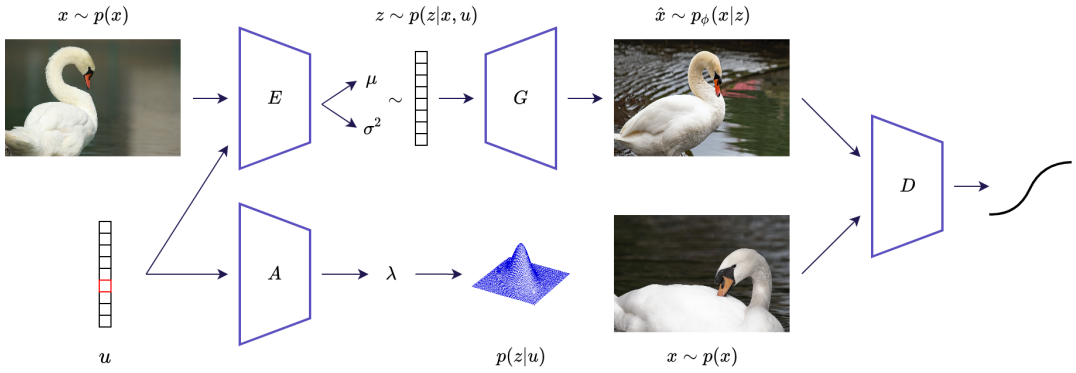
iVAE-GAN: Identifiable VAE-GAN Models for Latent Representation Learning

AAU Signal Processing and Computing Master Thesis



Bjørn Uttrup Dideriksen
Department of Electronic Systems
Aalborg University
Fredrik Bajers Vej 7
bdider16@student.aau.dk

Kristoffer Calundan Derosche
Department of Electronic Systems
Aalborg University
Fredrik Bajers Vej 7
kderos16@student.aau.dk

Zheng-Hua Tan
Department of Electronic Systems
Aalborg University
Fredrik Bajers Vej 7
zt@es.aau.dk
Supervisor

Electronics and IT
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
iVAE-GAN: Identifiable VAE-GAN Models for Latent Representation Learning

Theme:
Master thesis

Project Period:
September 2020 to June 2021

Project Group:
976

Participant(s):
Kristoffer Calundan Derosche
Bjørn Uttrup Dideriksen

Supervisor(s):
Zheng-Hua Tan

Copies: 1

Page Numbers: 77

Date of Completion:
June 10, 2021

Abstract:

Remarkable progress has been made within nonlinear Independent Component Analysis (ICA) and identifiable deep latent variable models. Formally, the latest identifiability theory enables us to recover the true latent variables up to a linear transformation by leveraging unsupervised deep learning. This is of significant importance for unsupervised learning in general as the true latent variables are of principal interest for meaningful representations. These theoretical results stand in stark contrast to the mostly heuristic approaches used for representation learning which do not provide analytical relations to the true latent variables. We extend the family of identifiable models by proposing an identifiable GAN model using variational inference we name iVAE-GAN. With iVAE-GAN we show the first principal approach to a theoretically meaningful latent space by means of adversarial training. We implement the novel iVAE-GAN architecture and prove its identifiability, which is confirmed by experiments. The GAN objective is believed to be an important addition to identifiable models as it is one of the most powerful deep generative models. We hope such work can inspire other constructions of meaningful latent spaces not based solely on heuristic approaches.

Contents

Preface

We would like to thank our supervisor, Professor Zheng-Hua Tan, whose endless enthusiasm and support fueled our work. We received more than could be expected from any supervisor, and for this we are deeply grateful.

When we started looking into the iVAE model, which heavily inspired this work, we inevitably had a few questions. These were however quickly answered by the PhD student who worked on the model, Ilyes Khemakhem. Thank you for the suggestions and inspiration we received during our correspondence.

Last but not least we would like to extend an acknowledgement to Morten Østergaard Nielsen, a PhD student at Aalborg University, who provided greatly appreciated last minute help.

Aalborg University, June 10, 2021

Kristoffer Calundan Derosche
<kderos16@student.aau.dk>

Bjørn Uttrup Dideriksen
<bdider16@student.aau.dk>

Chapter 1

Introduction

Deep learning is an interesting and promising field that has received extreme attention due to astonishing results recently shown [24]. Deep learning is also a broad field spanning different algorithms, architectures and sub-fields. Generally deep learning can be divided into three large paradigms: Supervised-, semi-supervised- and self-supervised learning, which largely pertains to the use of the available datasets during training of the network.

In supervised learning, data in the dataset must be associated with a known label, which is the desired output of the network, such that the correct output for every input is known. Training is then performed by only feeding the network the input and then, based on the output, adjust network parameters towards the correct input-output relation. In semi-supervised learning only some of the input data has an associated label and the goal for training is to leverage the limited amount of labels to achieve a better result. In self-supervised learning, or unsupervised learning (the two carry different meaning but are often used interchangeably in literature), none of the input data has labels and the goal for training is to extract meaningful representation from the raw data.

Generally, supervised learning will perform better than both semi- and unsupervised learning because the desired network behaviour can be derived from the labels, which are only partly available or non-existent in the other paradigms. However, labelling a dataset is costly. The annotations are man-made and since training both require and benefit from large datasets it is typically a long process which also increases chance of error in the labelling. Unlabelled datasets are thus comparatively cheap and luckily also very abundant. Data is collected almost everywhere and in great quantities (Big Data). Therefore, the motivation for unsupervised learning is to enable training on such large, unlabelled datasets in order to skip the time-consuming and costly process of labelling.

An interesting question could be: Given all datasets were labelled, would unsupervised learning still have relevance?
To hopefully start a train of thought, a famous quote says, "a picture is worth a thousand words". In this context that could mean that if pictures were the data, then the label for a single picture would be a large collection of words and phrases. Some labels will be relevant in one application but not others and vice versa. So even if only labelled datasets existed a time-consuming process of selecting the right labels would arise unless explicit datasets for every possible application existed. This is further complicated by recent investigation that has shown that labels themselves can be problematic when evaluating performance [21]. As shown in Figure 1.1 it was found that label ambiguity caused networks to score lower on accuracy even though both the prediction and label could be perfectly reasonable. Even situations with a wrong label but correct network prediction exist. The result after manually rectifying the errors became that the networks showed significantly better performance. The immedeate conclusion became that classification challenges, such as ImageNet, could already nearly be solved or at least better solved than the reported state-of-the-art due to imperfect or incorrect labelling.

Figure 1.1: Predictions on ImageNet. Red is the predicted label, ground-truth label is black [21].

Therefore, it could seem fair to assume that unsupervised learning could still have relevance if all datasets were labelled. Because it circumvents the human error associated with labelling that otherwise leads to the research of robust supervised learning algorithms.

Another interesting and active field of research is transfer learning. The purpose in transfer learning is to enable trained networks to adapt to other domains and applications, also called downstream tasks, than they were originally trained on. This could be of interest for a number of reasons. Adapting an already trained network to a new domain should require significantly less examples in the new domain, than if a network had to be trained from scratch. Therefore, the required datasets become smaller and thus it takes less time to train

the network. Smaller datasets are also of interest in applications where datasets are very costly to obtain or very hard to even get e.g., classifying types of soils on Mars. Then the ability to take a network trained on types of soil from Earth and then use relatively few examples of soil from Mars to transfer the knowledge from one domain to another becomes very valuable. And could even allow networks to be used in applications that would otherwise not be possible. Both supervised- and self-supervised learning can be used in transfer learning. But intuitively self-supervised learning could be expected to perform especially well. This is because self-supervised learning often involves finding features, or latent variables, which encompasses descriptive power of the seen data. Whereas supervised learning allows for specialized networks that become exceptionally good at solving a specific task, likely at the expense of generality. Therefore, it is argued that since self-supervised learning has no labels telling the network what the right answers are, they are to a larger degree driven to find some general meaning within the data. Which could make them more suitable for transfer learning. One of the biggest challenges in self-supervised learning is representation learning.

Representation Learning is about "learning representations of the data that make it easier to extract useful information when building classifiers or other predictors" [26]. Previously, and even to this date, this definition could be attributed to the task of preprocessing the input data in order to improve the performance of neural networks. This task of feature engineering has shown its merits in many different applications where the raw data is first filtered or transformed before being fed to the network. The art of feature engineering has largely been a product of human ingenuity but in the field deep learning of representations, the goal is to leverage deep learning to produce useful features. Thereby alleviating the need for tailored features. Which are not only time-consuming to produce but also typically require deep expert knowledge of the data at hand, which in turn could slow the progress and development of machine learning in various applications. A central and open question in deep learning of representations is how to define a good representation. Different types of representations exist and most with intuitively appealing objectives, such as capturing a posterior distribution over latent variables, allowing supervised predictors to perform better when the representations are used as input, disentangling factors of variation etc. [26]. A common theme is however, that these objectives have been pursued very heuristically. This is not surprising given that the objectives differ much from that seen in supervised learning, where we can obtain/label the correct answers and thus define good criteria for learning. Representation learning aims to infer meaningful information about the latent variables of datasets that, by definition, are never observed or even known. Therefore, it is not straightforward or even possible to compare two learned representations, simply because no universal metric is available - unlike supervised learning where state-of-the-art can be determined by evaluating accuracy or prediction error.

However, a very recent and promising leap in representation learning of meaningful latent variables is the understanding of identifiability in deep latent variable models [14, 15].

Identifiability is closely related to the notion of latent variables. A latent, or hidden, variable is a variable that is not directly observable. Instead, it is assumed that the observed data is a function of the latent variables. There are many reasons why it could be of interest to infer the latent variables from observed data. The latent variables are arguably one of the most complete representations of data as they hold all the information necessary to generate the data. The latent variables can also be very powerful for explaining causation in observed data as well as dimensionality reduction. As an example, the mean and variance of a Normal distribution could be considered latent variables. The mean and variance are never directly observed, only samples which are a function of the mean and variance can be observed. But inferring the mean and variance from observed data can be used to represent an infinite amount of samples and important aspects of the data can be explained from the mean and variance. Models which aim to infer the latent variables from observed data are broadly called latent variable models.

The recent identifiability results in deep latent variable models enables learning of the true latent variables up to a linear transformation [15]. The identifiability results require a prior distribution conditioned on an additionally observed variable, but this is shown to be a quite mild requirement. This is, to the best of our knowledge, the first line of work to produce rigorous proof of identifiability and a principled approach to disentanglement in representation learning and will form the basis of this Master Thesis.

1.1 Problem Formulation

One of the biggest challenges in machine learning is unsupervised representation learning. Most unsupervised models are heuristically derived and provide no analytical relation to the true latent variables they aim to infer. Recent identifiability results in deep latent variable models have paved the way for a theoretically rigorous and principled approach to representation learning. The theoretical results are very general yet only a few estimation models have been proposed [15, 14, 20].

We hypothesize that identifiability can be proven for the deep generative model GAN using variational inference thereby extending the existing family of identifiable deep latent variable models.

Proving identifiability in GANs is of interest because GAN is one of the most powerful generative models and by proving identifiability in a GAN model a flexible framework that can simultaneously i) recover the original latent variables, ii) approximate the seen data distribution and iii) generate novel samples is obtained. In the following chapter we will outline our proposed architecture iVAE-GAN.

Chapter 2

Proposed iVAE-GAN architecture

In this chapter we will briefly introduce our proposed iVAE-GAN architecture. An illustration of the proposed iVAE-GAN architecture can be seen on Figure 2.1 and the paper we have submitted to the NeurIPS 2021 conference [18] can be found in Appendix B.

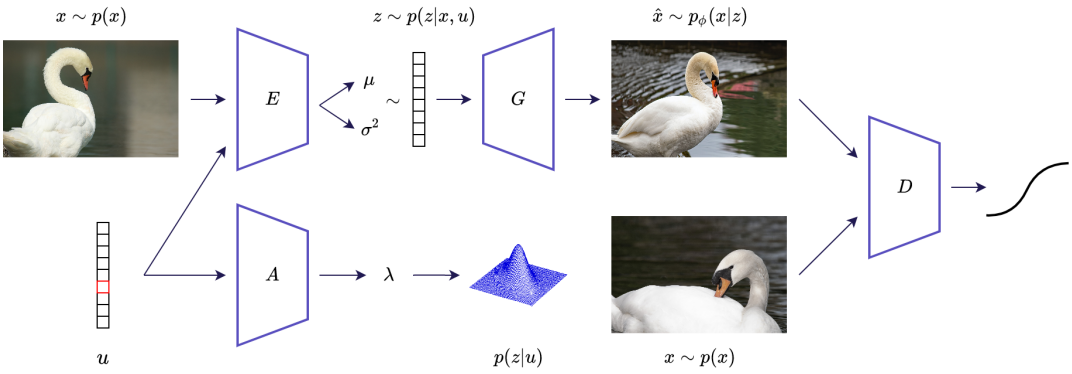


Figure 2.1: The proposed iVAE-GAN architecture. The deep latent variable model consists of four neural networks: an encoder E that learns the latent variables, a decoder/generator G that generates data in the original data space, a discriminator D that discriminates generated data from real data, and an auxiliary network A that learns the natural parameters, $\lambda_i(u)$, of an exponential family from the observed auxiliary variables.

The data, x, is presented to the encoder, E, along with an auxiliary variable u, the purpose of which shall become clear in Chapter 3 on page 7. The encoder uses the input to produce a mean and variance which is used to sample a latent variable, z, from the latent space using the reparameterization trick [16]. The latent variable is then passed through the generator, G, which produces an output which is classified as either real or fake by the discriminator, D. The process of encoding data, sampling a latent variable using the reparameterization

trick and decoding the latent variable is very similar to the working principle of a Variational Autoencoder (VAE) which is why VAE is part of the architecture name. Producing a sample from a latent variable with the goal of fooling a discriminator to mistake the generated output for real data is the working principle of GAN which, to no surprise, is why GAN is part of the name. It can be seen that the two models overlap at the generator where a normal VAE would use a reconstruction loss instead of an adversarial loss and GAN would not normally sample the latent space with variational inference. Thus, we named the architecture VAE-GAN. The i for identifiability in iVAE-GAN trivially implies the purpose of our work and is to a large extent associated with the auxiliary variable u, the auxiliary network, A, and the prior distribution $p(z|u)$. During training the encoded distribution $p(z|x, u)$ is forced towards the prior distribution $p(z|u)$ which in turn depends on the parameter λ, which will also be introduced in Chapter 3 on the next page, that is learnt by the auxiliary network, A, from the auxiliary variable, u. Through a combination of proper construction of the prior distribution and training guarantees of the iVAE-GAN model we will in the next chapter prove how our novel iVAE-GAN model is identifiable.

Chapter 3

Theory

In this chapter we describe the theoretical foundation and findings of this project. Specifically, we will show the recent theoretical framework of *identifiability* in the context of deep latent variable models and non-linear Independent Component Analysis (ICA). This will be the basis on which we subsequently show that iVAE-GAN is also an identifiable deep latent variable model by combining relevant proofs from both GAN and VAE.

3.1 Identifiability

Identifiability has origins in early econometrics as shown by the ”problem of confluent relations (or problem of arbitrary parameters)” [8]. This came at a time where much skepticism about the use of statistics within the field of economics existed - which is ironic because much of modern economic theory is deeply rooted in statistical methods. The notion of identifiability, or confluent relations as it was called, seemed to be motivated by the observation that many economic models were derived by a ”passive observer”. By passive observer it was understood that very few controlled experiments were used to test or validate the developed models. This was in contrast to other, mostly scientific, fields where a vital part of hypothesis testing and model design included carefully designing controlled experiments with which the hypothesis could be examined and thus it was common in other fields to not only passively observe data but also actively design representative experiments. The latter part seemed to be near non-existent in econometrics either because of difficulties in designing representative experiments or it simply was not a used practice. Therefore, they became passive observers as they only used the observations that could be found in e.g., existing price histories to derive useful models but rarely designed experiments that could replicate or explain the phenomena. The formulation of confluent relations thus became of interest because it states

that if two or more parametrizations of the same model lead to the same joint distribution over observed random variables they are indistinguishable on the basis of observations and therefore *unidentifiable*. Therefore, if an unidentifiable model were used to describe the observations there would exist many different parametrizations of that model that would be equally descriptive and thus conclusions based on any parametrization can not be said to be unambiguous. For example, if an unidentifiable model was used to explain an increase in consumption of some commodity and one parametrization attributed it to an increase in demand but another attributed it to an increase in disposable income either explanation would be equally likely, but the decision based upon them may be very different. If an increase in disposable income was the cause a similar increase in consumption could be expected in other commodities. However, if an increase in demand was the cause it could be expected that the consumption will decrease in some other commodities since the buyers do not have more money, they simply spend it differently. If an identifiable model could have been used to explain the increase in consumption, there would be no such ambiguity because only one parametrization of the model would exist.

We argue that in unsupervised learning we, as researchers, are oftentimes passive observers because the data we consider are not the product of our own experiments. Instead, it is typically existing data for which we would like to infer some meaningful representations that allows us to explain and draw meaningful conclusions from the data. In order to do so faithfully it would make sense to be cautious of unidentifiable models and instead harvest the benefits of identifiable models.

Up until recently, the general consensus in literature agreed that arbitrary nonlinear functions, such as those modeled by neural networks, were almost surely unidentifiable under the assumption of independent latent variables [12]. It is now understood that identifiability results can be achieved if the model assumes *conditionally* independent latent variables. That is, given an additionally observed variable under which the latent variables are independent, the latent variables may be estimated up to a linear transformation and in certain cases reduced to a simple scaled permutation. The nonlinear functions that map the observed data to latent variables need not preserve dimensionality, but if they do it is worth noting that the identifiability results become interpretable as nonlinear Independent Component Analysis (ICA) [12].

Formally, a model is said to be identifiable if only one parametrization of the model can lead to the observed data distribution, i.e.

$$if \quad p_{\boldsymbol{\theta}_1}(\mathbf{x}) = p_{\boldsymbol{\theta}_2}(\mathbf{x}) \quad \Rightarrow \quad \boldsymbol{\theta}_1 = \boldsymbol{\theta}_2 \qquad \forall(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) \tag{3.1}$$

On the other hand if $p_{\boldsymbol{\theta}_1}(\mathbf{x}) = p_{\boldsymbol{\theta}_2}(\mathbf{x})$ but the parametrization is not unique, $\boldsymbol{\theta_1} \neq \boldsymbol{\theta_2}$, the model is said to be unidentifiable on the basis of observations.

In our case we wish to define an identifiable deep latent variable model, which means that the joint distribution over observed variables, $\mathbf{x}$, and latent variables, $\mathbf{z}$, should be identifiable. Latent variable models like those used in e.g. variational autoencoders (VAE) commonly model the joint distribution as:

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z}) \tag{3.2}$$

for $\mathbf{x} \in \mathbb{R}^d$, and $\mathbf{z} \in \mathbb{R}^n$ (lower-dimensional, $n \leq d$), but only provide training guarantees on the marginal distribution of the model (such as learning a lower bound on the observed marginal distribution):

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})\,\mathrm{d}\mathbf{z} \tag{3.3}$$

Unfortunately, deep latent variable models as in Equation (3.2) do not learn the true joint distribution from the marginal distribution and as a result can not recover the original latent variables. In contrast, it is sufficient for identifiable models to learn the marginal distribution because only one parametrization of the joint distribution produces the seen marginal distribution and therefore the latent variables can be recovered. The reason models such as Equation (3.2) are not identifiable can be understood through the "impossibility result" [17]. It shows that even under the assumption that $p(\mathbf{z})$ admits any distribution for which the densities are independent i.e. $p(\mathbf{z}) = \prod_{i=1}^{n} p(z_i)$ there exists an infinite family of bijective functions, f, such that:

$$P(\mathbf{z} \leq \mathbf{u}) = P(f(\mathbf{z}) \leq \mathbf{u}) \quad \forall \mathbf{u} \in \text{supp}(\mathbf{z}) \tag{3.4}$$

Which means that $p(\mathbf{z}) = p(\hat{\mathbf{z}})$, where $p(\hat{\mathbf{z}}) = f(\mathbf{z})$. This would also imply that the latent variable model in Equation (3.2) will have the same marginal distribution over x since:

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} = \int p(\mathbf{x}|\hat{\mathbf{z}})p(\hat{\mathbf{z}})\,\mathrm{d}\hat{\mathbf{z}} \tag{3.5}$$

Therefore the model is unidentifiable because there are more than one parametrization, $\mathbf{z}$ and $\hat{\mathbf{z}}$, that are observationally identical. This shows that models with factorized priors are fundamentally unidentifiable by construction and require some form of inductive bias if they are to become identifiable. In the following we shall introduce a recently discovered identifiable deep latent variable model.

Identifiable Model. In [15, 14] a very general deep latent variable model has been derived that is identifiable up to linear equivalence relations. The pivotal distinction between their

model and the one in Equation (3.2) on the previous page is that the prior distribution is conditioned on an additional observed variable, u. As we will show, this is sufficient inductive bias to prove that the model becomes identifiable up to the mentioned linear equivalence relation. The general form of the identifiable model is:

$$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}|\mathbf{u}) = p_{\mathbf{f}}(\mathbf{x}|\mathbf{z})p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u}) \tag{3.6}$$

where $\mathbf{u} \in \mathbb{R}^m$ is the auxiliary variable observed alongside the data and $\boldsymbol{\theta} = (\mathbf{f}, \mathbf{T}, \boldsymbol{\lambda})$ are the parameters of the conditional generative model. The conditional latent distribution, $p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u})$, is assumed to belong to an exponential family of conditionally independent variables:

$$p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u}) = \prod_{i}^{n} \frac{Q_i(z_i)}{Z_i(\mathbf{u})} \exp\left[\sum_{j=1}^{k} T_{i,j}(z_i)\lambda_{i,j}(\mathbf{u})\right] \tag{3.7}$$

where $Q_i(z_i)$ is the base measure, $Z_i(u)$ is the normalization coefficient, $T_{i,j}(z_i)$ are the sufficient statistics and $\lambda_{i,j}(u)$ are the natural parameters of the family. The natural parameters of the distribution depending on $\mathbf{u}$ is learnt by the network we name A (for Auxiliary) in our implementation. The assumption that the latent distribution must belong to an exponential family is not considered restrictive as it has been shown to have universal approximation capabilities in [23]. The decoder, $p_{\mathbf{f}}(\mathbf{x}|\mathbf{z})$, is defined as:

$$p_{\mathbf{f}}(\mathbf{x}|\mathbf{z}) = p_{\boldsymbol{\epsilon}}(\mathbf{x} - \mathbf{f}(\mathbf{z})) \tag{3.8}$$

allowing $\mathbf{x}$ to be decomposed into $\mathbf{x} = \mathbf{f}(\mathbf{z}) + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is a noise variable independent of $\mathbf{z}$ or $\mathbf{f}$ and distributed according to $p_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon})$. An example of such a distribution, $p_{\boldsymbol{\epsilon}}(\cdot)$, and decomposition could be a Normal distribution with mean $\mathbf{f}(\mathbf{z})$ and known variance $\boldsymbol{\epsilon}$. Then we can write $p_{\mathbf{f}}(\mathbf{x}|\mathbf{z})$ as:

$$\mathbf{X} \sim \mathcal{N}(f(z), \epsilon) \quad \Leftrightarrow \quad p_f(x|z) = \frac{1}{\sqrt{2\pi\epsilon^2}} e^{\frac{-(x-f(z))^2}{2\epsilon^2}} \tag{3.9}$$

For which it may be seen that $p_f(x|z)$ is only a function of $x - f(z)$, as seen in the exponent, and therefore we can write it as $p_{\boldsymbol{\epsilon}}(x - f(z))$, where the subscript $\boldsymbol{\epsilon}$ indicates that it holds true for a known variance, $\boldsymbol{\epsilon}$. The decomposition can be shown using the reparameterization trick [16]. This trick shows a neat alternative way to sample from a Normal distribution:

$$x = f(z) + \epsilon \cdot y \quad where \quad \mathbf{Y} \sim \mathcal{N}(0, 1) \quad \Leftrightarrow \quad \mathbf{X} \sim \mathcal{N}(f(z), \epsilon) \tag{3.10}$$

Therefore the decomposition may be written as $x = f(z) + \epsilon$ where the noise variable is distributed such that $E \sim \mathbf{Y} \cdot \epsilon$. Multiplying a Normal random variable, $\mathbf{Y}$, with a constant, ϵ, results in a new Normal variable, $\mathbf{V}$, distributed according to $\mathcal{N}(\epsilon \cdot 0, \epsilon^2 \cdot 1)$ and therefore $p_\epsilon(\varepsilon)$ becomes:

$$p_\epsilon(\varepsilon) = \frac{1}{\sqrt{2\pi\epsilon^2}} e^{\frac{-(\varepsilon)^2}{2\epsilon^2}} \tag{3.11}$$

The interpretation of this construction is that the decoder in the generative model outputs the mean of a Normal distribution used to sample the output. It is worth noting that in a probabilistic model such as the one we are considering the output of the decoder should also always be probabilistic. However, more often than not we have no interest in thinking of the output of the decoder as the mean of a distribution and even less interested in having to sample at the output of our network as that is problematic with respect to optimization. Instead, we would like to think of the output of the decoder as if it was a generated sample. Here the most common choice by far is to assume the output distribution of the decoder to be a Normal distribution with mean $\mathbf{f}(\mathbf{z})$ and identity covariance I, which is also a valid choice in the decoder we have introduced. If the output distribution is chosen to have identity covariance we can show that the log probability of seeing a sample under the assumed model is proportional to the mean squared error plus some constant:

$$\begin{aligned}\log p_{\mathbf{f}}(\mathbf{x_i}|\mathbf{z_i}) &= \log\left(\frac{1}{\sqrt{(2\pi)^d|I|}} e^{-\frac{1}{2}(\mathbf{x_i}-\mathbf{f}(\mathbf{z_i}))^T I(\mathbf{x_i}-\mathbf{f}(\mathbf{z_i}))}\right) \\ &= -\frac{1}{2}||\mathbf{x_i} - \mathbf{f}(\mathbf{z_i})||^2 + \log\frac{1}{\sqrt{(2\pi)^d|I|}}\end{aligned} \tag{3.12}$$

Since we would like to maximize the probability of seeing a sample under our model this would correspond to driving the model to output $\mathbf{f}(\mathbf{z_i})$ that are as close to $\mathbf{x_i}$ as possible. Therefore, we can easily interpret the output of the decoder as generated samples instead of means and we avoid the need to sample from a distribution altogether. As noted in the original paper a property of the decoder model introduced in Equation (3.8) on the facing page is that it includes the non-noisy case if $p_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon})$ is chosen to be a Gaussian with infinitesimal variance such that $\mathbf{x} = \mathbf{f}(\mathbf{z})$ [15]. This is of importance for flow-based generative models, but we will not make further use of this property.

