VRIJE
UNIVERSITEIT
BRUSSEL

Master thesis submitted in partial fulfilment of the requirements for the degree of
Master of Science In de Ingenieurswetenschappen: Computerwetenschappen

VARIATIONAL GREEDY INFOMAX

Interpretable representation learning with latent space constraints

Fabian Denoodt

2022-2023

Promotor(s): Prof. Dr. Bart de Boer

Science and Bio-Engineering Sciences

VRIJE
UNIVERSITEIT
BRUSSEL

Proefschrift ingediend met het oog op het behalen van de graad van Master of Science In de Ingenieurswetenschappen: Computerwetenschappen

[DUTCH] VARIATIONAL GREEDY INFOMAX

[Dutch] Towards independent and interpretable representations

Fabian Denoodt

2022-2023

Promotor(s): Prof. Dr. Bart de Boer

Wetenschappen en Bio-ingenieurswetenschappen

Contents

Chapter 1

Introduction

In recent years, CPC has been shown to be a successful self-supervised learning approach in a wide range of domains (Stacke et al., 2020; de Haan and Löwe, 2021; Lu et al., 2019; Bhati et al., 2021; Deldari et al., 2021; Henaff, 2020). Löwe et al. contribute to this work by showing that modules can be trained greedily, each with their own instance of the InfoNCE loss, enabling modules to be trained in parallel, or sequentially. Moreover, Wang (Meihan Wang, 2019) empirically demonstrates that GIM learns good speech representations for the speech dataset we use in our own experiments 4.1.

However, the underlying representations obtained from optimising this InfoNCE objective have not yet been studied in great detail. In this thesis we aim to gain better understanding in these representations by enforcing constraints to the latent space, resulting in a space that is easier to analyse and understand.

———

FEEDBACK BART

!!! wat is + waarom het gedaan wordt.

should also explain in my work.

should be around 60 pages.

Discussion: Sectie 4.5: (Bart wil aparte sectie) - Hoe helpt deze techniek om het netwerk interpreteerbaarder te maken. - compare techniques: whether they are better explainable. - Zou na results moeten zijn. en kan zo inleiding zijn naar future work. - bart zou verwachten dat cha 3 zeer groot is.

- UVA: wordt geschreven voor begeleiders, hun hebben meer achtergrond. - Jury aan VUB andere verwachtingen, wil meer uitleg.

- mijn discussie sectie moet langer! en zeker in vertellen waarom het beter is voor visualities.

— Defense: - verduideluikende vragen - critiek als gaten in argumentatie - hun komen met suggestie die ze uit literatuur kennen, en moet bv mening over geven. - "zou dat ook anders gekund hebben" !! —

- Context: !E GIM for representation learning: generates representations that simplify classification tasks vs when done on raw data
- Problem: If wants to know for what tasks applicable... must know what is present in data
- Solution: Analysis of learned representations on speech data
- My contributions:

- Decoder ANN for each layer of GIM: Shows what information is maintained through the layers
- Search correlations between kernels weights and signal features
- Extension on CPC via VAE

Chapter 2

Background

[TODO] OUTLINE + WHY IMPORTANT TO UNDERSTAND

2.1 Entropy, relative entropy and mutual information

We first discuss specific definitions from information theory. These concepts will be relevant to understand contrastive predictive coding, which we discuss in a following section. The formal definitions are obtained from the book "Elements of Information Theory" (Cover and Thomas, 2006). The equations that contain a log function are assumed to be under base two.

2.1.1 Shannon's Entropy

Entropy measures the average amount of information required to describe a random variable (Cover and Thomas, 2006). The entropy H(X) of a discrete random variable X, is formally defined as follows:

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

where $\mathcal{X}$ represents the set of events the random variable X can take, formerly known as the *sample space*. Additionally, $p : \mathcal{X} \to [0, 1]$ denotes the probability density function of X. Hence, given an event $x \in \mathcal{X}$, $p(x)$ corresponds to the probability of event x occurring.

Assume a random variable X with possible events x_1, x_2. Intuitively, when $p(x_1)$ is low, the surprise when the event x_1 occurs will be high. The surprise for one event is defined as follows:

$$-\log p(x) \tag{2.2}$$

Hence, entropy can also be considered as the weighted average surprise over each event (Data Science Courses, 2017). To understand why equation 2.2 does indeed correspond to a measure of surprise, consider an event $x \in \mathcal{X}$ with $p(x) = 1$. Note that $\log p(x) = 0$, and thus the surprise is zero. Meanwhile, if p(x) approaches 0, $\log p(x)$ goes to $-\infty$. And hence, by the negation sign in formula 2.1 the surprise is large.

Figure 2.1 displays when entropy reaches its maximum for the case of a random variable with 2 outcomes. We can see that the entropy, and thus, the information is largest when the probability of the two outcomes is equal to each other, namely $p(x_1) = p(x_2) = 0.5$. Note that for a random variable X with more than two events, $H(X)$ can be larger than one.

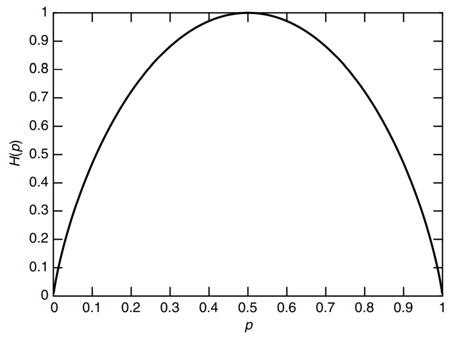


Figure 2.1: $H(p)$ vs p (originates from "Elements of Information Theory", page 16)

2.1.2 Relative entropy and mutual information

Relative entropy, also known as the Kullback Leibler (KL) divergence, is defined in equation 2.3, where p and q denote a probability density function over the same sample space $\mathcal{X}$ (Cover and Thomas, 2006). The KL divergence quantifies the "dissimilarity" or "divergence" between the two distributions. Note that $D(p||q)$ does not necessarily correspond to $D(q||p)$ and thus the metric is not symmetrical.

$$D_{KL}\left(q \mid\mid p\right) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} \tag{2.3}$$

The mutual information (MI) between two random variables X and Y can be computed as the KL divergence between their joint probability distribution, $p_{X,Y}(x,y)$, and the product of their marginal probability distributions, $p_X(x)$ and $p_Y(y)$, which is denoted as $p_X(x)p_Y(y)$ (Cover and Thomas, 2006). The equation for mutual information then becomes:

$$I(X;Y) = D_{KL}\left(p_{X,Y}(x,y) \mid\mid p_X(x)p_Y(y)\right) \tag{2.4}$$

As described by Cover and Thomas in their book "Elements of Information Theory" (**?**), MI quantifies the amount of information Y describes about X. An alternative definition is illustrated in 2.5. The equation provides us with an intuitive meaning for MI, corresponding to the surprise caused by X, which is reduced by the knowledge of Y. In sections 2.3 and 2.4, we discuss how these concepts from information theory are applied in representation learning.

$$I(X;Y) = H(X) - H(X|Y) \tag{2.5}$$

2.2 Supervised machine learning and generalisation challenges

We shall now discuss traditional supervised learning approaches, as these will lay the groundwork for the representation learning discussed in the following section.

2.2.1 Generalisation and curse of dimensionality

Consider a dataset $\mathcal{D} = \{\left(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}\right), \left(\mathbf{x}^{(2)}, \mathbf{y}^{(2)}\right), \ldots\}$, with $\mathbf{x}^{(i)} \in \mathcal{X}$ correspond to a feature vector from the feature space $\mathcal{X}$ and $\mathbf{y}^{(i)} \in \mathcal{Y}$ its class label. Supervised machine learning problems consider the task of finding a function

$$f : \mathcal{X} \rightarrow \mathcal{Y}$$

such that given a feature vector $\mathbf{x}^{(i)}$, a label $\mathbf{y}^{(i)}$ can be *inferred.* However, the entire dataset $\mathcal{D}$ is typically not available to us, and $f(\cdot)$ must be derived from only a partial dataset $\mathcal{D}_{\text{train}} \subset \mathcal{D}$. A good $f(\cdot)$ should not only ensure few errors on the training set $\mathcal{D}_{\text{train}}$, but it should also *generalise* well to unseen $\mathbf{x}^{(i)} \in \mathcal{X}$ (Neyshabur et al., 2017; Chung et al., 2019; Barbiero et al., 2020).

Furthermore, when the number of dimensions of the input space $\mathcal{X}$ increases, the supervised task becomes even more challenging, requiring a larger $\mathcal{D}_{\text{train}}$ to ensure good generalisation. This inability of Machine Learning algorithms to manage high-dimensional data is commonly referred to as the *curse of dimensionality* (Rust, 1997; Aremu et al., 2020; Stuart Russell and Peter Norvig, 2022).

2.2.2 Fully connected Neural Networks

Artificial Neural Networks (ANNs) address this problem of finding $f(\cdot)$, by representing it as a fixed set of parameters, known as layers, which consist of a series of transformation matrices and non-linearities (Jain et al., 1996; Krogh, 2008; Zhang, 2018). During inference, at each layer l a transformation matrix $\mathbf{W}^l$ is applied to the output vector from the previous layer $\mathbf{a}^{l-1}$. This is shown in the equation below.

$$\mathbf{z}^l = \mathbf{W}^l \mathbf{a}^{l-1}$$

A non-linear function $\sigma : R^d \rightarrow R^d$ is then applied to $\mathbf{z}^l$, as shown in equation 2.2.2. The resulting vector a^l may then again be the input for a following layer.

$$\mathbf{a}^l = \sigma(\mathbf{z}^l)$$

Hence, during inference, the input vector $\mathbf{x}$ is propagated through each layer, resulting in a final output $\hat{y}^L$. The equation for the forward pass of a neural network with L layers is described as follows:

$$f_{\mathbf{W}^1 \ldots \mathbf{W}^L}(x) = \hat{y}^L = \sigma(\ldots \sigma(\sigma(x^T \mathbf{W}^1)^T)\mathbf{W}^2 \ldots)^T \mathbf{W}^L$$

The complexity and power of a neural network are affected by the number of layers, and the choice of the number of parameters should depend on the complexity of the learning task. The architecture of a neural network, which includes the dimensions of the matrices, can be represented as a graph-like figure. An example is shown in 2.2 where the edges between $\mathbf{a}^{l-1}$ and $\mathbf{a}^l$ correspond to matrix $\mathbf{W}^l$.

During training the ANN's parameters $\mathbf{W} = \{\mathbf{W}^1 \ldots \mathbf{W}^L\}$, are optimised according to the learning problem. The viability of the parameters is quantified by the loss function $\mathcal{L}$ over a batch of n training samples, shown as follows:

$$\mathcal{L}(\mathbf{W}) = \frac{1}{n} \sum_{i=1}^{n} e(\mathbf{y}^{(i)}, \hat{\mathbf{y}}^{(i)})$$

where $\mathbf{y}^{(i)}$ corresponds to the ground truth label and $\hat{\mathbf{y}}^{(i)} = f_{\mathbf{W}}(\mathbf{x}^{(i)})$. The error function $e(\cdot)$ can be chosen depending on the task, for instance by mean squared error for regression or cross entropy for classification problems.

The parameters $\mathbf{W}^1 \ldots \mathbf{W}^L$ are typically achieved through iterative minimisation of the loss function. This is achieved using backpropagation, an efficient algorithm which computes the

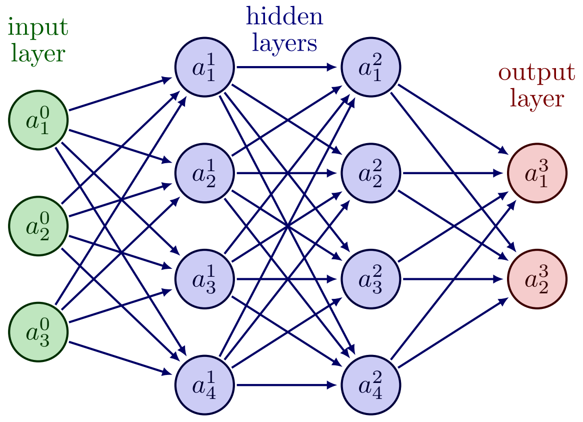


Figure 2.2: Visual representation of a neural network

partial derivates with respect to each parameter $\mathbf{W}^l_{ij}$. Updating the parameters in the opposite direction of the partial derivative allows for an iterative estimation of a local minimum, which is demonstrated in the following equation using stochastic gradient descent:

$$\mathbf{W}^l_{ij} \leftarrow \mathbf{W}^l_{ij} - \alpha \frac{\partial \mathcal{L}}{\partial \mathbf{W}^l_{ij}}$$

The details of computing the partial derivatives using backpropagation are beyond the scope of this thesis.

2.2.3 Towards lower dimensional feature spaces

Supervised learning through ANNs is often successful, even when dealing with complex problems. However, achieving high performance on the training set does not guarantee good generalisation to new data. As problems become more complex, the use of larger amounts of labelled data is required to ensure good generalisation. For complex feature vectors, a more sophisticated architecture may be necessary, which in turn requires even more data to prevent overfitting.

In cases where labelled data is scarce, ANNs may appear to perform well on the training set, but their generalisation performance to data outside the training set can be poor. The performance of ANNs is thus heavily dependent on the choice of data representation (Bengio et al., 2013). As a consequence, a lot of effort in the machine learning pipeline is invested in feature engineering (Zheng and Casari, 2018), which involves extracting useful features from the feature space $\mathcal{X}$, removing redundant information, and reducing the dimensionality of the feature space (Vali et al., 2020) to obtain a good representation $\mathbf{z} \in \mathcal{Z}$ that is easier to work with. Rather than learning the mapping function directly on the data, $\mathbf{x} \in \mathcal{X}$ is first transformed into a lower-dimensional representation $\mathbf{z} \in \mathcal{Z}$. The mapping function from equation 2.2.1 is now:

$$f : \mathcal{Z} \to \mathcal{Y}$$

However, feature engineering is a time-consuming and often manual process that requires domain knowledge and expertise (Przybyszewski et al., 2017; Wei et al., 2021). In the following section we discuss how part of the manual labour of finding good representations from data can be relieved with unsupervised learning approaches, which automate this process. These representations could then be used as the input for a supervised predictor $f(\cdot)$ or *downstream task* (Zhang et al., 2021).

2.3 Representation learning through reconstruction error

One of the challenges in supervised learning is the constant need of large amounts of labelled data $\mathcal{D}_{\text{train}} = \{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \dots\}$. When this set is scarce, or the task becomes too complex to ensure good generalisation, transforming the feature space $\mathcal{X}$ to a lower-dimensional space $\mathcal{Z}$ can be a good solution to aid with learning for the downstream task $f(\cdot)$. Meanwhile, an abundant amount of unlabelled data may already exist. Hence, when a labelled dataset is small, we would like to leverage a larger unlabelled dataset as basis for learning.

In this section we investigate two self-supervised learning paradigms which learn the following transformation function solely from a feature space $\mathcal{X}$ and thus do not require any labels:

$$T : \mathcal{X} \to \mathcal{Z}$$

such that the simplified representations obtained from $T(\cdot)$ can serve as the input for a downstream task as follows:

$$T(\mathbf{x}) = \mathbf{z}$$
$$f(\mathbf{z}) = \mathbf{y}$$

This process of learning a mapping function $T(\cdot)$ which translates a feature vector $\mathbf{x}$ to a lower-dimensional representation $\mathbf{z}$ is commonly referred to as representation learning (Le-Khac et al., 2020).

In the following two subsections we discuss two paradigms of representation learning with ANNs. The first paradigm learns representations by minimising a reconstruction error. The second learns its representations by contrasting them against noise. These two paradigms will lay the basis for our own contributions in chapter three.

2.3.1 Autoencoders

The encoding problem, introduced by Ackley and Hinton in 1985 (Ackley et al., 1985), and later referred to as the autoencoder by Rumelhart et al. in 1986 (Rumelhart et al., 1988), considers the problem of learning compressed representations through neural networks (Rumelhart et al., 1988; Bank et al., 2021). This is achieved through an ANN architecture consisting of two functions, an *encoder* $E(\mathbf{x}) = \mathbf{z}$ and a *decoder* $D(\mathbf{z}) = \tilde{\mathbf{x}}$. The lower-dimensional *encoding* $\mathbf{z}$ obtained from $E(\mathbf{x})$ will then serve as representation for downstream task $f(\cdot)$. The functions $E(\cdot)$ and $D(\cdot)$ are simultaneously optimised by minimising the reconstruction error shown in the following equation:

$$\mathcal{L} = \sum_{i=1}^{N} l(\mathbf{x}^{(i)}, \tilde{\mathbf{x}}^{(i)}) \tag{2.6}$$

where l refers to the error for a single data point, for instance the l^2-norm and N the number data points. The dimension of $\mathbf{z}$ is typically smaller than the original dimension of $\mathbf{x}$. This results in the encoder having to define encodings that are as "informative" as possible to reconstruct the original data (Bank et al., 2021).

An example autoencoder architecture is depicted in figure 2.3. Since an autoencoder is in fact a single ANN and $\mathbf{z}$ corresponds to the output produced by an intermediate hidden layer, $\mathbf{z}$ is also commonly referred to as a *latent* representation. The space of all latent representations $\mathcal{Z}$ is known as the *latent* space. While capable of learning compressed representations, autoencoders do not pose any restrictions on the latent space $\mathcal{Z}$. As a result, the representations may be meaningful to computers, but non-interpretable to humans. For instance, given the left image

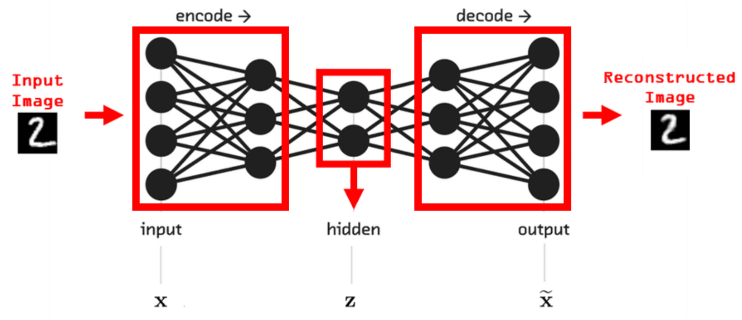


Figure 2.3: Autoencoder architecture, adapted from (Karagiannakos, 2018).

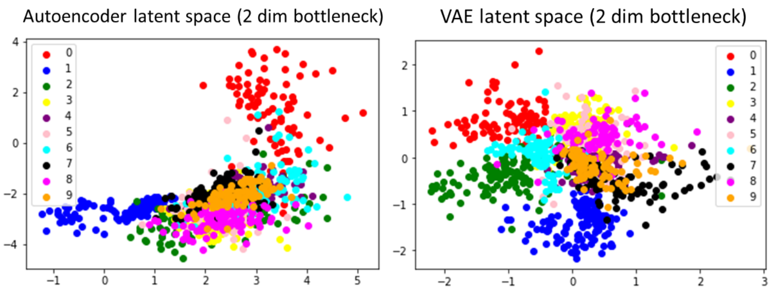


Figure 2.4: Latent space of autoencoder (left) and VAE (right), learned from the MNIST dataset (Lecun et al., 1998), a dataset consisting of images of handwritten digits between 0 and 9. The ANNs have not received any explicit information of the labels of the dataset.

depicted in figure 2.4 which depicts an autoencoder's two dimensional latent space, it is almost impossible to infer in advance what the corresponding digit would be when interpolating a new $\mathbf{z}$ somewhere in between the encodings corresponding to digit 0 (red) and 1 (blue). In addition, slight changes to $\mathbf{z}$ may result in significant changes to its decoded counterpart $\mathbf{x}$. Finally, attempting to understand an autoencoders latent space $\mathcal{Z}$ becomes even more infeasible as the dimension of $\mathcal{Z}$ increases.

