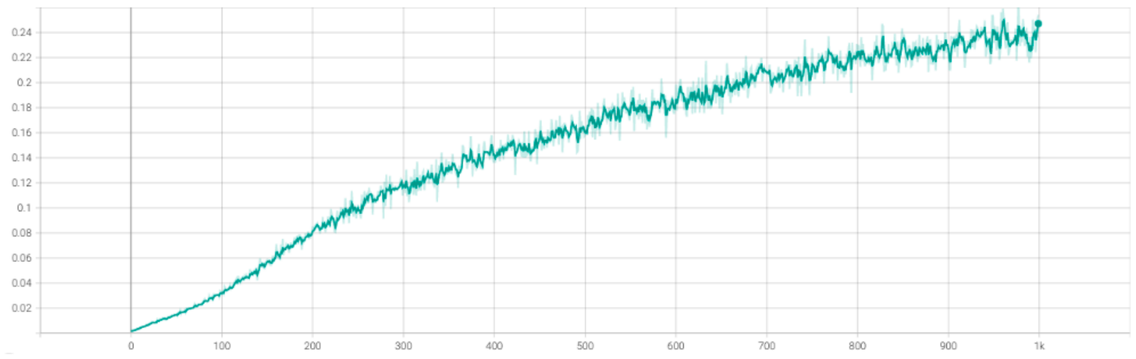


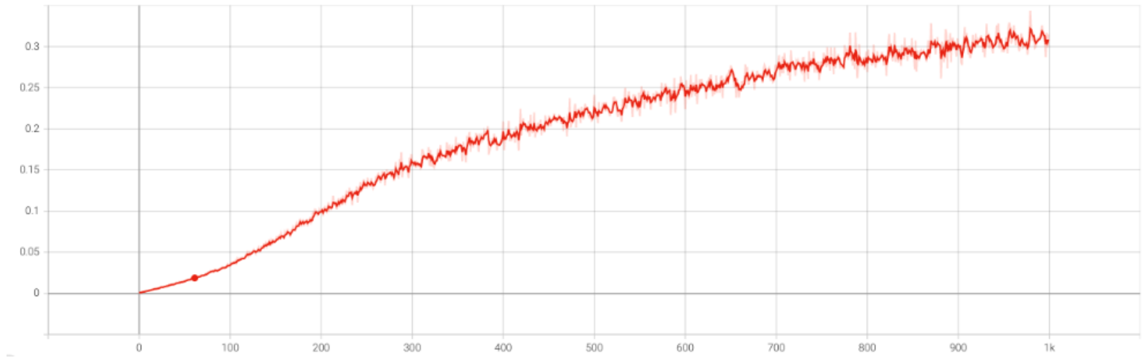
Figure 4-4: Encoder outputs from Section 4.2 with $\lambda_1 = 1, \lambda_2 = 1$ for epochs 0, 5, and 9 (left to right) between multivariate Gaussian components x_0 on the x-axis and x_1 on the y-axis.



(a)

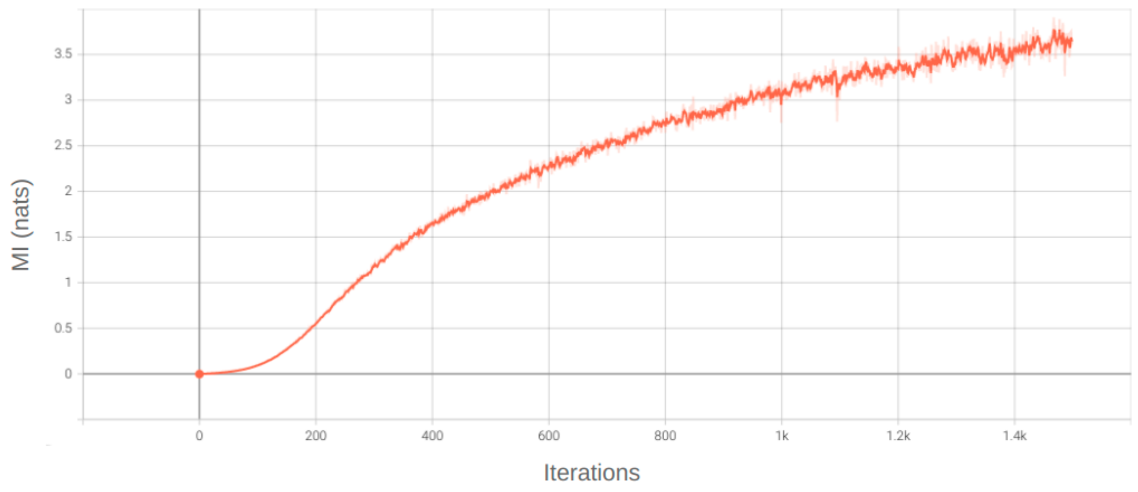


(b)