We place a rather intuitive assumption on the function $\mathbf{f}(\cdot)$, namely that the map $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^d$ should be injective. This is intuitive because the ultimate goal of our endeavor is to be able to recover the latent variables, $\mathbf{z}$, from the observed data, $\mathbf{x}$. If the function that maps the latent variable into the observed variables was not injective, meaning that each latent variable

maps to no more than one observed variable, we would have no hope of recovering the true latent variables. For example if $f(z) = z^2$, which is a surjection, we could never hope to learn the true latent variable, z, that generated the observed data as there will always be two distinct values of z that would give rise to the same observation (with the exception of $z = 0$). Therefore, if the data we observe is not the result of some injective function the problem is ill-posed and we can not hope to achieve identifiability. The reason we assume the function to be injective and not bijective is because a bijective function is an invertible function and such functions must map to and from the same dimension. In our case the dimension of the latent space and data space need not be the same as $n \leq d$. However, if $n = d$ the function $\mathbf{f}$ could become a bijection at which point the theory coincides with the definition of non-linear ICA, which is why the original authors has also highlighted their findings as very significant for non-linear ICA.

The auxiliary variable $\mathbf{u}$ of the exponential distribution is pivotal to the definition of the identifiable model as seen above. The crux of the matter when choosing the auxiliary variable is that the latent variables should be conditionally independent given $\mathbf{u}$ as seen from Equation (3.7) on page 10. Therefore, the key difference between the identifiable model and the unidentifiable model is the auxiliary variable. By introducing this variable, the model does not rely on factorized priors, which the impossibility result show will always be seriously unidentifiable, instead the identifiable model has conditionally factorized priors. Naturally, since the auxiliary variable should be observed this has implications for the used dataset. The dataset must contain pairwise observations of the data, $\mathbf{x}$, and $\mathbf{u}$ such that $\mathcal{D} = \{(\mathbf{x}^{(1)}, \mathbf{u}^{(1)}), \ldots, (\mathbf{x}^{(N)}, \mathbf{u}^{(N)})\}$. Since it is crucial to the identifiability of the model that $\mathbf{u}$ is chosen such that the latent variables become conditionally independent it would be reasonable to pose the question: How can we know the latent variables, which by definition are never observed, are independent given $\mathbf{u}$?

For most applications, the simple answer is that we can not know for certain that the latent variables become independent given $\mathbf{u}$. The choice of $\mathbf{u}$ will therefore often be application specific and rely on knowledge of the used data. Fortunately, there has yet to be reported any difficulty in choosing the auxiliary variable in relevant literature and the general claim is that the auxiliary variable can be chosen relatively freely to satisfy Equation (3.7) on page 10. Examples of choices for the auxiliary variable include the segment index of a time series divided into segments which is the basis of Time-Contrastive Learning (TCL) [11], a random sample from the same time series which is used in Permutation-Contrastive Learning (PCL) [10] or even a combination of the two. The labels of existing datasets such as EMNIST has also been used [20]. Thus, there exists different methods for obtaining the auxiliary variable which could either be existing labels or more heuristic approaches like segmenting a time series.

Equivalence relation. The newly discovered identifiability result we make use of in our novel iVAE-GAN model is the equivalence relation shown in [15, 14]. The equivalence relation is defined as:

$$(\mathbf{f}, \mathbf{T}, \boldsymbol{\lambda}) \sim (\tilde{\mathbf{f}}, \tilde{\mathbf{T}}, \tilde{\boldsymbol{\lambda}}) \quad \Leftrightarrow \quad \exists A, \mathbf{c} \mid \mathbf{T}(\mathbf{f}^{-1}(\mathbf{x})) = A\tilde{\mathbf{T}}(\tilde{\mathbf{f}}^{-1}(\mathbf{x})) + \mathbf{c}, \quad \forall x \in \mathcal{X} \tag{3.13}$$

where $\sim$ denotes the equivalence relation, A is an $nk \times nk$ invertible matrix and $\mathbf{c}$ is an nk-dimensional vector.

What this equivalence relation establishes is that the true latent variables, which by definition are equal to $\mathbf{z} = \mathbf{f}^{-1}(\mathbf{x})$, are related to the recovered latent variables, $\tilde{\mathbf{f}}^{-1}(\mathbf{x})$, by a linear transformation. Specifically, we see that the true sufficient statistic, which is a point-wise transformation [15], of the exponential family is a linear combination of the estimated sufficient statistic.

This is to the best of the authors knowledge the only identifiability proof in literature that has been able to explicitly prove a relation between the estimated latent variables and the true latent variables in a deep latent variable model. The line of work presented above has also been noted as a principled approach to disentanglement due to the rigorous theoretical framework that leads to this equivalence relation. In the following section we shall introduce the relevant theory for our model, iVAE-GAN, and prove that the equivalence in Equation (3.13) also applies to our new model such that we develop the first identifiable GAN.

3.2 iVAE-GAN

iVAE-GAN has a hybrid loss function that consists of a divergence loss, as known from VAEs, for the encoding of the latent space with respect to the prior (conditional) distribution and an adversarial loss for the generated samples such that:

$$\mathcal{L}_{iVAE-GAN} = \mathcal{L}_{prior} + \mathcal{L}_{GAN} \tag{3.14}$$

where we define:

$$\mathcal{L}_{prior} = -KL(q_\phi(z|x,u)||p_\theta(z|u)) \tag{3.15}$$

and

$$\mathcal{L}_{GAN} = V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim q_\phi(z|x,u)}[\log(1-D(G(z)))] \tag{3.16}$$

Training is then performed according to:

$$\begin{aligned}\max_\phi \mathcal{L}_{prior} + \min_G \max_D \mathcal{L}_{GAN} \\ = \max_\phi -KL(q_\phi(z|x,u)||p_\theta(z|u)) + \min_G \max_D V(G,D)\end{aligned} \tag{3.17}$$

Based on the loss function the iVAE-GAN architecture can either be thought of as a VAE (more specifically the iVAE [15]) where the reconstruction loss is replaced with an adversarial loss or it can be thought of as a regular GAN to which a variational latent encoding is added, whichever the reader prefers. The former is beneficial as an intuitive way to understand identifiability in iVAE-GAN since the iVAE is identifiable and the adversarial loss can be viewed as a substitute for the reconstruction loss. The latter can be used as an argument to argue that the identifiability results we show in GAN likely extend to a wide variety of GAN flavors since we place no assumptions or restrictions on the adversarial loss. Thus, most other GANs can easily replace $\mathcal{L}_{GAN}$ without loss of generality. Therefore our findings are also of significant interest to the remaining active research field of GANs and their applications.

In the following we show that the loss is a lower bound on the difference between the log probability of the data and the expected log likelihood of the data generated by the decoder:

$$\log p(x) - \mathbb{E}_{z\sim q_\phi(z|x,u)}[\log p_\Phi(x|z)] \geq \mathcal{L}_{iVAE-GAN} \tag{3.18}$$

and that by maximizing $\mathcal{L}_{iVAE-GAN}$ the expected log likelihood of the data generated by the decoder approaches the log probability of the data.

The first term in our loss is a maximization over a negative KL divergence for which it is relatively straightforward to recognize that the maximum will be zero, since a KL divergence

is always non-negative and we are dealing with a negated version. A KL divergence is a measure that quantifies the dissimilarity between two distributions. Apart from its non-negative property the KL divergence equals zero if and only if the two distributions are identical. Therefore we can recognize that the effect of maximizing the KL divergence between the two distributions $q_\phi(z|x,u)$ and $p_\theta(z|u)$ will be to force our encoded distribution, $q_\phi(z|x,u)$, to become more similar to the prior distribution, $p_\theta(z|u)$. The second term in the loss function is harder to interpret at first glance, since that term contains a sum of two expectations which is to be both maximized and minimized. Our first step towards showing that the loss function is a lower bound is to prove that the second term can be rewritten to the following (under the assumption of an optimal discriminator):

$$\min_G \max_D \mathcal{L}_{GAN} = \min_G V(G,D) = \min_G -\log(4) + 2 \cdot JSD(p(x)||p_\Phi(x)) \tag{3.19}$$

The two-player minimax game, $\mathcal{L}_{GAN}$, is defined as in [7] and the derivation we show in the following is an explicit proof of Theorem 1. $\mathcal{L}_{GAN}$ is formulated as a two-player minimax game according to:

$$\min_G \max_D V(G,D) = \min_G \max_D \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim q_\phi(z|x,u)}[\log(1 - D(G(z)))] \tag{3.20}$$

To simplify the derivations that are to follow, the second term is rewritten using the *law of the unconscious statistician*:

$$\mathbb{E}_{f_Y}[Y] = \mathbb{E}_{f_{h(W)}}[h(W)] = \mathbb{E}_{f_W}[h(W)] \tag{3.21}$$

where f is a probability density function. By choosing $h(W) = G(z)$ such that $Y = G(z) = x$ we rewrite the expectation over z in Equation (3.20) to be an expectation over x:

$$\min_G \max_D V(G,D) = \min_G \max_D \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{x\sim p_\Phi(x)}[\log(1 - D(x))] \tag{3.22}$$

To express this in a way where the behavior of the generator can be examined an optimal discriminator is assumed. This means that conceptually the discriminator is allowed to converge to an optimal discriminator at every step in the training. The optimal discriminator will be the discriminator that maximizes:

$$C(G) = \max_D V(G,D) = \max_D \int_x p_{data}(x)\log(D(x)) + p_\Phi(x)\log(1 - D(x))\,\mathrm{d}x \tag{3.23}$$

This integrand can be recognized as a function of the form $f(y) = a\log(y) + b\log(1-y)$ which attains a maximum at $y = \frac{a}{a+b}$, which implies that the optimal discriminator, $D^*(x)$, is given by:

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_\Phi(x)} \tag{3.24}$$

We can use this result to check what happens if the generator learns the data distribution perfectly i.e. $p_\Phi = p_{data}$. The optimal discriminator would then become $D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x)+p_{data}(x)} = \frac{1}{2}$ and by Equation (3.22) on the previous page we can find the optimum value at which the generated data will be indistinguishable from the true data:

$$V^*(G, D) = \mathbb{E}_{x\sim p_{data}(x)}[\log(\frac{1}{2})] + \mathbb{E}_{x\sim p_\Phi(x)}[\log(1 - \frac{1}{2})] = -\log 4 \tag{3.25}$$

By inserting the expression for the optimal discriminator in Equation (3.22) on the preceding page we can derive the expression, $V(G, D^*)$, which the generator tries to minimize. Furthermore, if we can show that the minimum of $V(G, D^*)$ is $-\log 4$ we can conclude that the optimal generator will be the generator that perfectly learns to generate samples from the seen data distribution. The expression for $V(G, D^*)$ can be written as:

$$\min_G V(G, D^*) = \min_G \int_x p_{data}(x) \log(\frac{p_{data}(x)}{p_{data}(x) + p_\Phi(x)}) + p_\Phi(x) \log(1 - \frac{p_{data}(x)}{p_{data}(x) + p_\Phi(x)})\, \mathrm{d}x \tag{3.26}$$

The first term can be recognized as a Kullback-Leibler divergence, since $KL(p_{data}||p_{data} + p_\Phi) = \int_x p_{data}(x) \log(\frac{p_{data}(x)}{p_{data}(x)+p_\Phi(x)})\, \mathrm{d}x$, and the second term can also be rewritten to a KL divergence by using:

$$1 - \frac{p_{data}(x)}{p_{data}(x) + p_\Phi(x)} = \frac{p_{data}(x) + p_\Phi(x)}{p_{data}(x) + p_\Phi(x)} - \frac{p_{data}(x)}{p_{data}(x) + p_\Phi(x)} = \frac{p_\Phi(x)}{p_{data}(x) + p_\Phi(x)} \tag{3.27}$$

Therefore Equation (3.26) can be written as:

$$\min_G V(G, D^*) = \min_G \; KL(p_{data}||p_{data} + p_\Phi) + KL(p_\Phi||p_{data} + p_\Phi) \tag{3.28}$$

The last step of the proof is achieved by rewriting the equation such that the two KL divergences can be expressed as a Jensen-Shannon divergence between p_{data} and p_Φ by multiplying the equation with $\frac{\log(2)}{\log(2)}$ and distributing terms:

$$\min_G -\log(4) + KL(p_{data}||\frac{p_{data} + p_\Phi}{2}) + KL(p_\Phi||\frac{p_{data} + p_\Phi}{2}) = \min_G -\log(4) + 2 \cdot JSD(p_{data}||p_\Phi) \tag{3.29}$$

Since the Jensen-Shannon divergence is always non-negative and attains a minimum only when $p_{data} = p_\Phi$, it is concluded that $\min_G V(G, D^*) = V^*(D, G) = -\log(4)$ if and only if $p_{data} = p_\Phi$. In other words, the global optimum of adversarial training, under the

assumption of an optimal discriminator, occurs only when the generated data follows the same distribution as that of the observed data. Thereby we have now obtained a new and more interpretable expression for $\mathcal{L}_{GAN}$ as seen in Equation (3.29) on the facing page. The Jensen-Shannon divergence is, like the KL divergence, a measure of how similar two distributions are. Although the JS divergence and KL divergence essentially measure the same thing there are some notable differences between the two, including that the JS divergence is symmetric, but this is not of further relevance in this context. Instead, we can now interpret the iVAE-GAN loss, $\mathcal{L}_{iVAE-GAN} = \mathcal{L}_{prior} + \mathcal{L}_{GAN}$, as a function of two divergences. One divergence that forces the encoded distribution towards the prior distribution and one divergence that forces the generated distribution towards the distribution of the observed data. This result is now used to derive the iVAE-GAN lower bound.

iVAE-GAN lower bound. $\mathcal{L}_{prior}$ of the iVAE-GAN loss is related to the ELBO loss [16] such that:

$$\log p(x) - \mathbb{E}_{z\sim q_\phi(z|x,u)}[\log p_\Phi(x|z)] \geq -KL(q_\phi(z|x,u)||p_\theta(z|u)) = \mathcal{L}_{prior} \tag{3.30}$$

Now we show that the same inequality is also fulfilled by $\mathcal{L}_{prior} + \mathcal{L}_{GAN}$, but in contrast to Equation (3.30) the data distribution may be learnt by maximizing $\mathcal{L}_{prior} + \mathcal{L}_{GAN}$. We use the result of Equation (3.29) on the facing page to write the optimization of $\mathcal{L}_{GAN}$ as:

$$\min_G \max_D \mathcal{L}_{GAN} = \min_G V(G, D^*) = \min_G -\log(4) + 2 \cdot JSD(p(x)||p_\Phi(x)) \tag{3.31}$$

To write our loss function as a function that is only to be maximized we pose the minimization over G as a maximization:

$$\min_G V(G, D^*) \equiv \max_G -V(G, D^*) = \max_G \log(4) - 2 \cdot JSD(p(x)||p_\Phi(x)) \tag{3.32}$$

Since we mean to maximize this function using a deep neural network the constant $\log(4)$ is inconsequential to the loss function. Therefore:

$$\min_G V(G, D^*) \equiv \max_G -2 \cdot JSD(p(x)||p_\Phi(x)) \tag{3.33}$$

The next step may be understood using the transitive property of inequalities that says:

$$if \quad y \geq x \quad \& \quad x \geq w \quad \Rightarrow \quad y \geq w \tag{3.34}$$

Similarly, for a variable u that can vary between 0 and any non-negative number we always have that:

$$x \geq x - u \qquad \forall u \quad 0 \leq u \leq c, \quad c \in \mathbb{R}_{0+} \tag{3.35}$$

Notice that it is important for the inclusive inequality that u has a lower limit of 0, otherwise it would be a strict inequality. The Jensen-Shannon divergence fulfills the conditions for such a variable u since it is always non-negative, therefore:

$$-KL(q_\phi(z|x,u)||p_\theta(z|u)) \geq -KL(q_\phi(z|x,u)||p_\theta(z|u)) - 2 \cdot JSD(p(x)||p_\Phi(x)) \tag{3.36}$$

Thus we recover our lower bound by using the transitive property of inequalities combined with Equations 3.30 and 3.36 to write:

$$\log p(x) - \mathbb{E}_{z\sim q_\phi(z|x,u)}[\log p_\Phi(x|z)] \geq \underbrace{-KL(q_\phi(z|x,u)||p_\theta(z|u))}_{\mathcal{L}_{prior}} \underbrace{- 2 \cdot JSD(p(x)||p_\Phi(x))}_{V(G,D^*)} \tag{3.37}$$

The right-hand side of Equation (3.37) can be recognized as the iVAE-GAN loss (updated according to Equation (3.31) on the preceding page and (3.33)):

$$\mathcal{L}_{iVAE-GAN} = \mathcal{L}_{prior} - V(G, D^*) \tag{3.38}$$

Thus, we conclude that the iVAE-GAN loss is a lower bound on the difference between the log probability of observed data and expected log likelihood of the data generated by the decoder. To, hopefully, make Equation (3.37) a little more interpretable we can make use of Jensen's inequality:

$$\mathbb{E}[\log(X)] \leq \log \mathbb{E}[X] \tag{3.39}$$

Therefore we can write:

$$\begin{aligned} \log p(x) - \mathbb{E}_{z\sim q_\phi(z|x,u)}[\log p_\Phi(x|z)] &\leq \log p(x) - \log \mathbb{E}_{z\sim q_\phi(z|x,u)}[p_\Phi(x|z)] \\ &= \log p(x) - \log \int p_\Phi(x|z) p_\phi(z)\, \mathrm{d}z = \log p(x) - \log p_\Phi(x) \end{aligned} \tag{3.40}$$

By using the transitive property of inequalities again we may write the lower bound as:

$$\log p(x) - \log p_\Phi(x) \geq -KL(q_\phi(z|x,u)||p_\theta(z|u)) - 2 \cdot JSD(p(x)||p_\Phi(x)) \tag{3.41}$$

The Jensen-Shannon divergence measures the distance between two distributions and is therefore closely related to the difference of log probabilities, so as the lower bound is maximized the difference between the log probabilities is minimized. In fact, the only condition for which the Jensen-Shannon divergence vanishes is $p(x) = p_\Phi(x)$, at which point the left-hand side becomes zero and the lower bound becomes:

$$0 \geq -KL(q_\phi(z|x,u)||p_\theta(z|u)) \tag{3.42}$$

Which is of course the normal bound for the negative KL divergence. Therefore, by maximizing $\mathcal{L}_{iVAE-GAN}$ we learn the data distribution while simultaneously maximizing the negative KL divergence between the encoded distribution, $q_\phi(z|x,u)$, and the prior distribution, $p_\theta(z|u)$.

We shall now use this result to prove identifiability in our novel iVAE-GAN architecture. Namely, because our lower bound allows us to jointly learn the prior distribution and data distribution we can now show that the parameters of our identifiable model will fulfill the equivalence relation shown in Equation (3.13) on page 13 and thus the recovered latent variables will be related to the true latent variables by a linear transformation.

Identifiability result. Given the data we observe is the marginal distribution of some generating process with a joint distribution conditioned on $\mathbf{u}$ with the true parameters $(\mathbf{f}, \mathbf{T}, \boldsymbol{\lambda})$, such that the marginal distribution is given by:

$$p_{(\mathbf{f},\mathbf{T},\boldsymbol{\lambda})}(\mathbf{x}|\mathbf{u}) = \int p_{\mathbf{f}}(\mathbf{x}|\mathbf{z}) p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u}) \, \mathrm{d}\mathbf{z} \tag{3.43}$$

and a deep generative model of the same form, iVAE-GAN, learns to approximate the marginal distribution of observed data with parameters $(\tilde{\mathbf{f}}, \tilde{\mathbf{T}}, \tilde{\boldsymbol{\lambda}})$ such that:

$$p_{(\mathbf{f},\mathbf{T},\boldsymbol{\lambda})}(\mathbf{x}|\mathbf{u}) = p_{(\tilde{\mathbf{f}},\tilde{\mathbf{T}},\tilde{\boldsymbol{\lambda}})}(\mathbf{x}|\mathbf{u}) \tag{3.44}$$

Then, under appropriate assumptions, $(\mathbf{f}, \mathbf{T}, \boldsymbol{\lambda}) \sim (\tilde{\mathbf{f}}, \tilde{\mathbf{T}}, \tilde{\boldsymbol{\lambda}})$ where $\sim$ is the equivalence relation of Equation (3.13) on page 13.

We can recall that the decoder is defined to include noise i.e. $p_{\mathbf{f}}(\mathbf{x}|\mathbf{z}) = p_{\boldsymbol{\epsilon}}(\mathbf{x} - \mathbf{f}(\mathbf{z}))$. The first step of the proof shows that the noise-free distributions, $\tilde{p}$ as they will be shown below, are also equal. We start by writing the marginal distributions of the generative models:

$$\int_{\mathcal{Z}} p_{\mathbf{f}}(\mathbf{x}|\mathbf{z}) p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u}) \, \mathrm{d}\mathbf{z} = \int_{\mathcal{Z}} p_{\tilde{\mathbf{f}}}(\mathbf{x}|\mathbf{z}) p_{\tilde{\mathbf{T}},\tilde{\boldsymbol{\lambda}}}(\mathbf{z}|\mathbf{u}) \, \mathrm{d}\mathbf{z} \tag{3.45}$$

Substituting the decoder with the definition from Equation (3.8) on page 10 yields:

$$\int_{\mathcal{Z}} p_{\boldsymbol{\epsilon}}(\mathbf{x} - \mathbf{f}(\mathbf{z})) p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u}) \, \mathrm{d}\mathbf{z} = \int_{\mathcal{Z}} p_{\boldsymbol{\epsilon}}(\mathbf{x} - \tilde{\mathbf{f}}(\mathbf{z})) p_{\tilde{\mathbf{T}},\tilde{\boldsymbol{\lambda}}}(\mathbf{z}|\mathbf{u}) \, \mathrm{d}\mathbf{z} \tag{3.46}$$

We now change the domain of the integral from $\mathcal{Z}$ to $\mathcal{X}$, by introducing $\bar{\mathbf{x}} = \mathbf{f}(\mathbf{z})$. We also introduce the notion of matrix volume denoted by $\mathrm{vol}\, A = \sqrt{\det A^T A}$, which acts as a replacement for the absolute determinant of the Jacobian [3] introduced by the change of variable formula for multivariate densities:

$$p_{\mathcal{X}}(\bar{\mathbf{x}}) = p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{f}^{-1}(\bar{\mathbf{x}}))\,|\det\,\mathbf{J}_{\mathbf{f}^{-1}}(\bar{\mathbf{x}})| = p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{f}^{-1}(\bar{\mathbf{x}}))\,\mathrm{vol}\,\mathbf{J}_{\mathbf{f}^{-1}}(\bar{\mathbf{x}}) \tag{3.47}$$

Therefore we may write the marginal distributions as:

$$\int_{\mathcal{X}} p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{f}^{-1}(\bar{\mathbf{x}})|\mathbf{u})\,\mathrm{vol}\,J_{\mathbf{f}^{-1}}(\bar{\mathbf{x}})p_{\epsilon}(\mathbf{x}-\bar{\mathbf{x}})\,\mathrm{d}\bar{\mathbf{x}} = \int_{\mathcal{X}} p_{\mathbf{T},\boldsymbol{\lambda}}(\tilde{\mathbf{f}}^{-1}(\bar{\mathbf{x}})|\mathbf{u})\,\mathrm{vol}\,J_{\tilde{\mathbf{f}}^{-1}}(\bar{\mathbf{x}})p_{\epsilon}(\mathbf{x}-\bar{\mathbf{x}})\,\mathrm{d}\bar{\mathbf{x}} \tag{3.48}$$

We now introduce the following shorthand:

$$\tilde{p}_{\mathbf{T},\boldsymbol{\lambda},\mathbf{f},\mathbf{u}}(\mathbf{x}) = p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{f}^{-1}(\mathbf{x})|\mathbf{u})\,\mathrm{vol}\,J_{\mathbf{f}^{-1}}(\mathbf{x})\mathbb{1}_{\mathcal{X}}(\mathbf{x}) \tag{3.49}$$

where $\mathbb{1}_{\mathcal{X}}$ is the indicator function, assuring that the expression is zero if x is not contained in the image of $\mathbf{f}$:

$$\int_{\mathbb{R}^d} \tilde{p}_{\mathbf{T},\boldsymbol{\lambda},\mathbf{f},\mathbf{u}}(\bar{\mathbf{x}})p_{\epsilon}(\mathbf{x}-\bar{\mathbf{x}})\,\mathrm{d}\bar{\mathbf{x}} = \int_{\mathbb{R}^d} \tilde{p}_{\tilde{\mathbf{T}},\tilde{\boldsymbol{\lambda}},\tilde{\mathbf{f}},\mathbf{u}}(\bar{\mathbf{x}})p_{\epsilon}(\mathbf{x}-\bar{\mathbf{x}})\,\mathrm{d}\bar{\mathbf{x}} \tag{3.50}$$

We recognize this to be the convolution between $\tilde{p}_{\mathbf{T},\boldsymbol{\lambda},\mathbf{f},\mathbf{u}}$ and p_{ϵ}:

$$(\tilde{p}_{\mathbf{T},\boldsymbol{\lambda},\mathbf{f},\mathbf{u}} * p_{\epsilon})(\mathbf{x}) = (\tilde{p}_{\tilde{\mathbf{T}},\tilde{\boldsymbol{\lambda}},\tilde{\mathbf{f}},\mathbf{u}} * p_{\epsilon})(\mathbf{x}) \tag{3.51}$$

Transforming the functions to the Fourier domain allows us to simplify the expression further:

$$F[\tilde{p}_{\mathbf{T},\boldsymbol{\lambda},\mathbf{f},\mathbf{u}}](\omega)\phi_{\epsilon}(\omega) = F[\tilde{p}_{\tilde{\mathbf{T}},\tilde{\boldsymbol{\lambda}},\tilde{\mathbf{f}},\mathbf{u}}](\omega)\phi_{\epsilon}(\omega) \tag{3.52}$$

Note here that we assume the characteristic function $\phi_{\epsilon}(\mathbf{x})$ to be non-zero for $\mathbf{x} \in \mathcal{X}$, which means it can be factored out yielding the final result, from which it is evident that:

$$F[\tilde{p}_{\mathbf{T},\boldsymbol{\lambda},\mathbf{f},\mathbf{u}}](\omega) = F[\tilde{p}_{\tilde{\mathbf{T}},\tilde{\boldsymbol{\lambda}},\tilde{\mathbf{f}},\mathbf{u}}](\omega) \tag{3.53}$$

$$\Updownarrow$$

$$\tilde{p}_{\mathbf{T},\boldsymbol{\lambda},\mathbf{f},\mathbf{u}}(\mathbf{x}) = \tilde{p}_{\tilde{\mathbf{T}},\tilde{\boldsymbol{\lambda}},\tilde{\mathbf{f}},\mathbf{u}}(\mathbf{x}) \tag{3.54}$$