2.3.2 Variational autoencoders

Similar to traditional autoencoders, variational autoencoders (VAE) learn representations that contain the important information necessary to reconstruct the data. VAEs maintain the same structure from autoencoders, consisting of encoder $E(\mathbf{x}) = \mathbf{z}$ and decoder $D(\mathbf{z}) = \tilde{\mathbf{x}}$. However, an additional constraint is applied to the latent space $\mathcal{Z}$ (Doersch, 2021; David Foster, 2023; Kingma and Welling, 2022, 2019; Cinelli et al., 2021). In fact, the latent representations $E(\mathbf{x}) = \mathbf{z}$ are samples from multivariate distributions, which we will denote by

$$\mathbf{z} \sim q(\mathbf{z} \mid \mathbf{x})$$

Thus, given a feature vector $\mathbf{x}$, we obtain its corresponding encoding or latent representation $\mathbf{z}$ by taking a sample from a distribution. We will define q in more detail in a following paragraph. The encoder function $E(\cdot)$ in VAEs is thus nondeterministic and computing $E(\mathbf{x}) = \mathbf{z}$ twice may result in two different $\mathbf{z}$s both corresponding to the same $\mathbf{x}$. In a following paragraph we will discuss how to achieve this nondeterministic behaviour using ANNs.

By defining the latent representations as samples from distributions, we can pose the same constraints on the quality of the representations $\mathbf{z} \in \mathcal{Z}$ as in traditional autoencoders, but also additional constraints on how the latent space $\mathcal{Z}$ is organised. The additional constraints posed by VAEs on $\mathcal{Z}$ will result in a more interpretable space, where $\mathbf{z}$'s underlying components can be better understood. This will be achieved by enforcing each distribution $q(\cdot \mid \mathbf{x}^{(i)})$ corresponding to a particular $\mathbf{x}^{(i)}$, to be "similar" to a chosen prior distribution $p(\mathbf{z})$. Typically, the prior $p(\mathbf{z})$ is set to the standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ (David Foster, 2023). The result of this setup can be observed in the latent space depicted in the right plot of figure 2.4 where all $\mathbf{z}$s are attracted to the origin.

In the following subsections, we will explore VAEs in more detail. We begin with a discussion on the process of imposing constraints on $\mathcal{Z}$, which leads to the VAE loss function. Next, we investigate how ANNs can be utilised to simulate sampling from distributions. Finally, we examine how VAEs enable the creation of interpretable representations and how they can be used to generate novel data.

The learning objective

So far, we have learned that Variational Autoencoders (VAEs) use an encoder $E(\mathbf{x}) = \mathbf{z}$ to create a latent representation $\mathbf{z}$ of the input feature vector $\mathbf{x}$, where $\mathbf{z}$ is a sample from the distribution $\mathbf{z} \sim q(\mathbf{z} \mid \mathbf{x})$. Equivalently, for each $\mathbf{x}$, there exists a unique distribution $q(\mathbf{z})$ that characterizes $q(\mathbf{z} \mid \mathbf{x})$ (Volodymyr Kuleshov and Stefano Ermon, 2023). We can then define $q(\mathbf{z} \mid \mathbf{x})$ as a Gaussian with independent components, parametrised by $\boldsymbol{\mu}$ and covariance matrix $\operatorname{diag}(\boldsymbol{\sigma})$. We obtain the the following definition:

$$q(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \operatorname{diag}(\boldsymbol{\sigma})) \tag{2.7}$$

where $\operatorname{diag}(\boldsymbol{\sigma})$ denotes the covariance matrix with $\boldsymbol{\sigma}$ on the diagonal and zeroes elsewhere. It is important to recognise that $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are different for every $\mathbf{x}$. The two parameters will be computed by an ANN.

The representations are optimised to minimise two measurements: one, the reconstruction error, and secondly, the dissimilarity from the latent distributions to a chosen prior $p(\mathbf{z})$. The loss function to be optimised for a single data point $\mathbf{x}^{(i)}$ is shown in the equation below.

$$\mathcal{L}(\mathbf{x}^{(i)}) = \underset{\mathbf{z}^{(i)} \sim q(\cdot \mid \mathbf{x}^{(i)})}{\mathbb{E}} \left[-\log p(\mathbf{x}^{(i)} \mid \mathbf{z}^{(i)}) \right] + D_{KL} \left(q(\cdot \mid \mathbf{x}^{(i)}) \mid\mid p(\cdot) \right) \tag{2.8}$$

Here, $p(\cdot)$ refers to the prior distribution, which is different from $p(\mathbf{x}^{(i)} \mid \mathbf{z}^{(i)})$. Although $\mathcal{L}$ may seem daunting at first, we will decompose its components. The loss function is made up of two terms, the left term corresponds to the reconstruction error, while the second term poses constraints on the latent space. Let us focus on the reconstruction term first.

$$\underset{\mathbf{z}^{(i)} \sim q(\cdot \mid \mathbf{x}^{(i)})}{\mathbb{E}} \left[-\log p(\mathbf{x}^{(i)} \mid \mathbf{z}^{(i)}) \right] \tag{2.9}$$

In a traditional autoencoder we optimise $D(E(\mathbf{x}^{(i)})) = \tilde{\mathbf{x}}^{(i)}$ such that $\mathbf{x}^{(i)} \approx \tilde{\mathbf{x}}^{(i)}$. We will now show that the reconstruction term in equation 2.9 achieves a similar goal for VAEs. Here, $p(\mathbf{x}^{(i)} \mid \mathbf{z}^{(i)})$ is a distribution over $\mathbf{z}^{(i)}$ where $\mathbf{x}^{(i)}$ is a constant, and the latent representation $E(\mathbf{x}^{(i)}) = \mathbf{z}^{(i)}$ is sampled from the Gaussian distribution $\mathbf{z}^{(i)} \sim q(\cdot \mid \mathbf{x}^{(i)})$. The objective is to ensure that $D(\mathbf{z}^{(i)}) \approx \mathbf{x}^{(i)}$. However, since $\mathbf{z}^{(i)}$ is sampled from a Gaussian distribution, the $\mathbf{z}^{(i)}$'s close to the mean $\boldsymbol{\mu}^{(i)}$ are more likely to be sampled than those further away. Thus, for these $\mathbf{z}^{(i)}$'s, we'd like the probability of corresponding to the actual $\mathbf{x}^{(i)}$ to be high, or equivalently

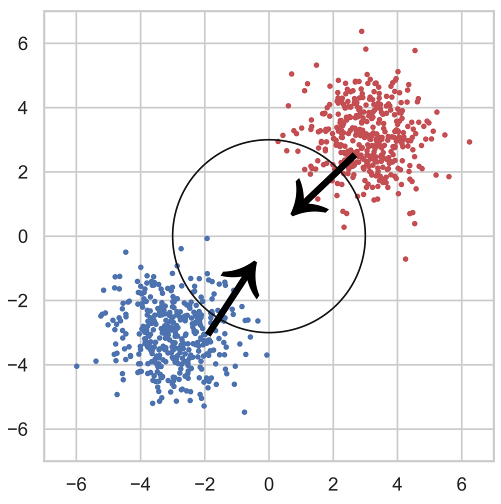


Figure 2.5: 200 samples $\mathbf{z}^{(i)}$ and $\mathbf{z}^{(j)}$ from feature vectors $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$, respectively, represented by red and blue data points. The prior $p(\mathbf{z})$ is set to the standard normal, and the regularisation term of VAE's loss pushes the latent space towards the origin.

we want $p(\mathbf{x}^{(i)} \mid \mathbf{z}^{(i)}) \approx 1$ (Cinelli et al., 2021). Finally, by adding a negative sign in front, maximising this probability is equivalent to minimising the negative probability. In practice this term is approximated through mini-batches with the mean squared error.

Optimising the second term of equation 2.8 poses the constraints on the distributions $q(\cdot \mid \mathbf{x}^{(i)})$.

$$D_{KL}\left(q(\cdot \mid \mathbf{x}^{(i)}) \mid\mid p(\cdot)\right) \tag{2.10}$$

Again, this metric should be minimised. As we discussed in chapter 2.1.2, KL divergence can be considered as a dissimilarity measure between two distributions. Hence, this value is small when the two distributions are similar. The prior distribution $p(\cdot)$ is a fixed distribution which is chosen by the user, typically defined as $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Minimising this metric will result in moving each distribution $q(\mathbf{z} \mid \mathbf{x}^{(i)})$, corresponding to a value $\mathbf{x}^{(i)}$, close to $\mathcal{N}(\mathbf{0}, \mathbf{I})$. This idea is depicted in figure 2.5. When $p(\cdot) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, equation 2.10 becomes:

$$D_{KL}\left(q(\cdot \mid \mathbf{x}^{(i)}) \mid\mid \mathcal{N}(\mathbf{0}, \mathbf{I})\right)$$

When $q(\cdot \mid \mathbf{x}^{(i)}) = \mathcal{N}(\boldsymbol{\mu}^{(i)}, \mathrm{diag}(\boldsymbol{\sigma}^{(i)}))$, one can algebraically prove, that optimising the above equation is equivalent to optimising the following (Kingma and Welling, 2022):

$$D_{KL}\left(\mathcal{N}(\boldsymbol{\mu}^{(i)}, \mathrm{diag}(\boldsymbol{\sigma}^{(i)})) \mid\mid \mathcal{N}(\mathbf{0}, \mathbf{I})\right) = \frac{1}{2}\sum_{k=1}^{D}\left(-\log(\sigma_k^{(i)})^2 - 1 + (\sigma_k^{(i)})^2 + (\mu_k^{(i)})^2\right) \tag{2.11}$$

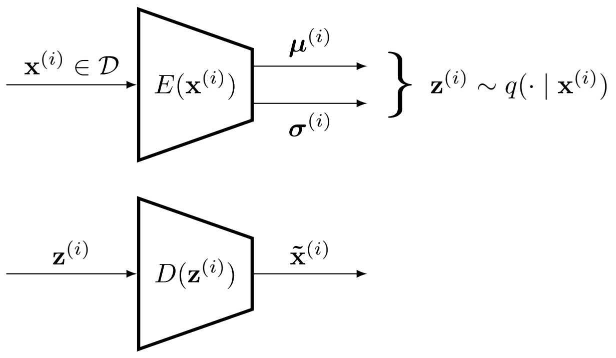
where $\sigma_k^{(i)}$ and $\mu_k^{(i)}$ correspond to the components of the predicted vectors $\boldsymbol{\sigma}^{(i)}$ and $\boldsymbol{\mu}^{(i)}$, respectively. The latent space has dimensionality D.

Simulating distributions through neural networks

Computing $E(\mathbf{x}^{(i)})$ results in encoding $\mathbf{z}^{(i)}$ which is sampled from $\mathbf{z}^{(i)} \sim q(\cdot \mid \mathbf{x}^{(i)})$. We will now discuss how ANNs can emulate the stochastic behaviour of sampling from a distribution. As discussed in equation 2.7, distribution $q(\cdot \mid \mathbf{x}^{(i)})$ is parametrised as a Gaussian with mean $\boldsymbol{\mu}^{(i)}$ and covariance matrix $\text{diag}(\boldsymbol{\sigma}^{(i)})$. The entire distribution can thus be modelled by an ANN which takes as input $\mathbf{x}^{(i)}$ and generates the two corresponding vectors $\boldsymbol{\mu}^{(i)}$ and $\boldsymbol{\sigma}^{(i)}$. Finally, a sample $\mathbf{z}^{(i)}$ can be obtained as follows:

$$\mathbf{z}^{(i)} = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(i)} \tag{2.12}$$

where $\boldsymbol{\epsilon}^{(i)}$ corresponds to a sampled value $\boldsymbol{\epsilon}^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\odot$ is element-wise multiplication. Computing $\mathbf{z}^{(i)}$ through $\boldsymbol{\epsilon}^{(i)}$, rather than directly sampling from $\mathbf{z}^{(i)} \sim \mathcal{N}(\boldsymbol{\mu}^{(i)}, \text{diag}(\boldsymbol{\sigma}^{(i)}))$ is referred to as the parametrisation trick and allows for gradients to freely backpropagate through the layer (David Foster, 2023). The decoder $D(\mathbf{z}^{(i)}) = \tilde{\mathbf{x}}^{(i)}$ can remain identical to the traditional autoencoder's decoder. An overview is presented in figure 2.6.



$\mathbf{x}^{(i)} \in \mathcal{D}$
$E(\mathbf{x}^{(i)})$
$\boldsymbol{\mu}^{(i)}$
$\boldsymbol{\sigma}^{(i)}$
$\mathbf{z}^{(i)} \sim q(\cdot \mid \mathbf{x}^{(i)})$
$\mathbf{z}^{(i)}$
$D(\mathbf{z}^{(i)})$
$\tilde{\mathbf{x}}^{(i)}$

Figure 2.6: High level view of a VAE.

Generating new data and interpretability

Due to the VAE's regularisation term all encodings are pushed towards the centre as depicted in figure 2.5. Additionally, the space of encodings corresponding to a single $\mathbf{x}^{(i)}$ is an entire neighbourhood around a particular $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. These two properties ensure a well covered and uninterrupted space around the origin with smooth transitions on the generative factors between latent representations (David Foster, 2023). As a result, new samples from the dataset can be generated by discarding the VAE's encoder and providing standard normal noise to the decoder instead. It is important to understand that this method only works for VAEs and not for the traditional autoencoder. This is because autoencoders do not pose any additional constraints on the latent space. Generating new data points from random noise via a traditional autoencoder may result in nonsense generations as the random noise may be too different from the latent space it was trained on. As a result, there are no guarantees that the autoencoder's decoder will generalise to these random representations.

Additionally, for the same reason, we can interpolate between encodings obtained from the data. The interpolated encoding can then decoded to the original space. We can observe the specific information contained in each of the representations' features through the decoder, resulting in a significant improvement in interpretability.

Disentanglement of latent representations and posterior collapse

Higgins et al. extend the VAE framework through the introduction of a slight modification of the loss function, resulting in β-VAE (Higgins et al., 2022). An additional hyper-parameter β is introduced in VAE's loss function to control the weight of the regularisation term:

$$\mathcal{L}_{\theta,\phi}(\mathbf{x}^{(i)}) = \underset{\mathbf{z}^{(i)} \sim q(\cdot \mid \mathbf{x}^{(i)})}{\mathbb{E}} \Big[-\log p(\mathbf{x}^{(i)} \mid \mathbf{z}^{(i)}) \Big] + \beta \; D_{KL} \Big(q(\cdot \mid \mathbf{x}^{(i)}) \mid\mid p(\cdot) \Big)$$

Setting the prior to a standard normal distribution ($p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$) encourages disentangled representations, where each variable is sensitive to a specific generative factor (Higgins et al., 2022; Bengio et al., 2013). For example, in an image of a face, altering one component of the representation would only change the smile, while other features like skin colour and brightness remain constant. Such disentangled representations are interpretable, making it easier to understand the underlying structure of the representation and what information each variable contains.

However, a trade-off must be made between accuracy and disentanglement. A small β value (close to 0) results in high reconstruction accuracy but little constraint on disentanglement, while a large β value (above 1) emphasises disentanglement at the cost of accuracy. If β becomes too large, the posteriors $q(\mathbf{z} \mid \mathbf{x}^{(i)})$ corresponding to each data point $\mathbf{x}^{(i)}$ can become equal to the prior $p(\mathbf{z})$, resulting in no information contribution and posterior collapse (Lucas et al., 2022).

2.4 Representation learning through Noise-contrastive estimation

2.4.1 Contrastive predictive coding

In what follows next, we discuss Contrastive Predictive Coding (CPC), a representation learning approach that we use as the basis for our own experiment in the following chapter. CPC is an unsupervised learning approach, again with the objective of learning (lower dimensional) representations from high dimensional data (van den Oord et al., 2019). While the objective is thus the same as for the autoencoders discussed in the previous section, CPC achieves its representations entirely differently. An autoencoder's objective is to define a compressed representation from which the original data can be recovered. However, when working with sequential data $\mathbf{x}_1, \mathbf{x}_2, \ \ldots, \ \mathbf{x}_t, \ \ldots,$ simply compressing patches $\mathbf{x}_i$ of the sequence without considering the relation with nearby patches can result in suboptimal encodings, as contextual information between patches is not encoded directly into the representation (Shah, 2020).

CPC deals with these context issues by maximising the shared information between the extracted representations of temporally nearby patches (Löwe et al., 2020). We will discuss this concept in more detail in this section. For now, we would like to draw the reader's attention to figure 2.7. The figure depicts a high level view of how this idea is achieved by producing two latent latent representations $\mathbf{z}_i$ and $\mathbf{c}_i$. An audio sequence of undefined length is split up into patches $\mathbf{x}_1 \ldots \mathbf{x}_n$ where each $\mathbf{x}_i$ is a vector of fixed length, containing for instance 10ms of speech audio. Each patch $\mathbf{x}_i$ is encoded into latent representation $\mathbf{z}_t$, defined as follows:

$$\mathbf{z}_t = g_{enc}(\mathbf{x}_t).$$

$g_{enc}(\cdot)$ is for instance a Convolutional Neural Network (CNN). The latent representations $\mathbf{z}_1..\mathbf{z}_n$ are obtained independently from each other and do not yet contain any contextual information. This is achieved through $g_{ar}(\cdot)$, an auto-regressor, which encodes all previous $\mathbf{z}_1 \ldots \mathbf{z}_t$ into a single representation $\mathbf{c}_t$:

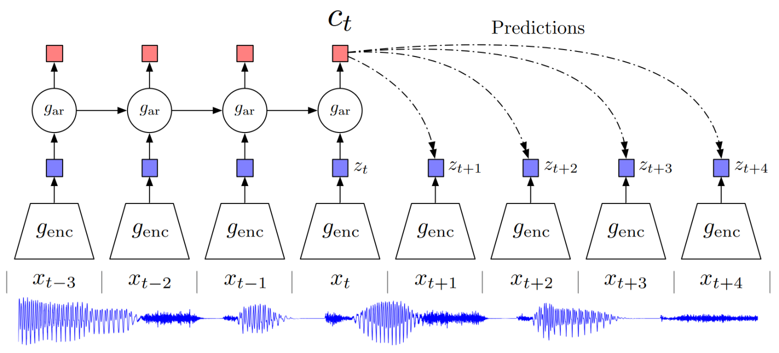


Figure 2.7: Overview of Contrastive Predictive Coding, originates from (van den Oord et al., 2019).

$$\mathbf{c}_t = g_{ar}(\mathbf{z}_1 \ldots \mathbf{z}_t)$$

Either $\mathbf{z}_t$ or $\mathbf{c}_t$ could be used as latent representation for downstream tasks. Oord et al. suggest to use $\mathbf{c}_t$ for tasks where context about the past is useful, for instance speech recognition, and $\mathbf{z}_t$ when historic context is not useful (van den Oord et al., 2019). As shown in figure 2.7, the encodings from sequential data of undefined length, may correspond a series of latent representations $\mathbf{c}_1, \mathbf{c}_2, \ldots$ or $\mathbf{z}_1, \mathbf{z}_2, \ldots$. In the case of downstream tasks which require a single representation vector, Oord et al. propose to pool the sequence of vector representations into a single vector.

Slowly varying features

The temporally nearby patches $\mathbf{z}_{t+1}$ and $\mathbf{c}_t$ are optimised to preserve shared information, while discarding differences. Before we discuss how to obtain such representations, we first motivate why defining representations in this fashion makes sense.

Consider we would like to define useful representations for sequential data such as speech signals. Then it is not unlikely to believe that the conveyed information at time step t and $t + k$ contains some redundancy, such as pitch, frequency, tone, etc. (Rao, 2020). Meanwhile, large changes of the signal in a small time window, may be the result of noise. Sequential data which poses these slowly varying features, are commonly referred to as “slow-features” (Zhang and Tao, 2012). CPC leverages these slowly varying features, by encoding the underlying shared information between different patches, while at the same time discarding low-level information and noise that is more local (van den Oord et al., 2019).

