UTRECHT UNIVERSITY

MASTER'S THESIS

Mapping Neuronal Activity in the Brain Using Hemodynamic Activity to Quantify a Disconnect Between Normal and Abnormal Coupling

Author:
Ing. Frans IRGOLITSCH

Institutional Supervisor:
Dr. Frédéric LESAGE

Supervisors:
Dr. Martijn MULDER
Dr. Ben HARVEY

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Graduate School of Natural Sciences

July 11, 2022

UTRECHT UNIVERSITY

Abstract

Faculty of Science

Graduate School of Natural Sciences

Master of Science

Mapping Neuronal Activity in the Brain Using Hemodynamic Activity to Quantify a Disconnect Between Normal and Abnormal Coupling

by Ing. Frans IRGOLITSCH

In this project we tried to develop a machine learning method for learning the neuronal activation patterns based on hemodynamic activity in the mouse brain, both represented as functional connectivity optical intrinsic signal (fcOIS) imaging. The goal of this method is to quantify the hemodynamic coupling in the brain and use it to look for a disconnect in brains that are subject to pathology or aged.

For this, two deep learning architectures were developed which are both suited to recreating data based on an encoded input. These architectures are the variational autoencoder (VAE) and the variational autoencoding generative adversarial networks (VAE-GAN). These architectures use the hemodynamic data to create a reconstruction of the neuronal activity. As the fcOIS data consist of data which have both spatial and temporal elements, a combination of convolutional and recurrent layers is used in the architectures to try and learn the features in the data.

Unfortunately, the architectures did not produce satisfactory results. The samples generated by the VAE suffered from mode collapse and the VAE-GAN produced blurry results. The samples generated by the networks are not suited for further investigation into the quantification of the coupling. We do show, however, that the VAE performs slightly better in this task with the current parameters. Improvements might be made by obtaining more data from the mice and by pre-training parts of the architectures used.

Preface

Here I present my thesis titled "Mapping neuronal activity in the brain using hemodynamic activity to quantify a disconnect between normal and abnormal coupling" that details my work from September 2021 until July 2022 at the Laboratoire d'imagerie optique et moléculaire at Polytechnique Montréal, Canada. This thesis is submitted for the fulfilment of the requirements for my Master's degree in Artificial Intelligence.

During the past two years of my Master's at Utrecht University I took my first steps into the world of academia and I can now say that I feel like I have found a place in which I can do what I feel is the most satisfactory to me: discovering new insights and contributing to the sciences. Even though, I wished I could have spend on campus, which was unfortunately not possible due to the global pandemic, I still had a good time studying and meeting a lot of students who have taught me many things.

Furthermore, even though my move to Canada was delayed, and I had to start my thesis proposal in the Netherlands, I have enjoyed my time in Montréal immensely. It has been a great adventure and I got to experience a lot of wonderful things there. I am glad to say that it already feels a bit like home and I am very much looking forward to what the future will bring.

More importantly, would like to express my sincere gratitude to my supervisor Frédéric Lesage. I would like to thank you for teaching me much about the context of this project, which I had very limited experience in (and to some degree still have) when I started this project. I really appreciate that you always make time to discuss the project and answer any questions I have. I am looking forward to our collaboration in the future.

I also would like to thank all my colleagues at Polytchnique Montréal for making me feel welcome and providing me with a lot of insight in everything I was working on and giving feedback and ideas for my research. I especially would like to thank Marleen Bakker and Samuel Bélanger for collecting and processing the data used in this project, as it was of excellent quality, making my work a lot easier.

Finally, I would like to thank my friends and family for keeping me sane throughout the process of moving to Canada and writing this thesis. Without all of your support, this thesis would not have been finished.