Therefore the noise-free distributions has to be identical. In the following we wish to examine the relationship between the true parameters $(\mathbf{T}, \boldsymbol{\lambda}, \mathbf{f})$ and the estimated parameters $(\tilde{\mathbf{T}}, \tilde{\boldsymbol{\lambda}}, \tilde{\mathbf{f}})$ given that our model learns to accurately approximate the true data distribution $\tilde{p}_{\mathbf{T},\boldsymbol{\lambda},\mathbf{f},\mathbf{u}}(\mathbf{x})$. First we use Equation (3.49) to write the expression for the marginal distribution:

$$\tilde{p}_{\mathbf{T},\boldsymbol{\lambda},\mathbf{f},\mathbf{u}}(\mathbf{x}) = p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{f}^{-1}(\mathbf{x})|\mathbf{u}) \text{ vol } J_{\mathbf{f}^{-1}}(\mathbf{x}) \mathbb{1}_{\mathcal{X}}(\mathbf{x}) = p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u}) \text{ vol } J_{\mathbf{f}^{-1}}(\mathbf{x}) \mathbb{1}_{\mathcal{X}}(\mathbf{x}) \tag{3.55}$$

Since $\mathbf{f}^{-1}(\mathbf{x}) = \mathbf{z}$ by definition. By inserting the expression for the prior distribution given an auxiliary variable, u: $p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u}) = \prod_i^n \frac{Q_i(z_i)}{Z_i(\mathbf{u})} \exp\left[\sum_{j=1}^k T_{i,j}(z_i)\lambda_{i,j}(\mathbf{u})\right]$ from Equation (3.7) on page 10, we can write the marginal distribution over x as:

$$\tilde{p}_{\mathbf{T},\boldsymbol{\lambda},\mathbf{f},\mathbf{u}}(\mathbf{x}) = \prod_i^n \frac{Q_i(z_i)}{Z_i(\mathbf{u})} \exp\left[\sum_{j=1}^k T_{i,j}(z_i)\lambda_{i,j}(\mathbf{u})\right] \text{ vol } J_{\mathbf{f}^{-1}}(\mathbf{x}) \mathbb{1}_{\mathcal{X}}(\mathbf{x}) \tag{3.56}$$

Here we can safely drop the indicator function, $\mathbb{1}_{\mathcal{X}}(\mathbf{x})$, as the expression no longer contains integration and we will write $\mathbf{z} = \mathbf{f}^{-1}(\mathbf{x})$ again to emphasize that latent variables are inferred from data. To make the derivation more tractable we use the log pdf since it simplifies the exponential term:

$$\log \tilde{p}_{\mathbf{T},\boldsymbol{\lambda},\mathbf{f},\mathbf{u}}(\mathbf{x}) = \sum_i^n (\log Q_i(\mathbf{f}_i^{-1}(\mathbf{x})) - \log Z_i(\mathbf{u}) + \sum_{j=1}^k (T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u}))) + \log \text{ vol } J_{\mathbf{f}^{-1}}(\mathbf{x}) \tag{3.57}$$

Thus we can use Equation (3.54) on the preceding page to investigate the relation between true and estimated parameters:

$$\begin{aligned} &\sum_i^n (\log Q_i(\mathbf{f}_i^{-1}(\mathbf{x})) - \log Z_i(\mathbf{u}) + \sum_{j=1}^k (T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u}))) + \log \text{ vol } J_{\mathbf{f}^{-1}}(\mathbf{x}) \\ &= \sum_i^n (\log \tilde{Q}_i(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x})) - \log \tilde{Z}_i(\mathbf{u}) + \sum_{j=1}^k (\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u}))) + \log \text{ vol } J_{\tilde{\mathbf{f}}^{-1}}(\mathbf{x}) \end{aligned} \tag{3.58}$$

Each side of the equation contain nk unknown parameters in $T_{i,j}$ and $\tilde{T}_{i,j}$ respectively, since they are summed over n latent variables and k sufficient parameters per latent variable. Each side of the equation also has nk unknown parameters in $\lambda_{i,j}$ and $\tilde{\lambda}_{i,j}$. Therefore a system of equations is created for $nk + 1$ different points $u^{(0)}, ..., u^{(nk)}$. In time series data divided into segments this step may intuitively be thought of as calculating the probability of seeing a given sample in each of the $nk + 1$ segments. It can also be seen as a consequence of Equation (3.54) on the facing page where we have equality between the marginal distribution for all choices of u. Thus we get the following system of equations:

$$\sum_{i}^{n}(\log Q_i(\mathbf{f}_i^{-1}(\mathbf{x})) - \log Z_i(\mathbf{u_0}) + \sum_{j=1}^{k}(T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u_0}))) + \log \text{ vol } J_{\mathbf{f}^{-1}}(\mathbf{x})$$
$$= \sum_{i}^{n}(\log \tilde{Q}_i(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x})) - \log \tilde{Z}_i(\mathbf{u_0}) + \sum_{j=1}^{k}(\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u_0}))) + \log \text{ vol } J_{\tilde{\mathbf{f}}^{-1}}(\mathbf{x}) \tag{3.59}$$

$$\sum_{i}^{n}(\log Q_i(\mathbf{f}_i^{-1}(\mathbf{x})) - \log Z_i(\mathbf{u_1}) + \sum_{j=1}^{k}(T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u_1}))) + \log \text{ vol } J_{\mathbf{f}^{-1}}(\mathbf{x})$$
$$= \sum_{i}^{n}(\log \tilde{Q}_i(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x})) - \log \tilde{Z}_i(\mathbf{u_1}) + \sum_{j=1}^{k}(\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u_1}))) + \log \text{ vol } J_{\tilde{\mathbf{f}}^{-1}}(\mathbf{x}) \tag{3.60}$$

$$\vdots$$

$$\sum_{i}^{n}(\log Q_i(\mathbf{f}_i^{-1}(\mathbf{x})) - \log Z_i(\mathbf{u_{nk}}) + \sum_{j=1}^{k}(T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u_{nk}}))) + \log \text{ vol } J_{\mathbf{f}^{-1}}(\mathbf{x})$$
$$= \sum_{i}^{n}(\log \tilde{Q}_i(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x})) - \log \tilde{Z}_i(\mathbf{u_{nk}}) + \sum_{j=1}^{k}(\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u}_{nk}))) + \log \text{ vol } J_{\tilde{\mathbf{f}}^{-1}}(\mathbf{x}) \tag{3.61}$$

A nifty trick is used to simplify this system of equations by using any of the $nk+1$ equations as pivot. We simply use the equation for $\mathbf{u_0}$. By pivot it is understood that we consider a ratio of pdfs or in this case, since we are dealing with logarithms, a difference of log pdfs. This is the motivation for using $nk+1$ points in our system of equations as we use one equation to pivot such that we end up with a system of nk equations. The consequence of this choice is that our equations no longer express how likely a sample is with a given $\mathbf{u_l}$ but rather how likely it is compared to $\mathbf{u_0}$, but this is of little importance as we are interested in the relation between parameters of the models and not the exact probability of seen samples. Therefore the system of equations becomes:

$$0 = 0 \tag{3.62}$$

$$\begin{aligned}
&\sum_i^n(\log Q_i(\mathbf{f}_i^{-1}(\mathbf{x})) - \log Z_i(\mathbf{u_1}) + \sum_{j=1}^k(T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u_1}))) + \log\ \text{vol}\ J_{\mathbf{f}^{-1}}(\mathbf{x}) \\
&-(\sum_i^n(\log Q_i(\mathbf{f}_i^{-1}(\mathbf{x})) - \log Z_i(\mathbf{u_0}) + \sum_{j=1}^k(T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u_0}))) + \log\ \text{vol}\ J_{\mathbf{f}^{-1}}(\mathbf{x})) = \\
&\sum_i^n(\log \tilde{Q}_i(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x})) - \log \tilde{Z}_i(\mathbf{u_1}) + \sum_{j=1}^k(\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u_1}))) + \log\ \text{vol}\ J_{\tilde{\mathbf{f}}^{-1}}(\mathbf{x}) \\
&-(\sum_i^n(\log \tilde{Q}_i(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x})) - \log \tilde{Z}_i(\mathbf{u_0}) + \sum_{j=1}^k(\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u_0}))) + \log\ \text{vol}\ J_{\tilde{\mathbf{f}}^{-1}}(\mathbf{x}))
\end{aligned} \tag{3.63}$$

$$\vdots$$

$$\begin{aligned}
&\sum_i^n(\log Q_i(\mathbf{f}_i^{-1}(\mathbf{x})) - \log Z_i(\mathbf{u_{nk}}) + \sum_{j=1}^k(T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u_{nk}}))) + \log\ \text{vol}\ J_{\mathbf{f}^{-1}}(\mathbf{x}) \\
&-(\sum_i^n(\log Q_i(\mathbf{f}_i^{-1}(\mathbf{x})) - \log Z_i(\mathbf{u_0}) + \sum_{j=1}^k(T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u_0}))) + \log\ \text{vol}\ J_{\mathbf{f}^{-1}}(\mathbf{x})) \\
&=\sum_i^n(\log \tilde{Q}_i(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x})) - \log \tilde{Z}_i(\mathbf{u_{nk}}) + \sum_{j=1}^k(\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u}_{nk}))) + \log\ \text{vol}\ J_{\tilde{\mathbf{f}}^{-1}}(\mathbf{x}) \\
&-(\sum_i^n(\log \tilde{Q}_i(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x})) - \log \tilde{Z}_i(\mathbf{u_0}) + \sum_{j=1}^k(\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u_0}))) + \log\ \text{vol}\ J_{\tilde{\mathbf{f}}^{-1}}(\mathbf{x}))
\end{aligned} \tag{3.64}$$

By eliminating terms we get rid of $\log \tilde{Q}_i(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x})$ and interestingly $\log\ \text{vol}\ J_{\tilde{\mathbf{f}}^{-1}}(\mathbf{x})$ which is typically notoriously difficult to evaluate, therefore these equations can be reduced to:

$$0 = 0 \tag{3.65}$$

$$\begin{aligned}
&\sum_i^n(-\log Z_i(\mathbf{u_1}) + \sum_{j=1}^k(T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u_1}))) - (\sum_i^n(-\log Z_i(\mathbf{u_0}) + \sum_{j=1}^k(T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u_0}))) \\
&=\sum_i^n(-\log \tilde{Z}_i(\mathbf{u_1}) + \sum_{j=1}^k(\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u_1}))) - (\sum_i^n(-\log \tilde{Z}_i(\mathbf{u_0}) + \sum_{j=1}^k(\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u_0})))
\end{aligned} \tag{3.66}$$

$$\vdots$$

$$\begin{aligned}
&\sum_{i}^{n}(-\log Z_i(\mathbf{u_{nk}})+\sum_{j=1}^{k}(T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u_{nk}})))-(\sum_{i}^{n}(-\log Z_i(\mathbf{u_0})+\sum_{j=1}^{k}(T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u_0})))\\
&=\sum_{i}^{n}(-\log \tilde{Z}_i(\mathbf{u_{nk}})+\sum_{j=1}^{k}(\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u}_{nk})))-(\sum_{i}^{n}(-\log \tilde{Z}_i(\mathbf{u_0})+\sum_{j=1}^{k}(\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u_0})))
\end{aligned} \tag{3.67}$$

By factoring terms and distributing sums for an arbitrary point $u^{(l)}$ we get:

$$\begin{aligned}
&\sum_{i}^{n}(\sum_{j=1}^{k}(T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))(\lambda_{i,j}(\mathbf{u_l})-\lambda_{i,j}(\mathbf{u_0})))+\sum_{i}^{n}\log\frac{Z_i(\mathbf{u_0})}{Z_i(\mathbf{u_l})}\\
&\qquad=\sum_{i}^{n}(\sum_{j=1}^{k}(\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))(\tilde{\lambda}_{i,j}(\mathbf{u_l})-\tilde{\lambda}_{i,j}(\mathbf{u_0})))+\sum_{i}^{n}\log\frac{\tilde{Z}_i(\mathbf{u_0})}{\tilde{Z}_i(\mathbf{u_l})}
\end{aligned} \tag{3.68}$$

$T_{i,j}$ and $\lambda_{i,j}$ are elements from the tall vectors $\mathbf{T}$ and $\boldsymbol{\lambda}$ therefore the first term can be recognized as the inner product between $\mathbf{T(f^{-1}(x))}$ and $\bar{\boldsymbol{\lambda}}(\mathbf{u_l})$ where $\bar{\boldsymbol{\lambda}}(\mathbf{u_l})$ is defined as:

$$\bar{\boldsymbol{\lambda}}(\mathbf{u_l})=\boldsymbol{\lambda}(\mathbf{u_l})-\boldsymbol{\lambda}(\mathbf{u_0}) \tag{3.69}$$

Therefore Equation (3.68) can be written as:

$$\left\langle \mathbf{T(f^{-1}(x))},\bar{\boldsymbol{\lambda}}(\mathbf{u_l})\right\rangle=\left\langle \tilde{\mathbf{T}}(\tilde{\mathbf{f}}^{-1}(\mathbf{x})),\bar{\tilde{\boldsymbol{\lambda}}}(\mathbf{u_l})\right\rangle+b_l \quad where \quad b_l=\sum_{i}^{n}\log\frac{\tilde{Z}_i(\mathbf{u_0})Z_i(\mathbf{u_l})}{\tilde{Z}_i(\mathbf{u_l})Z_i(\mathbf{u_0})} \tag{3.70}$$

Across all nk equations $\mathbf{T(f^{-1}(x))}$ will be the same, therefore we can collect all the equations in a single matrix product by defining the $nk\times nk$ matrix $L=\left[\bar{\boldsymbol{\lambda}}(\mathbf{u_1}),\bar{\boldsymbol{\lambda}}(\mathbf{u_2})\dots\bar{\boldsymbol{\lambda}}(\mathbf{u_{nk}})\right]$ and $\mathbf{b}=\left[b_1,b_2,\dots,b_{nk}\right]$ such that:

$$L^T\mathbf{T(f^{-1}(x))}=\tilde{L}^T\tilde{\mathbf{T}}(\tilde{\mathbf{f}}^{-1}(\mathbf{x}))+\mathbf{b} \tag{3.71}$$

In the final step we assume that there exists at least nk choices of u, $u^{(1)},...,u^{(nk)}$, such that the columns of the matrix containing the true natural parameters, $\mathbf{L}$, are linearly independent and thus invertible to obtain the following result:

$$\mathbf{T}(\mathbf{f}^{-1}(\mathbf{x})) = A\tilde{\mathbf{T}}(\tilde{\mathbf{f}}^{-1}(\mathbf{x})) + \mathbf{c} \tag{3.72}$$

Where $A = L^{T^{-1}}\tilde{L}^T$ and $\mathbf{c} = L^{T^{-1}}\mathbf{b}$. Thus, we see that the true latent variables are linear transformation of the recovered latent variables. The last step is to prove an equivalence relation such that the opposite is also true. Namely, that the recovered latent variables are also a linear transformation of the true latent variables. To do so it is assumed that the Jacobian of $\mathbf{T}$ exists and has full rank n. By using that by definition $\mathbf{x} = \mathbf{f}(\mathbf{z})$ Equation (3.72) can be rewritten to be a function of the n variables of $\mathbf{z}$, $z_1, \ldots, z_n$. By taking the derivative with respect to $\mathbf{z}$ on both sides of the equation we get the following where J is the $nk \times n$ Jacobian matrix:

$$J_{\mathbf{T}\circ\mathbf{f}^{-1}\circ\mathbf{f}}(\mathbf{z}) = AJ_{\tilde{\mathbf{T}}\circ\tilde{\mathbf{f}}^{-1}\circ\mathbf{f}}(\mathbf{z}) \tag{3.73}$$

By using the following inequality for the rank of a matrix multiplication we may deduce that the rank of both A and $J_{\tilde{\mathbf{T}}\circ\tilde{\mathbf{f}}^{-1}\circ\mathbf{f}}$ is at least n because the rank of $J_{\mathbf{T}\circ\mathbf{f}^{-1}\circ\mathbf{f}}(\mathbf{z})$ is equal to n:

$$Rank(AB) \leq min(Rank(A), Rank(B)) \tag{3.74}$$

Since $J_{\tilde{\mathbf{T}}\circ\tilde{\mathbf{f}}^{-1}}$ is a $nk \times n$ matrix we can conclude that it exists and has full rank. And if $k = 1$ then A will be a square $n \times n$ matrix with full rank and thus invertible, such that Equation (3.72) can be shown to be true in both directions. Therefore the equivalence relation for $k = 1$ has been proved.

For $k > 1$ the matrix A must be invertible in order to establish the equivalence relation. In the following we show how A is invertible under the assumption that each latent variable follow a *strongly exponential distribution*. A strongly exponential distribution is one that almost certainly contains the exponent and thus can not be reduced to the base measure. Formally a strong exponential distribution fulfills:

$$(\exists \boldsymbol{\theta} \in \mathbb{R}^k \mid \forall x \in \mathcal{X}, \langle \mathbf{T}(\mathbf{x}), \boldsymbol{\theta} \rangle = const) \quad \Rightarrow \quad (l(\mathcal{X}) = 0 \;\; or \;\; \boldsymbol{\theta} = \mathbf{0}) \tag{3.75}$$

Which means that the exponent, $\langle \mathbf{T}(\mathbf{x}), \boldsymbol{\theta} \rangle$, of a strongly exponential distribution only reduces to a constant if $\boldsymbol{\theta} = 0$ which means the inner product becomes zero, $\langle \mathbf{T}(\mathbf{x}), \mathbf{0} \rangle = 0$, or if the set $\mathcal{X}$ has Lebesgue measure 0. The following three Lemmas is used to derive useful properties for the derivative of the sufficient statistic, $\mathbf{T}'(x)$, from a strongly exponential distribution that is of relevance for the Jacobian matrix. The dimension, k, of all considered distributions is assumed minimal. That is, the distributions can not be rewritten with a $k' < k$.

Lemma 1 Consider an exponential family distribution with $k \geq 2$ components. [...], the components of the sufficient statistic $\mathbf{T}$ are linearly independent.

If the components of $\mathbf{T}$ are not linearly independent then one of the components, $T_k(x)$, could be written as a combination of the remaining components for an $\mathbf{a} \neq \mathbf{0}$.

$$T_k(x) = \sum_{i}^{k-1} a_i T_i(x) \tag{3.76}$$

If that was possible, we would have contradicted the assumption that the dimension of the distribution, k, is minimal.

Lemma 2 Consider a strongly exponential family distribution such that its sufficient statistic T is differentiable almost surely. Then $T_i' \neq 0$ almost everywhere on $\mathbb{R}$ for all $1 \leq$ i $\leq$ k

We provide an alternate proof than the original, simply because we used this alternate proof to verify our understanding of the original proof. If we consider an exponential distribution that is *not* strongly exponential then we necessarily have:

$$\langle \mathbf{T(x)}, \boldsymbol{\theta} \rangle = T_1(x)\theta_1 + T_2(x)\theta_1 + \cdots + T_k\theta_k = const \tag{3.77}$$

The derivative would then become:

$$\frac{\mathrm{d}}{\mathrm{d}x} \langle \mathbf{T(x)}, \boldsymbol{\theta} \rangle = \langle \mathbf{T(x)'}, \boldsymbol{\theta} \rangle = T_1'(x)\theta_1 + T_2'(x)\theta_2 + \cdots + T_k'(x)\theta_k = 0 \tag{3.78}$$

Thus for an exponential distribution that is not strongly exponential the derivative of the exponent must be equal to zero. Which can be achieved in many different ways having either $\boldsymbol{\theta} = \mathbf{0}$, $\mathbf{T}(x) = \mathbf{0}$ or their weighted sum equal to zero. For a strongly exponential distribution the exponent can only equal a constant if $\boldsymbol{\theta} = \mathbf{0}$ (see Equation (3.75) on the preceding page) and therefore the derivative can also only be 0 if $\boldsymbol{\theta} = \mathbf{0}$. From Lemma 1 we have that the components of the sufficient statistic can not be written as a function of each other. Therefore it can be seen that $\mathbf{T}'(x) \neq \mathbf{0}$ and even $T_i'(x) \neq 0$. Because, if any $T_i'(x)$ was equal to zero the corresponding θ_i could be an arbitrary number different from zero while the rest of $\boldsymbol{\theta}$ is zero and thus $\boldsymbol{\theta} \neq \mathbf{0}$ but the derivative would equal zero. which violates the statement that the distribution is strongly exponential. Thus, we may conclude that for a strongly exponential distribution $T_i'(x)$ must be different from zero.

Lemma 3 Consider a strongly exponential distribution of size $k \geq 2$ with sufficient statistic $\mathbf{T}(x) = (T_1(x), ..., T_k(x))$. Further assume that $\mathbf{T}$ is differentiable almost everywhere. Then there exist k distinct values x_1 to x_k such that $(\mathbf{T}'(x_1), ..., \mathbf{T}'(x_k))$ are linearly independent in $\mathbb{R}^k$

Recall that in order for the distribution to be strongly exponential then the only choice of parameter that can lead to the exponent being constant for all x is $\boldsymbol{\theta} = \mathbf{0}$. Since both $\mathbf{T'(x)}$ and $\boldsymbol{\theta}$ is in $\mathbb{R}^k$ this necessarily means that $\mathbf{T'(x)}$ must be able to span the full $\mathbb{R}^k$. That is, there exists at least k vectors of $\mathbf{T'(x)}$ in k points $x_1, \ldots, x_k$ such that the matrix $B = [\mathbf{T}'(x_1)\ \mathbf{T}'(x_2)\ \ldots\ \mathbf{T}'(x_k)]$ has full rank:

$$Rank(B) = Rank(\begin{bmatrix}\mathbf{T}'(x_1) & \mathbf{T}'(x_2) & \ldots & \mathbf{T}'(x_k)\end{bmatrix}) = k \tag{3.79}$$

If $Rank(B) \neq k$ then the nullity of B will be greater than 1 and thus any vector from the orthogonal complement of the column space of B can be picked as $\boldsymbol{\theta}^*$ such that $\boldsymbol{\theta}^* \neq \mathbf{0}$ and $\langle \mathbf{T(x)'}, \boldsymbol{\theta}^* \rangle = 0$ for all x. However, if that is the case then the distribution is not strongly exponential as seen from Lemma 2 that shows that only a distribution which is *not* strongly exponential will have $\langle \mathbf{T(x)'}, \boldsymbol{\theta} \rangle = 0$ for $\boldsymbol{\theta} \neq \mathbf{0}$. Therefore, in a strongly exponential distribution there must exist k points, $x_1, \ldots, x_k$ such that the column vectors of B are linearly independent.

These three Lemmas have been used to derive the important property that in univariate exponential distributions which are minimal in k and strongly exponential there exist at least k points, $x_1, \ldots, x_k$ such that the vectors $\mathbf{T}'(x_1), \ldots, \mathbf{T}'(x_k)$ are linearly independent. We can now use this to show that the $nk \times nk$ matrix A in Equation (3.80) is invertible under the assumption that the $nk \times n$ Jacobian matrix of $\mathbf{T}(\mathbf{f}^{-1}(x))$, $J_{\mathbf{T}}(\mathbf{f}^{-1}(x))$, exists and is of rank n:

$$\mathbf{T(f^{-1}(x))} = A\tilde{\mathbf{T}}(\tilde{\mathbf{f}}^{-1}(\mathbf{x})) + \mathbf{c} \tag{3.80}$$

To make the proof easier to follow we examine the form the Jacobian matrix will have. First we write the expression for the Jacobian matrix of $\mathbf{T}(\mathbf{f}^{-1}(x))$ (remembering that $\mathbf{f}^{-1}$ is a function that maps x to $\mathbb{R}^n$, such that $\mathbf{T}$ is a function of n (latent) variables, $f_1^{-1}(x), \ldots, f_n^{-1}(x)$. Therefore the Jacobian matrix of $\mathbf{T}$ is given by:

$$J_{\mathbf{T}}(\mathbf{f}^{-1}(x)) = \frac{\mathrm{d}\mathbf{T}(\mathbf{f}^{-1}(x))}{\mathrm{d}(f_1(x), f_2(x), \ldots, f_n(x))} = \begin{bmatrix} \frac{\mathrm{d}\mathbf{T}(\mathbf{f}^{-1}(x))}{\mathrm{d}f_1(x)} & \frac{\mathrm{d}\mathbf{T}(\mathbf{f}^{-1}(x))}{\mathrm{d}f_2(x)} & \cdots & \frac{\mathrm{d}\mathbf{T}(\mathbf{f}^{-1}(x))}{\mathrm{d}f_n(x)} \end{bmatrix} \tag{3.81}$$

$$= \begin{bmatrix} \frac{\mathrm{d}T_{1,1}(f_1^{-1}(x))}{df_1^{-1}(x)} & \frac{\mathrm{d}T_{1,1}(f_1^{-1}(x))}{df_2^{-1}(x)} & \cdots & \frac{\mathrm{d}T_{1,1}(f_1^{-1}(x))}{df_n^{-1}(x)} \\ \vdots & \vdots & & \vdots \\ \frac{\mathrm{d}T_{1,k}(f_1^{-1}(x))}{df_1^{-1}(x)} & \frac{\mathrm{d}T_{1,k}(f_1^{-1}(x))}{df_2^{-1}(x)} & \cdots & \frac{\mathrm{d}T_{1,k}(f_1^{-1}(x))}{df_n^{-1}(x)} \\ \frac{\mathrm{d}T_{2,1}(f_2^{-1}(x))}{df_1^{-1}(x)} & \frac{\mathrm{d}T_{2,1}(f_2^{-1}(x))}{df_2^{-1}(x)} & \cdots & \frac{\mathrm{d}T_{2,1}(f_2^{-1}(x))}{df_n^{-1}(x)} \\ \vdots & \vdots & & \vdots \\ \frac{\mathrm{d}T_{2,k}(f_2^{-1}(x))}{df_1^{-1}(x)} & \frac{\mathrm{d}T_{2,k}(f_2^{-1}(x))}{df_2^{-1}(x)} & \cdots & \frac{\mathrm{d}T_{2,k}(f_2^{-1}(x))}{df_n^{-1}(x)} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \frac{\mathrm{d}T_{n,1}(f_n^{-1}(x))}{df_1^{-1}(x)} & \frac{\mathrm{d}T_{n,1}(f_n^{-1}(x))}{df_2^{-1}(x)} & \cdots & \frac{\mathrm{d}T_{n,1}(f_n^{-1}(x))}{df_n^{-1}(x)} \\ \vdots & \vdots & & \vdots \\ \frac{\mathrm{d}T_{n,k}(f_n^{-1}(x))}{df_1^{-1}(x)} & \frac{\mathrm{d}T_{n,k}(f_n^{-1}(x))}{df_2^{-1}(x)} & \cdots & \frac{\mathrm{d}T_{n,k}(f_n^{-1}(x))}{df_n^{-1}(x)} \end{bmatrix}^{nk \times n} \tag{3.82}$$