The learning objective

CPC will learn to preserve information between temporally nearby representations, by solving another task. In particular, CPC learns to discriminate subsequent *positive* samples $\mathbf{z}_{t+k}$ from *negative* random samples $\mathbf{z}_j$. This is achieved through a similarity function $f_k(\cdot)$, which scores the similarity between two latent representations (Löwe et al., 2020). It is defined as a log bilinear model as follows:

$$f_k(\mathbf{z}_j, \mathbf{c}_t) = \exp(\mathbf{z}_j^T W_k \mathbf{c}_t) \tag{2.13}$$

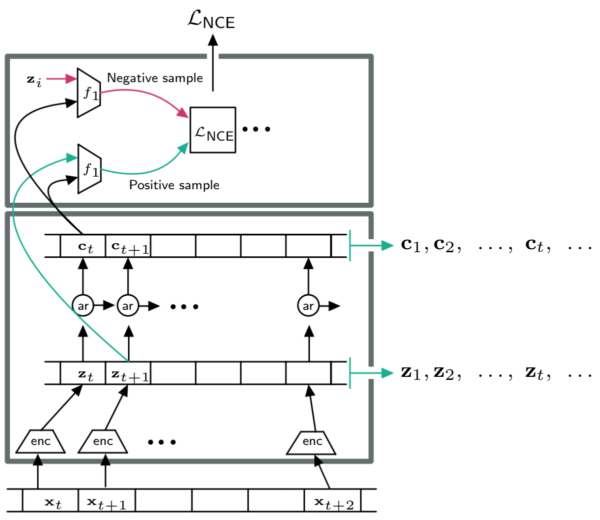


Figure 2.8: Overview of CPC. The encoder, autoregressor and f are ANNs and their parameters are optimised via $\mathcal{L}_{\text{NCE}}$.

where W_k is a weight matrix which is learned. $f_k(\mathbf{z}_j, \mathbf{c}_t)$ thus quantifies how likely the context $\mathbf{c}_t$ corresponds to a random vector $\mathbf{z}_j$. Due to the slowly varying data assumption, a good representation for successive representations $\mathbf{z}_{t+k}$ and $\mathbf{c}_t$ is one where $f_k(\mathbf{z}_{t+1}, \mathbf{c}_{t+1})$ is high and $f_k(\mathbf{z}_j, \mathbf{c}_t)$ is small for random $\mathbf{z}_j$. Or equivalently, maximising the shared information between temporally nearby patches, while discarding the temporal noise results in large values $f_k(\mathbf{z}_{t+1}, \mathbf{c}_{t+1})$.

The InfoNCE loss, used to optimise g_{enc}, g_{ar} and W_k simultaneously is shown below.

$$\mathcal{L}_{\text{NCE}} = -\sum_k \mathop{\mathbb{E}}_{X}\left[\log \frac{f_k(\mathbf{z}_{t+k}, \mathbf{c}_t)}{\sum_{\mathbf{z}_j \in X} f_k(\mathbf{z}_j, \mathbf{c}_t)}\right] \tag{2.14}$$

Here, X corresponds to the set $\{\mathbf{z}_{t+k}, \mathbf{z}_1, \mathbf{z}_2, \ldots\}$. Notice that there exists exactly one $\mathbf{z}_{t+k} \in X$, which corresponds to a positive sample for $\mathbf{c}_t$ and all other $\mathbf{z}_j \in X$ are negative samples. Hence good representations should result in a large similarity score $f_k(\mathbf{z}_{t+k}, \mathbf{c}_t)$, while $f_k(\mathbf{z}_i, \mathbf{c}_t) \approx 0$ for negative samples, resulting in a fraction equal to 1. This would lead to a loss of 0 by the $\log(\cdot)$ function. On the other hand, $\mathcal{L}_{\text{NCE}}$ is large when the denominator is large, which occurs when $f(\cdot)$ is often unsure about negative samples and produces large values for those as well. An overview of the three ANNs which each must be optimised is depicted in figure 2.8.

Ties with mutual information

Earlier we argued that CPC's encodings will preserve shared information between temporally nearby patches, while discarding the local noise. Oord et al. make this claim even stronger by making ties with mutual information, which we discussed in a previous chapter. In particular, Oord et al. prove that optimising InfoNCE is equivalent to maximising the mutual information between $\mathbf{c}_t$ and $\mathbf{z}_{t+1}$ (van den Oord et al., 2019).

$$I(\mathbf{z}_{t+1}; \mathbf{c}_t) = \sum_{\mathbf{z}_{t+1}, \mathbf{c}_t} p(\mathbf{z}_{t+1}, \mathbf{c}_t) \log \frac{p(\mathbf{z}_{t+1} \mid \mathbf{c}_{t+1})}{p(\mathbf{z}_{t+1})} \tag{2.15}$$

This proof is available in their appendix. Although, we do not repeat the proof here, we give a high level overview.

The first step in proving the relation between the InfoNCE loss and mutual information is to model $f_k(\mathbf{z}_{t+k}, \mathbf{c}_t)$ in a probabilistic manner. The InfoNCE loss is in fact the categorical cross-entropy of classifying the positive sample correctly with $\frac{f_k}{\sum_X f_k}$ as the predicted model (van den Oord et al., 2019). Since this equation may take values between zero and one, it can be considered as a probability. In particular, the optimal probability for the loss can then be written as

$$p(i \mid X, \mathbf{c}_t)$$

where X corresponds the set of samples $\{\mathbf{z}_{t+k}, \mathbf{z}_1, \mathbf{z}_2, \dots\}$ as discussed in the InfoNCE loss, and i corresponds to indicator that sample $\mathbf{z}_i$ is the "positive" sample. By doing so, one can eventually obtain a proportionality relation to the density distribution presented below.

$$f_k(\mathbf{z}_{t+k}, \mathbf{c}_t) \propto \frac{p(\mathbf{z}_{t+k} \mid \mathbf{c}_t)}{p(\mathbf{z}_{t+k})} \tag{2.16}$$

Oord et al. utilise this proportionality relation to reformulate $-\mathcal{L}_{\text{NCE}}$ as a lower bound on the mutual information between $\mathbf{z}_{t+1}$ and $\mathbf{c}_t$ as follows (Löwe et al., 2020; van den Oord et al., 2019):

$$I(\mathbf{z}_{t+1}; \mathbf{c}_t) \geq \log(N) - \mathcal{L}_{\text{NCE}} \tag{2.17}$$

Since the number of samples N is a constant, the mutual information between $\mathbf{z}_{t+1}$ and $\mathbf{c}_t$ becomes greater when $\mathcal{L}_{\text{NCE}}$ becomes smaller. Additionally, when the number of samples N increases, the bound becomes tighter.

2.4.2 Greedy InfoMax

So far we discussed how CPC encodes patches of sequential data into latent representations by maximising the mutual information between temporally nearby representations. This method has shown great success in recent years and is considered state-of-the art in self-supervised learning for encoding sequential data (Stacke et al., 2020). Additionally, CPC has been successfully applied to multiple use cases (Stacke et al., 2020; de Haan and Löwe, 2021; Lu et al., 2019; Bhati et al., 2021; Deldari et al., 2021; Henaff, 2020). This is achieved by minimising the InfoNCE loss discussed earlier in equation 2.14. Through this *global* loss function all parameters are optimised end-to-end via backpropagation.

Even though empirical evidence has shown that backpropagation is highly effective (Krizhevsky et al., 2012; Ioffe and Szegedy, 2015), it still suffers from multiple constraints. Firstly, there is a biological perspective to consider, as the human brain lacks a global objective function that can be optimised by backpropagating an error signal (Marblestone et al., 2016). This is especially significant when considering how children can learn to categorise from a few examples, whereas end-to-end backpropagation often requires extensive datasets to achieve good generalisation (Löwe et al., 2020). Moreover, end-to-end backpropagation also suffers from computational constraints, such as requiring the entire computational graph, including all parameters, activations and gradients to fit in memory (Löwe et al., 2020). Additionally, during the training process of a neural network, each layer has to wait for the gradients of its subsequent layer, which reduces locality and impedes the efficiency of hardware accelerator design (Löwe et al., 2020).

Towards greedy learning

To overcome these computational constraints, Löwe et al. introduce Greedy InfoMax (GIM) (Löwe et al., 2020), an extension on CPC inspired from the biological brain. Whereas CPC obtains representations $\mathbf{z}_t$ and $\mathbf{c}_t$ through encoder $g_{enc}(\cdot)$ and autoregressor $g_{ar}(\cdot)$, Löwe et al. split up $g_{enc}(\cdot)$'s neural network architecture by depth into M so called "*modules*":

$$g^1_{enc}(\cdot),\ g^2_{enc}(\cdot),\ \ldots,\ g^M_{enc}(\cdot)$$

A single module may for instance represent one or more layers. Each module's output is the input of the successive module.

$$g^m_{enc}(\mathbf{z}^{m-1}_t) = \mathbf{z}^m_t$$
$$g_{ar}(\mathbf{z}^M_1\ \ldots\ \mathbf{z}^M_t) = \mathbf{c}_t$$

The final representation $\mathbf{z}^M_t$ is obtained by propagating $\mathbf{x}_t$ through each module as follows:

$$g^M_{enc}(\ldots g^2_{enc}(g^1_{enc}(\mathbf{x}_t))) = \mathbf{z}^M_t$$

Both $\mathbf{z}^M_t$ or $\mathbf{c}_t$ may serve as representation for following downstream tasks.

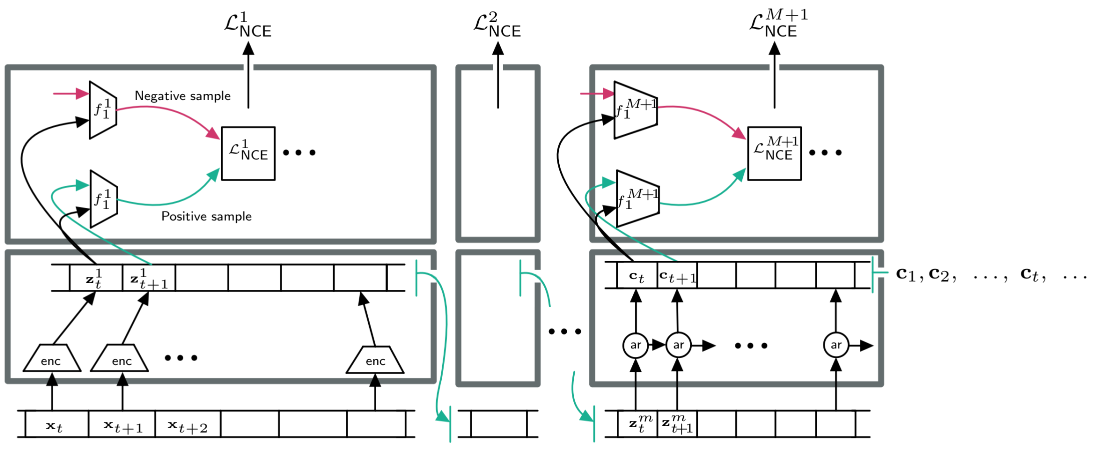


Figure 2.9: Overview of Greedy InfoMax

Each module $g^m_{enc}(\cdot)$ is *greedily* optimised with an adaptation of the InfoNCE Loss. This idea is depicted in figure 2.9, which displays M modules, each trained with their own instance of the InfoNCE loss.

In contrast to CPC where a single loss function $\mathcal{L}_{\text{NCE}}$ and scoring function $f_k(\cdot)$ are required, we now require $\mathcal{L}^m_{\text{NCE}}$ and $f^m_k(\cdot)$ for each module. The equations to optimise for $g^m_{enc}(\cdot)$ then become:

$$f^m_k(\mathbf{z}^m_{t+k}, \mathbf{z}^m_t) = \exp(\mathbf{z}^m_{t+k}{}^T W^m_k \mathbf{z}^m_t) \tag{2.18}$$

$$\mathcal{L}^m_{\text{NCE}} = -\sum_k \mathop{\mathbb{E}}_X \left[\log \frac{f^m_k(\mathbf{z}^m_{t+k}, \mathbf{z}^m_t)}{\sum_{\mathbf{z}^m_j \in X} f^m_k(\mathbf{z}^m_j, \mathbf{z}^m_t)} \right] \tag{2.19}$$

From equation 2.18 we observe, $f^m_k(\cdot)$ no longer receives as input an aggregate $\mathbf{c}_t$, but instead a second $\mathbf{z}_t$. This is in contrast to CPC where $f^m_k(\mathbf{z}^m_{t+k}, \mathbf{c}^m_t)$ is maximised instead. Löwe et al.

thus omit the autoregressive function within each module. Optionally, for downstream tasks where broad context is helpful, such as classification of phonetic structures in speech recognition, an autoregressive model $g_{ar}(\mathbf{z}_1^M \ \ldots \ \mathbf{z}_t^M) = \mathbf{c}_t$ can be appended as a final $M+1$'th module. The altered scoring function then becomes:

$$f_k^{M+1}\left(\mathbf{z}_{t+k}^M, \mathbf{c}_t\right) = \exp\left({\mathbf{z}_{t+k}^M}^T W_k^{M+1} c_t\right)$$

As a result of this approach, the mutual information $I(\mathbf{z}_{t+k}^m, \mathbf{z}_t^m)$ is maximised, rather than $I(\mathbf{z}_{t+k}, \mathbf{c}_t)$ as in CPC.

As a result of GIM's greedy approach, modules can be trained in parallel, but also sequentially. By training one module after the other, the memory cost can be decreased during training which is beneficial for memory-constrained scenarios. Furthermore, when a higher level of abstraction is needed, additional modules can be appended to the architecture later on without having to fine-tune previous modules.

Chapter 3

Variational Greedy InfoMax

3.1 Motivation

In the previous section we discussed two categories of representation learning through deep learning. First, we discussed the autoencoder and its variational counterpart, which minimise the reconstruction error. Secondly, we discussed Contrastive Predictive Coding and Greedy InfoMax, both of which optimise the InfoNCE objective. This category seeks to maximise the mutual information between the encodings of data patches that are temporally nearby. The latent representations obtained from all four methods can then be utilised for downstream tasks (Bengio et al., 2013; Wei and Mahmood, 2021; van den Oord et al., 2019; Löwe et al., 2020)

The autoencoder's sole objective is to learn lower dimensional representations of which the original data can be reconstructed. As a result, the representations may serve well for data compression, but no additional constraints are enforced, such as feature disentanglement and thus the latent space may still be hard to work with for downstream tasks (Tschannen et al., 2018). Meanwhile, VAEs with a standard normal prior, enforce representations which break down or disentangle each feature into a narrowly defined variable and encodes them as separate dimensions (Wei and Mahmood, 2021). This additional constraint could potentially result in better suited representations for downstream tasks. To understand why, consider for instance a classifier which is tasked to predict the pronounced syllables corresponding to the latent representation of a speech wave. It would then be beneficial to have all syllable related information encoded in specific dimensions, which are not influenced by other audio features, such as the speaker's identity.

Both autoencoders and VAEs assess the quality of their representations by comparing the original data with its reconstruction. Consequently, their architectures require an additional decoder block, which must also fit into memory. This can be a disadvantage for resource-constrained devices, as the overhead from the decoder restricts the architectures that these devices can train. Meanwhile, CPC and V-GIM learn representations without having to reconstruct the original data. As a result they do not require a decoder and can benefit from a simplified architecture, while, maintaining state-of-the-art performance (Stacke et al., 2020). A second benefit for CPC and GIM is that they are directly compatible with sequential data.

Both categories (reconstruction and information maximisation algorithms) possess the ability to obtain useful representations for various downstream tasks. However, the content of these representations may not always be intuitive to humans and their structure may be difficult to comprehend. While CPC and GIM are considered state-of-the-art, their performance comes at a cost of having the least interpretable representations. Autoencoders maintain interpretability because a decoder can be used to reveal the information contained in the latent representation. The same transparency can also be achieved with VAEs. Additionally, by using a standard Gaussian as a prior and constraining the latent distributions to be similar to this prior, we can interpolate between representations and observe the effects through the decoder. As such, we can observe the specific information that is contained in each of the representation's features. VAEs can also result in disentangled features, further enhancing interpretability (Grossutti et al., 2022). In contrast, CPC and GIM do not contain a built in decoder mechanism, nor pose constraints on the latent space, significantly reducing interpretability.

3.2 Towards decoupled training for probabilistic representations

In what follows next we introduce Variational Greedy InfoMax (V-GIM), maintaining the state-of-the-art performance obtained from optimising InfoNCE, while leveraging the interpretable and disentangled benefits from VAEs. This is achieved by optimising a novel loss function, *Variational-InfoNCE*, a combination of InfoNCE and the regularisation term from VAEs. Additionally, by splitting up the neural network into modules, as introduced in (Löwe et al., 2020), we greedily optimise each module with its own instance of this loss function. As a result, the interpretability benefits from VAEs will also be applicable in-between modules. This is in contrast to VAEs where solely the final output representations are interpretable.

As discussed in the section on Contrastive Predictive Coding (CPC), a patch of sequential data $\mathbf{x}_t$ is encoded through $g_{enc}(\mathbf{x}_t) = \mathbf{z}_t$ and aggregated over previous encodings through autoregressor $g_{ar}(\mathbf{z}_1 \dots \mathbf{z}_t) = \mathbf{c}_t$, where both $\mathbf{z}_t$ or $\mathbf{c}_t$ may serve as representations for downstream tasks. The encoder function $g_{enc}(\cdot)$ is represented as neural network, eg via a CNN, and $g_{ar}(\cdot)$ for instance as a Gated Recurrent Unit (GRU). Finally, the encoding functions $g_{enc}(\cdot)$ and $g_{ar}(\cdot)$ are obtained by optimising a global loss function, the InfoNCE loss, end-to-end via backpropagation.

Instead, in this study, we split up $g_{enc}(\cdot)$'s network architecture by depth into M modules

$$g^1_{enc}(\cdot),\ g^2_{enc}(\cdot),\ \dots,\ g^M_{enc}(\cdot)$$

and prevent gradients from flowing between modules, as introduced in (Löwe et al., 2020). An additional optional $M+1$'th module $g_{ar}(\cdot)$ can be appended to the architecture. Each module is greedily optimised via a novel loss function, $\mathcal{L}_{\text{V-NCE}}$, which we will define in section 3.3. Each module's output serves as input for the successive module, as presented in the following equations.

$$\begin{aligned} g^1_{enc}(\mathbf{x}_t) &= \mathbf{z}^1_t \\ g^m_{enc}(\mathbf{z}^{m-1}_t) &= \mathbf{z}^m_t \\ g_{ar}(\mathbf{z}^M_1\ \dots\ \mathbf{z}^M_t) &= \mathbf{c}_t \end{aligned}$$

The final representation $\mathbf{z}^M_t$ is obtained by propagating $\mathbf{x}_t$ through each modules as follows:

$$g^M_{enc}(\cdots g^2_{enc}(g^1_{enc}(\mathbf{x}_t))) = \mathbf{z}^M_t$$

Additionally, taking inspriation from VAEs, the outputs from $g^m_{enc}(\cdot)$ and $g_{ar}(\cdot)$ are in fact samples from a distribution denoted by $q(\mathbf{z}^m_t \mid \mathbf{z}^{m-1}_t)$, defined as a multivariate Gaussian with diagonal covariance matrix, as follows:

$$q(\cdot \mid \mathbf{z}^{m-1}_t) = \mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2)) \tag{3.1}$$

with $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ dependent on $\mathbf{z}_t^{m-1}$, specified in more detail in a following subsection. The outputs for $g_{enc}^m(\cdot)$ and $g_{ar}(\cdot)$ are obtained by sampling from this distribution, denoted respectively, as follows:

$$\mathbf{z}_t^m \sim \ q^m(\cdot \mid \mathbf{z}_t^{m-1}) \tag{3.2}$$

$$\mathbf{c}_t \sim \ q^{M+1}(\cdot \mid \mathbf{z}_t^{M}) \tag{3.3}$$

Modules are thus stochastic and computing $g_{enc}^m(\mathbf{z}_t^{m-1})$ twice will likely result in two different representations of $\mathbf{z}_t^m$. This is in contrast to CPC and GIM's latent representations which remain fixed depending to the input (van den Oord et al., 2019; Löwe et al., 2020).