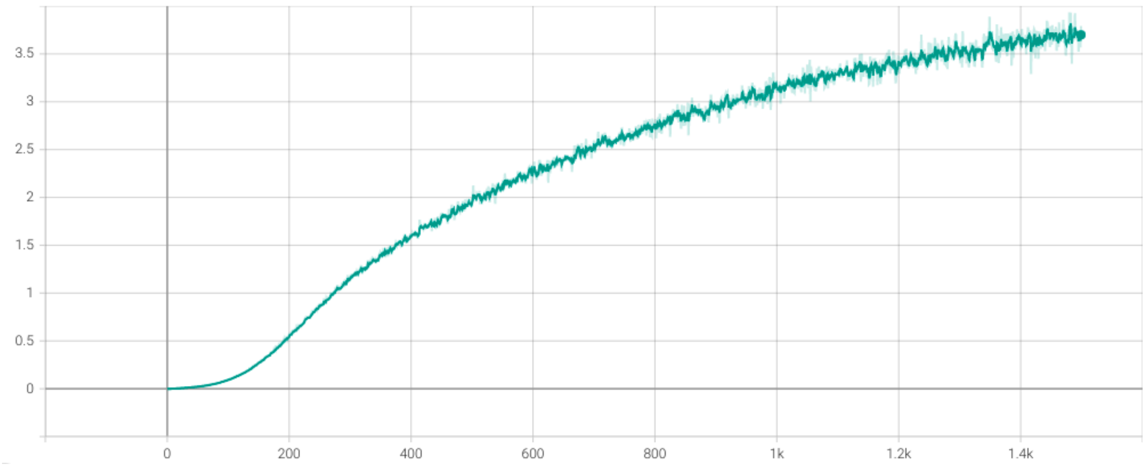


(c)

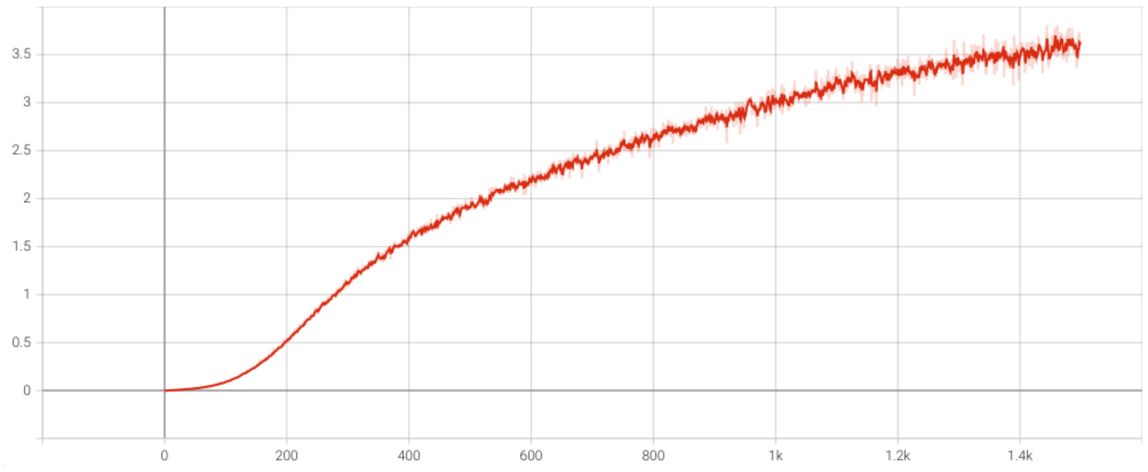
Figure 4-5: Utility score: $\hat{I}(Z; L(X))$ across encoder training epochs, with $\lambda_1 = 1$ and $\lambda_2 = 1$. Figure 4-5a shows the ReMINE algorithm convergence of $\hat{I}(Z; L(X)) = 0.13$ on *encoder training* epoch 0 (ie Z is the untrained encoder-transformed data). Figure 4-5b shows $\hat{I}(Z; L(X)) = 0.24$ for encoder training epoch 5. Figure 4-5c shows $\hat{I}(Z; L(X)) = 0.30$ for epoch 9.