We can notice that this matrix will have a particular shape since many of the entries will become zero as they are not a function of the variable with which the derivative is taken. Therefore the Jacobian matrix will have the shape:

$$J_{\mathbf{T}}(\mathbf{f}^{-1}(x)) = \begin{bmatrix} T'_{1,1}(f_1^{-1}(x)) & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ T'_{1,k}(f^{-1}(x)) & 0 & \dots & 0 \\ 0 & T'_{2,1}(f_2^{-1}(x)) & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & T'_{2,k}(f_2^{-1}(x)) & \dots & 0 \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & T'_{n,1}(f_n^{-1}(x)) \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & T'_{n,k}(f_n^{-1}(x)) \end{bmatrix} \tag{3.83}$$

The expression we used to derive invertibility of A for $k = 1$ was found in Equation (3.73) on page 25 as seen below (for simplicity we write $\mathbf{x}$ remembering that $\mathbf{x} = \mathbf{f}(\mathbf{z})$:

$$J_{\mathbf{T}\circ\mathbf{f}^{-1}}(\mathbf{x}) = AJ_{\tilde{\mathbf{T}}\circ\tilde{\mathbf{f}}^{-1}}(\mathbf{x}) \tag{3.84}$$

For $k = 1$ we used the fact that $J_{\mathbf{T}}(x)$ becomes a full rank square matrix to prove A is invertible. The same approach is used now. However, since for $k > 1$ $J_{\mathbf{T}}(x)$ is an $nk \times n$ matrix which is not square and thus not invertible, we use that the above expression is true for all x. In particular, we choose k points $x_1, \dots, x_k$ according to Lemma 3. The k points serve two purposes. First, k Jacobians are needed to have enough entries to fill an $nk \times nk$ square matrix of Jacobians and secondly by choosing the points according to Lemma 3 invertibility can be proved. By concatenating the Jacobian matrices evaluated at the k points the matrix Q can be formed:

$$Q = \begin{bmatrix} & | & & | & \\ J_{\mathbf{T}\circ\mathbf{f}^{-1}}(x_1) & | & \dots & | & J_{\mathbf{T}\circ\mathbf{f}^{-1}}(x_k) \\ & | & & | & \end{bmatrix} \tag{3.85}$$

And similarly for $J_{\tilde{\mathbf{T}}\circ\tilde{\mathbf{f}}^{-1}}(x)$:

$$\tilde{Q} = \begin{bmatrix} & | & & | & \\ J_{\tilde{\mathbf{T}}\circ\tilde{\mathbf{f}}^{-1}}(x_1) & | & \dots & | & J_{\tilde{\mathbf{T}}\circ\tilde{\mathbf{f}}^{-1}}(x_k) \\ & | & & | & \end{bmatrix} \tag{3.86}$$

Thus for $k \geq 1$ we may write:

$$Q = A\tilde{Q} \tag{3.87}$$

To verify that the concatenated system can indeed be written as in Equation (3.87) we can recognize the expression as block matrix multiplication where A is a matrix with $q = 1$ row partition and $s = 1$ column partition and $\tilde{Q}$ a block matrix with $s = 1$ row partition and $r = k$ column partitions. Thus, the partitions of A and $\tilde{Q}$ are conformable and the resulting matrix, Q, will have 1 row partition and k column partitions, as in the definition above. We may also confirm that the new matrix multiplication maps every column partition i in $\tilde{Q}$ to column partition i in Q as it should be from Equation (3.84) on the preceding page. Block matrix multiplication is defined as:

$$Q_{qr} = \sum_{i=1}^{s} A_{qi}\tilde{Q}_{ir} = A_{11}\tilde{Q}_{1r} = A\tilde{Q}_{1r} = Q_{1r} \tag{3.88}$$

Therefore each column partition of $\tilde{Q}$ is mapped to the same column partition in Q by A. Every column partition in Q has the form of Equation (3.83) on the previous page and by rearranging the columns of Q such that all the nonzero elements are grouped it can be seen that Q can be written as a block diagonal matrix:

$$Q = \begin{bmatrix} B_{f_1^{-1}} & & & 0 \\ & B_{f_2^{-1}} & & \\ & & \ddots & \\ 0 & & & B_{f_n^{-1}} \end{bmatrix} \tag{3.89}$$

Where B is defined as in Lemma 3, $B = [\mathbf{T}'(x_1)\ \mathbf{T}'(x_2)\ \ldots\ \mathbf{T}'(x_k)]$, and the subscript f_i^{-1} is used to emphasize $\mathbf{T(f^{-1}(x))}$ is a function of n variables, $f_1^{-1}(x), \ldots, f_n^{-1}(x)$, such that $B_{f_i^{-1}}$ is the $k \times k$ matrix containing the nonzero derivatives with respect to f_i^{-1} as seen in Equation (3.83) on the preceding page for all k points $x_1, \ldots, x_k$. If Q is invertible that would imply the invertibility of both A and $\tilde{Q}$ as it can be seen from Equation (3.87) which is what we would like to prove. A block diagonal matrix is invertible if all the diagonal matrices are invertible. That is:

$$Q^{-1} = \begin{bmatrix} B_{f_1^{-1}} & & & 0 \\ & B_{f_2^{-1}} & & \\ & & \ddots & \\ 0 & & & B_{f_n^{-1}} \end{bmatrix}^{-1} = \begin{bmatrix} B^{-1}_{f_1^{-1}} & & & 0 \\ & B^{-1}_{f_2^{-1}} & & \\ & & \ddots & \\ 0 & & & B^{-1}_{f_n^{-1}} \end{bmatrix} \tag{3.90}$$

Since the points $x_1, \ldots, x_k$ are chosen as in Lemma 3 every diagonal matrix of Q is exactly identical to B in Lemma 3 (one for each of the n latent variables). Therefore every diagonal matrix of Q is invertible because B has full rank, as proven in Lemma 3, and thus Q is also invertible. Since Q is invertible we can write:

$$Q^{-1} = (A\tilde{Q})^{-1} = \tilde{Q}^{-1}A^{-1} \tag{3.91}$$

Which means that both A and $\tilde{Q}$ are invertible. Since A is invertible we have proven the equivalence relation for $k \geq 1$. Therefore we have proven for all k that $(\mathbf{f}, \mathbf{T}, \boldsymbol{\lambda}) \sim (\tilde{\mathbf{f}}, \tilde{\mathbf{T}}, \tilde{\boldsymbol{\lambda}})$ which means that:

$$\exists A, \mathbf{c} \mid \mathbf{T}(\mathbf{f^{-1}(x)}) = A\tilde{\mathbf{T}}(\tilde{\mathbf{f}}^{-1}(\mathbf{x})) + \mathbf{c}, \quad \forall x \in \mathcal{X} \tag{3.92}$$

Where we have shown that $A = L^{T^{-1}}\tilde{L}^T$ and $\mathbf{c} = L^{T^{-1}}\mathbf{b}$ as seen from Equation (3.72) on page 25 with L, $\mathbf{b}$ defined as:

$$L = \left[\bar{\boldsymbol{\lambda}}(\mathbf{u_1}), \bar{\boldsymbol{\lambda}}(\mathbf{u_2}) \ldots \bar{\boldsymbol{\lambda}}(\mathbf{u_{nk}})\right], \qquad \bar{\boldsymbol{\lambda}}(\mathbf{u_l}) = \boldsymbol{\lambda}(\mathbf{u_l}) - \boldsymbol{\lambda}(\mathbf{u_0}) \tag{3.93}$$

$$\mathbf{b} = \left[b_1, b_2, \ldots, b_{nk}\right], \qquad b_l = \sum_i^n \log \frac{\tilde{Z}_i(\mathbf{u_0})Z_i(\mathbf{u_l})}{\tilde{Z}_i(\mathbf{u_l})Z_i(\mathbf{u_0})} \tag{3.94}$$

Thus we may conclude, on the basis of the proof above and the lower bound derived for iVAE-GAN, that the iVAE-GAN model achieves identifiability in the form of the equivalence relation above. Because, provided that our function approximators for $q_\phi(z|x,u)$ (encoder) and $p_\Phi(x|z)$ (decoder) are complex enough to include $p_\theta(z|u)$ and $p(x)$, respectively. Optimizing the iVAE-GAN loss minimizes the KL divergence between $q_\phi(z|x,u)$ and $p_\theta(z|u)$ as well as minimizes the Jensen-Shannon divergence between $p_\Phi(x|z)$ and $p(x)$. Thus, in the limit of infinite data iVAE-GAN will have learnt the seen marginal distribution over data and encode the prior distribution thereby making the model identifiable up to the equivalence relation of Equation (3.92). This implies that the latent variables recovered by iVAE-GAN will be related to the true latent variables by a linear transformation.

This is, to the best of authors knowledge, the first work to not only extend and apply the theoretical framework to a deep latent variable model not contained in the original works by [15, 14] but also the first identifiability result shown in GAN. In the next section we will present our implementation of the iVAE-GAN model and our experiments.

Chapter 4

Implementation & Results

In this section we will explain our implementation along with the tools and resources we have used and made in the process.

The main basis for our code has been the code for the iVAE model, which has been made publicly available by the original authors (along with the ICE-BeeM and TCL model)[1]. Our motivation for doing so has been threefold: Early in the process we chose the iVAE model to be our baseline comparison since it is a state-of-the-art identifiable deep latent variable model which will be applicable in many of the same applications as iVAE-GAN - much like a regular VAE and GAN network can be used on the same data but typically with different results. Secondly, the iVAE and iVAE-GAN model share a similar architecture as they both implement an encoder, decoder and an auxiliary network, A. This means that architecturally it was straightforward to add the additional discriminator to arrive at the iVAE-GAN architecture, which of course eases implementation. Lastly, the code for the iVAE model also included implementations for the metric used to quantify identifiability in literature, the MCC metric, and a data generator for the nonstationary Gaussian time series data used in literature.

Therefore it was a natural choice for us to base our implementation on the existing implementation of the iVAE model as it not only shares a similar architecture with iVAE-GAN but it also allowed for direct comparison of the two models both under and after development, since the models were built to share all the same hyperparameters for the architecture and data generation. The iVAE model and therefore in extension iVAE-GAN is implemented in Python using the PyTorch package. We have had no previous experience in neither neural networks nor PyTorch but apart from a slight learning curve we have only been positively surprised by the used framework.

[1] `https://github.com/ilkhem/icebeem`

CLAAUDIA has been used to run most of our experiments to speed up the process. We were fortunate to be granted access to Aalborg University's High-Performance Computing (HPC) cluster called CLAAUDIA. At the core CLAAUDIA contains two NVIDIA DGX-2 which each consists of 16 NVIDIA Tesla V100 GPUs that can be remotely accessed. The use of CLAAUDIA allowed us to train models that would otherwise have been too big for our own machines but also train faster in general.

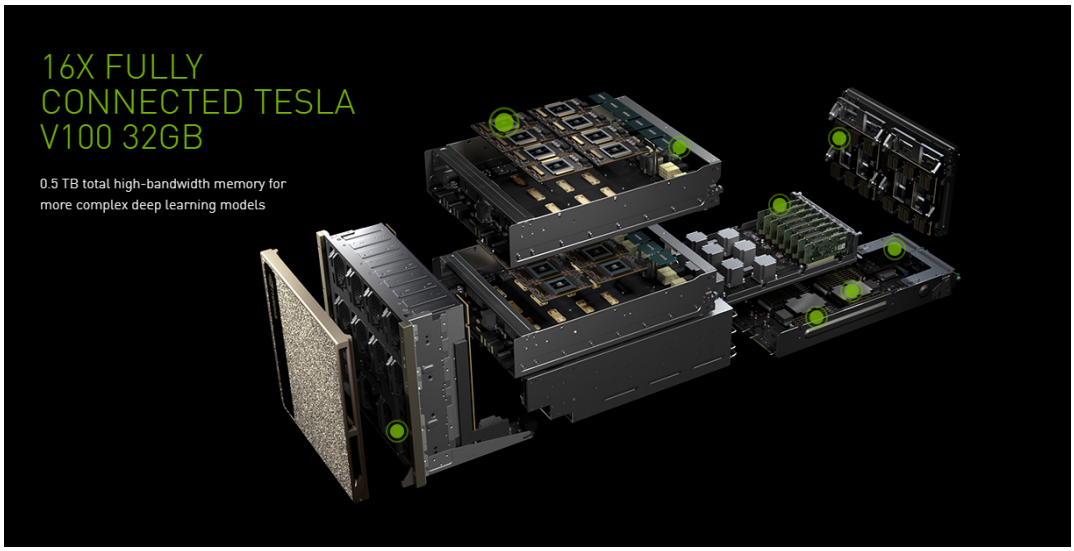


Figure 4.1: DGX-2 used in CLAAUDIA [19]

Overall CLAAUDIA has been a great resource for the project although the learning curve and setup associated with getting started was quite steep and lengthy. Many of the problems were rooted in how Docker and Singularity needed to be set up in order to transfer not only the required Python files but also environments such that the code may be executed and the logs and images generated during training saved. After many attempts we found the solution to be the build option - -sandbox for singularity images that allowed us to create a folder on CLAAUDIA which behaves like a normal file system as seen on Figure 4.2 on the facing page. The sandbox option made it easy to run code, upload and download files, install additional dependencies etc. from the created folder.

Since training is done remotely and could take several hours it was both inconvenient and inefficient to monitor progress manually. In an effort to ease planning around experiments, track errors that could occur during training and make the progress visible to all we found a useful Python package called tqdm. Using tqdm we were able to set up a bot in Discord that would provide a live updated progress bar that would display an estimate of the remaining training time, the current training time and how long the average iteration takes as seen in Figure 4.3 on the next page. Since CLAAUDIA is accessed by several researchers there is typically a queue to access the hardware meaning that jobs will have to wait a while before

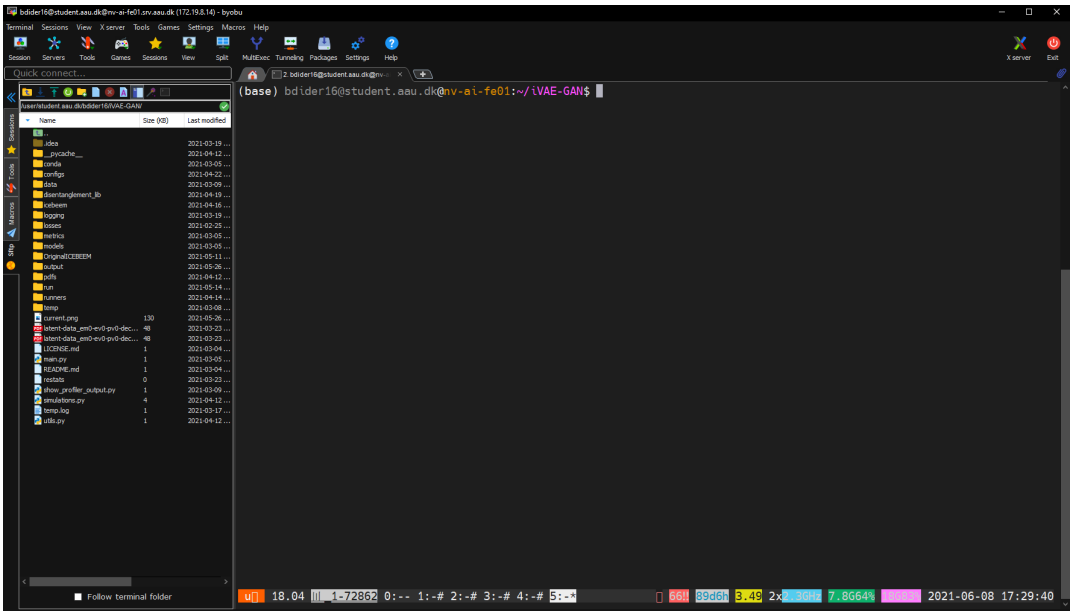


Figure 4.2: Remote interface to CLAAUDIA (using the MobaX terminal)

being started. Therefore, it was also beneficial to receive notifications through tqdm instead of monitoring the remote interface in order to probe the estimated training time. During high demand the queue for resources would typically become so long that we found it more efficient to use CLAAUDIAs CPU resources since they were more readily available and then run multiple smaller experiments in parallel.

Figure 4.3: Live tqdm progress bar in Discord [25]

Throughout our experiments we also came to draw good use of CLAAUDIAs storage capacity to save logged training data and generated images. So much so that we accumulated in excess of 100 GB of logged data and at one point hit the upper limit for stored data per user. The large amount of logged data is a testament to the notoriously tricky convergence of GAN training [6] and in the following we will highlight some of the various approaches we have attempted in order to stabilize training.

Model iterations. We have had no prior experience training GANs and from literature it quickly became apparent that a variety of different things can be attempted to stabilize GAN training - but none that seems to work universally well in all applications. A common guideline, recommended by most, seemed to be to follow the design of DCGAN [22]. Therefore our baseline became using the parameters of DCGAN for e.g. optimizers. Also commonly recommended was the list of "hacks" researchers had collected from past experiences [9]. Since we do not use convolutional layers as in DCGAN and our model architecture differs from traditional GANs because we also learn an encoding of the latent space it was uncertain whether or not, and to which extent, common GAN hacks would apply to our model. After having experienced how our first naive implementation of iVAE-GAN hopelessly failed to converge no matter what, we generated a list of promising approaches to stabilize training as seen in Appendix A. Using the list we could iteratively work our way towards a model that would converge while keeping track of what works and what does not. Naturally, we did not exhaust the options in the list, and neither will we describe every attempted method in nitty gritty detail. Instead, we will briefly highlight some of the options we have explored before arriving at our final model to hopefully convince the reader that a functioning GAN model does not come without considerable experimentation - especially not when experiments can take hours each. In the following we highlight some experiences we have gained with respect to loss functions, batch normalization, discriminator implementations, initial weights, CUDA libraries and preprocessing of input data.

Loss functions are of course essential to any deep learning model, GAN and iVAE-GAN included. The iVAE-GAN loss is a hybrid loss consisting of a loss for the encoded distribution, $\mathcal{L}_{prior}$, and a loss for the adversarial training, $\mathcal{L}_{GAN}$. For the prior loss we identified two choices: The KL divergence loss as implemented by PyTorch and a custom implementation of the KL divergence used in the iVAE model. For the adversarial loss we iterated over 4 different losses: The original GAN loss [7], two variations of the relativistic GAN loss, standard and average, [13] and the Wasserstein loss [2]. Interestingly, we found that the using the KL divergence loss from PyTorch and the custom KL divergence loss produced different results, even though they, to our knowledge, should calculate the same thing. We were unable to find any satisfactory explanation for this behavior and do not know whether or not it can be attributed to PyTorch specific details. However, throughout experiments we found the custom loss from the iVAE model to produce better results and kept that.

The loss functions for the adversarial training were used at different times throughout development, typically when it was no longer believed that performance could be improved through hyperparameter tuning. The different losses were chosen because they had appealing theoretical and practical properties. The relativistic GAN loss attempts to force the discriminator to decide between real and fake samples using the a priori knowledge that there are equally many real and fake samples. It is argued that this is important for training as it forces the discriminator to perceive real samples as "less" real when the generator becomes better at

generating realistic samples. The Wasserstein loss is a very popular GAN loss that provides stable gradients during training in order to improve convergence. This remedies the known problem of vanishing gradients that often lead to poor optimization at the later stages of training. We experimented with all four adversarial losses without seeing any significant boost in performance so when we finally achieved a functioning model this choice did not seem to be of much consequence. The reason could possibly be that our data are simpler than real images typically encountered with GAN or we are less perceptive of the quality of time series. For example, we could probably quickly look at an image of a face and determine that an eyebrow is slightly misplaced or missing, but we would probably not notice if some frequency components of a time series were missing. So for no apparent reason we found the model to work well using the standard GAN loss to update the discriminator and the Wasserstein loss to update the generator.

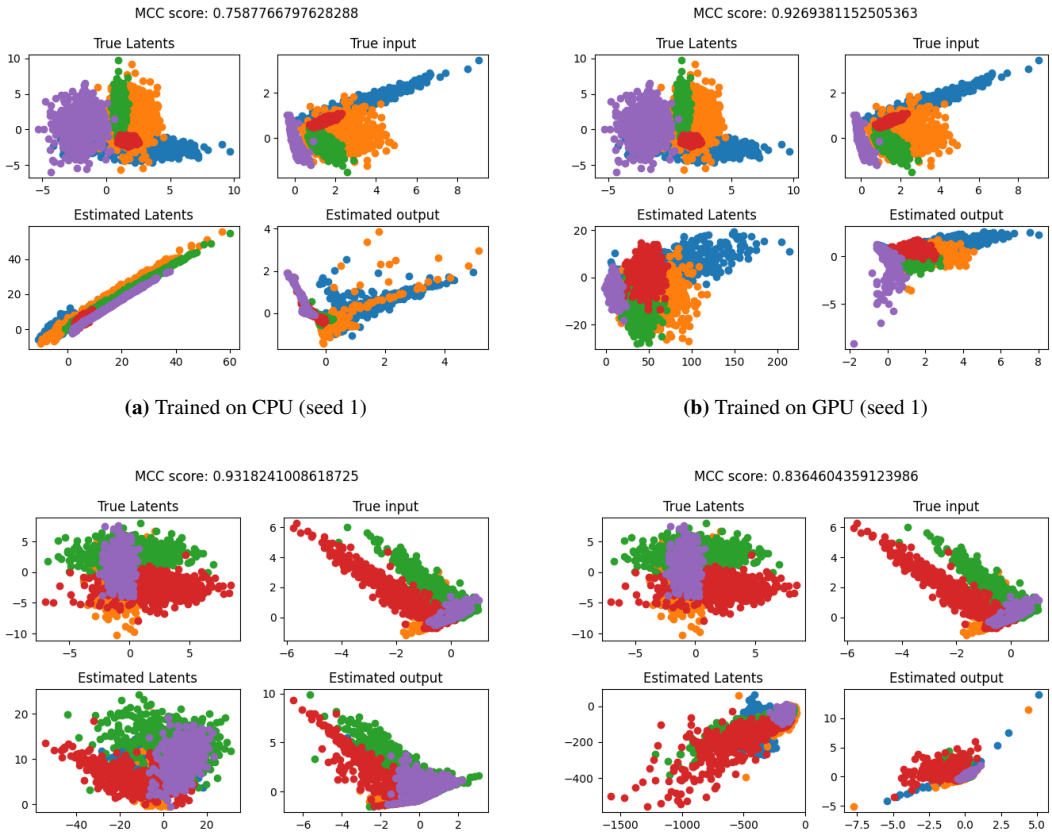
Batch normalization is very common for GANs, but we found it to be detrimental to identifiability in our model. We also implemented batch normalization in the iVAE model to compare and found similar results. We were not able to find out exactly why this was the case, but we also found similar tendencies when using dropout layers in the model. The quality of the generated outputs did not seem to suffer greatly from either method so perhaps the poor performance is a result of violating a model assumption such as the injectivity of $\mathbf{f}$, which could certainly be true for dropout layers.

The discriminator is essential for proper performance in iVAE-GAN and the most promising progress typically originated here. We found that it is key that the discriminator is allowed to discriminate based on full, ordered time series even though the decoder only produces individual data points. This is to ensure that enough information is presented to the discriminator. In early stages, the discriminator was fed individual data points but that only allowed the discriminator to determine whether a generated sample was within the range of real samples and thus outputs were heavily clustered and lacked any dependency on time. To introduce time dependency the discriminator was also tasked with predicting the auxiliary variable, which in our case is a segment index. This provided better performance, but the only solution was to present it with full time series data. Adding noise to the input of the discriminator also improves performance, although we have found that when using minibatches the effect of adding noise diminishes and can be removed altogether.

Initial weights seem to be very important for the rate of convergence in iVAE-GAN. Initially we were puzzled by very small gradients in the auxiliary network as compared to the rest of the networks and suspected that the auxiliary network remained mostly constant throughout training. To test the hypothesis we changed the initial weights of the networks to those suggested in DCGAN and found that performance decreased quite significantly.

To verify the result we changed the initial weights in the iVAE model, which has the same auxiliary network, but found no decrease in performance. After several experiments we found that the performance was regained if only iVAE-GAN was allowed to train for a lot more iterations. Therefore we concluded that, although small, the gradients update the auxiliary network appropriately and that the convergence rate of iVAE-GAN seems to be very sensitive to the choice of initial weights.

CUDA is used by PyTorch when a model is trained on an NVIDIA GPU which the Tesla V100 GPU we had access to in this project is. Using a GPU for training is of course desirable as a GPU is more optimized for the operations involved in machine learning and can thus train models faster. For some reason still unknown to us, we could expect widely different results depending on whether our model was trained on CPU or GPU as illustrated in Figure 4.4.



(a) Trained on CPU (seed 1)

(b) Trained on GPU (seed 1)

(c) Trained on CPU (seed 2)

(d) trained on GPU (seed 2)

Figure 4.4: The same iVAE-GAN models trained on either CPU or GPU

As seen from the figure, performance is not necessarily better or worse on either CPU or GPU. Only different. We also noticed different results if we used a different version of the

CUDA library. Such discrepancies are hard to identify but were most notable when a model would return NaN values in some runs but give reasonable results in another even though all hyperparameters were identical. We have found no good strategy for handling such behavior other than to exclusively obtain results using either the CPU or GPU.

Data preprocessing is the first thing recommended in [9]. Specifically, it is recommended to normalize inputs between [-1, 1] and use a hyperbolic tangent activation function as the last layer of the generator. We spend a good amount of time trying to improve performance of the model to no avail, until by chance we discovered that for some reason the model simply does not work with normalized inputs. After removing normalization model behavior returned to normal.

As we hope to have shown with the above findings the intricacies of GAN training often result in more questions than answers and reasonable explanations for why some things work and others do not can be few and far between.

The iVAE-GAN training scheme we have found to perform best and used to achieve our results is shown in Algorithm 1. It is worth noting that the discriminator is trained only if epoch modulo 4 is equal to 0. This has been necessary to avoid the common issue that the discriminator quickly becomes too strong and can discriminate between real and fake samples so confidently that the training signal for the generator becomes useless. Therefore, the generator is allowed to train four times as much as the discriminator. At the later stages of training the ratio can be reduced, but we found no reliable strategy for doing so and kept a

static ratio throughout training.