We achieve these stochastic modules by defining each module $g_{enc}^m(\cdot)$ consisting of two blocks. The first block receives as input $\mathbf{z}_t^{m-1}$ and predicts the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, which fully describe the distribution $q^m(\cdot \mid \mathbf{z}_t^{m-1})$ defined in equation 3.1. The second block samples $\mathbf{z}_t^m \sim q^m(\cdot \mid \mathbf{z}_t^{m-1})$ from this distribution and produces an output representation. This is depicted in figure 3.1.

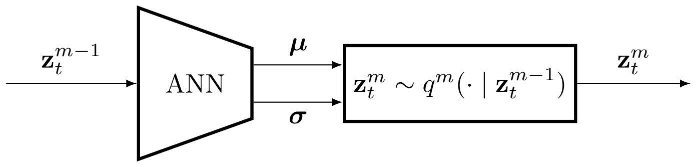


Figure 3.1: A single module.

In practice, sampling from q^m is achieved through a reparametrisation trick, as introduced in (Kingma and Welling, 2022). The equation to compute $\mathbf{z}_t^m$ then becomes:

$$\mathbf{z}_t^m = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$$

where $\boldsymbol{\epsilon}$ corresponds to a sampled value $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\odot$ is element-wise multiplication. The procedure to obtain $\mathbf{c}_t$ is analogous to $\mathbf{z}_t^m$.

3.3 The learning objective

Instead of training the neural network's modules end-to-end with a global loss function, each module is optimised greedily with its own personal loss function. Through the introduction of the novel *Variational-InfoNCE* loss, mutual information between temporally nearby representations is maximised, while regularising the latent space to be approximate to the standard Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The Variational-InfoNCE loss is defined as follows:

$$\mathcal{L}_{\text{V-NCE}}^m = \underbrace{\sum_k \mathop{\mathbb{E}}_{\substack{\mathbf{z}_{t+k}^m \sim q^m(\cdot \mid \mathbf{z}_{t+k}^{m-1}) \\ \mathbf{z}_t^m \sim q^m(\cdot \mid \mathbf{z}_t^{m-1})}} \left[\log \frac{f_k^m(\mathbf{z}_{t+k}^m, \mathbf{z}_t^m)}{\sum_{\mathbf{z}_j^m \in X} f_k^m(\mathbf{z}_j^m, \mathbf{z}_t^m)} \right]}_{\text{Maximise } I(\mathbf{z}_{t+k}^m, \mathbf{z}_t^m)} + \underbrace{\beta \ D_{KL}\left(q^m(\cdot \mid \mathbf{z}_t^{m-1}) \mid\mid \mathcal{N}(\mathbf{0}, \mathbf{I})\right)}_{\text{Regularisation}} \tag{3.4}$$

Here, $m \in \mathbb{N}$ refers to the m'th module and $k \in \mathbb{N}$ the number of patches in the future the similarity score $f_k^m(\mathbf{z}_{t+k}^m, \mathbf{z}_t^m)$ must rate. The latent representations $\mathbf{z}_{t+k}^m$ and $\mathbf{z}_t^m$ are encoded samples produced by $g_{enc}^m(\mathbf{z}_{t+k}^{m-1})$ and $g_{enc}^m(\mathbf{z}_t^{m-1})$, respectively and X is a set of samples $\{\mathbf{z}_{t+k}^m, \mathbf{z}_1^m, \mathbf{z}_2^m, \dots\}$ where $\mathbf{z}_j^m$ with $j \neq t+k$ are random samples.

The similarity score $f_k^m(\cdot)$'s definition is identical to (Löwe et al., 2020):

$$f_k^m(\mathbf{z}_{t+k}^m, \mathbf{z}_t^m) = \exp({\mathbf{z}_{t+k}^m}^T W_k^m \mathbf{z}_t^m)$$

$\mathcal{L}_{\text{V-NCE}}^m$ consists of two terms. The first term ensures that latent representations of temporally nearby patches contain maximised mutual information. The second pushes the latent representations close to the origin. Finally, β is a hyper-parameter which decides the relative importance between the two terms. $\beta >> 1$ will weight more importance to regularisation, but may result in posterior collapse (Lucas et al., 2022). On the other hand $\beta \approx 0$ will attach more importance to the mutual information maximisation term while forgetting about the regularisation term. When $\beta = 0$, V-GIM is identical to GIM but with an altered neural network architecture which supports probabilistic latent representations.

3.3.1 Gradient

To estimate the expectation term in $\mathcal{L}_{\text{V-NCE}}$, we apply the same approximation method as in VAEs, achieved through Monte Carlo estimates (Kingma and Welling, 2022). The first term $\mathcal{L}_{\text{V-NCE}}$ in then becomes:

$$\sum_k \frac{1}{L}\left[\sum_{l=1}^{L} \log \frac{f_k^m({\mathbf{z}_{t+k}^m}^{(l)}, \mathbf{z}_t^{m(l)})}{\sum_{\mathbf{z}_j^m \in X} f_k^m(\mathbf{z}_j^m, \mathbf{z}_t^{m(l)})}\right]$$

Here, L refers to the number of samples drawn for a single data point and each ${\mathbf{z}_{t+k}^m}^{(l)}$ or $\mathbf{z}_t^{m(l)}$ is a different sample. However, as argued by Kingma and Welling, choosing a large batch size will allow this value to be set to 1 without harming performance (Kingma and Welling, 2022).

With regards to the second term, since $q^m(\cdot \mid \mathbf{z}_t^{m-1})$ is a Gaussian defined by parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, a closed form solution exists (Kingma and Welling, 2022). The closed form equation is the following:

$$D_{KL}\left(q^m(\cdot \mid \mathbf{z}_t^{m-1}) \mid\mid \mathcal{N}(\mathbf{0}, \mathbf{I})\right) = \frac{1}{2}\sum_{k=1}^{D}\left(-\log \sigma_k^2 - 1 + \sigma_k^2 + \mu_k^2\right)$$

As such, this term can be directly computed and does not need to be approximated through Monto Carlo. The gradient for the two terms can then be computed using an automatic differentiation tool such as PyTorch (Paszke et al., 2017).

3.3.2 Properties of the latent space

In this section, we present two important claims regarding the structure of the latent space defined by each of V-GIM's modules. For each module $g_{enc}^m(\cdot)$ (or analogously for $g_{ar}(\cdot)$), an input space $\mathcal{Z}^{m-1}$ is mapped to a latent space $\mathcal{Z}^m$ as follows:

$$g_{enc}^m : \mathcal{Z}^{m-1} \to \mathcal{Z}^m$$

Here, $\mathcal{Z}^0$ equals the feature space $\mathcal{X}$. The two claims hold true when $\mathcal{L}_{\text{V-NCE}}$ is optimal. Theey will serve as the main argument for why V-GIM's representations are interpretable, discussed in section 3.4. Meanwhile, traditional techniques such as CPC and GIM lack these interpretable benefits.

Claim 1: $\mathcal{Z}^m$ is uninterrupted and well-covered around the origin.

In V-GIM, a latent representation $\mathbf{z}_t^m \in \mathcal{Z}^m$ of a data point $\mathbf{z}_t^{m-1} \in \mathcal{Z}^{m-1}$ is a sample from the Gaussian distribution $q^m(\cdot \mid \mathbf{z}_t^{m-1})$. Thus, encoding the same $\mathbf{z}_t^{m-1}$ an infinite number of times results in an entire spherical region (around a particular mean $\boldsymbol{\mu}$) in $\mathcal{Z}^m$ that is entirely covered by the latent representations corresponding to $\mathbf{z}_t^{m-1}$, without any interruptions in this region. This is different from GIM and CPC where a data point merely covers a single point of the latent space (and not an entire region). Furthermore, because each region should be close to the origin, regions are more likely to utilise the limited space efficiently around the origin, resulting in a lower chance of obtaining large holes between two regions from different data points. As a result, the latent representations from the dataset should cover the entire latent space around the origin, such that all latent representations around the origin have a corresponding data point that is similar to a data point from the dataset.

Claim 2: $\mathcal{L}_{\text{V-NCE}}$ enforces smooth and consistent transitions in the latent space with respect to mutual information of the original data points.

As we discussed in section 3.3, the first term in $\mathcal{L}_{\text{V-NCE}}$ maximises the mutual information $I(\mathbf{z}_t^m, \mathbf{z}_{t+k}^m)$, which was proven by Oord et al. in (van den Oord et al., 2019). However, in addition, Löwe et al. argue that when modules are trained in a greedy setting using this loss function, the mutual information between outputs of successive modules $I(\mathbf{z}_t^{m-1}, \mathbf{z}_t^m)$ is also maximised (Löwe et al., 2020), and therefore also $I(\mathbf{x}_t, \mathbf{z}_t^m)$. Or equivalently, given a data point $\mathbf{x}_t^{(i)}$, the output produced by $g_{enc}^m(\dots g_{enc}^2(g_{enc}^1(\mathbf{x}_t))) = \mathbf{z}_t^m$ will ensure $I(\mathbf{x}_t^{(i)}, \mathbf{z}_t^m)$ to be maximum. This is an important observation. Now, since the latent representations corresponding to $\mathbf{x}_t^{(i)}$ cover an entire spherical region (as discussed in Claim 1), the mutual information $I(\mathbf{x}_t^{(i)}, \mathbf{z}_t^m)$ for all the latent representations $\mathbf{z}_t^m$ in this region should be maximised.

Let us now consider a second data point $\mathbf{x}_t^{(j)}$ whose corresponding regions in $\mathcal{Z}^m$ overlap with $\mathbf{x}_t^{(i)}$'s region. Since regions are fairly large and the latent space is small, this is likely to happen. Due to the overlap between the regions, there exists a latent representation $\mathbf{z}_t^{m\prime}$ that corresponds to both $\mathbf{x}_t^{(i)}$ and $\mathbf{x}_t^{(j)}$. Since $\mathbf{z}_t^{m\prime}$ is lower dimensional and is defined such that $I(\mathbf{z}_t^{m\prime}, \mathbf{x}_t^{(i)})$ and $I(\mathbf{z}_t^{m\prime}, \mathbf{x}_t^{(j)})$ are both maximised, $\mathbf{z}_t^{m\prime}$ contains information that is common to both $\mathbf{x}_t^{(i)}$ and $\mathbf{x}_t^{(j)}$ (if not, $\mathbf{z}_t^{m\prime}$ would not have been in the two regions).

If small changes to the components of $\mathbf{z}_t^{m\prime}$ caused abrupt changes in the corresponding feature vectors in $\mathcal{X}$, this would be heavily penalised by $\mathcal{L}_{\text{V-NCE}}$ due to the width of the Gaussian regions. Therefore, the transitions in the latent space must be smooth to ensure that small changes in $\mathbf{z}_t^{m\prime}$ lead to small changes in the corresponding feature vectors in $\mathcal{X}$.

Furthermore, due to the definition of q^m with covariance matrix containing only non zero values on the diagonal, the components of q^m are independent. Therefore, moving a latent representation in a single direction of a particular dimension should cause the same effect on the mutual information with respect to the corresponding data point in $\mathcal{X}$ as moving it in a different dimension. This property ensures that the transitions in the latent space are consistent.

Finally, the result of these two claims is an uninterrupted latent space around the origin where moving a latent representation towards a particular dimension in the latent space causes smooth and consistent changes in the corresponding data point from the previous module's latent space. This crucial observation will serve as the main argument for why V-GIM's representations are interpretable, while traditional techniques such as CPC and GIM do not have these guarantees.

3.4 Computational benefits

Each module in V-GIM is greedily trained through the variational InfoNCE loss. This approach, as suggested by Löwe et al. (Löwe et al., 2020), offers several computational benefits, including decoupled and asynchronous distributed training. Modules can then be trained in parallel on multiple devices, or sequentially on a single device. This is particularly interesting for memory constrained devices, as it enables the training of architectures that exceed the available memory capacity. Furthermore, V-GIM mitigates the vanishing gradient problem, a well-known issue in deep architectures, by preventing the flow of gradients between modules (Hu et al., 2021).

In addition to the benefits associated with greedy training, V-GIM adds a constraint to the latent space produced by each module, resulting in further practical benefits. Since V-GIM adopts a greedy training approach, these benefits apply not only to the final module's output but also to the outputs of intermediate modules.

Interpretability of final and intermediate modules

By regularising the latent space resulting in the properties discussed in section 3.3.2, each of V-GIM's modules define a space where sampling from a point around the origin is likely to correspond to a data point that is similar to the dataset. A decoder trained on this space will generalise well to unseen data when given a latent representations around the origin. This has significant implications to V-GIM's interpretability. Not only can we assess the information contained in a latent representation through a decoder, we can also attempt to understand the underlying structure of the latent representation. This is achieved in a similar way as with VAEs, where manipulating individual dimension's value and observing the effect through the decoder gives insight in the specific information contained in that dimension.

Furthermore, since V-GIM's neural network architecture consists of a variable number of modules, each with their own interpretable latent space, these benefits are applicable to the latent representations produced by intermediate modules, allowing us to observe the internal mechanism of the neural network. Additionally, due to the CNN working, different modules encode different sequence lengths, encouraging different abstractions to be learned at different levels. By having intermediate modules be interpretable, we can analyse these abstractions as well. This is in contrast to VAEs, where only the output latent representation is interpretable, and intermediate representations in the architecture are not.

In contrast, GIM and CPC do not impose constraints on the latent space. Although a decoder can still be trained on the latent representations of GIM and CPC, the latent space is highly unpredictable. Consequently, it is not guaranteed that the reconstructed output from a randomly sampled latent representation will be meaningful, as it may be very dissimilar from the the original data. Moreover, attempting to decode an interpolated representation from two existing latent representations may not lead to a meaningful output either, since the latent space the decoder was trained on may contain large gaps. This makes it challenging to interpret the underlying structure of the latent representations defined by CPC and GIM.

Disentanglement

In section 2.3.2 on β-VAEs and disentanglement, Higgins et al. argue that setting the prior $p(\mathbf{z})$ to an isotropic Gaussian encourages disentanglement in the encodings (Higgins et al., 2022). When encodings are fully disentangled, this results in each dimension from the encoding to capture a different feature from the original data. Since $\mathcal{L}_{\text{V-NCE}}$ enforces the latent space to be standard normal, this theorem is also applicable to V-GIM and choosing a large value for β in $\mathcal{L}_{\text{V-NCE}}$ applies more pressure for encodings to be disentangled further increasing interpretability.

Improved generalisation performance through representation variance

V-GIM's encodings are samples from a distribution, which means that a single patch of data $\mathbf{x}_t$ may have multiple encoded representations $\mathbf{z}_t^M$. For downstream tasks with very little labelled data, the variability in representations could serve as an internal data augmentation method, which may improve generalisation to unseen data for downstream tasks.

Internal batch normalisation mechanism

During training of ANNs, the weights of different layers undergo changes, which can lead to a problem known as "internal covariate shift" (Ioffe and Szegedy, 2015). This shift causes the distributions of each layer's input to change over time, which in turn can negatively affect subsequent layers, slowing down the training process (Bjorck et al., 2018; LeCun et al., 1998). This issue is typically addressed via batch normalisation (Santurkar et al., 2018; Bjorck et al., 2018), a mechanism which normalises the activations of internal layers, allowing for a higher learning rate during training and thus accelerating the process.

Batch normalisation is especially important in a greedy setting such as GIM and V-GIM, where modules are independently trained in parallel. Without intermediate normalisation, subsequent modules may learn slower than previous modules, resulting in the subsequent modules not being able to catch up in time to the changes from preceding modules made during training. The result would be that modules may have to be trained with different learning rates, resulting in an additional hyperparameter that must be obtained for each module.

In contrast, V-GIM already contains an internal normalisation mechanism in-between modules. This is indirectly caused by regularising each module's latent space to have mean zero and standard deviation of one. Finally, the normalised encodings can also be beneficial for downstream tasks. If normalisation is desired, computing the mean and standard deviation over a potentially very large dataset is not required, as this is already built in to the encodings.

Chapter 4

Experiments

In this chapter, we evaluate the effectiveness of the latent representations obtained from Variational Greedy InfoMax (V-GIM) in comparison to its non-variational counterpart, GIM, by training both models on sequential data from the audio domain. To assess the quality of the representations, we project them into a two-dimensional space using t-SNE and analyse the emergence of potential clusters. Moreover, we use a linear classifier that takes these representations as input, and evaluate its accuracy to gain insights into the performance of these models. To further examine the efficacy of these representations, we investigate their potential for downstream task generalisation by training the linear classifier on smaller subsets of the dataset and assessing the amount of labelled data required to achieve satisfactory performance. Finally, we delve into the underlying structure of V-GIM's representations by training a decoder on top of each of V-GIM's modules, providing insights into their composition.

4.1 Experimental details GIM and V-GIM

GIM and V-GIM are trained on raw speech signals of fixed length sampled at 16kHz. The dataset consists of 851 audio files and is randomly shuffled and split up into 80% training data (680 files) and 20% test data (171 files). Each audio file consists of a single spoken sound consisting of three consonants and three vowels, where the consonants and vowels alternate with each other. Some examples are the sounds "gi-ga-bu" and "ba-bi-gu". The words consist of three syllables from the following list: ba, bi, bu, da, di, du, ga, gi and gu. All the sounds are spoken by the same person, in a constant and peaceful tone. We crop the files to a fixed length of 10240 samples, or 640 milliseconds, which is slightly longer than half a second.

GIM and V-GIM are trained on the same architecture, consisting of two encoder modules $g^1_{enc}(\cdot)$ and $g^2_{enc}(\cdot)$, no autoregressor module $g_{ar}(\cdot)$ is used. Modules are trained in parallel. Each module consists of solely convolution and pooling layers. Since the architecture contains no fully connected layers, the length of the audio files can therefore be variable during inference. The architecture details for the two modules are presented in tables 4.1 and 4.2. ReLu non-linearity is applied after each convolution, except in the last layer in each module. No batch normalisation is used. The number of hidden channels produced by each convolution and pooling layer remains constant, at 32 channels. In each module, data flows synchronously from one layer to the successive layer, except in the final two layers. These run in parallel and both take the same input from the preceding layer. This architecture choice is related to the parametrisation trick we discussed in section 3.2. One layer generates $\boldsymbol{\mu}$ and the other $\boldsymbol{\sigma}$ such that $\mathbf{z}^m_t$ can be computed via the reparametrisation trick. The architecture is visualised in figure 4.1. An input

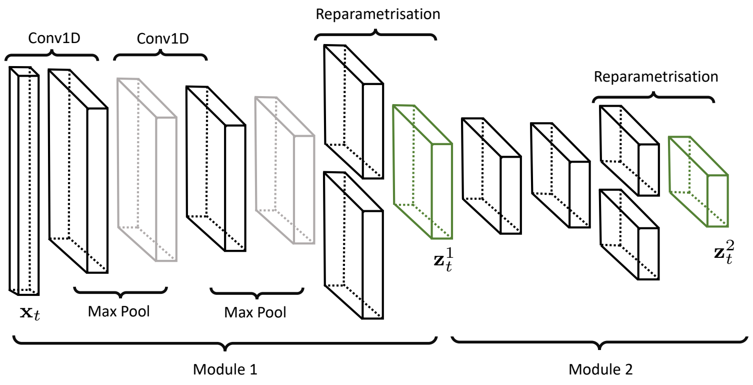


Figure 4.1: Visualisation of V-GIM's architecture.

signal of length 1×10240 is downsampled to 32×52 by the first module and to 32×13 by the second module. Due to overlap, each latent representation $\mathbf{z}_t^M = \mathbf{z}_t^2$ captures 64ms of speech.