(a)



(b)

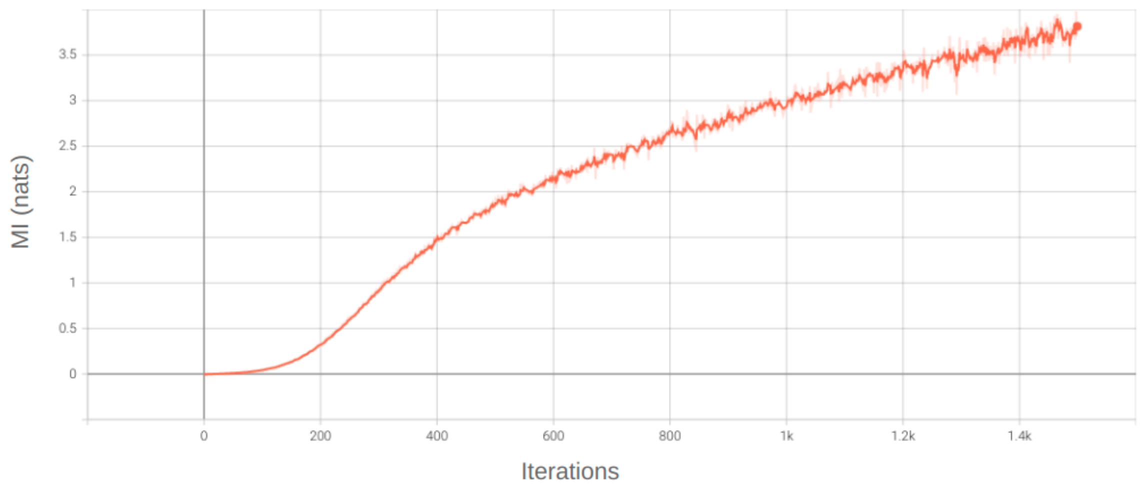


(c)

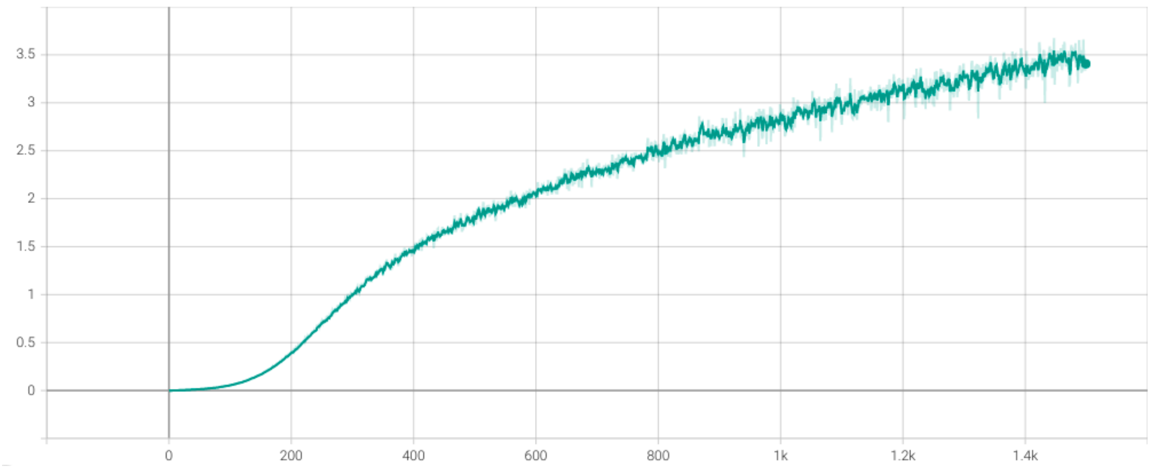
Figure 4-6: $\hat{I}(X;Z)$ across encoder training epochs. Figure 4-6a shows $\hat{I}(X;Z) = 3.61$ after epoch 0. Figure 4-6b shows $\hat{I}(Z;L(X)) = 3.67$ after epoch 5. Figure 4-6c shows $\hat{I}(Z;L(X)) = 3.52$ after epoch 9.

we simultaneously extended the number of training epochs to 25 and increased the number of ReMINE epochs to 2500. With these new sets of parameters, we were able to observe a small yet significant consistent decrease in the privacy leakage over time. Despite fluctuations still causing certain epochs to display inconsistent behavior, overall we can see that *InfoShape* is also able to optimize for a privacy leakage term.