Algorithm 1: iVAE-GAN training scheme

Create DataLoader with shuffle=False, drop_last=True
Initialize the model with the chosen hidden dimensions and xavier_uniform weights
Initialize ADAM optimizers with lr= 0.001 and β = (0.5, 0.999)
`/* Training loop */`
while *Training* **do**
  if *epoch* mod *4* == *0* **then**
    `/* Train discriminator */`
    - Sample minibatch of *m* samples from the training dataset $((x,u)^{(1)},...,(x,u)^{(m)})$
    - Encode *m* samples into *m* latent variable vectors $(z^{(1)},...,z^{(m)})$
    - Perform gradient descent with ADAM

    $$\mathcal{L}_{\mathcal{D}} = BinaryCrossEntropy(D(x),1)+BinaryCrossEntropy(D(G(z),0)$$

  end
  `/* Train generator */`
  - Sample minibatch of *m* samples from the training dataset $((x,u)^{(1)},...,(x,u)^{(m)})$
  - Encode *m* samples into *m* latent variable vectors $(z^{(1)},...,z^{(m)})$
  - Perform gradient descent with ADAM

  $$\mathcal{L}_{\mathcal{G}} = -\text{mean}(D(G(z))) - KL(q_\phi(z|x,u)||p_\theta(z|u))$$

end

Hyperparameter tuning is a necessary evil - especially with adversarial training. This is primarily caused by the competing nature of the generator and discriminator which enable GANs to generate exceptionally realistic outputs. However, good results can only be achieved if there is some balance between the networks such that neither is too complex nor simple. This balance is not trivial and can dynamically shift throughout training such that methods that stabilize training during some epochs can suddenly destabilize training during other epochs. To monitor the effect of different changes in hyperparameters and reliably seek better parameters we have logged various metrics of interest such as the loss, MCC, percentage of fake images classified as real etc. We also saved generated images along with the magnitude of the gradients in each hidden layer of the network. To visualize and draw meaningful conclusions from our logged data we created the interactive plotting tool shown in Figures

4.5 and 4.6.

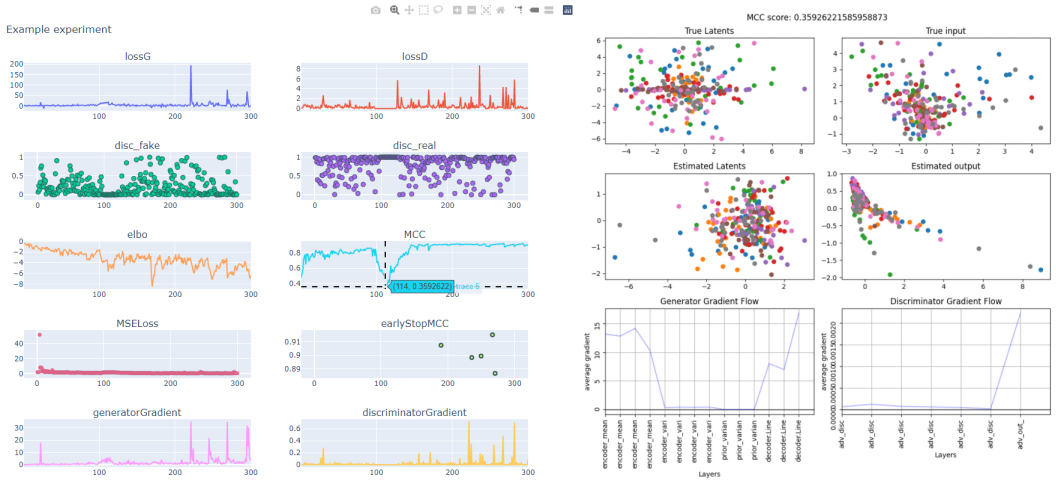


Figure 4.5: Interactive plots with updating images on hover

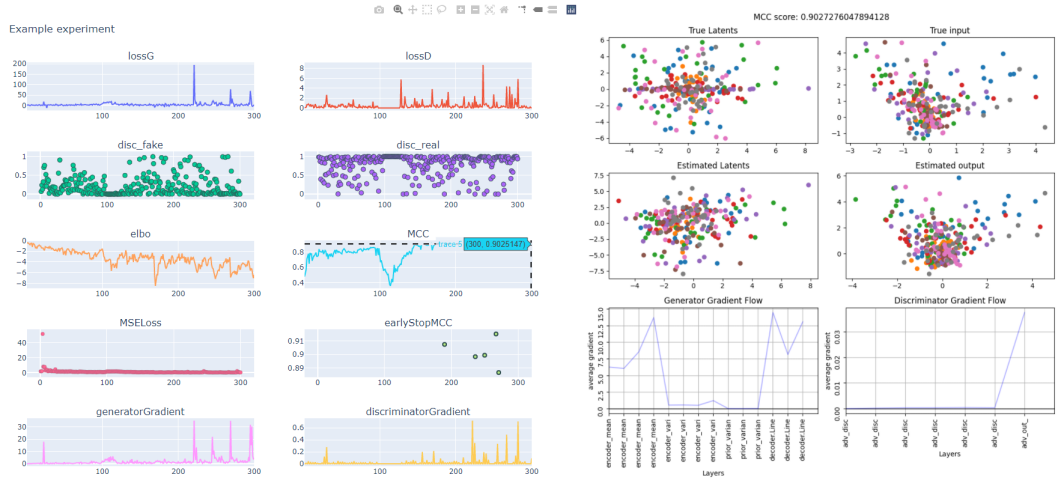


Figure 4.6: Interactive plots with updating images on hover

As the figures above show, all the logged metrics are displayed on the left in interactive graphs which can be zoomed, dragged etc. and on mouse hover each data point shows from which iteration it originates and what the logged value is. The right-hand side shows the images saved during training such that when the mouse hovers a datapoint on the interactive graphs the right hand side is updated to show the images generated for that specific datapoint. For each datapoint the saved images include the true latent variables and input, estimated latent variables and output as well as the magnitude of the gradients in each hidden layer. One

of the crucial aspects of hyperparameter tuning has been the convergence of the model. In Figures 4.7 and 4.8 we show the training convergence for iVAE-GAN across the 10 different seeds to highlight the training characteristics.

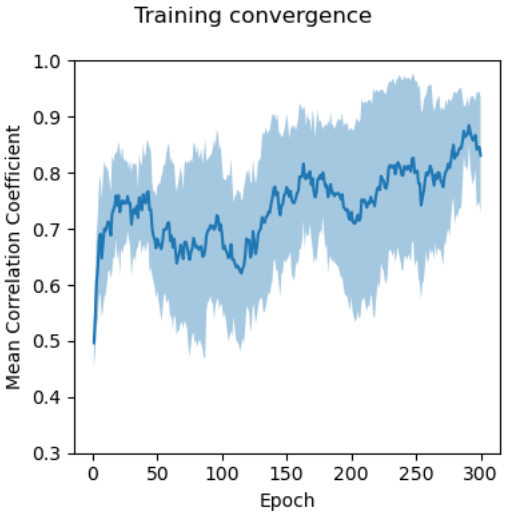


Figure 4.7: Mean and standard deviation of MCC during training

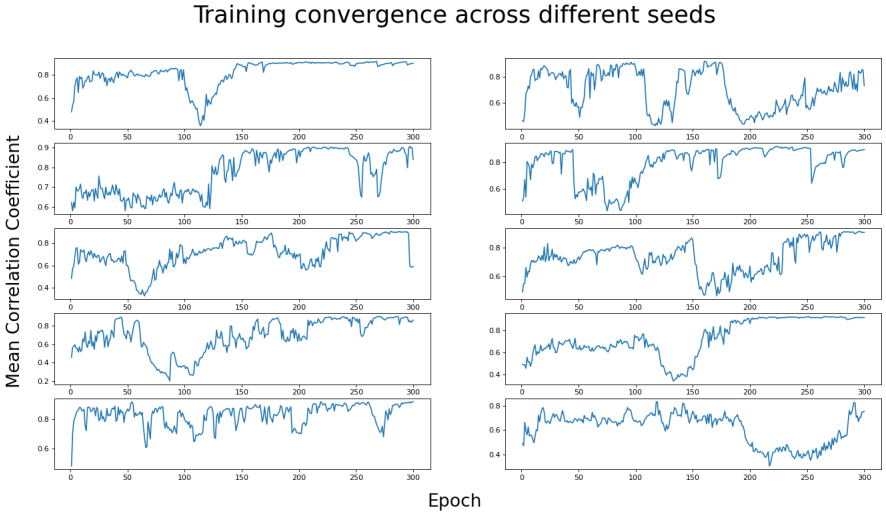


Figure 4.8: MCC during training for 10 different seeds

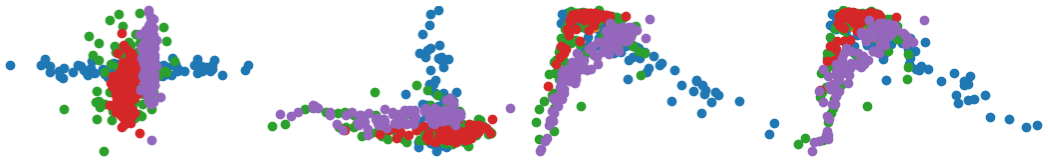
From the figures above it is obvious that convergence is not monotonic in iVAE-GAN as it has been shown to be in the iVAE model [15]. From Figure 4.7 a slight upwards trend can be seen such that more iterations result in better MCC on average, but as shown in Figure 4.8

on the facing page performance can vary drastically from seed to seed. We attempted to mitigate the non-monotonic convergence by using early stopping which is also common practice in regular GANs. However, we found no metric that could provide consistently better performance than simply allowing the model to train longer.

In the next section we will show our results and compare our model to other state-of-the-art identifiable models in order to validate our theoretical proofs and show that iVAE-GAN is indeed identifiable in practice.

4.1 Results

The core premise of the problem we aim to solve is that the latent variables are, by definition, never observed yet carry meaningful information about the seen data. In fact, we could argue that the latent variables are the most complete representation of data since it is from these variables that the seen data originates. However, because we do not have access to the latent variables they must be inferred from the observed data which are a nonlinear function of the latent variables. This premise is of great practical interest because it almost always reflects the true nature of data collection. In our case where we also learn a generative model not only can we infer about the origin of the data but also generate unseen data as seen on Figure 4.9.



(a) $p^*(z|u)$ **(b)** $p(z|x,u)$ **(c)** $p^*(x)$ **(d)** $p(x|z)$

Figure 4.9: 2-Dimensional data and latent spaces. We have omitted axes to emphasize the linear indeterminacy as a rotation. a) The original generating latent variables. b) The latent variables recovered by iVAE-GAN. c) Input data. d) Data generated by iVAE-GAN.

This unobservable nature of latent variables also has implications for our experiments since it means that datasets with known latent variables are very limited. Even for datasets where the latent variables would intuitively be very simple. Consider e.g. MNIST. It would be very intuitive to expect the true latent space to consist of ten independent distributions - one for each number. Yet, because it is very difficult to observe the latent space we cannot use such datasets to validate our model. Therefore we have strictly used a synthetic dataset such that there is no ambiguity with respect to the true latent variables. Of course the latent variables are of greatest interest in real data but the scope of this work has been to show that identifiability is possible in adversarial networks.

The synthetic dataset used in our experiments has been made with the same data generator used to validate the iVAE model [15]. This decision was made early in the process when we chose the iVAE model to be our baseline model as explained in the beginning of Chapter 4 on page 33. Combined with the discussed reality of datasets with known latent variables it was a natural choice both for practical reasons and to accurately compare the iVAE-GAN model to existing identifiable models.

The generated data are non-stationary Gaussian time series divided into segments. All segments are generated with equally many samples and can be changed freely through the parameters of the data generator. The latent variables used to generate the samples are drawn from an exponential family distribution with λ_i generated randomly and independent for each segment and passed through an uninitialized Multilayer Perceptron MLP to produce data that are a nonlinear function of the latent variables. The parameters of the data generator used in the experiments are shown in Table 4.1.

Table 4.1: Data generator parameters

Data dimension	Number of segments	Number of observations per segment	Mixing layers
2	5	{100, 200, 500, 1000, 2000}	3

The MCC metric was used to quantify identifiability in [15, 14] and we adopt the same metric to quantify identifiability in iVAE-GAN.

Given two sets of observations of m random variables each, the Mean Correlation Coefficient (MCC) metric calculates the interclass correlation coefficients (either Pearson or Spearman's correlation coefficients) between the m random variables of each set. Since every recovered latent variable should correspond to exactly one true latent variable a linear sum assignment problem is solved such that each recovered latent variable is assigned to exactly one true latent variable and the sum of the assigned correlation coefficients is maximized. The MCC score is then the mean of the correlation coefficients after assignment. A high MCC score thus reflects that the recovered latent variables are highly correlated with the true latent variables. This is a reasonable metric to use in our case instead of e.g., a mean square error between the true and estimated latent variables because our theory guarantees identifiability up to an equivalence relation. This means that our estimated latent variables may be a linear transformation of the true variables such that the estimated latent variables could for example just be a scaled version of the true latent variables. That would result in a mean square error proportional to the scaling which is misleading as we would have recovered the latent variables up to a linear transformation. Correlation coefficients are invariant to change in origin and scale and would therefore not be proportional to the scaling but instead measure the strength of the linear relationship between the estimated and true latent variables.

To provide as fair and objective comparison as possible we have made the network sizes between iVAE and iVAE-GAN as identical as possible because they share a somewhat similar architecture. The ICE-BeeM model does not share a similar architecture, so we have simply used the default model. All networks are fully connected MLPs with Leaky ReLU activation functions with a negative slope of 0.1. The hidden dimensions of the networks can be seen in Table 4.2.

Table 4.2: Hidden dimension sizes

Model	Encoder	Decoder	Auxiliary	Discriminator
iVAE-GAN	{4, 4, 4}	{4, 4, 4}	{4, 4, 4}	$6 \times \begin{Bmatrix} 32 \\ 32 \\ 256 \\ 1024 \\ 1024 \end{Bmatrix}$
iVAE	{4, 4, 4}	{4, 4, 4}	{4, 4, 4}	-
		ICE-BeeM		
Model	n_layers_flow	ebm_hidden_size		Comment
ICE-BeeM	10	32		Default parameters

We state multiple values for the discriminator because different widths were required for 100, 200, 500, 1000 and 2000 number of observations per segment, respectively. We believe this to be a result of a more complex discrimination task as the number of observations per segment increases. This seems to be consistent with common findings in GAN where more data points result in more complex networks, e.g., it is harder to get reasonable results with higher resolution images than with lower resolution images.

By using the data generated according to Table 4.1 on the preceding page combined with the network sizes of Table 4.2 we trained the iVAE-GAN model for 300000 iterations across 10 different seeds while the iVAE and ICE-BeeM model were trained for 70000 iterations across the same 10 seeds. The batch size used in iVAE and ICE-BeeM were 256 and 128, respectively, while the iVAE-GAN model was simply trained on the full dataset each iteration because it could fit inside the Tesla V100 GPU. The results are shown in Figure 4.10 on the next page.

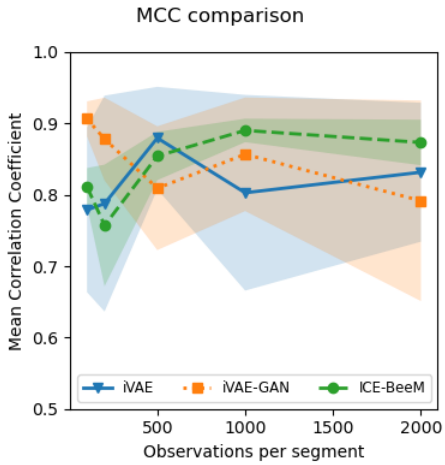


Figure 4.10: Comparison of iVAE-GAN with iVAE and ICE-BeeM

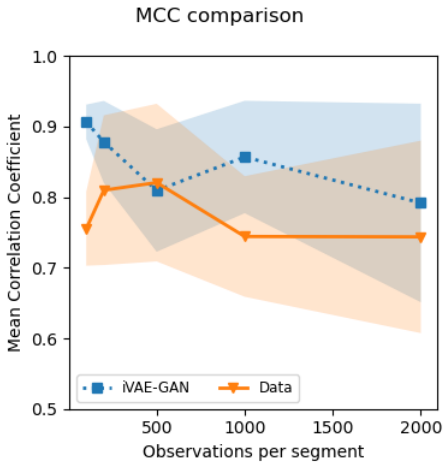


Figure 4.11: iVAE-GAN and MCC if data itself is interpreted as the latent representation

As it can be seen from Figure 4.10 iVAE-GAN can not be said to have strictly better or worse performance than iVAE and ICE-BeeM. Especially with fewer observations per segment the iVAE-GAN model seems to perform very well and as the number of observations is increased performance takes a hit and the standard deviation increases likewise. It is unclear whether or not this drop in performance can be attributed to the scaling properties of GAN [4]. But it supports our findings that the model become increasingly difficult to train and requires a lot more hyperparameter tuning as the number of observations increase. We include Figure 4.11 to provide a reference for the MCC metric and the used data. It may be natural to consider the range of the MCC metric, [0, 1], and immediately conclude that MCC > 0.8 must be way above chance level. In our case we interpret chance level as the MCC score between the true latent variables and the seen data. Because, if a model does not recover better estimates of the true latent variables than the raw input, it would have been better to interpret the raw input as observations from the latent space. The MCC between data and true latent variables also reflect the strength of the linear relationship between the two and is thus also a measure of how nonlinear the used mixing function is. Therefore, if the data MCC becomes too high it may also indicate that the mixing is trivial. This chance level is of course identical for all three models as they use the same data, and because iVAE-GAN performs competitively with iVAE and ICE-BeeM we conclude that our experiments validate our theoretical findings of identifiability.

Chapter 5

Conclusion

In this work we have proven that the novel iVAE-GAN model is identifiable and validated that our implementation achieves competitive identifiability results when compared to state-of-the-art identifiable models. By placing no assumptions or restrictions on the nature of the adversarial training our theoretical results extend to a wide variety of GAN flavors, meaning our results are general and of interest to the active research fields of both unsupervised learning and GANs. We have provided observations of the training behavior our model have exhibited during experimentation and find that the iVAE-GAN model share many training characteristics with regular GAN. This would suggest performance could potentially be improved in iVAE-GAN by leveraging methods that improve convergence known from other GAN research.

Future Work

Our contribution has shown the first identifiability proof in adversarial training. However, identifiability in the context of deep learning is still a very recent topic with many avenues to pursuit. We recognize one interesting direction to be applying and comparing the existing identifiable models, including iVAE-GAN, on datasets which are not synthetic to obtain more tangible results that are more readily understood. Had time allowed, we would have liked to apply our model on the public dataset DiPCo for the Cocktail Party problem provided by Amazon [1]. This Dinner Party Corpus (DiPCo) is an excellent target for nonlinear ICA which the theory of both this and previous works trivially includes [15]. The dataset contains clean speech as latent variables from each speaker as well as the speech perceived in the room, which is presumably a nonlinear mixing of the individual speech signals. Thus it is a dataset with known latent variables, which as discussed in the thesis are hard to come by. It is

a known problem i.e., results are readily understood by a broad community and well-known linear methods such as linear ICA has been applied to the problem, which means that it is also ideal to not only quantify the performance between the new nonlinear and identifiable models but also quantify the performance of the nonlinear methods to that of the known linear methods.

Theoretically we think it would have been interesting to examine whether some of the concepts from Bidirectional Generative Adversarial Networks (BiGAN) could be transferred to the theoretical framework of identifiability [5]. Specifically Theorem 2 states that if E and G of Figure 5.1 are an optimal encoder and decoder, respectively, then the encoder is the inverse of the decoder, $E = G^{-1}$.

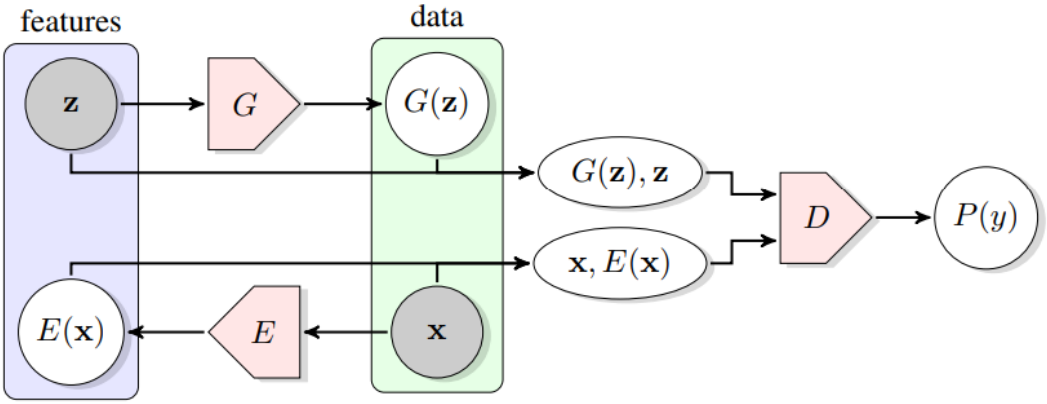


Figure 5.1: Structure of BiGAN [5]

iVAE-GAN already incorporates an encoder from data space to latent space as BiGAN, only in iVAE-GAN it is used to generate the latent space and in BiGAN it is used to infer where in latent space a sample resides (typically for the purpose of latent space exploration). If a similar proof could be made for the conceptual architecture in Figure 5.2 on the next page it would seem that the encoders E and E_2 must be equal if E is to succeed in estimating the correct latent variable. If that is the case and $E = G^{-1}$ then necessarily we must have that $E_2 = G^{-1}$. For an eventual implementation it is likely not even necessary to implement the encoder E in order to ensure $E_2 = G^{-1}$, but it seems practical to build the proof. It is of course not controversial to view the encoder and decoder as inverse mappings, but having an explicit proof may help in obtaining stronger identifiability proofs as it proves that $\tilde{\mathbf{f}}$ is an injective function and perhaps that can be used to relax the assumption that $\mathbf{f}$ is injective. Naturally, it would seem impossible to recover the latent variables if $\mathbf{f}$ is not injective so perhaps it can subsequently be proven that estimation is impossible given $\mathbf{f}$ is not injective. At the very least it is believed that such a proof would aid the understanding of the estimation achieved by iVAE-GAN.

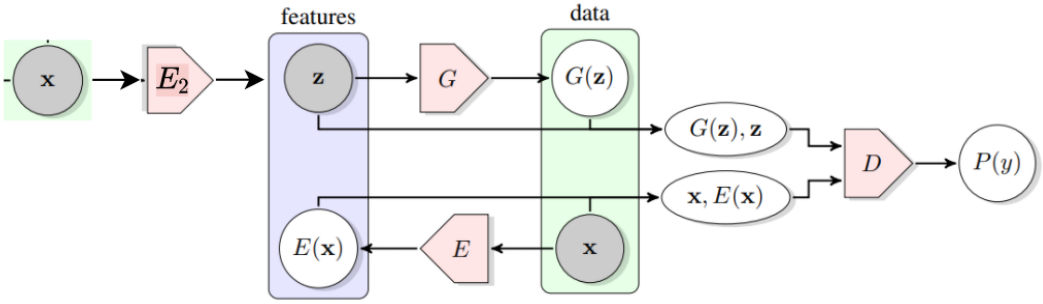


Figure 5.2: Conceptual drawing of BiGAN structure together with iVAE-GAN (omitting the auxiliary network for clarity)

Bibliography

[1] *Amazon Releases New Public Data Set to Help Address "Cocktail Party" Problem*. https://www.amazon.science/blog/amazon-releases-new-public-data-set-to-help-address-cocktail-party-problem. Accessed: 09-06-2021.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. PMLR, 2017, pp. 214–223. URL: http://proceedings.mlr.press/v70/arjovsky17a.html.

[3] Adi Ben-Israel. "The Change-of-Variables Formula Using Matrix Volume". In: *Siam Journal on Matrix Analysis and Applications* 21 (1999), pp. 300–312. DOI: 10.1137/S0895479895296896.

[4] Andrew Brock, Jeff Donahue, and Karen Simonyan. *Large Scale GAN Training for High Fidelity Natural Image Synthesis*. 2019. arXiv: 1809.11096 [cs.LG].

[5] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. *Adversarial Feature Learning*. 2017. arXiv: 1605.09782 [cs.LG].

[6] *GAN common problems*. https://developers.google.com/machine-learning/gan/problems. Accessed: 08-06-2021.

[7] Ian J Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. 2014.

[8] Trygve Haavelmo. *The Probability Approach in Econometrics*. http://fitelson.org/woodward/haavelmo.pdf. 1944.

[9] *How to Train a GAN? Tips and tricks to make GANs work*. https://github.com/soumith/ganhacks. Accessed: 08-06-2021.

[10] Aapo Hyvarinen and Hiroshi Morioka. "Nonlinear ICA of Temporally Dependent Stationary Sources". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Ed. by Aarti Singh and Jerry Zhu. Vol. 54. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, 20–22 Apr 2017, pp. 460–469. URL: http://proceedings.mlr.press/v54/hyvarinen17a.html.

[11] Aapo Hyvarinen and Hiroshi Morioka. "Unsupervised Feature Extraction by Time-Contrastive Learning and Nonlinear ICA". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper/2016/file/d305281faf947ca7acade9ad5c8c818c-Paper.pdf.

[12] Aapo Hyvärinen and Petteri Pajunen. "Nonlinear independent component analysis: Existence and uniqueness results". In: *Neural networks* 12.3 (1999), pp. 429–439.

[13] Alexia Jolicoeur-Martineau. *The relativistic discriminator: a key element missing from standard GAN*. 2018. arXiv: 1807.00734 [cs.LG].

[14] Ilyes Khemakhem et al. "ICE-BeeM: Identifiable Conditional Energy-Based Deep Models Based on Nonlinear ICA". In: *Advances in Neural Information Processing Systems* 33 (2020).

[15] Ilyes Khemakhem et al. "Variational autoencoders and nonlinear ica: A unifying framework". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 2207–2217.

[16] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *stat* 1050 (2014), p. 1.

[17] Francesco Locatello et al. "Challenging common assumptions in the unsupervised learning of disentangled representations". In: *international conference on machine learning*. PMLR. 2019, pp. 4114–4124.

[18] *NeurIPS 2021 conference*. https://nips.cc/Conferences/2021. Accessed: 10-06-2021.

[19] *NVIDIA DGX-2*. https://www.nvidia.com/en-us/data-center/dgx-2/. Accessed: 08-06-2021.

[20] Ullrich Kothe Peter Sorrenson Carsten Rother. *Disentanglement by Nonlinear ICA with General Incompressible-flow Networks (GIN)*. https://arxiv.org/pdf/2001.04872.pdf. 2020.

[21] Moustapha Cisse Pierre Stock. *ConvNets and ImageNet Beyond Accuracy: Understanding Mistakes and Uncovering Biases*. https://arxiv.org/pdf/1711.11443.pdf. 2018.

[22] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016. arXiv: 1511.06434 [cs.LG].

[23] Bharath Sriperumbudur et al. "Density estimation in infinite dimensional exponential families". In: *Journal of Machine Learning Research* 18 (2017).