Layer	**Kernel Size**	**Stride**	**Padding**
Conv1D	10	4	2
Max Pool	8	4	0
Conv1D	8	3	2
Max Pool	8	4	0
In parallel:			
Conv1D	3	1	2
Conv1D	3	1	2

Table 4.1: Module 1 architecture.

Layer	**Kernel Size**	**Stride**	**Padding**
Conv1D	6	2	2
Conv1D	6	2	2
In parallel:			
Conv1D	3	1	1
Conv1D	3	1	1

Table 4.2: Module 2 architecture.

Importantly, the reparametrisation trick is included in our implementation of GIM's architecture as well. However, by enforcing no constraints on the latent space, which is achieved by assigning $\beta = 0$ in $\mathcal{L}_{\text{V-NCE}}$, we observed that $\boldsymbol{\sigma}$ moves towards $\mathbf{0}$ such that all the randomness from sampling from a Gaussian distribution is removed. As such, our implementation of GIM is equivalent to GIM introduced in (Löwe et al., 2020) but with an altered architecture.

Both GIM and V-GIM are trained using the Adam optimiser for 800 epochs with an exponential learning rate scheduler (Bhargav Lad, 2020), with initial learning rate $lr_{init} = 0.01$ and $\gamma = 0.995$, such that lr is updated as follows:

$$lr_{epoch} = \gamma * lr_{epoch-1}$$

The batch size is set to 171, which is exactly the size of the test set. The number of patches to predict in the future k, is set to 12. Implementation details with regards to drawing negative samples for $f_k^m(\cdot)$ remain identical to the experiments from (van den Oord et al., 2019) and (Löwe et al., 2020). The regularisation importance term β is set to 0 for GIM and 0.0035 for V-GIM, which is the largest value we could obtain without causing posterior collapse.

After training V-GIM and GIM for 800 epochs, we train linear multi-class classifiers for 100 epochs on their respective latent representations for every module, evaluated using Cross-Entropy (Ho and Wookey, 2020). We set $lr = 0.001$ and use the Adam optimiser without scheduler. The classifier is tasked to predict the syllable corresponding to its latent representation. Details on the algorithm used to split up audio files by individual syllables are provided in appendix A. Speech waves are zero-padded at both the beginning and the end to achieve a fixed length between syllables. The corresponding latent representations thus all consist of a fixed number of time frames $\mathcal{T}$ and 32 channels. Here, $\mathcal{T}$ is different depending on the module. The classifier consists of a single connected layer with $32 \times \mathcal{T}$ input nodes and 9 output nodes, matching the number of input features and number of classes respectively. We reshuffle the dataset and use 80% for training and 20% for testing.

4.2 Results GIM and V-GIM

4.2.1 Training error

Figure 4.2 displays the loss curves corresponding to GIM ($\beta = 0$) and V-GIM ($\beta = 0.0035$) for both modules. In the first module, the test loss for GIM is lower compared to V-GIM. This is expected because V-GIM includes an additional regularisation term in its loss function. In the second module, both GIM and V-GIM have loss curves that converge to lower values than their previous module. This suggests that the task becomes easier for the subsequent modules, indicating that the preceding module does indeed make simplified representations of the speech data.

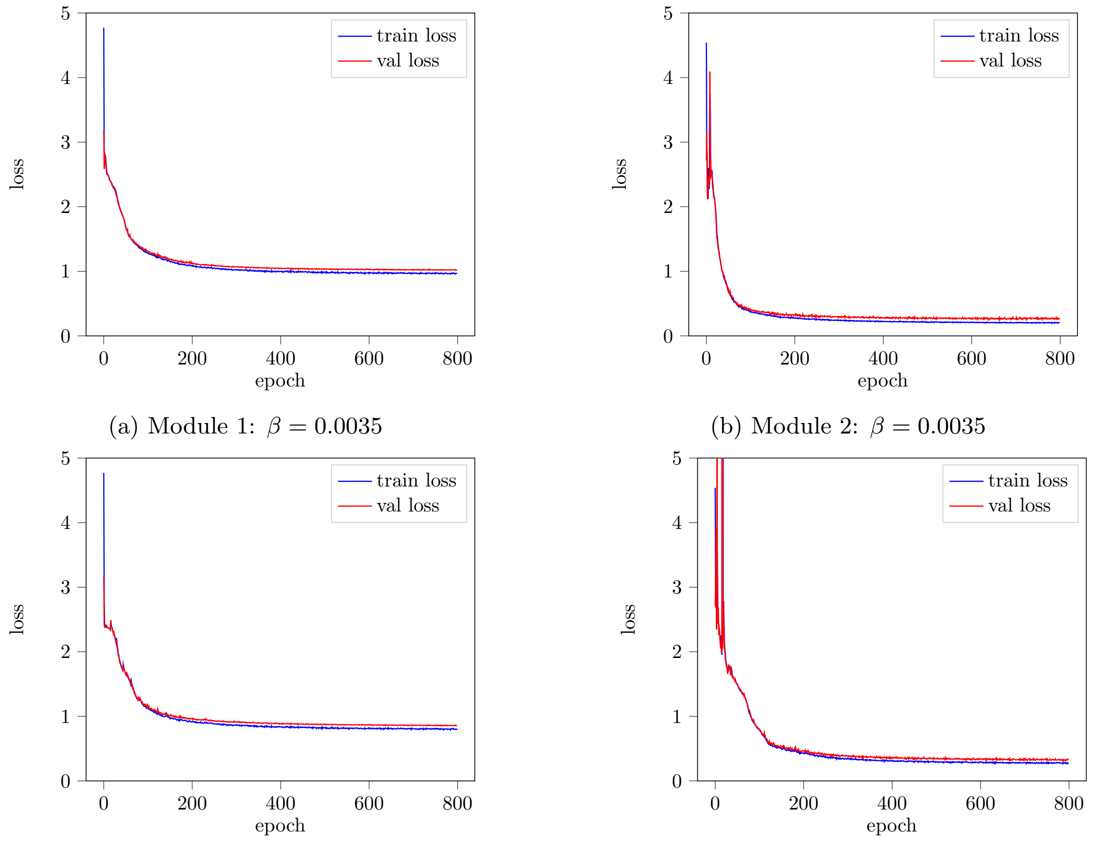
Interestingly, in the second module, V-GIM has a lower loss compared to GIM, even though V-GIM includes the regularisation term in its loss function. We argue that this is related to internal covariate shift, which we discussed in section 3.4. Using a relatively large learning rate for GIM and V-GIM leads to a significant “drift” in the distribution of activations during training, making it difficult for GIM’s successive modules to keep up with these changes. V-GIM is less susceptible to this issue due to the internal batch-normalisation mechanism that is built into the regularisation term. Consequently, V-GIM’s modules can train with a larger learning rate without affecting the performance of successive modules, resulting in faster convergence.

4.2.2 t-SNE

We project the latent representations obtained from each module to a two-dimensional plane using t-SNE (van der Maaten and Hinton, 2008). This enables us to observe potential clusters, as similar data points are mapped close together, while dissimilar points remain far away. Similar to the classifier discussed in section 4.1, t-SNE is trained on flattened representations that are split into individual syllables, without performing any pooling. We run t-SNE with random initialisation, perplexity of 30 and a learning rate of 200 for 1000 iterations.

The graphs for GIM and V-GIM are displayed in figure 4.3. T-SNE was not provided with any information about the class labels. Syllables containing the vowel “a” are represented with a red tint, “u” with green, and “i” with blue. We observe similar results in the first module of both GIM and V-GIM. T-SNE identifies two clusters, with one cluster corresponding to syllables containing an “a” and the other cluster to containing “u” or “i”. Within the “u/i” cluster, there is still a grouping of “u” and “i” data points. However, distinguishing between “b”, “d” or “g” is more challenging as data points within the clusters are entangled.

In the second module, we observe more significant differences between GIM and V-GIM. In V-GIM’s second module, there is a clearer separation between the “i” and “u” syllables, indicating that the second module contributes to improved latent representations. However, distinguishing between the pronounced consonant remains difficult. On the other hand, GIM’s second module does not appear to have converged well, as t-SNE struggles to separate the representations into



(a) Module 1: $\beta = 0.0035$

(b) Module 2: $\beta = 0.0035$

(c) Module 1: $\beta = 0$

(d) Module 2: $\beta = 0$

Figure 4.2: Training and validation loss (GIM: $\beta = 0$, V-GIM: $\beta = 0.0035$).

meaningful clusters. Consequently, syllables are mixed without much structure. This further emphasises how GIM is affected by internal covariate shift, while V-GIM is not.

4.2.3 Classification performance

Table 4.3 presents the accuracies of the linear classifier, which aims to predict the syllable corresponding to the latent representation. There are a total of 9 different syllables, such that a random model would obtain an accuracy of 11%. While the t-SNE plots in figure 4.3 have demonstrated that vowels can be distinguished relatively easily, differentiating consonants poses a greater challenge. As a result, both GIM and V-GIM show mediocre test accuracies, ranging between 53% and 54% in the first module, and further decreasing in the second module. These accuracy values, coupled with the t-SNE plots, indicate that information regarding the consonants may no longer be adequately captured in GIM and V-GIM's latent representations. Moreover, the performance continues to decline when considering the accuracies obtained from the two second modules.

We believe that these mediocre accuracies can be attributed to the limited mutual information between temporally nearby patches. GIM and V-GIM assume data that adheres to the slowly changing features assumption, which is necessary to maximise the mutual information between

Method	**Accuracy Module 1 (%)**	**Accuracy Module 2 (%)**
GIM	54.39	28.07
V-GIM	52.92	41.15

Table 4.3: Accuracies obtained on the test dataset.

the latent representations of temporally nearby data patches. Consequently, abrupt changes in the patches are discarded from the latent representations. However, in general, the words spoken in the dataset consist of longer durations for vowels compared to consonants. For instance, in the syllable "ba", the phoneme "b" is usually pronounced for a shorter duration than the "a". This can cause a problem when the phone "b" is not captured over multiple patches, as only the mutual information between the patches is kept. Additionally, latent representations are optimised to maximise the mutual information with the future k patches. Since k remains fixed over the different modules and deeper modules capture longer time windows, the latent representations of deeper modules must capture more information, while the number of channels remains relatively small, at 32.

4.2.4 Distributions

Figures 4.4, 4.5, 4.6 and 4.6 depict the distributions of activations corresponding to an individual dimension. Each sub-figure depicts the distributions for a single module. We show distributions for the first 6 dimensions. We observe that V-GIM does indeed learn to constrain the latent space to the standard normal for most dimensions. Meanwhile, in GIM's first module, we observe high and thin peaks, indicating that the same value is regularly predicted with little noise due to $\boldsymbol{\sigma} \approx 0$. The peaks are less dominant in the second module, which may be attributed to suboptimal convergence.

4.3 Generalisation study

In contrast to GIM, V-GIM's latent representations are samples from a distribution, resulting in a single patch of data having multiple latent representations. In this section, we examine whether a linear classifier with little annotated training data can benefit from this representation variability introduced by V-GIM. We train multiple multi-class classifiers with the same experimental details as discussed in section 4.1 but with a modification to the annotated training set. Each classifier is trained on a subset of the dataset, varying between 1 data point per class, all the way up to 128 data points. The batch size is set to the size of the subset. Training details for GIM and V-GIM remain unchanged, including the size of the training set, which does not require any labels.

The test accuracies are shown in figure 4.8. Overall, we observe no performance benefit from V-GIM's representation variance. Performance in GIM and V-GIM's first modules remains consistent, regardless of the subset size. Meanwhile, differences in the second module are more prominent. However, this is related to the internal batch normalisation mechanism, discussed in section 4.2.1, resulting in faster convergence of V-GIM's modules.

4.4 V-GIM's Interpretability analysis

In the following sections, we delve deeper into V-GIM's latent representations, aiming to gain an understanding of the captured information and understand underlying structures. This is achieved by employing a decoder on top of each of V-GIM's modules. By altering a representation's component values and observing the effect through the decoder, we can analyse the contained information in each individual dimension. As we argued in section 3.4, this is only possible because V-GIM's latent space is optimised to be approximate to the standard normal. The decoder can then generalise to the altered representations as long as the representations are close to the origin.

4.4.1 Decoders for V-GIM

We employ two decoders, one for each module, which can be represented as follows: $D(\mathbf{z}_t^1) = \widetilde{\mathbf{x}}_t$ for the intermediate decoder and $D(\mathbf{z}_t^2) = \widetilde{\mathbf{x}}_t$ for the final decoder. This allows us to assess the information in both the final and the intermediate representations. Architecture details for the two decoders are provided in tables 4.4 and 4.5. An intermediate representation $\mathbf{z}_t^1$ with a shape of 32×1, capturing a single time step, is transformed into a speech signal $\widetilde{\mathbf{x}}_t$ with a shape of 1×448 (or 28ms). Similarly, the final representation $\mathbf{z}_t^2$ is transformed into a shape of 1×1024 (or 64ms).

Layer	Kernel Size	Stride	Padding
ConvTrans	3	1	1
ConvTrans	8	4	0
ConvTrans	8	3	2
ConvTrans	8	4	0
ConvTrans	10	4	2

Table 4.4: Decoder architecture for intermediate latent representations.

Layer	Kernel Size	Stride	Padding
ConvTrans	3	1	1
ConvTrans	8	3	2
ConvTrans	8	3	2

Table 4.5: Decoder architecture for final latent representations:

While the decoders can be optimised by minimising the Mean Squared Error (MSE) of the speech waves, this metric may not reflect well with the natural biases in human hearing. Humans perceive certain frequencies to be louder than others (Radkoff, 2021; Li et al., 2020). To account for this, we instead minimise the MSE on the mel-frequency spectrograms, which are an adaptation of linear spectrograms that emphasise frequency bins based on perceptual hearing biases (Shen et al., 2018). Additionally, we employ a logarithmic transformation to account for humans' logarithmic perceptual hearing (Braun and Tashev, 2020). Figure 4.9 illustrates an example of a log mel-spectrogram.

The loss function we use is the following:

$$\mathcal{L}_{\text{decoder}} = \frac{1}{n} \sum_{i=1}^{n} \Big(\log(\text{MEL}(\mathbf{x}_t^{(i)})) - \log(\text{MEL}(\widetilde{\mathbf{x}}_t^{(i)})) \Big)^2$$

In this equation, $\text{MEL}(\mathbf{x}_t^{(i)})$ represents the mel-spectrogram of the original signal $\mathbf{x}_t^{(i)}$, computed using the following parameters: the number of FFT bins set to 4096, the length of the hop between STFT windows set to 2048, and the number of filter banks set to 128. Other parameters are kept at their default values in PyTorch's MelSpectrogram implementation (Paszke et al., 2017). Logarithms in the equation are computed in base 10.

4.4.2 V-GIM representation analysis through decoder

Decoding latent representations into their original representation as a speech wave allows us to perceive the remaining information in the representation. It is worth mentioning that the reliability of this analysis is dependent on the performance of the decoder. If it is not able to reconstruct certain features, this does not necessarily mean the information is not contained in the latent representation.

Training and validation loss curves for both decoders are shown in figure 4.10. Overall, in both decoders, we observed that audio files could be reconstructed and the pronounced syllables were recognisable when listening to them. However, reconstructed sounds were noisier and information about the speaker's identity was unrecognisable, suggesting that this information is either no longer contained in the latent representations, or the decoders are not able to replicate it. Additionally, while vowels were easily identifiable by listening to the audio, we observed that the decoder occasionally generated incorrect consonants. For instance, decoding the latent representation for "gu" could occasionally result in an incorrectly generated "bu" or "du". These observations are in line with the t-SNE plots classification performance we discussed in the previous section, further suggesting that consonant information may no longer be present in the latent representations.

By decoding the latent representations of existing speech data, we gain insight in the remaining information in the representations. However, our goal goes further: we would like to also understand the underlying structure of these representations, and attempt to understand the precise information contained in each dimension. We achieve this by starting from the zero-vector **0** as latent representation, and manipulating a single dimension at a time. We can then observe the effect in the reconstructed speech signals, by looking at the it from the time and frequency domain. Figures 4.11, 4.12 and 4.13 show reconstructed speech signals, obtained from the *intermediate* decoder. We show the reconstructed audio waves, both in the time and frequency domain. In each figure, a single dimension is being manipulated, ranging between -2.68 and 2.68. Since the latent space is optimised to conform to the standard normal distribution, this would mean that 99% of the data points are contained within this range (Bhandari, 2020).

We categorise the dimensions into one of three groups based on their contained information. Figures 4.11, 4.12 and 4.13 show an example of each category. The first category contains information about frequencies for a specific frequency bin. Altering this dimension will a have strong influence on the magnitude of one or two frequency bins while other bins remain relatively unchanged. This is shown in figure 4.13, where changing the value of the 12th dimension towards -2.68 causes a peek around the 100 Hz frequency bin. The peek disappears when approaching 0. Meanwhile, when approaching +2.68, the magnitude of the 150 Hz frequency bin increases. Overall, we observed this kind of behaviour for roughly half of the 32 dimensions. The majority of dimensions were sensitive in the 100 Hz and 150 Hz frequency bins, while a few were in the 175 Hz bin. These ranges are to be expected as the pitch range for men's voices is between 60 and 180 Hz (Re et al., 2012).

Dimensions that belong to the second category, not only influence the magnitude of frequencies in a specific bin, but also the neighbouring bins. In this category, modifying a dimension will cause an increase in the magnitude of a particular bin, while simultaneously influencing the surrounding frequencies. This results in a smooth envelope with a single peak which gradually decreases across the neighbouring bins. This behaviour is shown in figure 4.11.

The final category includes the dimensions which do not seem to make any contribution to the reconstructed signal. Changes to these cause very little change to the resulting speech wave, both in the time and frequency domain. These dimensions are either useless, or the decoder was not able to capture their contribution. An example is shown in figure 4.12.

Regarding the second decoder, which was trained with V-GIM's *final* representations, we observe that syllables are audible, but due to disturbances, it becomes more challenging to distinguish the precise syllables and consonants. This is to be expected, as latent representations in the second module capture speech sequences of a longer time frame (64ms instead of 28ms) while remaining the same size at 32 dimensions.

In the final representations, we observed more dimensions of the second category, containing the information of an entire neighbourhood of frequency bins, influencing frequencies ranging between 80 to 300 Hz. Additionally, we observed peaks in the frequency domain with much stronger magnitudes, compared to those in the intermediate representations. An example is shown in figure 4.14, displaying the reconstructed speech wave in the time and frequency domain for a final representation.

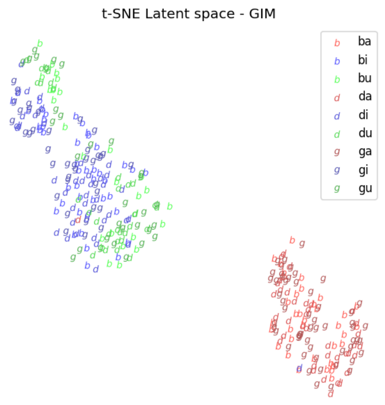
4.4.3 Consonants

To gain better insight in which dimensions incorporate vowel information, we train a linear classifier to predict whether the vowel corresponding to a syllable's latent representation is an "a", "i" or "u". Implementation details are the same as the linear classifier in section 4.1, but with only the three classes and without a bias term. The classifier is trained on the latent representations from V-GIM's first module. Additionally, representations are max pooled over the time domain, resulting in a single 32 dimensional feature vector for each syllable. After training, a Softmax function is appended to normalise predictions into probability values. After convergence, the classifier obtained a validation accuracy of ??% [**Insert exact value here (80% roughly)**].