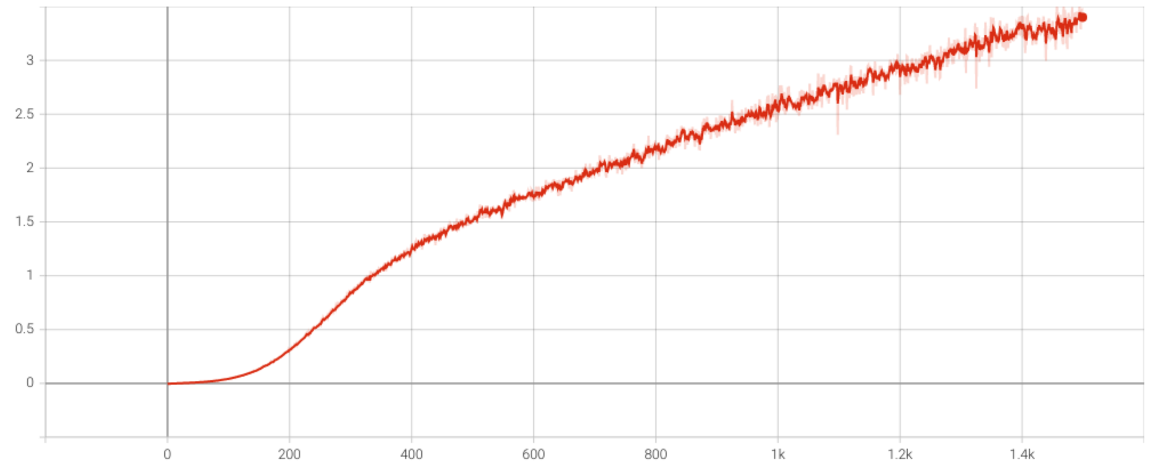
We conclude and discuss potential future paths of work in the next section.



(a)

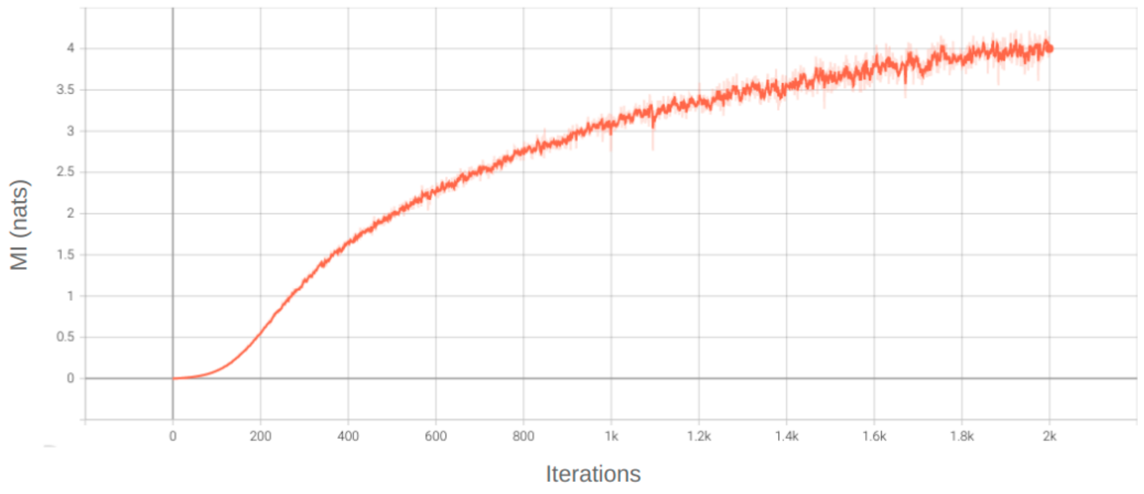


(b)

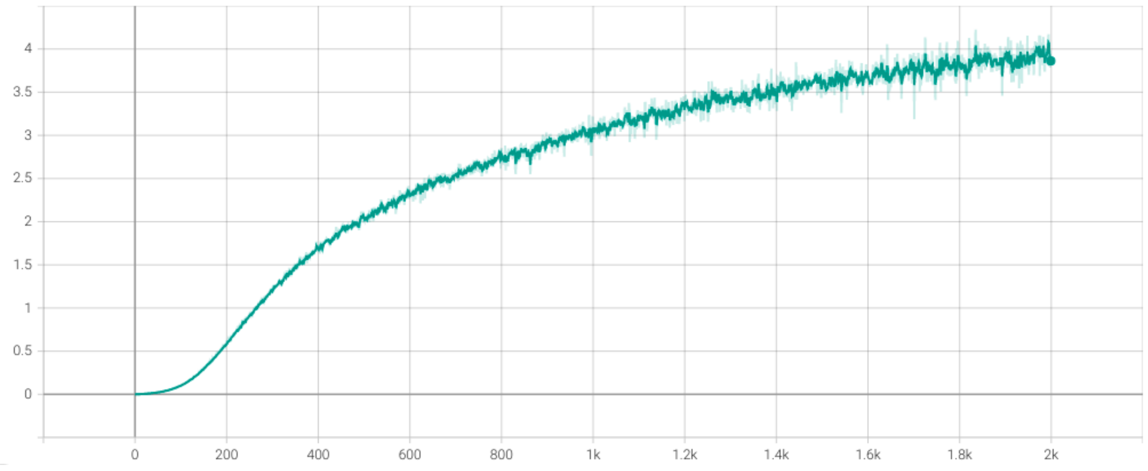


(c)