[24] *This X does not exist*. https://thisxdoesnotexist.com/. 2021.

[25] *tqdm Discord contrib*. https://tqdm.github.io/docs/contrib.discord/. Accessed: 08-06-2021.

[26] Pascal Vincent Yoshua Bengio Aaron Courville. *Representation Learning: A Review and New Perspectives*. `https://arxiv.org/pdf/1206.5538.pdf`. 2012.

Appendix A

GAN training tips

1. Label smoothing – labels should be close to 0 or 1, not binary values.
2. Occasionally flip labels – real -> 0, fake -> 1
3. Relativistic GAN objective – Use that we know we provide 50/50 of real and fake samples.
4. Historical averaging – Use a historical loss to optimize
5. Activation functions – ReLU, leakyReLU, tanh (normalize input to [-1, 1]?)
6. No batchnorm in generator output layer and discriminator input layer
7. Mixing inputs to the discriminator – try not giving it all real or all fake samples at a time
8. Mini batches of all real or fake samples to calculate batch norm statistics
9. Discriminator loss of 0.0 is a failure mode
10. Add noise to inputs to the discriminator and decay the noise over time.
11. Use dropout of 50 percent during train and generation.
12. ADAM optimizer – DCGAN learning rate -> 0.0002 momentum -> 0.5
13. Perhaps SGD in discriminator
14. Weight initialization – DCGAN normal distribution (0, 0.02)
15. (Remove fully connected layers)

16. Layer type – Linear, convolutional, RNN etc.
17. Batch size – Perhaps below 64 and even as low as 8 16
18. Plot continuously – performance can degrade with longer training time
19. Scheduler – adjust learning rate while running.
20. Ratio of losses
21. Training ratio of generator and discriminator – Not generally advised unless good reasoning
22. Experience replay – Past checkpoints of models, swap them out for a few iterations and let them train a few cycles.+
23. Gradients seem to be a good indicator of training
24. Try to use auxiliary variable as label
25. Pretrain generator or discriminator
26. Pass auxiliary variable (u) as an int instead of one-hot-encoded variable.

Appendix B

Paper submitted to NeurIPS

iVAE-GAN: Identifiable VAE-GAN Models for Latent Representation Learning

Bjørn Uttrup Dideriksen*
Department of Electronic Systems
Aalborg University
Fredrik Bajers Vej 7, Aalborg 9220, Denmark
bdider16@student.aau.dk

Kristoffer Calundan Derosche*
Department of Electronic Systems
Aalborg University
Fredrik Bajers Vej 7, Aalborg 9220, Denmark
kderos16@student.aau.dk

Zheng-Hua Tan
Department of Electronic Systems
Aalborg University
Fredrik Bajers Vej 7, Aalborg 9220, Denmark
zt@es.aau.dk

Abstract

Remarkable progress has been made within nonlinear Independent Component Analysis (ICA) and identifiable deep latent variable models. Formally, the latest nonlinear ICA theory enables us to recover the true latent variables up to a linear transformation by leveraging unsupervised deep learning. This is of significant importance for unsupervised learning in general as the true latent variables are of principal interest for meaningful representations. These theoretical results stand in stark contrast to the mostly heuristic approaches used for representation learning which do not provide analytical relations to the true latent variables. By combining the training guarantees of Generative Adversarial Network (GAN) and the latest nonlinear ICA theory we extend the family of identifiable models by proposing an identifiable Variational Autoencoder (VAE) based GAN model we name iVAE-GAN. The latent space of most GANs, including VAE-GAN, is generally unrelated to the true latent variables. With iVAE-GAN we show the first principal approach to a theoretically meaningful latent space by means of adversarial training. We implement the novel iVAE-GAN architecture and show its identifiability, which is confirmed by experiments. The GAN objective is believed to be an important addition to identifiable models as it is one of the most powerful deep generative models. We hope such work can inspire other constructions of meaningful latent spaces not based solely on heuristic approaches. Furthermore, existing GANs may be reformulated to learn identifiable latent variables with only slight additions to architecture and dataset.

1 Introduction

One of the biggest challenges facing machine learning and unsupervised learning in particular is representation learning. A very recent leap in meaningful representation learning is the understanding of identifiability in deep latent variable models by Khemakhem et al. [2020a,b]. *Identifiability* has origins in early econometrics as shown by the "problem of confluent relations (or problem of arbitrary parameters)" Le Gall [2002]. The formulation states that if two or more parametrizations of the same

*These authors contributed equally

Preprint. Under review.

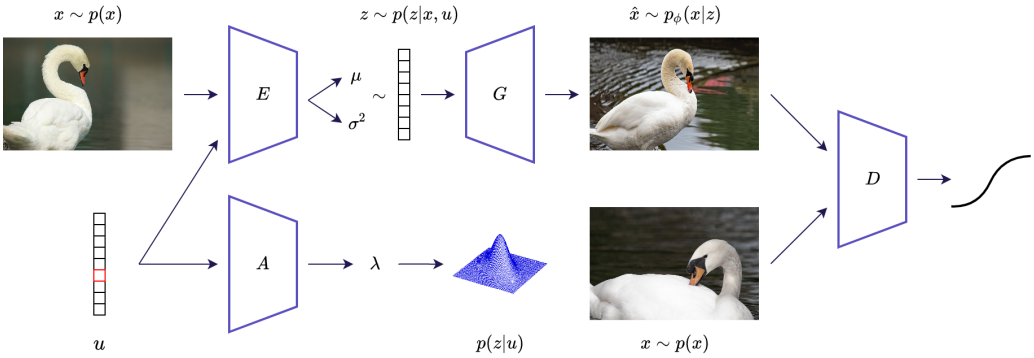


Figure 1: The proposed iVAE-GAN architecture. The latent variable model consists of four neural networks: an encoder E that learns the latent variables, a decoder/generator G that generates data in the original data space, a discriminator D that discriminates generated data from real data, and an auxiliary network A that learns the natural parameters, $\lambda_i(u)$, of an exponential family from the observed auxiliary variables.

model lead to the same joint distribution over observed random variables they are indistinguishable on the basis of observations and therefore *unidentifiable*. Up until recently the general consensus in literature agreed that arbitrary nonlinear functions, such as those modeled by neural networks, were almost surely unidentifiable under the assumption of independent latent variables Locatello et al. [2019], Hyvärinen and Pajunen [1999a]. It is now understood that identifiability results can be achieved if the model assumes *conditionally* independent latent variables. That is, given an additionally observed variable under which the latent variables are independent, they may be estimated up to a linear transformation and in certain cases reduced to a simple scaled permutation. The nonlinear maps from observable to latent variables need not preserve dimensionality, but if they do it is worth noting that the identifiability results become interpretable as nonlinear Independent Component Analysis (ICA) Hyvärinen and Pajunen [1999b].

It has been shown that identifiability can be achieved in Variational Autoencoders (VAE) Khemakhem et al. [2020b], Energy-Based Models (EBM) Khemakhem et al. [2020a] and General Incrompressible-flow Networks (GIN) Sorrenson et al. [2020]. However, it has remained an open question whether identifiability is possible in Generative Adversarial Networks (GANs) Goodfellow et al. [2014]. In this work we leverage the new identifiability results to propose the first identifiable GAN by using variational inference (iVAE-GAN) as shown in Figure 1 and we validate it with experiments. The source code for the model implementation and the experiments will be made publicly available when the paper gets published.

2 Existing identifiability theory

This section reiterates the necessary theoretical results needed to propose iVAE-GAN. This is important, as it not only shows the theoretical justification for identifiability in iVAE-GAN, but also solidifies that the followed theoretical framework is general and could potentially be applied to a wider family of deep learning models.

Identifiability A model is said to be identifiable if only one parametrization of the model can lead to the observed data distribution, i.e.

$$if \quad p_{\boldsymbol{\theta}_1}(\mathbf{x}) = p_{\boldsymbol{\theta}_2}(\mathbf{x}) \quad \Rightarrow \quad \boldsymbol{\theta}_1 = \boldsymbol{\theta}_2 \qquad \forall(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) \tag{1}$$

On the other hand if $p_{\theta_1}(x) = p_{\theta_2}(x)$ but the parametrization is not unique, $\theta_1 \neq \theta_2$, the model is said to be unidentifiable on the basis of observations.

It is clear that identifiability is of interest in deep latent variable models as elaborated in the following. Latent variable models commonly model the joint distribution as:

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z}) \tag{2}$$

for $\mathbf{x} \in \mathbb{R}^d$, and $\mathbf{z} \in \mathbb{R}^n$ (lower-dimensional, $n \leq d$), but only provide training guarantees on the marginal distribution of the model (such as learning a lower bound on the observed marginal distribution):

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} \tag{3}$$

Unfortunately deep latent variable model as in Equation (2) do not learn the true joint distribution and as a result can not recover the original latent variables. In contrast, it is sufficient for identifiable models to learn the marginal distribution because only one parametrization of the joint distribution produces the seen marginal distribution and therefore the latent variables can be recovered.

Identifiable Model Khemakhem et al. [2020b,a] have derived a very general deep latent variable model that is identifiable up to linear equivalence relations. Their work highlights that it is pivotal that the prior distribution is conditioned on an additional observed variable, u. Therefore, the general form of the identifiable model becomes:

$$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}|\mathbf{u}) = p_{\mathbf{f}}(\mathbf{x}|\mathbf{z})p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u}) \tag{4}$$

where $\mathbf{u} \in \mathbb{R}^m$ is the auxiliary variable observed alongside the data and $\boldsymbol{\theta} = (\mathbf{f}, \mathbf{T}, \boldsymbol{\lambda})$ are the parameters of the conditional generative model. The conditional latent distribution, $p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u})$, is assumed to belong to an exponential family of independent variables:

$$p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u}) = \prod_{i}^{n} \frac{Q_i(z_i)}{Z_i(\mathbf{u})} \exp\left[\sum_{j=1}^{k} T_{i,j}(z_i)\lambda_{i,j}(\mathbf{u})\right] \tag{5}$$

where $Q_i(z_i)$ is the base measure, $Z_i(u)$ is the normalization coefficient, $T_{i,j}(z_i)$ are the sufficient statistics and $\lambda_{i,j}(u)$ are the natural parameters of the family. The natural parameters of the distribution depending on $\mathbf{u}$ is learnt by the network we name A in our implementation. The assumption that the latent distribution must belong to an exponential family is not considered restrictive as it has been shown to have universal approximation capabilities by Sriperumbudur et al. [2017]. The decoder, $p_{\mathbf{f}}(\mathbf{x}|\mathbf{z})$, is defined as:

$$p_{\mathbf{f}}(\mathbf{x}|\mathbf{z}) = p_{\boldsymbol{\epsilon}}(\mathbf{x} - \mathbf{f}(\mathbf{z})) \tag{6}$$

allowing $\mathbf{x}$ to be decomposed into $\mathbf{x} = \mathbf{f}(\mathbf{z}) + \boldsymbol{\epsilon}$, where the noise is distributed according to $p_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon})$ and $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^d$ is assumed injective.

The auxiliary variable $\mathbf{u}$ is pivotal to the definition of the identifiable model. The auxiliary variable is an observed variable such that the collected dataset contains pairwise observations of both the data, $\mathbf{x}$, and $\mathbf{u}$ such that $\mathcal{D} = \{(\mathbf{x}^{(1)}, \mathbf{u}^{(1)}), \ldots, (\mathbf{x}^{(N)}, \mathbf{u}^{(N)})\}$. From Equation (5) it can be seen that it is critical that the latent variables are independent given $\mathbf{u}$. It would be natural to wonder: How do we know the latent variables, which by definition are never observed, are independent given $\mathbf{u}$? In short; we do not know. Often this will be application specific and rely on knowledge of the data at hand. For most labeled datasets, such as MNIST, $\mathbf{u}$ could simply be the label.

The identifiability result states that, given the data we observe is the marginal distribution of some generating process with joint distribution conditioned on $\mathbf{u}$ with true generating parameters $(\mathbf{f}, \mathbf{T}, \boldsymbol{\lambda})$:

$$p_{(\mathbf{f},\mathbf{T},\boldsymbol{\lambda})}(\mathbf{x}|\mathbf{u}) = \int p_{\mathbf{f}}(\mathbf{x}|\mathbf{z})p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u})\,\mathrm{d}\mathbf{z} \tag{7}$$

and a deep generative model of the same form learns to approximate the marginal distribution of observed data with parameters $(\tilde{\mathbf{f}}, \tilde{\mathbf{T}}, \tilde{\boldsymbol{\lambda}})$ such that:

$$p_{(\mathbf{f},\mathbf{T},\boldsymbol{\lambda})}(\mathbf{x}|\mathbf{u}) = p_{(\tilde{\mathbf{f}},\tilde{\mathbf{T}},\tilde{\boldsymbol{\lambda}})}(\mathbf{x}|\mathbf{u}) = \int p_{\tilde{\mathbf{f}}}(\mathbf{x}|\mathbf{z}) p_{\tilde{\mathbf{T}},\tilde{\boldsymbol{\lambda}}}(\mathbf{z}|\mathbf{u}) \, \mathrm{d}\mathbf{z} \tag{8}$$

Then the parameters $(\mathbf{f}, \mathbf{T}, \boldsymbol{\lambda})$ and $(\tilde{\mathbf{f}}, \tilde{\mathbf{T}}, \tilde{\boldsymbol{\lambda}})$ are said to be $\sim_A -identifiable$ such that:

$$(\mathbf{f}, \mathbf{T}, \boldsymbol{\lambda}) \sim_A (\tilde{\mathbf{f}}, \tilde{\mathbf{T}}, \tilde{\boldsymbol{\lambda}}) \Leftrightarrow \mathbf{T}(\mathbf{f^{-1}(x)}) = A\tilde{\mathbf{T}}(\tilde{\mathbf{f}}^{-1}(\mathbf{x})) + \mathbf{c} \tag{9}$$

for some $nk \times nk$ invertible matrix A and vector $\mathbf{c}$. We provide a walk-through of the original proof in Appendix B. Main points from the proof include that with a small assumption on the nature of the noise in Equation (6) the underlying noiseless distributions of the models must be equal. By using said equality a system of equations can be constructed because the noiseless distributions are equal for all u. This system of equations has a matrix representation of the form $L^T\mathbf{T}(\mathbf{f^{-1}(x)}) = \tilde{L}^T\tilde{\mathbf{T}}(\tilde{\mathbf{f}}^{-1}(\mathbf{x})) + \mathbf{b}$. The entries of the matrix L are a function of points of u. It is assumed that there exists at least $nk + 1$ points of u such that the matrix L is invertible. And lastly an assumption on the sufficient statistics, T, is made to prove the final equivalence relation.

This theoretical result is significant because it states that the trained deep generative model will have recovered the original latent variables, $\mathbf{f}^{-1}(\mathbf{x}) = \mathbf{z}$, up to a linear transformation of the sufficient statistics.

The theory requires that estimation models must follow a deep latent variable model with a conditional prior as seen from Equation (4) and it must be able to approximate the seen marginal distribution. The proposed iVAE-GAN model learns both a variational approximation, $q_\phi(z|x,u)$, of the posterior, $p_\theta(z|u)$, and a generative model and is therefore an appropriate deep latent variable model. In the next section we show that the iVAE-GAN model also fulfills the second condition and thereby make the first link between identifiability and GAN.

3 iVAE-GAN

iVAE-GAN has a hybrid loss function that consists of a divergence loss for the encoding of the latent space with respect to the prior (conditional) distribution and an adversarial loss for the generated samples such that:

$$\mathcal{L}_{iVAE-GAN} = \mathcal{L}_{prior} + \mathcal{L}_{GAN} \tag{10}$$

where we define:

$$\mathcal{L}_{prior} = -KL(q_\phi(z|x,u)||p_\theta(z|u)) \tag{11}$$

and

$$\mathcal{L}_{GAN} = V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z|u)}[\log(1 - D(G(z)))] \tag{12}$$

Training is then performed according to:

$$\begin{aligned} &\max_\phi \mathcal{L}_{prior} + \min_G \max_D \mathcal{L}_{GAN} \\ &\qquad = \max_\phi -KL(q_\phi(z|x,u)||p_\theta(z|u)) + \min_G \max_D V(G,D) \end{aligned} \tag{13}$$

In the following we show that the loss is a lower bound on the difference between the log probability of the data and the expected log likelihood of the data generated by the decoder:

$$\log p(x) - \mathbb{E}_{z\sim q\phi(z|x,u)}[\log p_\Phi(x|z)] \geq \mathcal{L}_{iVAE-GAN} \tag{14}$$

and that by maximizing $\mathcal{L}_{iVAE-GAN}$ the expected log likelihood of the data generated by the decoder approaches the log probability of the data.

$\mathcal{L}_{prior}$ of the iVAE-GAN loss is related to the ELBO loss (Kingma and Welling [2014]) such that:

$$\log p(x) - \mathbb{E}_{z\sim q_\phi(z|x,u)}[\log p_\Phi(x|z)] \geq -KL(q_\phi(z|x,u)||p_\theta(z|u)) = \mathcal{L}_{prior} \tag{15}$$

Now we show that the same inequality is also fulfilled by $\mathcal{L}_{prior} + \mathcal{L}_{GAN}$, but in contrast to Equation (15) the data distribution may be learnt by maximizing $\mathcal{L}_{prior} + \mathcal{L}_{GAN}$. We assume an optimal discriminator, D^*, and use the result of Goodfellow et al. [2014]. Therefore we can write the optimization of $\mathcal{L}_{GAN}$ as (See Appendix A for proof):

$$\min_G \max_D \mathcal{L}_{GAN} = \min_G V(G, D^*) = \min_G -\log(4) + 2 \cdot JSD(p(x)||p_\Phi(x)) \tag{16}$$

To write our loss function only as a function that is to be maximized we pose the minimization over G as a maximization:

$$\min_G V(G, D^*) \equiv \max_G -V(G, D^*) = \max_G \log(4) - 2 \cdot JSD(p(x)||p_\Phi(x)) \tag{17}$$

Since we mean to maximize this function using a deep neural network the constant $\log(4)$ is inconsequential to the loss function. Therefore:

$$\min_G V(G, D^*) \equiv \max_G -2 \cdot JSD(p(x)||p_\Phi(x)) \tag{18}$$

Since the negated Jensen-Shannon divergence is non-positive it can always be added to the lesser side of an inequality without altering the inequality. Therefore we recover our lower bound by adding $-2 \cdot JSD(p(x)||p_\Phi(x|z))$ to Equation (15):

$$\log p(x) - \mathbb{E}_{z\sim q_\phi(z|x,u)}[\log p_\Phi(x|z)] \geq \underbrace{-KL(q_\phi(z|x,u)||p_\theta(z|u))}_{\mathcal{L}_{prior}} \underbrace{- 2 \cdot JSD(p(x)||p_\Phi(x))}_{V(G,D^*)} \tag{19}$$

The right-hand side of Equation (19) can be recognized as the iVAE-GAN loss (updated according to Equation (16) and (18)):

$$\mathcal{L}_{iVAE-GAN} = \mathcal{L}_{prior} - V(G, D^*) \tag{20}$$

Thus the iVAE-GAN loss is a lower bound on the difference between the log probability of observed data and expected log likelihood of the data generated by the decoder. To, hopefully, make Equation (19) a little more interpretable we can make use of Jensen's inequality:

$$\mathbb{E}[\log(X)] \leq \log \mathbb{E}[X] \tag{21}$$

Therefore we can write:

$$\begin{aligned} \log p(x) - \mathbb{E}_{z\sim q_\phi(z|x,u)}[\log p_\Phi(x|z)] &\leq \log p(x) - \log \mathbb{E}_{z\sim q_\phi(z|x,u)}[p_\Phi(x|z)] \\ &= \log p(x) - \log \int p_\Phi(x|z) p_\phi(z) dz = \log p(x) - \log p_\Phi(x) \end{aligned} \tag{22}$$

By using the transitive property of inequalities we may write the lower bound as:

$$\log p(x) - \log p_\Phi(x) \geq -KL(q_\phi(z|x,u)||p_\theta(z|u)) - 2 \cdot JSD(p(x)||p_\Phi(x)) \tag{23}$$

The Jensen-Shannon divergence measures the distance between two distributions and is therefore closely related to the difference of log probabilities, so as the lower bound is maximized the difference between log probabilities is minimized. In fact, the only condition for which the Jensen-Shannon divergence vanishes is $p(x) = p_\Phi(x)$, at which point the left-hand side becomes zero and the lower bound becomes:

$$0 \geq -KL(q_\phi(z|x,u)||p_\theta(z|u)) \tag{24}$$

Which is of course the normal bound for the negative KL divergence. Therefore, by maximizing $\mathcal{L}_{iVAE-GAN}$ we learn the data distribution while simultaneously maximizing the negative KL divergence between the encoded distribution, $q_\phi(z|x,u)$, and the prior distribution, $p_\theta(z|u)$.

This is, to the best of authors knowledge, the first work to actually extend and apply the theoretical framework to a deep latent variable model not contained in the original works by Khemakhem et al.

[2020b,a]. The developed identifiability theory is claimed to be very general and extendable to a wide range of models and applications - a belief the authors of this paper share. Therefore it is not insignificant that we have shown how it extends to GAN. The training algorithm for iVAE-GAN is shown below.

Algorithm 1: iVAE-GAN training scheme

Initialize the model
Initialize the ADAM optimizer by Kingma and Ba [2014]
while *Training* **do**
  for *k discriminator steps* **do**
    - Sample minibatch of *m* samples from the training dataset $((x,u)^{(1)},...,(x,u)^{(m)})$
    - Encode *m* samples into *m* latent variable vectors $(z^{(1)},...,z^{(m)})$
    - Perform gradient ascent with ADAM

    $$\mathcal{L}_{\mathcal{D}} = BinaryCrossEntropy(D(x),1) + BinaryCrossEntropy(D(G(z),0) \quad (25)$$

  end
  for *i generator steps* **do**
    - Sample minibatch of *m* samples from the training dataset $((x,u)^{(1)},...,(x,u)^{(m)})$
    - Encode *m* samples into *m* latent variable vectors $(z^{(1)},...,z^{(m)})$
    - Perform gradient ascent with ADAM

    $$\mathcal{L} = BinaryCrossEntropy(D(G(z)),1) - KL(q_\phi(z|x,u)||p_\theta(z|u)) \quad (26)$$

  end
end

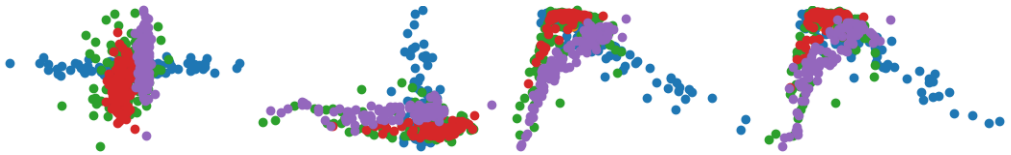
It is no coincidence that we consider GANs as a valuable framework to make identifiable. GANs are a very active area of research with state-of-the-art generative models and a wide range of applications. Providing proof of identifiability and initial experiments expand the toolkit researchers have at their disposal when meaningful latent spaces are desired in GANs. Importantly, the results we have shown do not impose or assume any restrictions on the adversarial training, therefore identifiability should be attainable in a large variety of GAN flavors that have other desirable properties such as stable training or alternative formulations of the minimax game.

4 Experiments

The core premise of the problem we aim to solve is that the latent variables are, by definition, never observed. Only the data which are a nonlinear function of the latent variables are observed. This premise is of great practical interest because it almost always reflects the true nature of data collection and the latent variables carry valuable information about the data. In our case where we also learn a generative model not only can we infer about the origin of the data but also generate unseen data.

However, this also means that datasets with known latent variables are very limited, even for datasets where the latent variables would intuitively be very simple. Consider e.g. MNIST. It would be very intuitive to expect the true latent space to consist of ten independent distributions - one for each number. Yet, because it is very difficult to observe the latent space we cannot use such datasets to validate our model. Therefore we have strictly used a synthetic dataset such that there is no ambiguity with respect to the true latent variables. Of course the latent variables are of greatest interest in real data but the scope of this work has been to show that identifiability is possible in adversarial networks.

Dataset We have created our dataset with the data generator graciously provided in Khemakhem et al. [2020a]. There are two main reasons for this choice: As discussed above datasets with known latent variables are very limited and secondly the same data generator has been used with other identifiable models, so it is a suitable generator to compare models across the same data and parameters. The data are generated in segments such that they become a non-stationary Gaussian time series. All segments are generated with equally many samples. The latent variables are drawn from an exponential family distribution with λ_i generated randomly and independent for each segment



(a) $p^*(z|u)$ (b) $p(z|x,u)$ (c) $p^*(x)$ (d) $p(x|z)$

Figure 2: 2-Dimensional data and latent spaces. We have omitted axes to emphasize the linear indeterminacy as a rotation. a) The original generating latent variables. b) The latent variables recovered by iVAE-GAN. c) Input data. d) Data generated by iVAE-GAN.

and passed through an uninitialized Multilayer Perceptron MLP to produce data that are a nonlinear function of the latent variables.

As it can be seen from Figure 2, iVAE-GAN generates similar but slightly different data, but most importantly it can be seen that the recovered latent variables are related to the true latent variables by a linear transformation, in this case a $90°$ clockwise rotation. To experimentally show identifiability we compare our model to iVAE and ICE-Beem Khemakhem et al. [2020b,a] as shown in Figure 3. The encoder, decoder and auxiliary network have the same size across all three models while the iVAE-GAN model additionally has the discriminator network.

During training we have noted the following remarks:

- iVAE-GAN inherits common training instabilities associated with adversarial training.
- Training is not consistently seen to monotonically converge. See Figures 4 and 5.
- Latent variables are only recovered well if the network learns to generate good data. Therefore the output of the model may act as a good proxy for early stopping.
- Discriminator size is vital for well-behaved training. In practice we have used a VAE model to find a decoder network sufficiently complex to express the data and then tuned discriminator hyperparameters.

Our experiments have convinced us that identifiability is achievable not only in the model we have presented here, but in adversarial training generally without sacrificing the desired properties that make adversarial training appealing. Since the training stability of our model greatly resembles the notoriously challenging training of most GANs, future work could greatly benefit from various developed methods aimed at stabilizing GAN training as described by Salimans et al. [2016], Gulrajani et al. [2017], Arjovsky et al. [2017], Xu et al. [2020], Jenni and Favaro [2019].

5 Discussion

The main contribution of this paper is to show that GAN models can be made identifiable. This was shown in a two-fold manner, firstly in the theoretical proof and secondly by experimentally validating our result. Our GAN implementation is quite simple and as such it has tendencies to easily become unstable on larger data, this could however be remedied using various methods to stabilize training.