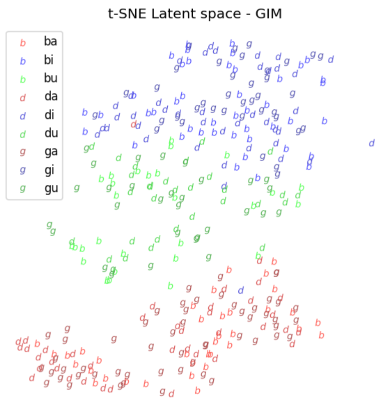
Figure 4.15 displays the weights of the linear classifier. The weights are scaled between -1 and 1. Each row corresponds to the weights corresponding to an individual prediction class. Equivalently, when considering a graph representation for the classifier, the weights on the edges corresponding to a particular output node, are the values in a single row. Consequently, values close to 1 or -1 indicate a high (positive or negative) correlation between the target class and the corresponding dimension.

Interestingly, most dimensions appear to contribute very little to the predicted outcome, indicating that these dimensions are either useless, or the classifier did not find value in their contained information. Consequently, most vowel information can be derived with relatively high accuracy from merely a small subset of dimensions.

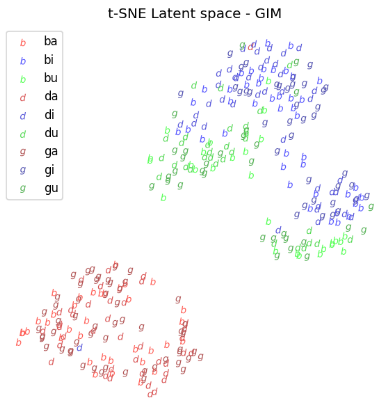
In addition, we observe that certain dimensions contain information for a single vowel, while others contain information for two. Values in the first dimension for instance, have a strong correlation with the vowel "a", while no significant correlation with "i" or "u" appears to be visible. Meanwhile, values in dimension 17 contain information for both "a" and "i", depending on whether the values approach 1 or -1.



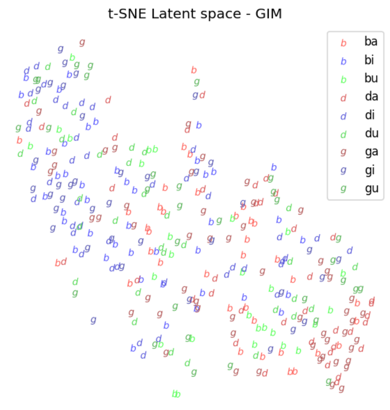
(a) Module 1: $\beta = 0.0035$



(b) Module 2: $\beta = 0.0035$



(c) Module 1: $\beta = 0$



(d) Module 2: $\beta = 0.0035$

Figure 4.3: t-SNE plots (GIM: $\beta = 0$, V-GIM: $\beta = 0.0035$).

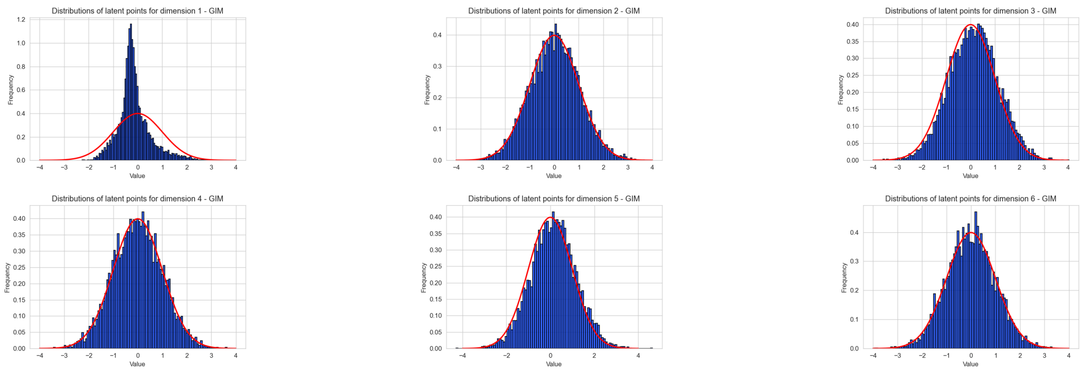


Figure 4.4: Module 1: $\beta = 0.0035$

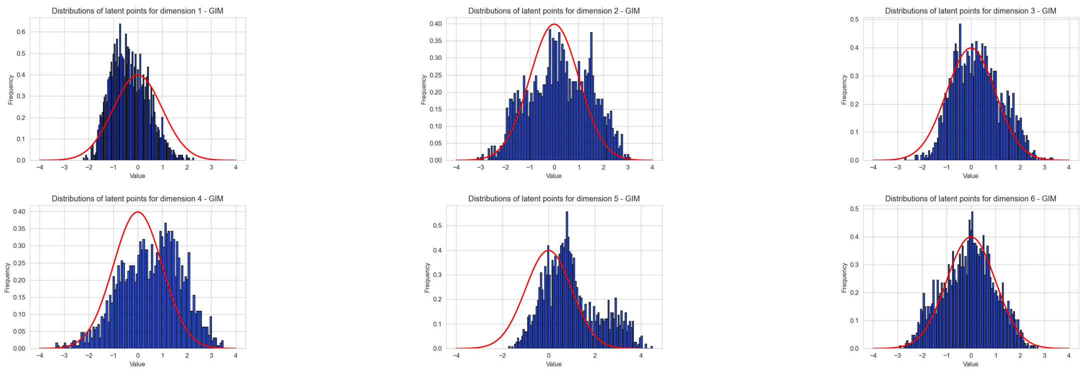


Figure 4.5: Module 2: $\beta = 0.0035$

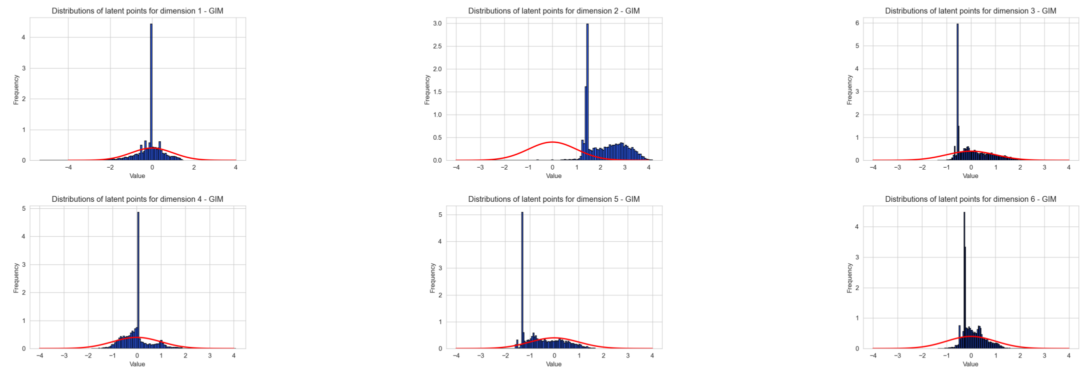


Figure 4.6: Module 1: $\beta = 0$

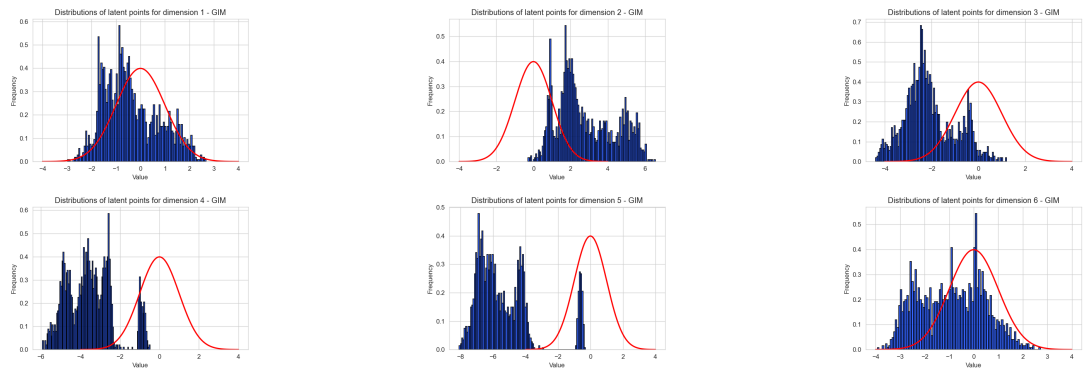


Figure 4.7: Module 2: $\beta = 0$

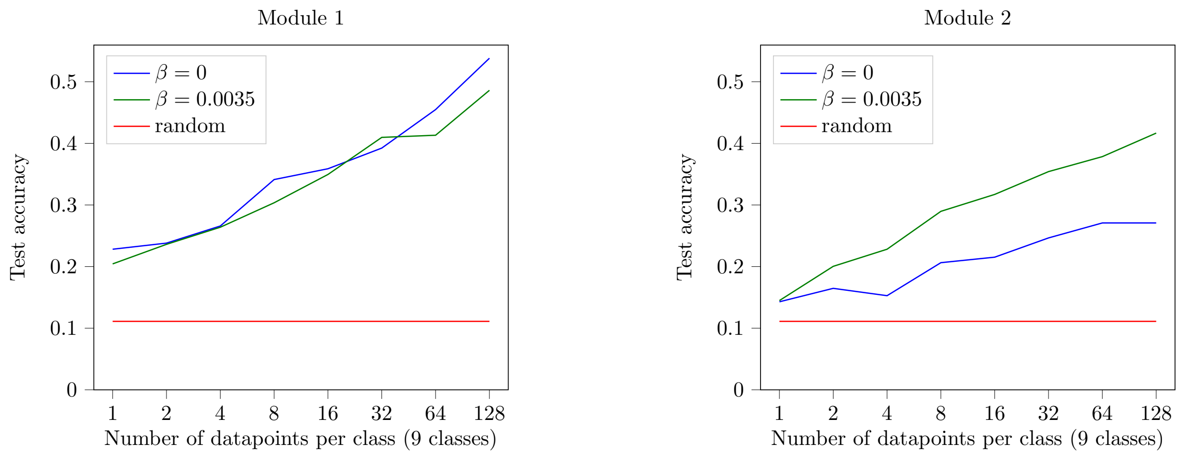


Figure 4.8: Test accuracy for different subset sizes

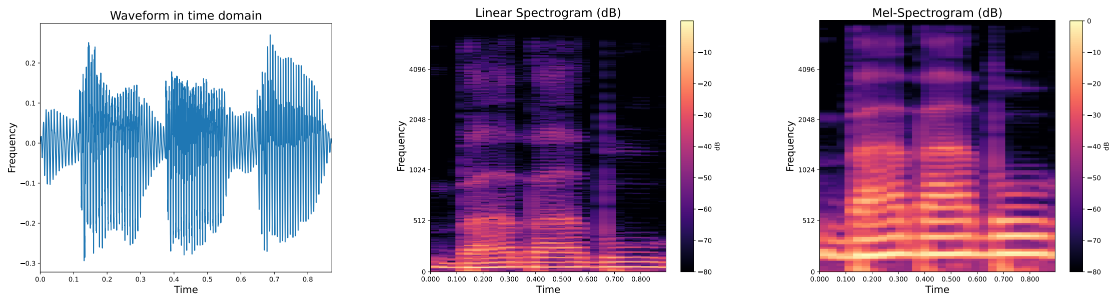


Figure 4.9: Example waveform and spectrogram for "bababu".

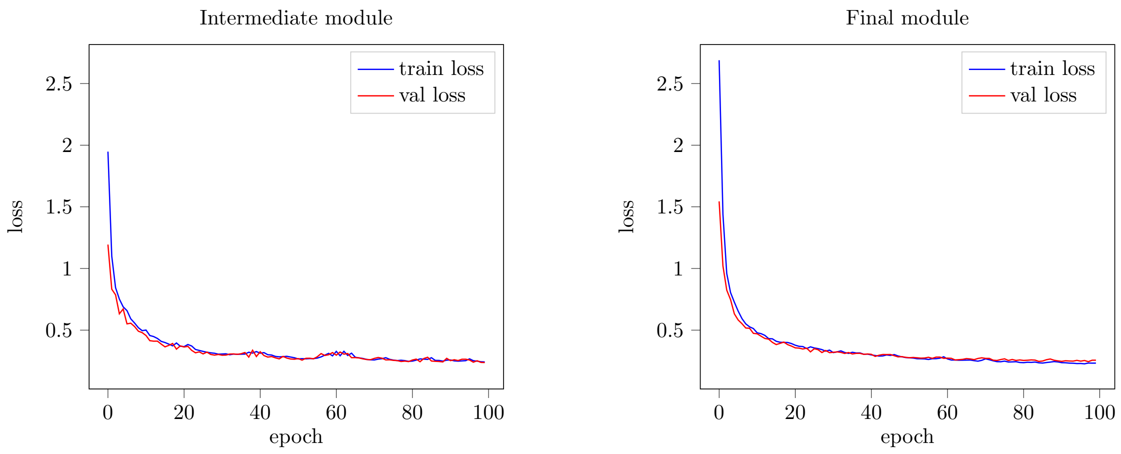


Figure 4.10: Training and validation loss for the two decoders, trained with V-GIM's representations.

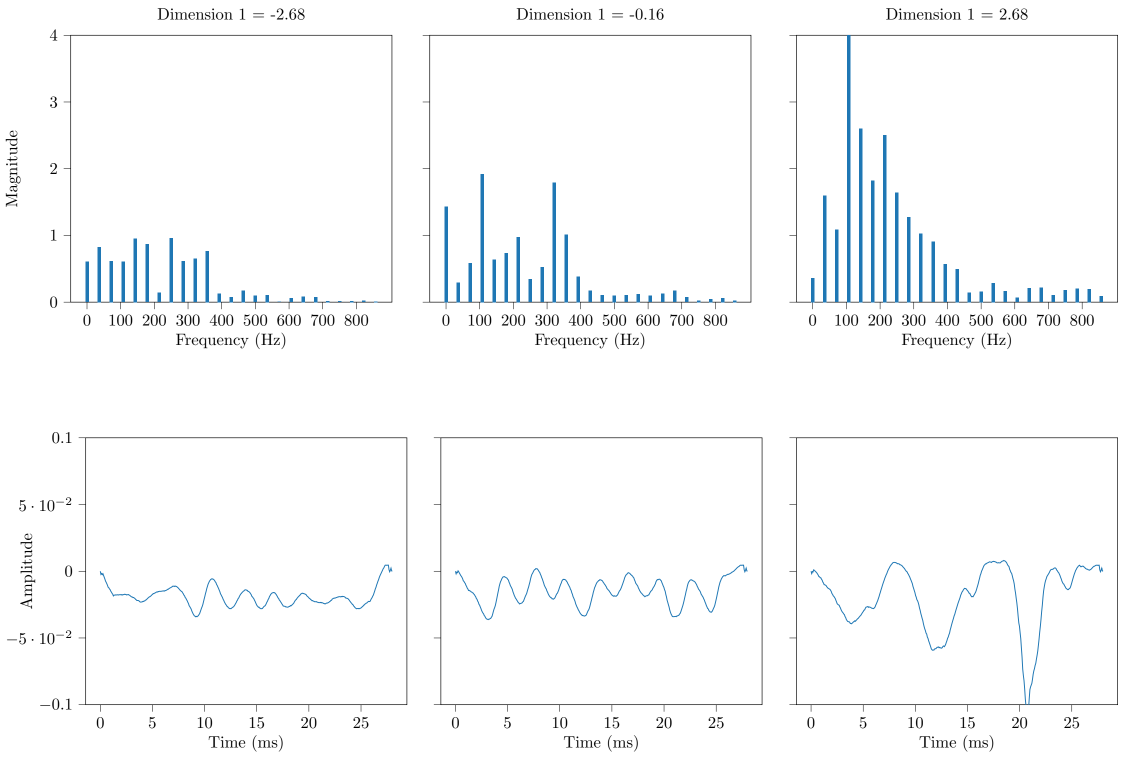


Figure 4.11: Reconstructed speech signal from *intermediate* latent representation. The first dimension is being modified, while other dimensions are fixed at 0. The dimension has influence on the 150Hz frequency band, but also neighbouring ranges.

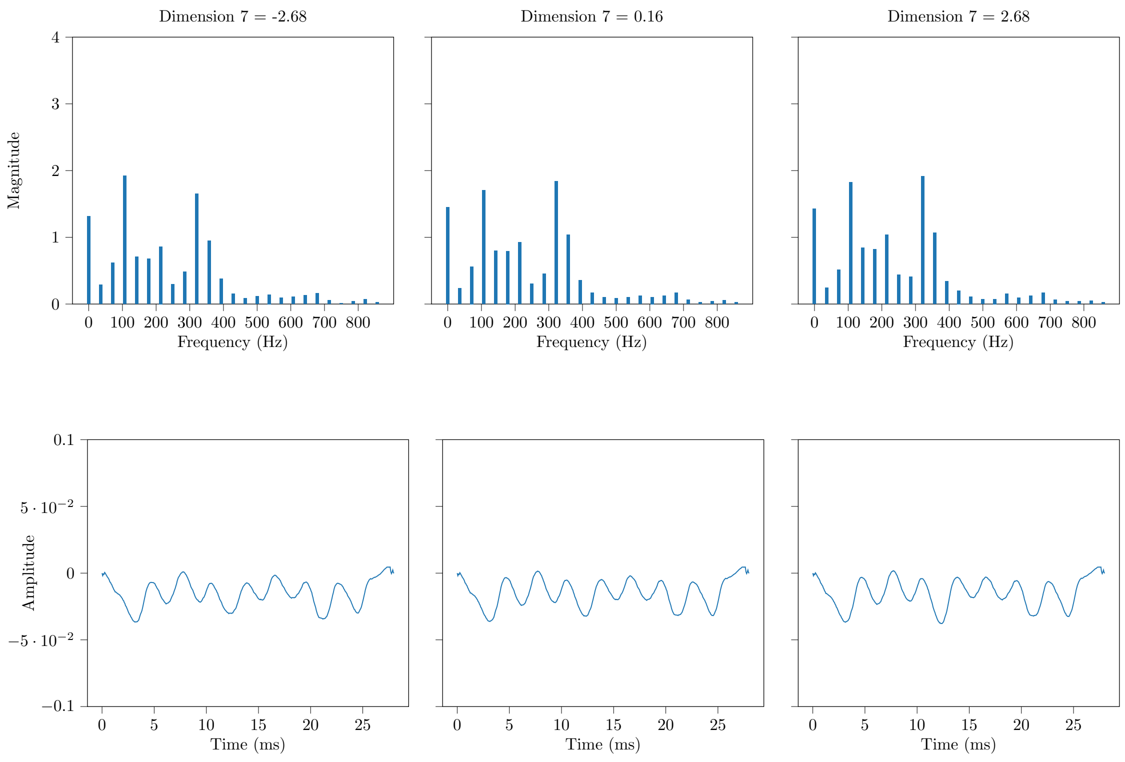


Figure 4.12: Reconstructed speech signal from *intermediate* latent representation. The seventh dimension is being modified, while other dimensions are fixed at 0. We observe no significant differences when adjusting, indicating that the dimension may not capture any information at all.