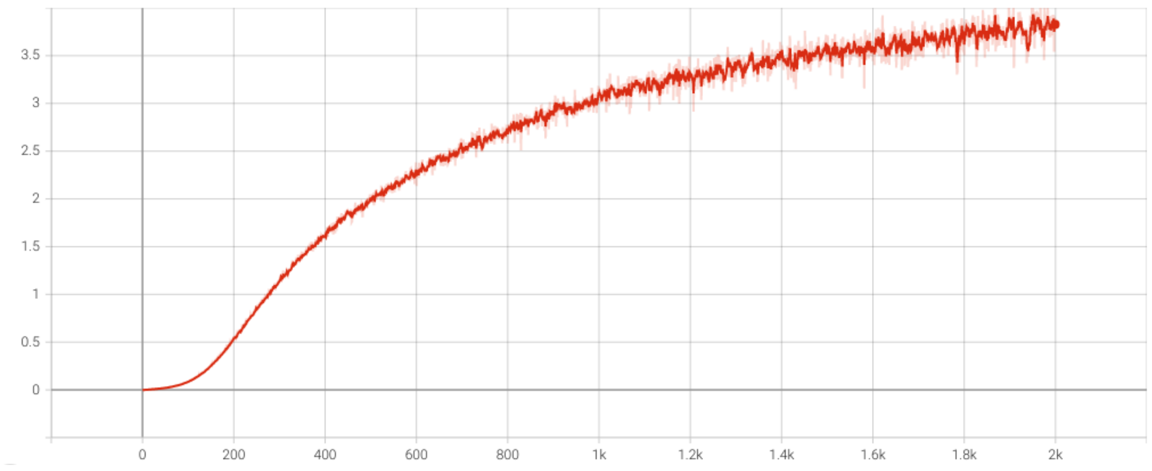
Figure 4-7: $\hat{I}(X;Z)$ across encoder training epochs with $\lambda_2 = 10$ (see the loss function in Equation 4.1). Figure 4-7a shows $\hat{I}(X;Z) = 3.66$ after epoch 0. Figure 4-7b shows $\hat{I}(Z;L(X)) = 3.46$ after epoch 5. Figure 4-7c shows $\hat{I}(Z;L(X)) = 3.38$ after epoch 9.



(a)

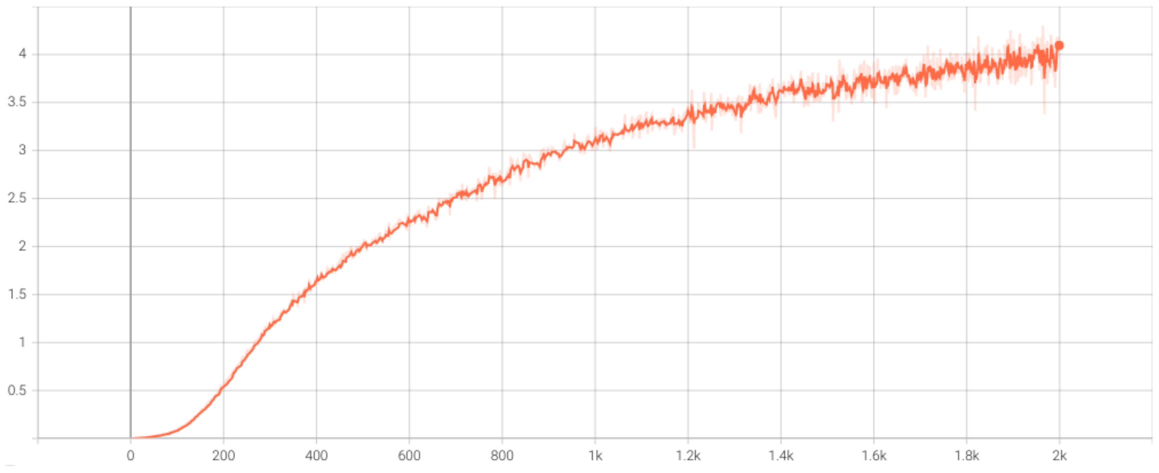


(b)