The auxiliary variable is selected to be the segment index in the experiments, and this is to incorporate the temporal structure which is vital to identifiability as mentioned by Hyvärinen and Pajunen [1999a]. As such it is imperative that if using the model on a dissimilar dataset, one chooses a substitution for the auxiliary variable which contains a temporal structure of some kind.

6 Conclusion

We have proven that Generative Adversarial Networks (GANs) can be made identifiable and proposed the identifiable model iVAE-GAN. As validation we have implemented the model and compared it to state-of-the-art identifiable models on the same data. This is the first proof of identifiability in GAN and it does not impose constraints on the adversarial training. Therefore, the results will apply broadly to a variety of different GAN flavors. We found through experiments, that the training

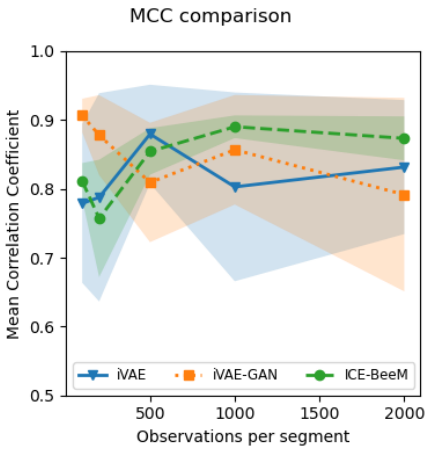


Figure 3: MCC comparison of iVAE, iVAE-GAN and ICE-BeeM with similar network parameters.

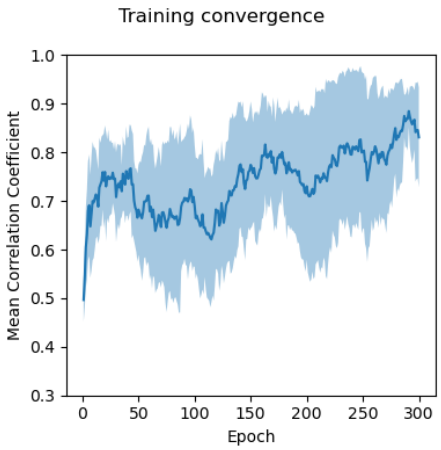


Figure 4: Statistics of the MCC convergence across seeds with standard deviation error bars included. (Epochs in thousands)

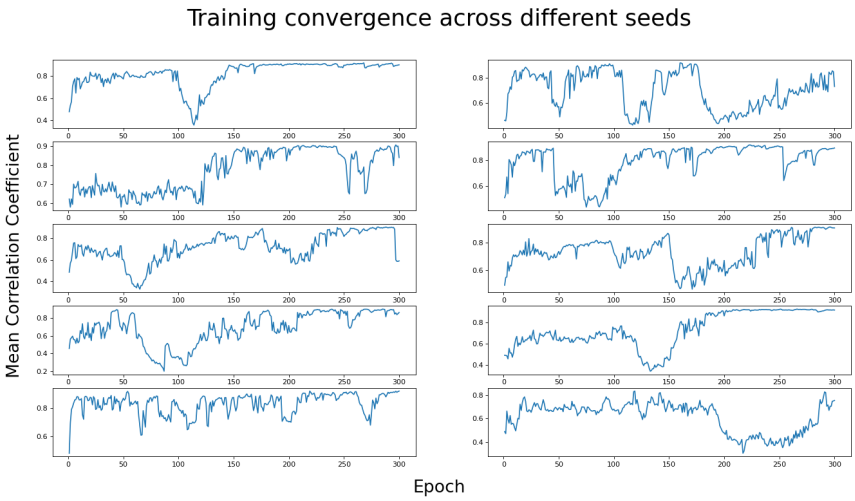


Figure 5: MCC scores vs epochs (in thousands) shown across ten different seeds.

dynamics of iVAE-GAN is prone to the same difficulties found in most GANs and is therefore a topic of interest for further work.

Societal Impact GAN has been associated with ethical concerns following the generation of images and speech that are virtually indistinguishable to the human eyes and ears. This work is not application specific and has therefore not produced any content that may be ethically concerning. Instead, we hypothesize that identifiability in GAN could be a remedying factor - not because it prohibits malicious applications, but because it forces the generation to be based on the true generating process. Therefore there should be nothing inherently evil about an identifiable model. Identifiable models could in fact be of interest when sensitive or safety critical data are used because the operator runs little to no risk of introducing unwanted bias in the model.

References

Ilyes Khemakhem, Ricardo Monti, Diederik Kingma, and Aapo Hyvarinen. ICE-BeeM: Identifiable Conditional Energy-Based Deep Models Based on Nonlinear ICA. *Advances in Neural Information Processing Systems*, 33, 2020a.

Ilyes Khemakhem, Diederik Kingma, Ricardo Monti, and Aapo Hyvarinen. Variational autoencoders and nonlinear ica: A unifying framework. In *International Conference on Artificial Intelligence and Statistics*, pages 2207–2217. PMLR, 2020b.

Philippe Le Gall. Trygve haavelmo, the probability approach in econometrics (1944). *Dictionnaire des Grandes Œuvres Economiques, Paris: Dalloz*, 2002.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pages 4114–4124. PMLR, 2019.

Aapo Hyvärinen and Petteri Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural networks*, 12(3):429–439, 1999a.

Aapo Hyvärinen and Petteri Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural networks*, 12(3):429–439, 1999b.

Peter Sorrenson, Carsten Rother, and Ullrich Köthe. Disentanglement by nonlinear ica with general incompressible-flow networks (gin). *arXiv preprint arXiv:2001.04872*, 2020.

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

Bharath Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Aapo Hyvärinen, and Revant Kumar. Density estimation in infinite dimensional exponential families. *Journal of Machine Learning Research*, 18, 2017.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *stat*, 1050:1, 2014.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Tim Salimans, Ian J Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, 2016.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5769–5779, 2017.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

Kun Xu, Chongxuan Li, Jun Zhu, and Bo Zhang. Understanding and stabilizing gans' training dynamics using control theory. In *International Conference on Machine Learning*, pages 10566–10575. PMLR, 2020.

Simon Jenni and Paolo Favaro. On stabilizing generative adversarial training with noise. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12137–12145, 2019. doi: 10.1109/CVPR.2019.01242.

A Derivation of global optimality in GAN

This derivation follows directly from Goodfellow et al. [2014] only stated more explicitly. GAN is formulated as a two-player minimax game according to:

$$\min_G \max_D V(G,D) = \min_G \max_D \; \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \tag{27}$$

To simplify the derivations that are to follow, the second term is rewritten using the *law of the unconscious statistician* such that:

$$\min_G \max_D V(G,D) = \min_G \max_D \; \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{x\sim p_g(x)}[\log(1-D(x))] \tag{28}$$

To express this in a way where the behaviour of the generator can be examined an optimal discriminator is assumed, such that the generator will try to minimize the following function:

$$C(G) = \max_D V(G,D) = \max_D \int_x p_{data}(x)\log(D(x)) + p_g(x)\log(1-D(x))\,\mathrm{d}x \tag{29}$$

This equation can be recognized as the function $f(y) = a\log(y) + b\log(1-y)$ which attains a maximum at $y = \frac{a}{a+b}$, which implies that the optimal discriminator is given by:

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \tag{30}$$

Thus, if $p_g = p_{data}$ the optimal discriminator will become $D^*_G(x) = \frac{1}{2}$ and by Equation (28) we can find the optimum value at which the generated data will be indistinguishable from the true data:

$$V^*(G,D) = \mathbb{E}_{x\sim p_{data}(x)}[\log(\frac{1}{2})] + \mathbb{E}_{x\sim p_g(x)}[\log(1-\frac{1}{2})] = -\log 4 \tag{31}$$

Now we need to verify that $-\log 4$ is indeed a minimum of $V(G,D^*)$:

$$\min_G V(G,D^*) = \min_G \int_x p_{data}(x)\log(\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}) + p_g(x)\log(1-\frac{p_{data}(x)}{p_{data}(x)+p_g(x)})\,\mathrm{d}x \tag{32}$$

The first term can be recognized as a Kullback-Leibler divergence $KL(p_{data}||p_{data}+p_g) = \int_x p_{data}(x)\log(\frac{p_{data}(x)}{p_{data}(x)+p_g(x)})\,\mathrm{d}x$ and the second term can also be rewritten to a KL divergence since:

$$1 - \frac{p_{data}(x)}{p_{data}(x)+p_g(x)} = \frac{p_{data}(x)+p_g(x)}{p_{data}(x)+p_g(x)} - \frac{p_{data}(x)}{p_{data}(x)+p_g(x)} = \frac{p_g(x)}{p_{data}(x)+p_g(x)} \tag{33}$$

Therefore Equation (32) can be written as:

$$\min_G V(G,D^*) = \min_G \; KL(p_{data}||p_{data}+p_g) + KL(p_g||p_{data}+p_g) \tag{34}$$

The last step is achieved by rewriting the equation such that the two KL divergences can be expressed as a Jensen-Shannon divergence between p_{data} and p_g by multiplying the equation with $\frac{\log(2)}{\log(2)}$ and distributing terms:

$$\min_G -\log(4) + KL(p_{data}||\frac{p_{data}+p_g}{2}) + KL(p_g||\frac{p_{data}+p_g}{2}) = \min_G -\log(4) + 2\cdot JSD(p_{data}||p_g) \tag{35}$$

Since the Jensen-Shannon divergence is always non-negative and attains a minimum only when $p_{data} = p_g$, it is concluded that $\min_G V(G,D^*) = V^*(D,G) = -\log(4)$ only when $p_{data} = p_g$. In other words, the global optimum of adversarial training, under the assumption of an optimal discriminator, occurs only when the generated data follows the same distribution as that of the observed data.

B Identifiability proof

All the proofs stated herein comes from Khemakhem et al. [2020b] only with a more explicit walk-through.

Given two sets of parameters $(\mathbf{f}, \mathbf{T}, \boldsymbol{\lambda})$ and $(\tilde{\mathbf{f}}, \tilde{\mathbf{T}}, \tilde{\boldsymbol{\lambda}})$ such that $p_{\mathbf{f},\mathbf{T},\boldsymbol{\lambda}}(x|u) = p_{\tilde{\mathbf{f}},\tilde{\mathbf{T}},\tilde{\boldsymbol{\lambda}}}(x|u)$ then the noise-free distributions, $\tilde{p}$ as they will be shown below, are also equal. The marginal distribution in the generative model can be written as:

$$\int_{\mathcal{Z}} p_{\mathbf{f}}(\mathbf{x}|\mathbf{z}) p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u}) \, \mathrm{d}\mathbf{z} = \int_{\mathcal{Z}} p_{\tilde{\mathbf{f}}}(\mathbf{x}|\mathbf{z}) p_{\tilde{\mathbf{T}},\tilde{\boldsymbol{\lambda}}}(\mathbf{z}|\mathbf{u}) \, \mathrm{d}\mathbf{z} \tag{36}$$

Substituting the decoder with the definition from Equation (6) yields:

$$\int_{\mathcal{Z}} p_{\boldsymbol{\epsilon}}(\mathbf{x} - \mathbf{f}(\mathbf{z})) p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u}) \, \mathrm{d}\mathbf{z} = \int_{\mathcal{Z}} p_{\boldsymbol{\epsilon}}(\mathbf{x} - \tilde{\mathbf{f}}(\mathbf{z})) p_{\tilde{\mathbf{T}},\tilde{\boldsymbol{\lambda}}}(\mathbf{z}|\mathbf{u}) \, \mathrm{d}\mathbf{z} \tag{37}$$

We now change the domain of the integral from $\mathcal{Z}$ to $\mathcal{X}$, by introducing $\bar{\mathbf{x}} = \mathbf{f}(\mathbf{z})$. We also introduce the notion of matrix volume denoted by $\mathrm{vol}\, A$, which acts as a replacement for the absolute determinant of the Jacobian introduced as a result of the change of variable:

$$\int_{\mathcal{X}} p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{f}^{-1}(\bar{\mathbf{x}})|\mathbf{u}) \, \mathrm{vol}\, J_{\mathbf{f}^{-1}}(\bar{\mathbf{x}}) p_{\epsilon}(\mathbf{x} - \bar{\mathbf{x}}) \, \mathrm{d}\bar{\mathbf{x}} = \int_{\mathcal{X}} p_{\mathbf{T},\boldsymbol{\lambda}}(\tilde{\mathbf{f}}^{-1}(\bar{\mathbf{x}})|\mathbf{u}) \, \mathrm{vol}\, J_{\tilde{\mathbf{f}}^{-1}}(\bar{\mathbf{x}}) p_{\epsilon}(\mathbf{x} - \bar{\mathbf{x}}) \, \mathrm{d}\bar{\mathbf{x}} \tag{38}$$

We now introduce the following shorthand:

$$\tilde{p}_{\mathbf{T},\boldsymbol{\lambda},\mathbf{f},\mathbf{u}}(\mathbf{x}) = p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{f}^{-1}(\mathbf{x})|\mathbf{u}) \, \mathrm{vol}\, J_{\mathbf{f}^{-1}}(\mathbf{x}) \mathbb{1}_{\mathcal{X}}(\mathbf{x}) \tag{39}$$

where $\mathbb{1}_{\mathcal{X}}$ is the indicator function, assuring that the expression has measure zero if x is not contained in the image of $\mathbf{f}$:

$$\int_{\mathbb{R}^d} \tilde{p}_{\mathbf{T},\boldsymbol{\lambda},\mathbf{f},\mathbf{u}}(\bar{\mathbf{x}}) p_{\epsilon}(\mathbf{x} - \bar{\mathbf{x}}) \, \mathrm{d}\bar{\mathbf{x}} = \int_{\mathbb{R}^d} \tilde{p}_{\tilde{\mathbf{T}},\tilde{\boldsymbol{\lambda}},\tilde{\mathbf{f}},\mathbf{u}}(\bar{\mathbf{x}}) p_{\epsilon}(\mathbf{x} - \bar{\mathbf{x}}) \, \mathrm{d}\bar{\mathbf{x}} \tag{40}$$

We recognize this to be the convolution between $\tilde{p}_{\mathbf{T},\boldsymbol{\lambda},\mathbf{f},\mathbf{u}}$ and p_{ϵ} as such:

$$(\tilde{p}_{\mathbf{T},\boldsymbol{\lambda},\mathbf{f},\mathbf{u}} * p_{\epsilon})(\mathbf{x}) = (\tilde{p}_{\tilde{\mathbf{T}},\tilde{\boldsymbol{\lambda}},\tilde{\mathbf{f}},\mathbf{u}} * p_{\epsilon})(\mathbf{x}) \tag{41}$$

Transforming the functions to the Fourier domain allows us to simplify the expression further:

$$F[\tilde{p}_{\mathbf{T},\boldsymbol{\lambda},\mathbf{f},\mathbf{u}}](\omega)\phi_{\epsilon}(\omega) = F[\tilde{p}_{\tilde{\mathbf{T}},\tilde{\boldsymbol{\lambda}},\tilde{\mathbf{f}},\mathbf{u}}](\omega)\phi_{\epsilon}(\omega) \tag{42}$$

Note here that we assume the characteristic function $\phi_{\epsilon}(\mathbf{x})$ to be non-zero for $\mathbf{x} \in \mathcal{X}$, which means it can be factored out yielding the final result, from which it is evident that:

$$F[\tilde{p}_{\mathbf{T},\boldsymbol{\lambda},\mathbf{f},\mathbf{u}}](\omega) = F[\tilde{p}_{\tilde{\mathbf{T}},\tilde{\boldsymbol{\lambda}},\tilde{\mathbf{f}},\mathbf{u}}](\omega) \tag{43}$$

$$\tilde{p}_{\mathbf{T},\boldsymbol{\lambda},\mathbf{f},\mathbf{u}}(\mathbf{x}) = \tilde{p}_{\tilde{\mathbf{T}},\tilde{\boldsymbol{\lambda}},\tilde{\mathbf{f}},\mathbf{u}}(\mathbf{x}) \tag{44}$$

Therefore the noise-free distributions has to be the same. In the following we wish to examine the relationship between the true parameters $(\mathbf{T}, \boldsymbol{\lambda}, \mathbf{f})$ and the estimated parameters $(\tilde{\mathbf{T}}, \tilde{\boldsymbol{\lambda}}, \tilde{\mathbf{f}})$ given that our model learns to accurately approximate the true data distribution $\tilde{p}_{\mathbf{T},\boldsymbol{\lambda},\mathbf{f},\mathbf{u}}(\mathbf{x})$. First we use Equation (39) to write the expression for the marginal distribution:

$$\tilde{p}_{\mathbf{T},\boldsymbol{\lambda},\mathbf{f},\mathbf{u}}(\mathbf{x}) = p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{f}^{-1}(\mathbf{x})|\mathbf{u}) \, \mathrm{vol}\, J_{\mathbf{f}^{-1}}(\mathbf{x}) \mathbb{1}_{\mathcal{X}}(\mathbf{x}) = p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u}) \, \mathrm{vol}\, J_{\mathbf{f}^{-1}}(\mathbf{x}) \mathbb{1}_{\mathcal{X}}(\mathbf{x}) \tag{45}$$

Since $\mathbf{f}^{-1}(\mathbf{x}) = \mathbf{z}$ by definition. By inserting the expression for the prior distribution given an auxiliary variable, u: $p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u}) = \prod_i^n \frac{Q_i(z_i)}{Z_i(\mathbf{u})} \exp\left[\sum_{j=1}^k T_{i,j}(z_i)\lambda_{i,j}(\mathbf{u})\right]$ from Equation (5), we can write the marginal distribution over x as:

$$\tilde{p}_{\mathbf{T},\boldsymbol{\lambda},\mathbf{f},\mathbf{u}}(\mathbf{x}) = \prod_i^n \frac{Q_i(z_i)}{Z_i(\mathbf{u})} \exp\left[\sum_{j=1}^k T_{i,j}(z_i)\lambda_{i,j}(\mathbf{u})\right] \text{ vol } J_{\mathbf{f}^{-1}}(\mathbf{x}) \mathbb{1}_{\mathcal{X}}(\mathbf{x}) \tag{46}$$

Here we can safely drop the indicator function, $\mathbb{1}_{\mathcal{X}}(\mathbf{x})$, as the expression no longer contains integration and we will write $\mathbf{z} = \mathbf{f}^{-1}(\mathbf{x})$ again to emphasize that latent variables are inferred from data. For simplicity we shall work with the log pdf since it greatly simplifies the exponential term:

$$\log \tilde{p}_{\mathbf{T},\boldsymbol{\lambda},\mathbf{f},\mathbf{u}}(\mathbf{x}) = \sum_i^n (\log Q_i(\mathbf{f}_i^{-1}(\mathbf{x})) - \log Z_i(\mathbf{u}) + \sum_{j=1}^k (T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u}))) + \log \text{ vol } J_{\mathbf{f}^{-1}}(\mathbf{x}) \tag{47}$$

Thus we can rewrite Equation (44) and investigate the relation between true and estimated parameters:

$$\begin{aligned}
&\sum_i^n (\log Q_i(\mathbf{f}_i^{-1}(\mathbf{x})) - \log Z_i(\mathbf{u}) + \sum_{j=1}^k (T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u}))) + \log \text{ vol } J_{\mathbf{f}^{-1}}(\mathbf{x}) \\
&\quad = \sum_i^n (\log \tilde{Q}_i(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x})) - \log \tilde{Z}_i(\mathbf{u}) + \sum_{j=1}^k (\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u}))) + \log \text{ vol } J_{\tilde{\mathbf{f}}^{-1}}(\mathbf{x})
\end{aligned} \tag{48}$$

Each side of the equation contain nk unknown parameters in $T_{i,j}$ and $\tilde{T}_{i,j}$ respectively, since they are summed over n latent variables and k sufficient parameters per latent variable. Each side of the equation also has nk unknown parameters in $\lambda_{i,j}$ and $\tilde{\lambda}_{i,j}$. Therefore a system of equations is created for $nk+1$ different points $u^{(0)}, ..., u^{(nk)}$. In time series data divided into segments this step may intuitively be thought of as calculating the probability of a seeing a given sample in each of nk segments. It can also be seen as a consequence of Equation (44) where we have equality between the marginal distribution for all choices of u. Thus we get the following system of equations:

$$\begin{aligned}
&\sum_i^n (\log Q_i(\mathbf{f}_i^{-1}(\mathbf{x})) - \log Z_i(\mathbf{u_0}) + \sum_{j=1}^k (T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u_0}))) + \log \text{ vol } J_{\mathbf{f}^{-1}}(\mathbf{x}) \\
&\quad = \sum_i^n (\log \tilde{Q}_i(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x})) - \log \tilde{Z}_i(\mathbf{u_0}) + \sum_{j=1}^k (\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u_0}))) + \log \text{ vol } J_{\tilde{\mathbf{f}}^{-1}}(\mathbf{x})
\end{aligned} \tag{49}$$

$$\begin{aligned}
&\sum_i^n (\log Q_i(\mathbf{f}_i^{-1}(\mathbf{x})) - \log Z_i(\mathbf{u_1}) + \sum_{j=1}^k (T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u_1}))) + \log \text{ vol } J_{\mathbf{f}^{-1}}(\mathbf{x}) \\
&\quad = \sum_i^n (\log \tilde{Q}_i(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x})) - \log \tilde{Z}_i(\mathbf{u_1}) + \sum_{j=1}^k (\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u_1}))) + \log \text{ vol } J_{\tilde{\mathbf{f}}^{-1}}(\mathbf{x})
\end{aligned} \tag{50}$$

$$\vdots$$

$$\begin{aligned}
&\sum_i^n (\log Q_i(\mathbf{f}_i^{-1}(\mathbf{x})) - \log Z_i(\mathbf{u_{nk}}) + \sum_{j=1}^k (T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u_{nk}}))) + \log \text{ vol } J_{\mathbf{f}^{-1}}(\mathbf{x}) \\
&= \sum_i^n (\log \tilde{Q}_i(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x})) - \log \tilde{Z}_i(\mathbf{u_{nk}}) + \sum_{j=1}^k (\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u}_{nk}))) + \log \text{ vol } J_{\tilde{\mathbf{f}}^{-1}}(\mathbf{x})
\end{aligned} \tag{51}$$

A neat trick is used to simplify this system of equations by using any of the $nk+1$ equations as pivot, we shall simply use $\mathbf{u_0}$. By pivot it is understood that we consider a ratio of pdfs or in this case, since we are dealing with logarithms, a difference of log pdfs. This is the motivation for using $nk+1$ points in our system of equations as we use one equation to pivot such that we end up with a system of nk equations. The consequence of this choice is that our equations no longer express how likely a sample is with a given $\mathbf{u_l}$ but rather how likely it is compared to $\mathbf{u_0}$, but this is of little importance as we are interested in the relation between parameters of the models and not the exact likelihood of seen samples. Therefore the system of equations becomes:

$$0 = 0 \tag{52}$$

$$\begin{aligned}
&\sum_i^n (\log Q_i(\mathbf{f}_i^{-1}(\mathbf{x})) - \log Z_i(\mathbf{u_1}) + \sum_{j=1}^k (T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u_1}))) + \log \text{ vol } J_{\mathbf{f}^{-1}}(\mathbf{x}) \\
&- (\sum_i^n (\log Q_i(\mathbf{f}_i^{-1}(\mathbf{x})) - \log Z_i(\mathbf{u_0}) + \sum_{j=1}^k (T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u_0}))) + \log \text{ vol } J_{\mathbf{f}^{-1}}(\mathbf{x})) = \\
&\quad \sum_i^n (\log \tilde{Q}_i(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x})) - \log \tilde{Z}_i(\mathbf{u_1}) + \sum_{j=1}^k (\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u_1}))) + \log \text{ vol } J_{\tilde{\mathbf{f}}^{-1}}(\mathbf{x}) \\
&- (\sum_i^n (\log \tilde{Q}_i(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x})) - \log \tilde{Z}_i(\mathbf{u_0}) + \sum_{j=1}^k (\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u_0}))) + \log \text{ vol } J_{\tilde{\mathbf{f}}^{-1}}(\mathbf{x}))
\end{aligned} \tag{53}$$

$$\vdots$$

$$\begin{aligned}
&\sum_i^n (\log Q_i(\mathbf{f}_i^{-1}(\mathbf{x})) - \log Z_i(\mathbf{u_{nk}}) + \sum_{j=1}^k (T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u_{nk}}))) + \log \text{ vol } J_{\mathbf{f}^{-1}}(\mathbf{x}) \\
&\quad - (\sum_i^n (\log Q_i(\mathbf{f}_i^{-1}(\mathbf{x})) - \log Z_i(\mathbf{u_0}) + \sum_{j=1}^k (T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u_0}))) + \log \text{ vol } J_{\mathbf{f}^{-1}}(\mathbf{x})) \\
&\quad = \sum_i^n (\log \tilde{Q}_i(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x})) - \log \tilde{Z}_i(\mathbf{u_{nk}}) + \sum_{j=1}^k (\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u}_{nk}))) + \log \text{ vol } J_{\tilde{\mathbf{f}}^{-1}}(\mathbf{x}) \\
&- (\sum_i^n (\log \tilde{Q}_i(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x})) - \log \tilde{Z}_i(\mathbf{u_0}) + \sum_{j=1}^k (\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u_0}))) + \log \text{ vol } J_{\tilde{\mathbf{f}}^{-1}}(\mathbf{x}))
\end{aligned} \tag{54}$$

By eliminating terms we get rid of $\log \tilde{Q}_i(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x})$ and interestingly $\log \text{ vol } J_{\tilde{\mathbf{f}}^{-1}}(\mathbf{x})$ which is typically notoriously difficult to evaluate, therefore these equations can be reduced to:

$$0 = 0 \tag{55}$$

$$\begin{aligned}
&\sum_i^n (-\log Z_i(\mathbf{u_1}) + \sum_{j=1}^k (T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u_1}))) - (\sum_i^n (-\log Z_i(\mathbf{u_0}) + \sum_{j=1}^k (T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u_0}))) \\
&= \sum_i^n (-\log \tilde{Z}_i(\mathbf{u_1}) + \sum_{j=1}^k (\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u_1}))) - (\sum_i^n (-\log \tilde{Z}_i(\mathbf{u_0}) + \sum_{j=1}^k (\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u_0})))
\end{aligned} \tag{56}$$

$$\vdots$$

$$\sum_{i}^{n}(-\log Z_i(\mathbf{u_{nk}})+\sum_{j=1}^{k}(T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u_{nk}})))-(\sum_{i}^{n}(-\log Z_i(\mathbf{u_0})+\sum_{j=1}^{k}(T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))\lambda_{i,j}(\mathbf{u_0})))$$
$$=\sum_{i}^{n}(-\log \tilde{Z}_i(\mathbf{u_{nk}})+\sum_{j=1}^{k}(\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(u_{nk})))-(\sum_{i}^{n}(-\log \tilde{Z}_i(\mathbf{u_0})+\sum_{j=1}^{k}(\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))\tilde{\lambda}_{i,j}(\mathbf{u_0}))) \tag{57}$$