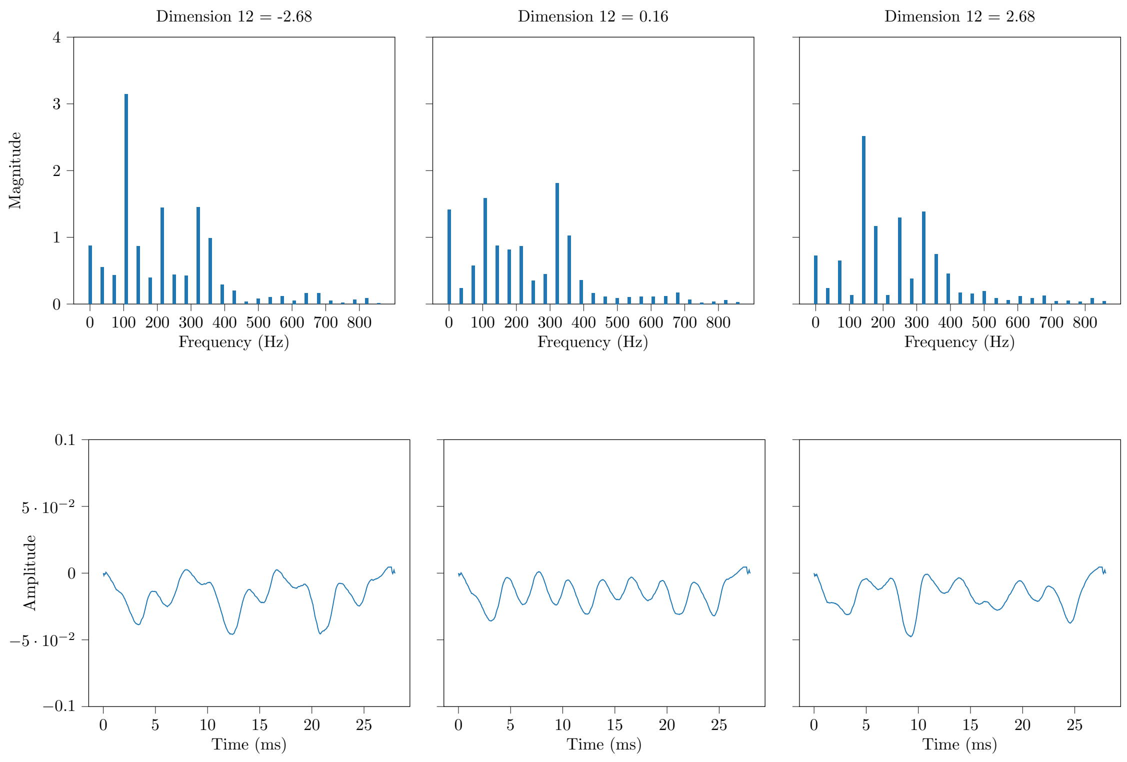


Figure 4.13: Reconstructed speech signal from *intermediate* latent representation. The twelfth dimension is being modified, while other dimensions are fixed at 0. Negative values cause a large spike around 100Hz while positive values fully remove the 100Hz and influences the 150Hz range instead.

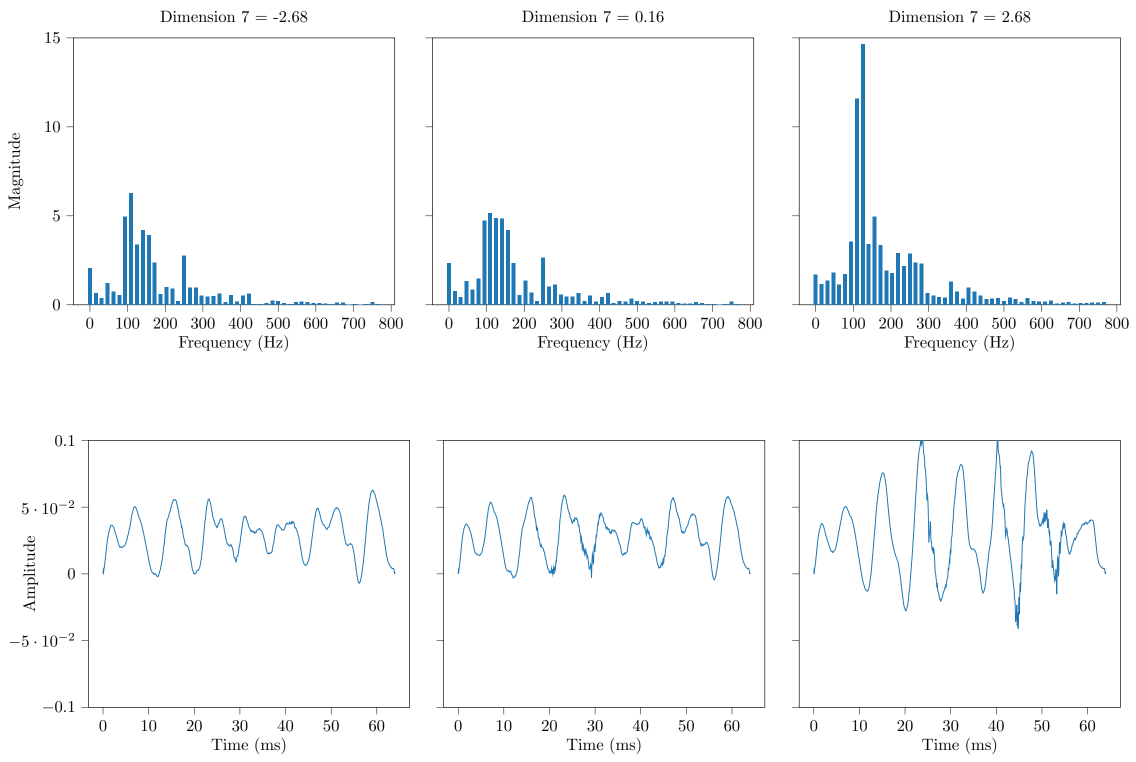


Figure 4.14: Reconstructed speech signal from *final* latent representation. The seventh dimension is being modified, while other dimensions are fixed at 0. The dimension has influence on the 125Hz frequency band, but also neighbouring ranges.

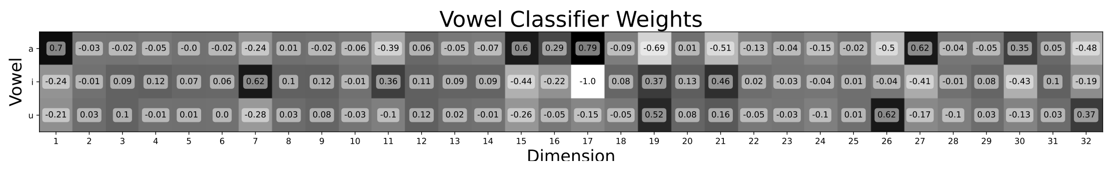


Figure 4.15: Visualisation of weights in the linear vowel classifier. Latent representations from V-GIM's first module are used. The classifier is trained without bias terms.

Chapter 5

Relation to existing work

We have studied the representations obtained by maximising the InfoNCE loss function. We achieved this through the addition of a regularisation term to the loss function, resulting in a space that is easier to analyse and understand. In this chapter, we give an overview of the existing literature that is relevant to our research. We begin with a discussion of representation learning techniques related to mutual information maximisation, and slowly digress into interpretable representations. Furthermore, we give an overview of the existing regularisation techniques used for both improved generalisation and better representations. Finally, we give an overview of alternative priors that have been introduced to the VAE framework in the past decade.

5.1 Mutual information and interpretable representation learning

V-GIM is based on the ideas of mutual information maximisation introduced in CPC and GIM (Löwe et al., 2020; van den Oord et al., 2019). This is achieved by maximising the mutual information between temporally nearby patches, assuming common information between nearby data (Löwe et al., 2020). Similar approaches utilising mutual information maximization for representation learning have also been explored.

Deep InfoMax incorporates an ANN encoder which maximises the mutual information between input and output. This is achieved by incorporating knowledge about locality in the representations, resulting in locally-consistent information across structural locations (Hjelm et al., 2019). Meanwhile, InfoGAN, C-DSVAE and S3VAE each introduce a different flavour on the mutual information maximisation scheme, while obtaining disentangled representations (Chen et al., 2016; Bai et al., 2021; Zhu et al., 2020). In addition, InfoGAN's learning approach results in interpretable representations. Manipulating an individual dimension in the latent space results in changes to a specific feature of the generated data while leaving other features unaffected. Building upon these concepts, Bridge-GAN introduces an intermediate latent space, or "bridge" between text and images. This approach enables the synthesis of interpretable and disentangled representations for text-to-image synthesis (Yuan and Peng, 2020).

Continuing in the line of interpretable representation learning, Timeline uses recurrent neural networks with an attention mechanism to aggregate sequential health data to interpretable representations (Bai et al., 2018). Its interpretability is achieved through analysis of the weights associated with different medical codes. Similarly, Agrawal and Ganapathy (2020) apply relevance weighting on raw speech data, allowing for interpretation of the representations during forward propagation .

Lastly, InfoVAEGAN combines the framework of Generative Adversarial Networks with concepts from VAEs, enabling the learning of interpretable data variations (Ye and Bors, 2021).

5.2 Regularisation

The practice of adding a penalty term to the loss function, known as regularization, has been widely used for various purposes. Kukačka et al. (2017) provide a survey categorising different regularisation techniques, including those that impose constraints on the weights, or on the activations.

Regularisation terms that enforce constraints on the weights typically aim to improve generalisation performance by penalising complexity. Weight decay, for example, achieves this by applying the l^2-norm on the network weights, encouraging smaller weights (Gnecco and Sanguineti, 2009). Another approach described by Kukačka et al. (2017) is weight smoothing which applies l^2-norm to the gradients during training. Weight elimination is similar to weight decay but favours sparse networks (Weigend et al., 1990). Soft weight-sharing clusters weights together, ensuring that weights within a cluster have similar values (Nowlan and Hinton, 1992).

Tian and Zhang (2022) discuss sparse vector-based regularisation, which imposes constraints on the activations. This is useful for applications requiring sparse representations, such as data compression. Continuing in the line of activation regularisation, Tomczak (2016) introduces a regularisation term that encourages activations to maximise entropy, resulting in uncorrelated and disentangled representations. Meanwhile, Wu et al. (2018) introduce a regularisation term putting constraints on the output activations to improve the interpretability of representations.

5.3 Alternative priors and posteriors in VAEs

Taking inspiration from VAEs, V-GIM minimises the KL-divergence with its posterior distribution and a fixed prior $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. This results in a latent space that can be better understood. In recent years, several contributions have been made to VAEs concerning different priors or posteriors.

In VAEs, the posterior $q(\mathbf{z} \mid \mathbf{x}^{(i)})$ serves as an approximation to the true posterior $p(\mathbf{z} \mid \mathbf{x}^{(i)})$ (Odaibo, 2019). This approximate posterior is most commonly chosen as a simple factorised Gaussian for mathematical convenience. However, this is often an oversimplification of the true posterior (Nalisnick et al., 2016). Kingma and Welling (2019) demonstrate that the approximate posterior can be extended to a Gaussian with a full covariance matrix. Additionally, Nalisnick et al. (2016) propose a Gaussian mixture model, which combines several Gaussians, as an approximate posterior, enabling the modelling of multimodal posterior distributions.

Continuing in the exploration of Gaussian mixture models, alternative priors have also been investigated. Guo et al. (2020) and Lee et al. (2021) experiment with Gaussian mixture model priors, resulting in improved performance. Additionally, Tomczak and Welling (2018) introduce VampPrior, choosing the prior as a mixture of variational posteriors. This approach improves performance and mitigates issues related to useless dimensions, which is a well-known problem in VAEs and also observed in V-GIM.

Chapter 6

Discussion

– decaying learing rate: we train using decaying lr, because models must first learn distributions and goes too slow if lr is too small. and a learning rate scheduler ExponentialLR decay rate 0.995

— batch norm: - sindy didn't have issues of batch norm, but believe this is because each module consisted of a single layer, ours contain a number of layers. potentially: outputs from first module change too fast for second module to catch up.

while GIM argues to resolve memory constraints, not entirely true. In fact we even countered the opposite as containing multiple neural networks, each with their own personal loss function (the loss function is based on fk which contains parameters that must be learned), and thus for early layers where the sequence is still long, a lot of memory is required. We went for a compromise on GIM by splitting up the architecture in merely two modules, significantly reducing the memory constraints.

— The second module in GIM clearly doesn't have as much effect. This can be explained because there may not be as much common information anymore between the patches. There may be a source that says that cpc learns low level features, but the second module is supposed to learn more high level features, which cpc may have trouble with? —

Future work: - Related work in VAE shows that gradually increasing regularisation term, results in better disentengledment, while avoiding posterior collapse. could have a kldweight scheduler.

- not constrained solely to InfoNCE loss, the GIM architecture could work for other losses too that allow for greedy optimisation.

- I didn't add an autoregressor as i didn't find a performance benefit. Potentially, with larger architecture could further improve performance.

—- Towards production setting: encodings are thus optimised to be close the standard normal. When in a production environment and new data is given, could in fact have an idea of how well generalisation to the production data: eg via anomaly detection if encodings are too far away from center. = gives automated way of verifying generalisation.

can then maybe see to which data that doesn't generalise well via outliers.

—-

future work: - disentanglement should do more investations

— GIM: Modular training could incrementally increase numb of modules and observe performance increase for downstream tasks. based on this, could find smallest gim architecture depth which satisfies required accuracies.

—- interpretability: most dimensions sensitive around 75 to 150 hz. this is as expected as the adult man speaks around 80 to 180 hz.

— interpretability is only as good as the decoder. if a shitty decoder doesn't construct well, doesn't necessarily give correct conclusions about V-GIM.

— Future work:alternatieve prior (zie related work.)

- Explainability of latents is dependent on the performance of the decoder.
- Intermediate loss function with kld resulted in similar behaviour as batch normalisation. Resulting in faster convergence than without kld.
- We observed no quality loss in the learned representations. Data was equally easily separable.

Bibliography

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985). A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147–169.

Agrawal, P. and Ganapathy, S. (2020). Interpretable Representation Learning for Speech and Audio Signals Based on Relevance Weighting. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2823–2836.

Aremu, O. O., Hyland-Wood, D., and McAree, P. R. (2020). A machine learning approach to circumventing the curse of dimensionality in discontinuous time series machine data. *Reliability Engineering & System Safety*, 195:106706.

Bai, J., Wang, W., and Gomes, C. P. (2021). Contrastively Disentangled Sequential Variational Autoencoder. In *Advances in Neural Information Processing Systems*, volume 34, pages 10105–10118. Curran Associates, Inc.

Bai, T., Zhang, S., Egleston, B. L., and Vucetic, S. (2018). Interpretable Representation Learning for Healthcare via Capturing Disease Progression through Time. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 43–51, New York, NY, USA. Association for Computing Machinery.

Bank, D., Koenigstein, N., and Giryes, R. (2021). Autoencoders.

Barbiero, P., Squillero, G., and Tonda, A. (2020). Modeling Generalization in Machine Learning: A Methodological and Computational Study.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation Learning: A Review and New Perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35:1798–1828.

Bhandari, P. (2020). The Standard Normal Distribution — Calculator, Examples & Uses. https://www.scribbr.com/statistics/standard-normal-distribution/.

Bhargav Lad (2020). Guide to Pytorch Learning Rate Scheduling. https://kaggle.com/code/isbhargav/guide-to-pytorch-learning-rate-scheduling.

Bhati, S., Villalba, J., Żelasko, P., Moro-Velazquez, L., and Dehak, N. (2021). Segmental Contrastive Predictive Coding for Unsupervised Word Segmentation.

Bjorck, N., Gomes, C. P., Selman, B., and Weinberger, K. Q. (2018). Understanding Batch Normalization. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Braun, S. and Tashev, I. (2020). A consolidated view of loss functions for supervised deep learning-based speech enhancement.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets.

Chung, Y., Haas, P. J., Upfal, E., and Kraska, T. (2019). Unknown Examples & Machine Learning Model Generalization.

Cinelli, L. P., Marins, M. A., Barros da Silva, E. A., and Netto, S. L. (2021). *Variational Methods for Machine Learning with Applications to Deep Networks*. Springer International Publishing, Cham, 1st ed. 2021 edition edition.

Cover, T. M. and Thomas, J. A. (2006). *Elements of Information Theory 2nd Edition*. Wiley-Interscience, Hoboken, N.J, 2nd edition edition.

Data Science Courses (2017). Ali Ghodsi, Lec : Deep Learning, Variational Autoencoder, Oct 12 2017 [Lect 6.2].

David Foster (2023). 3. Variational Autoencoders. In *Generative Deep Learning, 2nd Edition*. O'Reilly Media, Inc., second edition.

de Haan, P. and Löwe, S. (2021). Contrastive Predictive Coding for Anomaly Detection.

Deldari, S., Smith, D. V., Xue, H., and Salim, F. D. (2021). Time Series Change Point Detection with Self-Supervised Contrastive Predictive Coding. In *Proceedings of the Web Conference 2021*, WWW '21, pages 3124–3135, New York, NY, USA. Association for Computing Machinery.

Doersch, C. (2021). Tutorial on Variational Autoencoders.

Gnecco, G. and Sanguineti, M. (2009). The weight-decay technique in learning from data: An optimization point of view. *Computational Management Science*, 6:53–79.

Grossutti, M., D'Amico, J., Quintal, J., MacFarlane, H., Quirk, A., and Dutcher, J. R. (2022). Deep Learning and Infrared Spectroscopy: Representation Learning with a Beta-Variational Autoencoder. *The Journal of Physical Chemistry Letters*, 13(25):5787–5793.

Guo, C., Zhou, J., Chen, H., Ying, N., Zhang, J., and Zhou, D. (2020). Variational Autoencoder With Optimizing Gaussian Mixture Model Priors. *IEEE Access*, 8:43992–44005.

Henaff, O. (2020). Data-Efficient Image Recognition with Contrastive Predictive Coding. In *Proceedings of the 37th International Conference on Machine Learning*, pages 4182–4192. PMLR.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2022). Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *International Conference on Learning Representations*.

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. (2019). Learning deep representations by mutual information estimation and maximization.

Ho, Y. and Wookey, S. (2020). The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling. *IEEE Access*, 8:4806–4813.

Hu, Z., Zhang, J., and Ge, Y. (2021). Handling Vanishing Gradient Problem Using Artificial Derivative. *IEEE Access*, 9:22371–22377.

Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.

Jain, A., Mao, J., and Mohiuddin, K. (1996). Artificial neural networks: A tutorial. *Computer*, 29(3):31–44.

Karagiannakos, S. (2018). How to Generate Images using Autoencoders. https://theaisummer.com/Autoencoder/.

Kingma, D. P. and Welling, M. (2019). An Introduction to Variational Autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392.

Kingma, D. P. and Welling, M. (2022). Auto-Encoding Variational Bayes.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Krogh, A. (2008). What are artificial neural networks? *Nature Biotechnology*, 26(2):195–197.

Kukačka, J., Golkov, V., and Cremers, D. (2017). Regularization for Deep Learning: A Taxonomy.

Le-Khac, P. H., Healy, G., and Smeaton, A. F. (2020). Contrastive Representation Learning: A Framework and Review. *IEEE Access*, 8:193907–193934.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

LeCun, Y., Bottou, L., Orr, G. B., and Müller, K. R. (1998). Efficient BackProp. In Orr, G. B. and Müller, K.-R., editors, *Neural Networks: Tricks of the Trade*, Lecture Notes in Computer Science, pages 9–50. Springer, Berlin, Heidelberg.

Lee, D. B., Min, D., Lee, S., and Hwang, S. J. (2021). Meta-GMVAE: Mixture of Gaussian VAE for Unsupervised Meta-Learning. In *International Conference on Learning Representations*.

Li, A., Peng, R., Zheng, C., and Li, X. (2020). A Supervised Speech Enhancement Approach with Residual Noise Control for Voice Communication. *Applied Sciences*, 10(8):2894.

Löwe, S., O'Connor, P., and Veeling, B. S. (2020). Putting An End to End-to-End: Gradient-Isolated Learning of Representations.

Lu, M. Y., Chen, R. J., Wang, J., Dillon, D., and Mahmood, F. (2019). Semi-Supervised Histology Classification using Deep Multiple Instance Learning and Contrastive Predictive Coding.

Lucas, J., Tucker, G., Grosse, R., and Norouzi, M. (2022). Understanding Posterior Collapse in Generative Latent Variable Models.

Marblestone, A. H., Wayne, G., and Kording, K. P. (2016). Toward an Integration of Deep Learning and Neuroscience. *Frontiers in Computational Neuroscience*, 10.