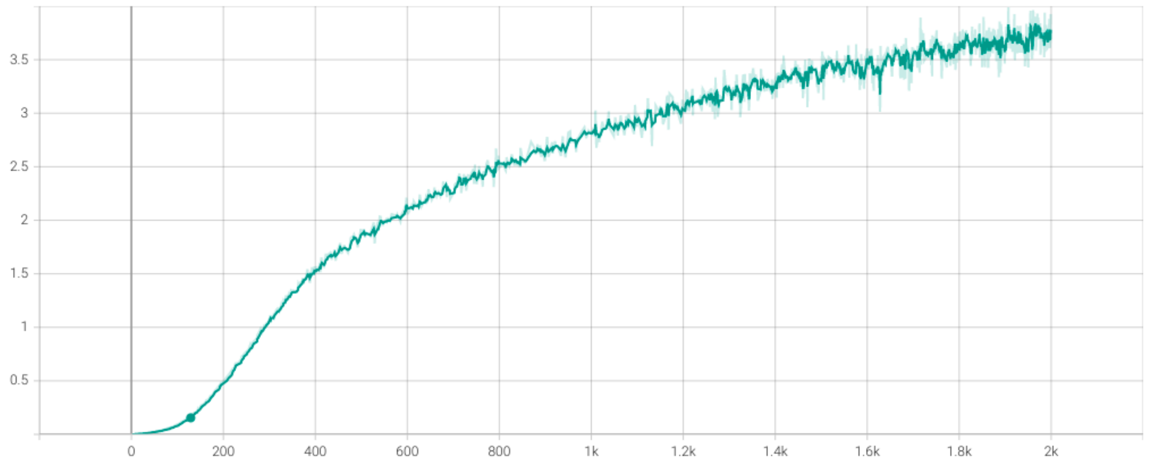


(c)

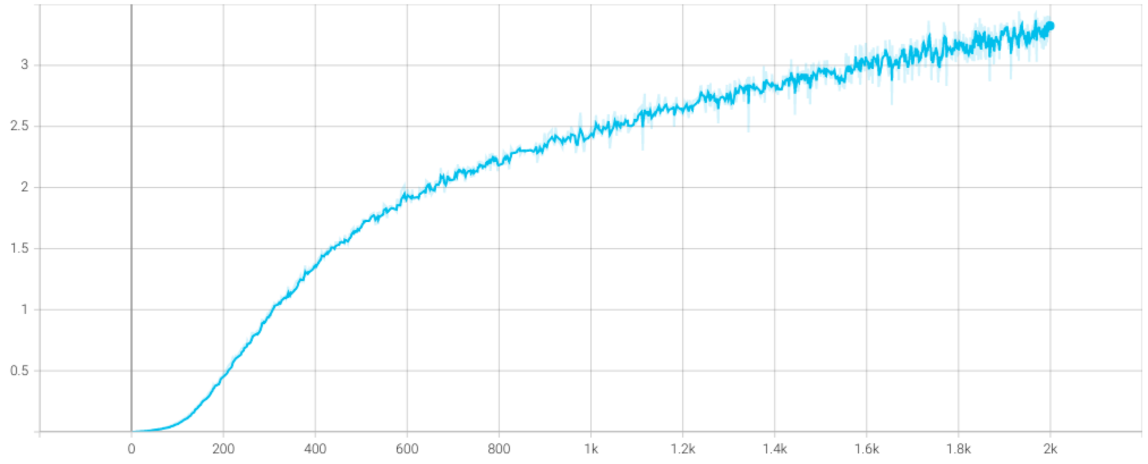
Figure 4-8: $\hat{I}(X;Z)$ across encoder training epochs with $\lambda_2 = 100$. We extended the number of ReMINE epochs to 2000 for this experiment for better convergence. Figure 4-7a shows $\hat{I}(X;Z) = 4.03$ after epoch 0. Figure 4-7b shows $\hat{I}(Z;L(X)) = 3.98$ after epoch 5. Figure 4-7c shows $\hat{I}(Z;L(X)) = 3.81$ after epoch 9.



(a)



(b)



(c)

Figure 4-9: $\hat{I}(X;Z)$ across encoder training epochs with $\hat{I}(X;Z)$ as the only term in the loss function, trained for 25 epochs and with 2000 ReMINE iterations. Figure 4-9a shows $\hat{I}(X;Z) = 3.94$ after epoch 0. Figure 4-9b shows $\hat{I}(Z;L(X)) = 3.69$ after epoch 5. Figure 4-9c shows $\hat{I}(Z;L(X)) = 3.32$ after epoch 9.