By factoring terms and distributing sums for an arbitrary point $u^{(l)}$ we get:

$$\sum_{i}^{n}(\sum_{j=1}^{k}(T_{i,j}(\mathbf{f}_i^{-1}(\mathbf{x}))(\lambda_{i,j}(\mathbf{u_l})-\lambda_{i,j}(\mathbf{u_0})))+\sum_{i}^{n}\log\frac{Z_i(\mathbf{u_0})}{Z_i(\mathbf{u_l})}$$
$$=\sum_{i}^{n}(\sum_{j=1}^{k}(\tilde{T}_{i,j}(\tilde{\mathbf{f}}_i^{-1}(\mathbf{x}))(\tilde{\lambda}_{i,j}(\mathbf{u_l})-\tilde{\lambda}_{i,j}(\mathbf{u_0})))+\sum_{i}^{n}\log\frac{\tilde{Z}_i(\mathbf{u_0})}{\tilde{Z}_i(\mathbf{u_l})} \tag{58}$$

$T_{i,j}$ and $\lambda_{i,j}$ are elements from the tall vectors $\mathbf{T}$ and $\boldsymbol{\lambda}$ therefore the first term can be recognized as the inner product between $\mathbf{T(f^{-1}(x))}$ and $\bar{\boldsymbol{\lambda}}(\mathbf{u_l})$ where $\bar{\boldsymbol{\lambda}}(\mathbf{u_l})$ is defined as:

$$\bar{\boldsymbol{\lambda}}(\mathbf{u_l})=\boldsymbol{\lambda}(\mathbf{u_l})-\boldsymbol{\lambda}(\mathbf{u_0}) \tag{59}$$

Therefore Equation (58) can be written as:

$$\left\langle \mathbf{T(f^{-1}(x))},\bar{\boldsymbol{\lambda}}(\mathbf{u_l})\right\rangle=\left\langle \tilde{\mathbf{T}}(\tilde{\mathbf{f}}^{-1}(\mathbf{x})),\bar{\tilde{\boldsymbol{\lambda}}}(\mathbf{u_l})\right\rangle+b_l \quad where \quad b_l=\sum_{i}^{n}\log\frac{\tilde{Z}_i(\mathbf{u_0})Z_i(\mathbf{u_l})}{\tilde{Z}_i(\mathbf{u_l})Z_i(\mathbf{u_0})} \tag{60}$$

Across all nk equations $\mathbf{T(f^{-1}(x))}$ will be the same, therefore we can collect all the equations in a single matrix product by defining the $nk\times nk$ matrix $L=\left[\bar{\boldsymbol{\lambda}}(\mathbf{u_1}),\bar{\boldsymbol{\lambda}}(\mathbf{u_2})\dots\bar{\boldsymbol{\lambda}}(\mathbf{u_{nk}})\right]$ and $\mathbf{b}=[b_1,b_2,\dots,b_{nk}]$such that:

$$L^T\mathbf{T(f^{-1}(x))}=\tilde{L}^T\tilde{\mathbf{T}}(\tilde{\mathbf{f}}^{-1}(\mathbf{x}))+\mathbf{b} \tag{61}$$

In the final step we assume that the true matrix of natural parameters, $\mathbf{L}$, is invertible to obtain the following result:

$$\mathbf{T(f^{-1}(x))}=A\tilde{\mathbf{T}}(\tilde{\mathbf{f}}^{-1}(\mathbf{x}))+\mathbf{c} \tag{62}$$

Where $A=L^{T^{-1}}\tilde{L}^T$ and $\mathbf{c}=L^{T^{-1}}\mathbf{b}$. Thus, we see that the true latent variables are linear transformation of the recovered latent variables. The last step is to prove an equivalence relation such that the opposite is also true. That the recovered latents are also a linear transformation of the true latent variables. To do so it is assumed that the Jacobian of $\mathbf{T}$ exists has full rank n. Therefore:

$$J_{\mathbf{T}}(x)=AJ_{\tilde{\mathbf{T}}\circ\tilde{\mathbf{f}}^{-1}}(x) \tag{63}$$

By using the following inequality for the rank of a matrix multiplication we may deduce that the rank of both A and $J_{\tilde{\mathbf{T}}\circ\tilde{\mathbf{f}}^{-1}}$ is at least n:

$$Rank(AB)\leq min(Rank(A),Rank(B)) \tag{64}$$

Since $J_{\tilde{\mathbf{T}}\circ\tilde{\mathbf{f}}^{-1}}$ is a $nk\times n$ matrix we can conclude that it exists and has full rank. And if $k=1$ then A will be a square $n\times n$ matrix with full rank and thus invertible, such that Equation (62) can be shown to be true in both directions.

For $k > 1$ the matrix A must be invertible in order to establish the equivalence relation. In the following we show how A is invertible under the assumption that each latent variable follows a *strongly exponential distribution*. A strongly exponential distribution is one that almost certainly contains the exponent and thus can not be reduced to the base measure. Formally:

$$(\exists \boldsymbol{\theta} \in \mathbb{R}^k \mid \forall x \in \mathcal{X}, \langle \mathbf{T}(\mathbf{x}), \boldsymbol{\theta} \rangle = const) \quad \Rightarrow \quad (l(\mathcal{X}) = 0 \;\; or \;\; \boldsymbol{\theta} = \mathbf{0}) \tag{65}$$

Which means that the exponent of a strongly exponential distribution only reduces to a constant if $\boldsymbol{\theta} = 0$ which means the inner product becomes zero, $\langle \mathbf{T}(\mathbf{x}), \mathbf{0} \rangle = 0$, or if the set $\mathcal{X}$ has Lebesgue measure 0. The following three Lemmas is used to derive useful properties for the derivate of the sufficient statistic, $\mathbf{T}'(x)$, from a strongly exponential distribution that is of relevance for the Jacobian matrix. The dimension, k, of all considered distributions is assumed minimal. That is, the distributions can not be rewritten with a $k' < k$.

Lemma 1 Consider an exponential family distribution with $k \geq 2$ components. [...], the components of the sufficient statistic $\mathbf{T}$ are linearly independent.

If the components of $\mathbf{T}$ are not linearly independent then one of the components, $T_k(x)$, could be written as a combination of the remaining components for an $\mathbf{a} \neq \mathbf{0}$.

$$T_k(x) = \sum_{i}^{k-1} a_i T_i(x) \tag{66}$$

If that was possible, we would have contradicted the assumption that the dimension of the distribution, k, is minimal.

Lemma 2 Consider a strongly exponential family distribution such that its sufficient statistic T is differentiable almost surely. Then $T_i' \neq 0$ almost everywhere on $\mathbb{R}$ for all $1 \leq$ i $\leq$ k

We provide an alternate proof than the original, simply because we used this alternate proof to verify our understanding of the original proof. If we consider an exponential distribution that is *not* strongly exponential then we necessarily have:

$$\langle \mathbf{T}(\mathbf{x}), \boldsymbol{\theta} \rangle = T_1(x)\theta_1 + T_2(x)\theta_1 + \cdots + T_k\theta_k = const \tag{67}$$

The derivative would then become:

$$\frac{\mathrm{d}}{\mathrm{d}x} \langle \mathbf{T}(\mathbf{x}), \boldsymbol{\theta} \rangle = \langle \mathbf{T}(\mathbf{x})', \boldsymbol{\theta} \rangle = T_1'(x)\theta_1 + T_2'(x)\theta_2 + \cdots + T_k'(x)\theta_k = 0 \tag{68}$$

Thus for an exponential distribution that is not strongly exponential the derivative of the exponent must be equal to zero. Which can be achieved in many different ways having either $\boldsymbol{\theta} = \mathbf{0}$, $\mathbf{T}(x) = \mathbf{0}$ or their weighted sum equal to zero. For a strongly exponential distribution the exponent can only equal a constant if $\boldsymbol{\theta} = \mathbf{0}$ (see Equation (65)) and therefore the derivative can also only be 0 if $\boldsymbol{\theta} = \mathbf{0}$. From Lemma 1 we have that the components of the sufficient statistic can not be written as a function of each other. Therefore it can be seen that $\mathbf{T}'(x) \neq \mathbf{0}$ and even $T_i'(x) \neq 0$. Because, if any $T_i'(x)$ was equal to zero the corresponding θ_i could be an arbitrary number different from zero while the rest of $\boldsymbol{\theta}$ is zero and thus $\boldsymbol{\theta} \neq \mathbf{0}$ but the derivative would equal zero. which violates the statement that the distribution is strongly exponential. Thus, we may conclude that for a strongly exponential distribution $T_i'(x)$ must be different from zero.

Lemma 3 Consider a strongly exponential distribution of size $k \geq 2$ with sufficient statistic $\mathbf{T}(x) = (T_1(x), ..., T_k(x))$. Further assume that $\mathbf{T}$ is differentiable almost everywhere. Then there exist k distinct values x_1 to x_k such that $(\mathbf{T}'(x_1), ..., \mathbf{T}'(x_k))$ are linearly independent in $\mathbb{R}^k$

Recall that in order for the distribution to be strongly exponential then the only choice of parameter that can lead to the exponent being constant for all x is $\boldsymbol{\theta} = \mathbf{0}$. Since both $\mathbf{T}'(\mathbf{x})$ and $\boldsymbol{\theta}$ is in $\mathbb{R}^k$ this necessarily means that $\mathbf{T}'(\mathbf{x})$ must be able to span the full $\mathbb{R}^k$. That is, there exists at least k vectors

of $\mathbf{T'(x)}$ in k points $x_1, \ldots, x_k$ such that the matrix $B = [\mathbf{T}'(x_1)\ \mathbf{T}'(x_2)\ \ldots\ \mathbf{T}'(x_k)]$ has full rank:

$$Rank(B) = Rank(\begin{bmatrix}\mathbf{T}'(x_1) & \mathbf{T}'(x_2) & \ldots & \mathbf{T}'(x_k)\end{bmatrix}) = k \tag{69}$$

If $Rank(B) \neq k$ then the nullity of A will be greater than 1 and thus any vector from the orthogonal complement of the column space of B can be picked as $\boldsymbol{\theta}^*$ such that $\boldsymbol{\theta}^* \neq \mathbf{0}$ and $\langle \mathbf{T(x)'}, \boldsymbol{\theta}^* \rangle = 0$ for all x. However, if that is the case then the distribution is not strongly exponential as seen from Lemma 2 that shows that only a distribution which is *not* strongly exponential will have $\langle \mathbf{T(x)'}, \boldsymbol{\theta} \rangle = 0$ for $\boldsymbol{\theta} \neq \mathbf{0}$. Therefore, in a strongly exponential distribution there must exist k points, $x_1, \ldots, x_k$ such that the column vectors of A are linearly independent.

These three Lemmas have been used to derive the important property that in univariate exponential distributions which are minimal in k and strongly exponential there exist at least k points, $x_1, \ldots, x_k$ such that the vectors $\mathbf{T}'(x_1), \ldots, \mathbf{T}'(x_k)$ are linearly independent. We can now use this to show that the $nk \times nk$ matrix A in Equation (70) is invertible under the assumption that the $nk \times n$ Jacobian matrix of $\mathbf{T}(\mathbf{f}^{-1}(x))$, $J_{\mathbf{T}}(\mathbf{f}^{-1}(x))$, exists and is of rank n:

$$\mathbf{T(f^{-1}(x))} = A\tilde{\mathbf{T}}(\tilde{\mathbf{f}}^{-1}(\mathbf{x})) + \mathbf{c} \tag{70}$$

To make the proof easier to follow we examine the form the Jacobian matrix will have. First we write the expression for the Jacobian matrix of $\mathbf{T}(\mathbf{f}^{-1}(x))$ (remembering that $\mathbf{f}^{-1}$ is a function that maps x to $\mathbb{R}^n$, such that $\mathbf{T}$ is a function of n (latent) variables, $f_1^{-1}(x), \ldots, f_n^{-1}(x)$:

$$J_{\mathbf{T}}(\mathbf{f}^{-1}(x)) = \frac{\mathrm{d}\mathbf{T}(\mathbf{f}^{-1}(x))}{\mathrm{d}(f_1(x), f_2(x), \ldots, f_n(x))} = \begin{bmatrix} \frac{\mathrm{d}\mathbf{T}(\mathbf{f}^{-1}(x))}{\mathrm{d}f_1(x)} & \frac{\mathrm{d}\mathbf{T}(\mathbf{f}^{-1}(x))}{\mathrm{d}f_2(x)} & \ldots & \frac{\mathrm{d}\mathbf{T}(\mathbf{f}^{-1}(x))}{\mathrm{d}f_n(x)} \end{bmatrix} \tag{71}$$

$$= \begin{bmatrix} \frac{\mathrm{d}T_{1,1}(f_1^{-1}(x))}{df_1^{-1}(x)} & \frac{\mathrm{d}T_{1,1}(f_1^{-1}(x))}{df_2^{-1}(x)} & \cdots & \frac{\mathrm{d}T_{1,1}(f_1^{-1}(x))}{df_n^{-1}(x)} \\ \vdots & \vdots & & \vdots \\ \frac{\mathrm{d}T_{1,k}(f_1^{-1}(x))}{df_1^{-1}(x)} & \frac{\mathrm{d}T_{1,k}(f_1^{-1}(x))}{df_2^{-1}(x)} & \cdots & \frac{\mathrm{d}T_{1,k}(f_1^{-1}(x))}{df_n^{-1}(x)} \\ \frac{\mathrm{d}T_{2,1}(f_2^{-1}(x))}{df_1^{-1}(x)} & \frac{\mathrm{d}T_{2,1}(f_2^{-1}(x))}{df_2^{-1}(x)} & \cdots & \frac{\mathrm{d}T_{2,1}(f_2^{-1}(x))}{df_n^{-1}(x)} \\ \vdots & \vdots & & \vdots \\ \frac{\mathrm{d}T_{2,k}(f_2^{-1}(x))}{df_1^{-1}(x)} & \frac{\mathrm{d}T_{2,k}(f_2^{-1}(x))}{df_2^{-1}(x)} & \cdots & \frac{\mathrm{d}T_{2,k}(f_2^{-1}(x))}{df_n^{-1}(x)} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \frac{\mathrm{d}T_{n,1}(f_n^{-1}(x))}{df_1^{-1}(x)} & \frac{\mathrm{d}T_{n,1}(f_n^{-1}(x))}{df_2^{-1}(x)} & \cdots & \frac{\mathrm{d}T_{n,1}(f_n^{-1}(x))}{df_n^{-1}(x)} \\ \vdots & \vdots & & \vdots \\ \frac{\mathrm{d}T_{n,k}(f_n^{-1}(x))}{df_1^{-1}(x)} & \frac{\mathrm{d}T_{n,k}(f_n^{-1}(x))}{df_2^{-1}(x)} & \cdots & \frac{\mathrm{d}T_{n,k}(f_n^{-1}(x))}{df_n^{-1}(x)} \end{bmatrix}^{nk \times n} \tag{72}$$

We can notice that this matrix will have a particular shape since many of the entries will become zero as they are not a function of the variable with which the derivative is taken. Therefore the Jacobian matrix will have the shape:

$$J_{\mathbf{T}}(\mathbf{f}^{-1}(x)) = \begin{bmatrix} T'_{1,1}(f_1^{-1}(x)) & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ T'_{1,k}(f^{-1}(x)) & 0 & \dots & 0 \\ 0 & T'_{2,1}(f_2^{-1}(x)) & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & T'_{2,k}(f_2^{-1}(x)) & \dots & 0 \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & T'_{n,1}(f_n^{-1}(x)) \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & T'_{n,k}(f_n^{-1}(x)) \end{bmatrix} \tag{73}$$

The expression we used to derive invertibility of A for $k = 1$ was found in Equation (63) as seen below:

$$J_{\mathbf{T}}(x) = A J_{\tilde{\mathbf{T}} \circ \tilde{\mathbf{f}}^{-1}}(x) \tag{74}$$

For $k = 1$ we used the fact that $J_{\mathbf{T}}(x)$ becomes an $n \times n$ matrix of rank n to prove A is invertible. The same approach is used now, but since for $k > 1$ $J_{\mathbf{T}}(x)$ is an $nk \times n$ matrix which is not square and thus not invertible we use that the above expression is true for all x. In particular, we choose k points $x_1, \dots, x_k$ according to Lemma 3. The k points serve two purposes. First, k Jacobians are needed to have enough entries to fill an $nk \times nk$ square matrix of Jacobians and secondly by choosing the points according to Lemma 3 invertibility can be proved. By concatenating the Jacobian matrices evaluated at the k points the matrix Q can be formed:

$$Q = \begin{bmatrix} & | & & | & \\ J_{\mathbf{T}}(x_1) & | & \dots & | & J_{\mathbf{T}}(x_k) \\ & | & & | & \end{bmatrix} \tag{75}$$

And similarly for $J_{\tilde{\mathbf{T}} \circ \tilde{\mathbf{f}}^{-1}}(x)$:

$$\tilde{Q} = \begin{bmatrix} & | & & | & \\ J_{\tilde{\mathbf{T}} \circ \tilde{\mathbf{f}}^{-1}}(x_1) & | & \dots & | & J_{\tilde{\mathbf{T}} \circ \tilde{\mathbf{f}}^{-1}}(x_k) \\ & | & & | & \end{bmatrix} \tag{76}$$

Thus for $k \geq 1$ we may write:

$$Q = A\tilde{Q} \tag{77}$$

To verify that the concatenated system can indeed be written as in Equation (77) we can recognize the expression as block matrix multiplication where A is a matrix with $q = 1$ row partition and $s = 1$ column partition and $\tilde{Q}$ a block matrix with $s = 1$ row partition and $r = k$ column partitions. Thus, the partitions of A and $\tilde{Q}$ are conformable and the resulting matrix, Q, will have 1 row partition and k column partitions, as in the definition above. We may also confirm that the new matrix multiplication maps every column partition i in $\tilde{Q}$ to column partition i in Q as it should be from Equation (74). Block matrix multiplication is defined as:

$$Q_{qr} = \sum_{i=1}^{s} A_{qi}\tilde{Q}_{ir} = A_{11}\tilde{Q}_{1r} = A\tilde{Q}_{1r} = Q_{1r} \tag{78}$$

Therefore each column partition of $\tilde{Q}$ is mapped to the same column partition in Q by A. Every column partition in Q has the form of Equation (73) and by rearranging the columns of Q such that all the nonzero elements are grouped it can be seen that Q can be written as a block diagonal matrix:

$$Q = \begin{bmatrix} B_{f_1^{-1}} & & & 0 \\ & B_{f_2^{-1}} & & \\ & & \ddots & \\ 0 & & & B_{f_n^{-1}} \end{bmatrix} \tag{79}$$

Where B is defined as in Lemma 3, $B = [\mathbf{T}'(x_1) \ \mathbf{T}'(x_2) \ \dots \ \mathbf{T}'(x_k)]$, and the subscript f_i^{-1} is used to emphasize $\mathbf{T}(\mathbf{f^{-1}(x)})$ is a function of n variables, $f_1^{-1}(x), \dots, f_n^{-1}(x)$, such that $B_{f_i^{-1}}$ is the $k \times k$ matrix containing the nonzero derivatives with respect to f_i^{-1} as seen in Equation (73) for all k points $x_1, \dots, x_k$. If Q is invertible that would imply the invertibility of both A and $\tilde{Q}$ as it can be seen from Equation (77) which is what we would like to prove. A block diagonal matrix is invertible if all the diagonal matrices are invertible. That is:

$$Q^{-1} = \begin{bmatrix} B_{f_1^{-1}} & & & 0 \\ & B_{f_2^{-1}} & & \\ & & \ddots & \\ 0 & & & B_{f_n^{-1}} \end{bmatrix}^{-1} = \begin{bmatrix} B_{f_1^{-1}}^{-1} & & & 0 \\ & B_{f_2^{-1}}^{-1} & & \\ & & \ddots & \\ 0 & & & B_{f_n^{-1}}^{-1} \end{bmatrix} \tag{80}$$

Since the points $x_1, \dots, x_k$ are chosen as in Lemma 3 every diagonal matrix of Q is exactly identical to B in Lemma 3 (one for each of the n latent variables). Therefore every diagonal matrix of Q is invertible because B has full rank, as proven in Lemma 3, and thus Q is also invertible. Since Q is invertible we can write:

$$Q^{-1} = (A\tilde{Q})^{-1} = \tilde{Q}^{-1}A^{-1} \tag{81}$$

Which means that both A and $\tilde{Q}$ are invertible. Since A is invertible we have proven the equivalence relation for $k \geq 1$.

C Experiment details

In this section we will provide the experimental setup used to achieve the reported results. Table 1 shows the size of the hidden dimensions of the networks, except for ICE-BeeM where we have left all parameters default, there we simply state the two parameters of interest. Hidden dimensions are fully connected MLPs with Leaky ReLU activation functions.

Table 1: Network sizes

Model	Encoder	Decoder	Auxiliary	Discriminator
iVAE-GAN	{4, 4, 4}	{4, 4, 4}	{4, 4, 4}	$\left\{\begin{matrix} 32, & 32, & 32, & 32, & 32, & 32 \\ 32, & 32, & 32, & 32, & 32, & 32 \\ 256, & 256, & 256, & 256, & 256, & 256 \\ 1024, & 1024, & 1024, & 1024, & 1024, & 1024 \\ 1024, & 1024, & 1024, & 1024, & 1024, & 1024 \end{matrix}\right\}$
iVAE	{4, 4, 4}	{4, 4, 4}	{4, 4, 4}	-
ICE-BeeM				

Model	n_layers_flow	ebm_hidden_size	Comment
ICE-BeeM	10	32	Default parameters

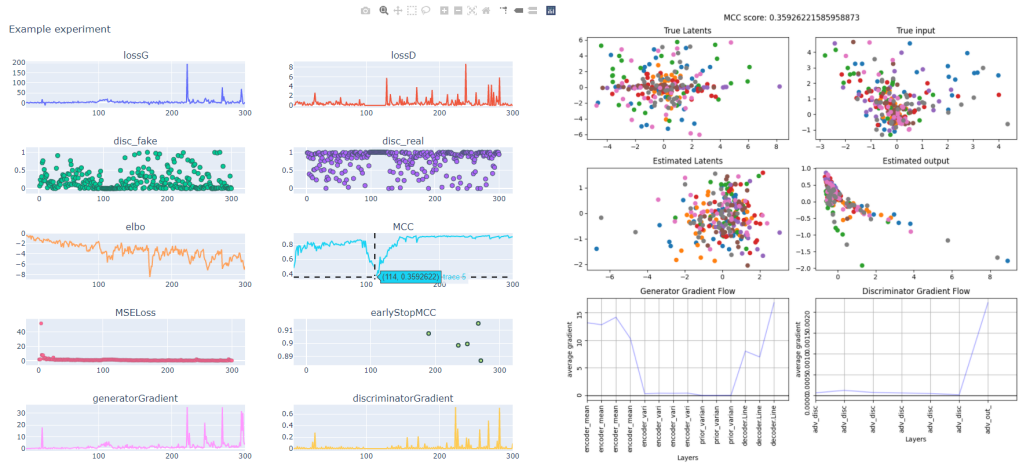


Figure 6: Interactive plots with updating images on hover

The reason multiple values are stated for the discriminator is because different widths were required for 100, 200, 500, 1000 and 2000 number of observations per segment. We believe this to be a result of more observations per segment means the discriminator has to discriminate based on more points which in turn requires a larger network in order to perform well. We also think this is consistent with common findings where more data points result in more complex GAN networks, e.g. it is harder to get reasonable results with higher resolution images than with lower resolution images.

Optimizer Two optimizers were used. One for the discriminator and one for the remaining three networks. The optimizers were both Adam optimizers with the same parameters: learning rate of 0.001 and $\beta = (0.5, 0.999)$.

Training All iVAE-GAN models were trained for 300000 iterations across 10 different seeds while iVAE and ICE-BeeM models were trained for 70000 iterations across 10 different seeds. iVAE used the default batch size of 256, ICE-BeeM also used the default batch size of 128. For this particular experiment of iVAE-GAN the batch size was simply set to the size of the dataset because the entire dataset could reside in the memory of the Tesla V100 32GB GPU made available by the university. To monitor training, we logged metrics of interest such as loss, MCC, percentage of how many generated images were classified as real etc... every n iterations. In addition, images of the generated output were saved along with plots of the magnitude of the gradients at each hidden dimension. To visualize and draw meaningful conclusion from all the logged data we created an interactive plot that simultaneously show interactive graphs of the logged data, meaning all graphs can be zoomed, dragged, scaled etc. and individual points can be inspected at cursor hover as well as the saved images. Since all datapoints are associated with an iteration and all images are also associated with an iteration whenever the mouse hovers a datapoint the plot is updated to show the images and gradients of that particular iteration as it can be seen in Figure 6 and 7.

Hyperparameter tuning Adversarial training is known to require extensive hyperparameter tuning since there needs to be a balance between the generator and discriminator. If either is too complex or simple the training will collapse. Our approach was to use the same decoder size as the iVAE model because it would allow direct comparison of the two models and it was clear that the iVAE decoder was sufficiently complex to represent the data faithfully. In this way the hyperparameter tuning was simplified to finding an adequate discriminator. By sweeping over different discriminator sizes a discriminator size that would not collapse training and produce faithful reconstruction of the data could be found for each number of observations per segment. The found discriminator sizes are those seen in Table 1. Batch size and discriminator size are also closely related as the batch size determines how many samples is presented to the discriminator and thus larger batch sizes tend to require a larger discriminator. We found no trivial tendency such that e.g. when the batch size is doubled so should the size of the discriminator be. In practice a batch size that seemed appropriate for the used device was chosen and then the above-mentioned sweep over discriminator size was performed.

Figure 7: Interactive plots with updating images on hover

Data generation The used data was generated using the same data generator as in Khemakhem et al. [2020b,a]. The parameters of the data generator is summarized in Table 2:

Table 2: Data generator parameters

Data dimension	Number of segments	Number of observations per segment	Mixing layers
2	5	{100, 200, 500, 1000, 2000}	3

The same parameters are used for the generated data in all three models.

MCC metric The Mean Correlation Coefficient (MCC) was used to quantify identifiability in Khemakhem et al. [2020b,a] and we adopt the same metric to quantify identifiability in iVAE-GAN.

Given two sets of observations of m random variables each, the MCC metric calculates the interclass correlation coefficients (either Pearson or Spearman's correlation coefficients) between the m random variables of each set. Since every recovered latent variable should correspond to exactly one true latent variable a linear sum assignment problem is solved such that each recovered latent variable is assigned to exactly one true latent variable and the sum of the assigned correlation coefficients is maximized. The MCC score is then the mean of the correlation coefficients after assignment. A high MCC score thus reflects that the recovered latent variables are highly correlated with the true latent variables.