Meihan Wang (2019). *Speech Representation Learning without Backpropagation.* PhD thesis, Vrije Universiteit Brussel.

Nalisnick, E. T., Hertel, L., and Smyth, P. (2016). Approximate Inference for Deep Latent Gaussian Mixtures.

Neyshabur, B., Bhojanapalli, S., Mcallester, D., and Srebro, N. (2017). Exploring Generalization in Deep Learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Nowlan, S. J. and Hinton, G. E. (1992). Simplifying Neural Networks by Soft Weight-Sharing. *Neural Computation*, 4(4):473–493.

Odaibo, S. (2019). Tutorial: Deriving the Standard Variational Autoencoder (VAE) Loss Function.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in PyTorch.

Przybyszewski, P., Dziewiatkowski, S., Jaszczur, S., Smiech, M., and Szczuka, M. (2017). Use of domain knowledge and feature engineering in helping AI to play Hearthstone. In *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 143–148.

Radkoff, E. (2021). Loss Functions in Audio ML. https://soundsandwords.io//audio-loss-functions/.

Rao, S. S. (2020). Understanding the Gradient-Isolated Learning of Representations and intuition to the Greedy. . . .

Re, D. E., O'Connor, J. J. M., Bennett, P. J., and Feinberg, D. R. (2012). Preferences for Very Low and Very High Voice Pitch in Humans. *PLoS ONE*, 7(3):e32719.

Rumelhart, D., Hinton, G., and Williams, R. (1988). Learning Internal Representations by Error Propagation. In *Readings in Cognitive Science*, pages 399–421. Elsevier.

Rust, J. (1997). Using Randomization to Break the Curse of Dimensionality. *Econometrica*, 65(3):487–516.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. (2018). How Does Batch Normalization Help Optimization? In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Shah, R. (2020). [AN #92]: Learning good representations with contrastive predictive coding.

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerrv-Ryan, R., Saurous, R. A., Agiomvrgiannakis, Y., and Wu, Y. (2018). Natural TTS Synthesis by Conditioning Wavenet on MEL Spectrogram Predictions. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4779–4783.

Stacke, K., Lundström, C., Unger, J., and Eilertsen, G. (2020). Evaluation of Contrastive Predictive Coding for Histopathology Applications. In *Proceedings of the Machine Learning for Health NeurIPS Workshop*, pages 328–340. PMLR.

Stuart Russell and Peter Norvig (2022). *Artificial Intelligence: A Modern Approach, 4th US Ed.* Fourth edition.

Tian, Y. and Zhang, Y. (2022). A comprehensive survey on regularization strategies in machine learning. *Information Fusion*, 80:146–166.

Tomczak, J. and Welling, M. (2018). VAE with a VampPrior. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, pages 1214–1223. PMLR.

Tomczak, J. M. (2016). Learning Informative Features from Restricted Boltzmann Machines. *Neural Processing Letters*, 44(3):735–750.

Tschannen, M., Bachem, O., and Lucic, M. (2018). Recent Advances in Autoencoder-Based Representation Learning.

Vali, A., Comai, S., and Matteucci, M. (2020). Deep Learning for Land Use and Land Cover Classification Based on Hyperspectral and Multispectral Earth Observation Data: A Review. *Remote Sensing*, 12(15):2495.

van den Oord, A., Li, Y., and Vinyals, O. (2019). Representation Learning with Contrastive Predictive Coding.

van der Maaten, L. and Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605.

Volodymyr Kuleshov and Stefano Ermon (2023). The variational auto-encoder. https://ermongroup.github.io/cs228-notes/extras/vae/.

Wei, R. and Mahmood, A. (2021). Recent Advances in Variational Autoencoders With Representation Learning for Biomedical Informatics: A Survey. *IEEE Access*, 9:4939–4956.

Wei, W., Hu, X., Liu, H., Zhou, M., and Song, Y. (2021). Towards Integration of Domain Knowledge-Guided Feature Engineering and Deep Feature Learning in Surface Electromyography-Based Hand Movement Recognition. *Computational Intelligence and Neuroscience*, 2021:e4454648.

Weigend, A., Rumelhart, D., and Huberman, B. (1990). Generalization by Weight-Elimination with Application to Forecasting. In *Advances in Neural Information Processing Systems*, volume 3. Morgan-Kaufmann.

Wu, C., Gales, M. J. F., Ragni, A., Karanasou, P., and Sim, K. C. (2018). Improving Interpretability and Regularization in Deep Learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(2):256–265.

Ye, F. and Bors, A. G. (2021). InfoVAEGAN: Learning Joint Interpretable Representations by Information Maximization and Maximum Likelihood. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 749–753.

Yuan, M. and Peng, Y. (2020). Bridge-GAN: Interpretable Representation Learning for Text-to-Image Synthesis. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(11):4258–4268.

Zhang, X., Xing, Y., Sun, K., and Guo, Y. (2021). OmiEmbed: A Unified Multi-Task Deep Learning Framework for Multi-Omics Data. *Cancers*, 13(12):3047.

Zhang, Z. (2018). Artificial Neural Network. In Zhang, Z., editor, *Multivariate Time Series Analysis in Climate and Environmental Research*, pages 1–35. Springer International Publishing, Cham.

Zhang, Z. and Tao, D. (2012). Slow Feature Analysis for Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):436–450.

Zheng, A. and Casari, A. (2018). *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. "O'Reilly Media, Inc.".

Zhu, Y., Min, M. R., Kadav, A., and Graf, H. P. (2020). S3VAE: Self-Supervised Sequential VAE for Representation Disentanglement and Data Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6538–6547.

Appendices

Appendix A

Syllable classification through histogram segmentation

probabily bs: During inference, (in this context obtaining the latent representations for our input signals), depending on the length of the input signal, the length of the output latent representation will differ. If we wish to look at how separable latent representations are for syllables, the length can be variable. Some input sounds could be 6,600 samples, while others 8,800 samples. We therefore pad the syllables with zeroes in front and end of the signal, to obtain fixed length of equal to that of the longest syllable; 8,800 samples.

Training happens on longer data samples, and every **X** epochs t-SNE visualisations are made to observe evolutional of dis-entanglement.

Since the recordings are very consistent in loudness and are noise free, we can split up the files per syllable, obtaining three files per original sound (one for each syllable).

A sliding window is used of size 0.02 seconds. With a sample rate of 22050, this corresponds to roughly 500 samples per window. The maximum is computed for each window. Speech signals can then be split up when a severe dip happens in the signal. Regions where the amplitude is greater than 0.2 are considered **klinkers**, the regions with with lower values are considered **medeklinkers**. Apart from a few edge cases, this technique worked well enough for this purpose. In those cases, the splitting points closest to the one-third and two-third splitting points were considered.

ERR: OUDE AFBEELDINGEN ZIJN WEG

note: we do need a hard threshold which is based on the signal's intentisity level. One could consider the alternative approach of looking at the gradient at each point and selecting the points with largest negative gradient. This will work in many cases, however, not for temporal envelops which gradually move towards zero, s.t: A.1. Instead we use a dynamic threshold. This threshold is computed by creating transforming the signal into bins of 90'th percentile, creating a histogram of the single signal and applying otsu's image segmentation algorithm to obtain the threshold of that single audio sample. We also tried directly applying otsu to the moving average and maximum of the bins. This either gave a threshold that was too small or too large. the 90th percent resulted in an acceptable compromise.

Example where the explained strategy does not work:

Reference images for in the text:

audio padded to maximum length. (added zeros in front and back)

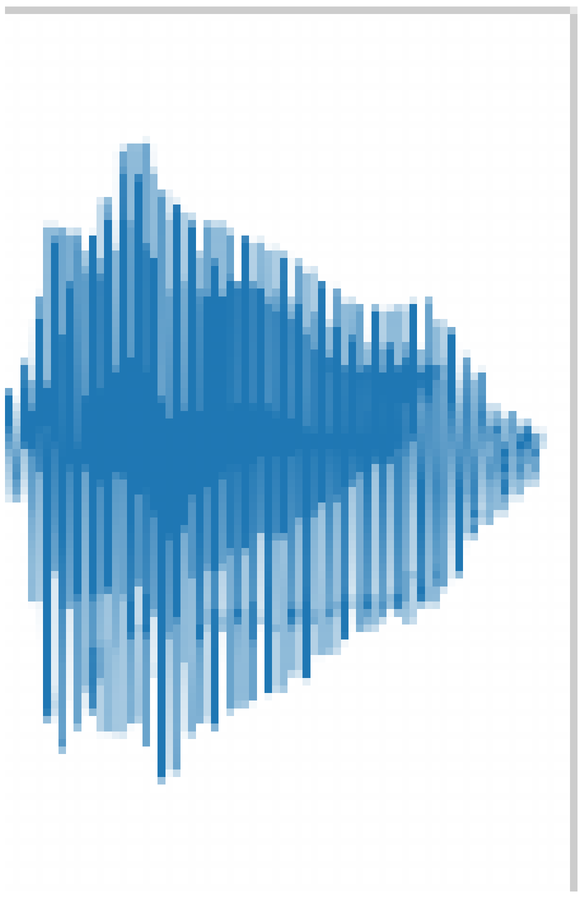


Figure A.1

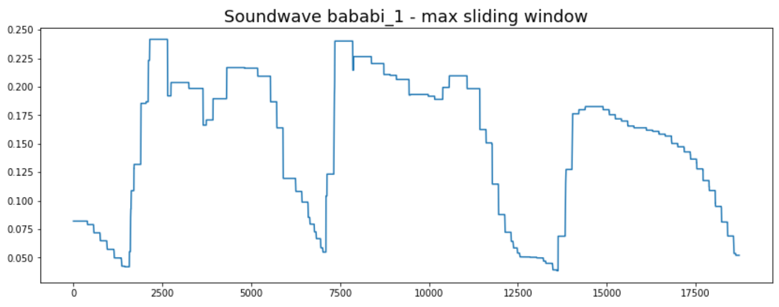


Figure A.2

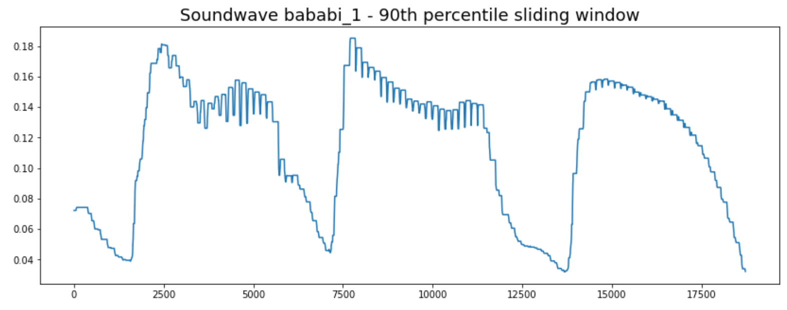


Figure A.3

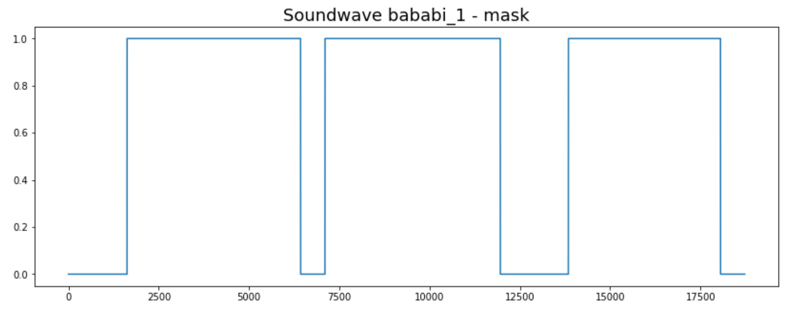


Figure A.4

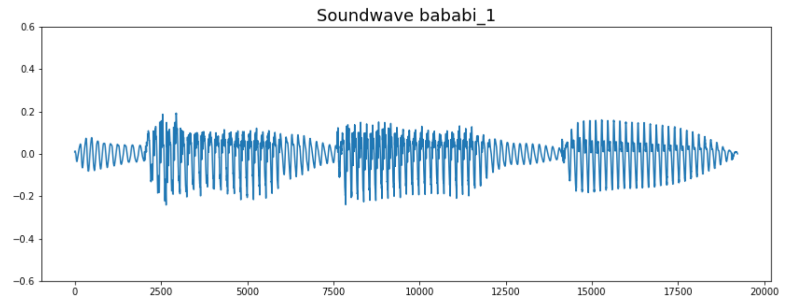


Figure A.5

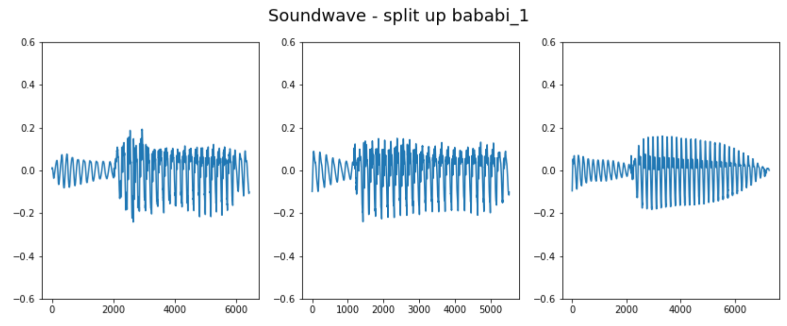


Figure A.6

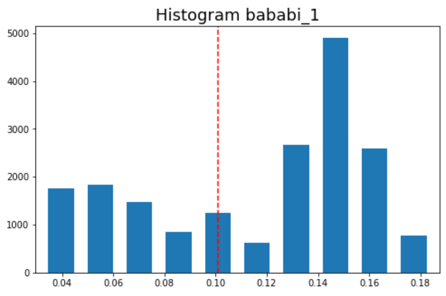

Histogram bababi_1
5000
4000
3000
2000
1000
0
0.04
0.06
0.08
0.10
0.12
0.14
0.16
0.18

Figure A.7

Appendix B

Some more appendix

B.1 GIM: Activations visualisations

thought for later: its actually weird i was able to play enc as audio as enc is 512 x something so huh? that means that a lot of info is already in first channel? what do other 511 channels then contain? """ Observations: First layer decoded still contains the same sound, but with some added noise (could be because decoder hasn't trained very). However, the encoded first layer, still contains the exact sound as the original sound. It is however downsampled a lot -¿ from 16khz to 3khz """ thought for later: its actually weird i was able to play enc as audio as enc is 512 x something so huh? that means that a lot of info is already in first channel? what do other 511 channels then contain?

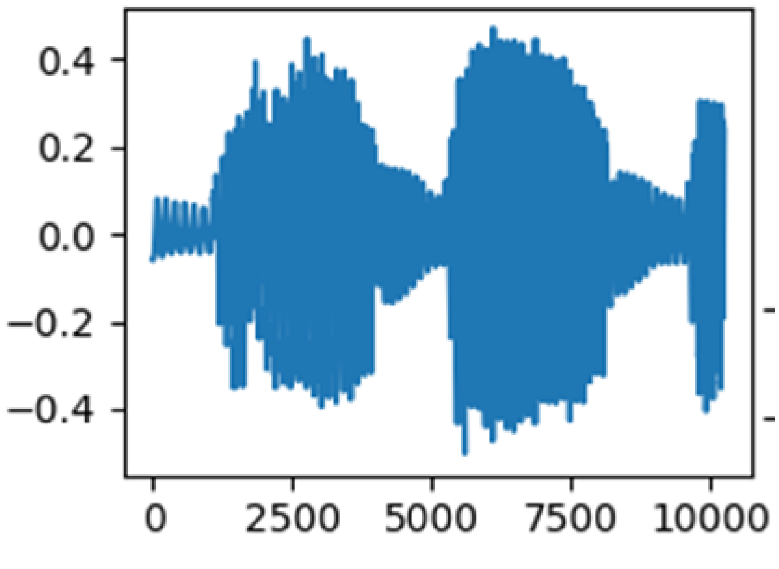


Figure B.1: "BA-BA-BA" time domain

No batch normalisation, so although channels appear to have larger activations than other channels, size of activation does not really say anything about information. eg activations 0.01 could still contain more information than 3.0 activation.

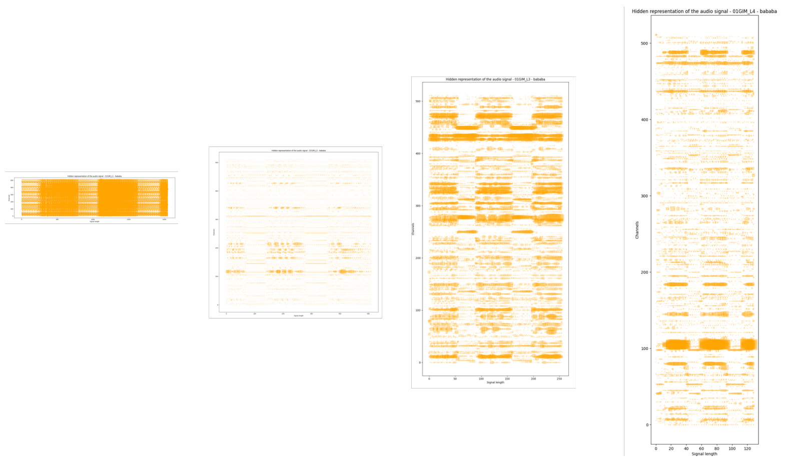


Figure B.2: Activations of the sound ”BA-BA-BA” through GIM

Since the activations from convolutional neural networks, the order is still maintained. Hence, can align activations with original signal.

Observations in latent representations:

Layer 1: The activations of the first decoder still contain a lot of similarity with the original signal, in terms of structure. There is a lot of redundant data within the representation. Eg: the one channel could be replied

Layer 2

Layer 3:

Layer 4: Still notices multiple channels which have high activations when signal is has high amplitudes and small activations when amplitude is low.

Also activations which are high when volume is low. −¿ indicates that certain kernel weights are sensitive for **”klinkers”** and other kernels for **medeklinkers**. see B.3.

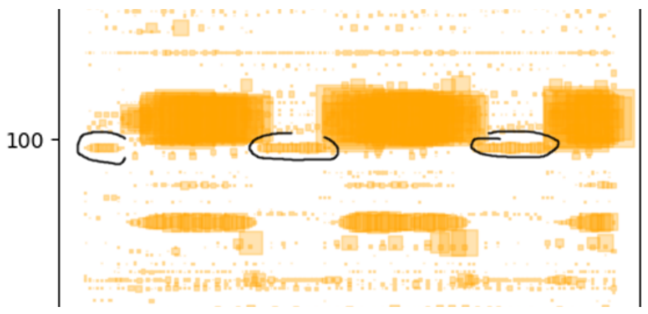


Figure B.3: zoomed in

Observe that activations happen in clusters/sequences. So it is usually a patch of signal samples that cause high activations. This could for instance indicate that both kernels are sensitive for the **medeklinker** ”b”, but sensitive for different features. eg the letter B has spoken sound ”buh”. so maybe one is sensitive for ”b” and other for ”uh”.

Figure B.4 also nicely shows how different channels have clusters of activations at slightly different times.

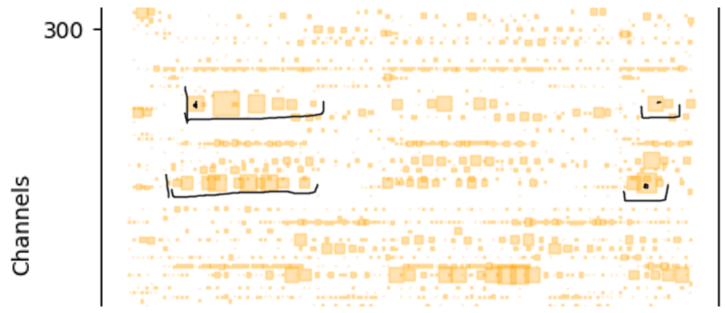


Figure B.4: Zoomed in