Chapter 5

Discussion and Conclusion

In this thesis, we proposed the novel idea of *InfoShape*, which, to the best of our knowledge, has not been studied in the literature in this practical setup. Our idea centers on two fundamental concepts. First, with certain architectural choices, a neural network encoder can be viewed as a lossy compressor that can potentially filter out certain information (such as private features of sensitive data, or information not needed for the downstream task, as in the Information Bottleneck) and keep information related to the inputs and intended labels. Second, mutual information is a general metric that has been shown to be an excellent theoretical substitute for the goal of privacy, and it is possible to estimate mutual information with neural estimation methods that produce low bias and low variance results. These results can then be used to train the encoder to simultaneously drive down the privacy leakage and increase the utility score.

Using ReMINE and carefully selected hyperparameters, we were able to train a simple encoder to increase the utility score and decrease privacy leakage over time, although not simultaneously. We are confident that, with further investigation, the dual optimization that is desired can be achieved. In the next section, we discuss potential avenues for future work that we believe can yield promising results, directly building off of our thesis and the surrounding literature.

5.1 Future Work

We propose five potential areas of future work, all of which could be important for developing the robustness of *InfoShape*. For each area of work, we first discuss the primary challenges that have not been addressed by this work and then suggest ideas to tackle them.

One of the largest issues that we faced during our experiments was high variance. Without a stable MI estimation, random fluctuations make the training of the encoder in *InfoShape* almost impossible. We believe that lowering the variance while keeping acceptable bias levels is a crucial next step. The authors of [25] presented many different MI estimators and examined their advantages and drawbacks. We believe that it is worth studying how I_α with $\alpha \in [0.01, 0.5]$ performs for our purposes, since a higher value of α runs into the limitation that InfoNCE experiences, namely a theoretical upper bound on the estimation: $O(\log K)$, K being the number of samples in each batch. Without a large enough batch size, if $I(X;Y) > \log K$, then the estimation would provide a loose bound. We wish to avoid this at all costs, since this could lead to the same loss values for our encoder despite the encoder weights being adjusted to change whichever metric is being calculated. We would prefer high variance and low bias over the flip-side; however, exploring I_α could potentially identify a range of α values for which the lowest variance is achieved.

Recent work by the authors of [31] suggest an interesting alternative to estimating MI itself. They claim that estimating the gradients of MI is more appealing for representation learning and that their Mutual Information Gradient Estimation (MIGE) method "exhibits a tight and smooth gradient estimation of MI in the high-dimensional and large-MI settings." These settings are precisely where existing MI estimators struggle, especially with the image domain, as we experienced in our experiments. We believe that if a custom training pipeline could be built for *InfoShape* where gradients are manually computed and applied in back-propagation, our learned representations from an encoder could be much more powerful, both in terms of privacy leakage and utility score. We would also have to investigate the difference between convergence rates when using MIGE vs ReMINE, and also MIGE's numerical stability.

One critical aspect of the project that would enhance the robustness of *InfoShape* would be the development of a suite of adversarial attacks, like in [32]. While we had a basic classifier to verify any increases in utility score via classification AUC changes, we would ideally also like some concrete method of measuring certain privacy metrics. Although there was not enough time for me to develop this during my year at MIT, I believe that this could be a relatively simple extension of the work that verifies the practicality of the privacy leakage results. This area of work could even potentially develop into adding specific terms in the *InfoShape* loss function to enforce that certain privacy goals are met, as opposed to reducing a general measure of privacy leakage.

Another possible path would be to study the convergence rates of MINE and ReMINE from a theoretical standpoint, since this directly influences how long the *InfoShape* training process takes. It would also be an interesting direction to be able to visualize the loss landscape of ReMINE or MINE when working in the image domain, in order to better understand why the results are so numerically unstable. This was yet another potential mini-experiment that I did not have the time to finish. In particular, I would have liked to have been able to visualize the loss surface for ReMINE in our synthetic Gaussian experiments to verify the effects of regularization that the authors in [8] claim that ReMINE has.

Finally, one could take a more theoretical direction and discover any potential connections between the work of [28] and *InfoShape*. From our results, especially those in Figure 4-5 and Figure 4-9, we can see that our encoder quickly doubled the utility score after just 10 epochs; however, it needed 25 epochs before any significant differences were to be observed for the decrease of privacy leakage. This is directly in line with the hypothesis of Tishby et. al: they believe that "deep networks undergo two distinct phases consisting of an initial fitting phase and a subsequent compression phase," which they believe "occurs due to the diffusion-like behavior of stochastic gradient descent." This paper has been supposedly refuted in [26], leading to some controversy, mostly from Tishby himself. This could be a remarkable area of study that provides another stepping stone in the direction of developing a theoretical understanding of neural networks, especially for our niche concerned with privacy.

Bibliography

[1] Mohammad Alomrani. A critical review of information bottleneck theory and its applications to deep learning, 05 2021.

[2] Ishmael Belghazi, Sai Rajeswar, Aristide Baratin, R. Devon Hjelm, and Aaron C. Courville. MINE: mutual information neural estimation. *CoRR*, abs/1801.04062, 2018.

[3] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: A new vision for public-key cryptography. *Commun. ACM*, 55(11):56–64, nov 2012.

[4] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on Computing*, 43(2):831–871, 2014.

[5] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan L. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, pages 207–222, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[6] Centers for Medicare & Medicaid Services. The Health Insurance Portability and Accountability Act of 1996 (HIPAA). Online at `http://www.cms.hhs.gov/hipaa/`, 1996.

[7] Hyunghoon Cho, David J Wu, and Bonnie Berger. Secure genome-wide association analysis using multiparty computation. *Nature biotechnology*, 36(6):547–551, 2018.

[8] Kwanghee Choi and Siyeong Lee. Regularized mutual information neural estimation, 2021.

[9] Council of European Union. Regulation (eu) 2016/679. Online at `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=\CELEX%3A32016R0679&qid=1639028015586`, 2016.

[10] Paul Cuff and Lanqing Yu. Differential privacy as a mutual information constraint. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 43–54, New York, NY, USA, 2016. Association for Computing Machinery.

[11] D.L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[12] M. D. Donsker and S. R. S. Varadhan. Asymptotic evaluation of certain markov process expectations for large time, i. *Communications on Pure and Applied Mathematics*, 28(1):1–47, 1975.

[13] Cynthia Dwork and Aaron Roth. 2014.

[14] Soheil Feizi-Khankandi and Muriel Médard. On network functional compression. *CoRR*, abs/1011.5496, 2010.

[15] Veerraju Gampala, M. Kumar, C. Sushama, and Fantin Irudaya Raj Edward Sehar. Deep learning based image processing approaches for image deblurring. *Materials Today: Proceedings*, 12 2020.

[16] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[17] Schulam P Beam AL Chen IY Ranganath R Ghassemi M, Naumann T. A review of challenges and opportunities in machine learning for health. In *AMIA Jt Summits Transl Sci Proc*, 2020.

[18] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951.

[19] Bo Liu, Ming Ding, Hanyu Xue, Tianqing Zhu, Dayong Ye, Li Song, and Wanlei Zhou. Dp-image: Differential privacy for image data in feature space. *CoRR*, abs/2103.07073, 2021.

[20] Yong Liu, Xinghua Zhu, Jianzong Wang, and Jing Xiao. A quantitative metric for privacy leakage in federated learning. *CoRR*, abs/2102.13472, 2021.

[21] Ali Makhdoumi, Salman Salamatian, Nadia Fawaz, and Muriel Médard. From the information bottleneck to the privacy funnel. In *2014 IEEE Information Theory Workshop (ITW 2014)*, pages 501–505, 2014.

[22] David McAllester and Karl Stratos. Formal limitations on the measurement of mutual information. *CoRR*, abs/1811.04251, 2018.

[23] Richard McPherson, Reza Shokri, and Vitaly Shmatikov. Defeating image obfuscation with deep learning. *CoRR*, abs/1609.00408, 2016.

[24] Kai Packhäuser, Sebastian Gündel, Nicolas Münster, Christopher Syben, Vincent Christlein, and Andreas Maier. Is medical chest x-ray data anonymous?, 03 2021.

[25] Ben Poole, Sherjil Ozair, Aäron van den Oord, Alexander A. Alemi, and George Tucker. On variational bounds of mutual information. *CoRR*, abs/1905.06922, 2019.

[26] Andrew Michael Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Daniel Tracey, and David Daniel Cox. On the information bottleneck theory of deep learning. In *International Conference on Learning Representations*, 2018.

[27] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[28] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *CoRR*, abs/1703.00810, 2017.

[29] Jiaming Song and Stefano Ermon. Understanding the limitations of variational mutual information estimators. *CoRR*, abs/1910.06222, 2019.

[30] Jiaming Song and Stefano Ermon. Understanding the limitations of variational mutual information estimators. In *International Conference on Learning Representations*, 2020.

[31] Liangjian Wen, Yiji Zhou, Lirong He, Mingyuan Zhou, and Zenglin Xu. Mutual information gradient estimation for representation learning, 2020.

[32] Adam Yala, Homa Esfahanizadeh, Rafael GL D' Oliveira, Ken R Duffy, Manya Ghobadi, Tommi S Jaakkola, Vinod Vaikuntanathan, Regina Barzilay, and Muriel Medard. Neuracrypt: Hiding private health data via random neural networks for public training. *arXiv preprint arXiv:2106.02484*, 2021.