With all that said, I hope you enjoy reading my Master's thesis,

Frans Irgolitsch

Contents

List of Figures

List of Tables

1 Introduction

In humans, brain function is imaged using functional magnetic resonance imaging (fMRI). FMRI measures the blood oxygen level dependent (BOLD) signal, which is the difference between how de-oxygenated blood (deoxyhemoglobin, HbR) in the brain responds to the MR signal compared to oxygenated blood (oxyhemoglobin, HbO) with MRI imaging, due to their different electromagnetic properties [1]. As the blood transport the oxygen needed for the activations of neurons, this creates increase in HbR when the neurons have fired as the oxygen is consumed and conversely an increase in HbO which carries the oxygen which aids the repolarisation of the neurons. The hemodynamic activity in the brain is thus used as a proxy for measuring the neuronal activity in the brain. The use of fMRI on small animals, like mice, is difficult however. The size of the animal requires specialised MRI equipment, making research on these animals difficult and expensive [2].

As an alternative to fMRI, functional connectivity optical intrinsic signal (fcOIS) imaging can be used. This technique records the brain using microscopy, and thus does not require expensive machinery like fMRI. The intrinsic signals are the measurement of the activity in the microvasculature in the brain cortex. FcOIS is not limited to just recording the hemodynaimcs of the brain, but can also be used to record neuronal activity directly. By being able to record the hemodynamic and neuronal activity in the mouse, we can create models in mice in regards to the function of the BOLD signal, which we can then apply to techniques used in humans.[3, 4]. Furthermore, fcOIS is an invasive procedure, requiring surgery and injection of a virus to modify the composition of the neurons. This is not desirable and humans, and therefore fcOIS is more suited in animal research.

To record the neuronal activity in the brain, several options are available. One of these is through the use of green fluorescent proteins (GFPs). GFPs emit a green fluorescence light when excited by a trigger, for example an influx of ions, like with GCaMP6. GCaMP6 is a an ultra-sensitive protein calcium (Ca^{2+}) sensor, which can be implemented in animals using either dye loading, a viral vector or through transgenics [5, 6, 7]. This allows for the recording of the entire brain cortex of awake mice. Since the mouse skull is very thin, the brain activity can be recorded directly through the skull after the skin is removed. The activity is then measured by recording the activation of the calcium sensor, which itself is activated by neuronal activity. During the depolarisation of the neuron, the concentration of Ca^{2+} in the neuron increases due to the opening of voltage-gated ion-channels, letting Ca^{2+} into the neuron. This leads to the fluorescence of the GCaMP6 sensor, which is then recorded using fcOIS and shows which part of the brain is active [6].

FcOIS can be used to record the resting state networks in the brain. The resting state network show the connectivity in the brain in the absence of external stimuli. These networks have become more important as they become altered with certain medical conditions, like Alzheimer's. Being able to record default networks, or seeing the

changes caused by neurological dysfunctions could help better understand these conditions [8].

With fMRI imaging, the assumption is made that the hemodynamic activity and the electrical activity in the brain are related (the neuro-vascular coupling), insofar that one can be predicted form the other. Aging and (vascular) diseases are believed to have impact on this coupling. Understanding the impact of these causes is important to correctly interpret data in both animal and human models. If the assumption of conserved neuro-vascular coupling is not correct, this will lead to erroneous interpretation of fMRI data in humans with these afflictions [9]. Developing a method that can quantify neurovascular coupling and investigate its changes with either aging or pathology would enable a better understanding of the coupling, and potentially a better interpretation of human data in populations that are subject to pathology or aged. Due to the size and complexity of the data, machine learning might pose an answer to this problem, and a method can be developed that can help in quantifying this coupling.

1.1 Thesis Objective

The aim of this research will be to develop a method using machine learning to quantify the mapping between the electrical activity and the hemodynamics in the brain. This mapping will be based on data gathered from mice exhibiting GCaMP6 via viral injection. Both neuronal (calcium) and hemodynamic (HbO and HbR) spontaneous activity will be recorded. Due to earlier success with electroencephalographic and functional near-infrared spectroscopy data, where the resting state of the fNIRS signal was predicted using the EEG data, and the complexity of the data, a deep neural network architecture will be preferred [10]. In the data used for this research however, spatial features are also available. Because of this, research will be focused on architectures containing convolutional networks to learn these spatial features. Since the data also has temporal features, some way to model a sequence will also be needed. 3D convolutional networks have shown success in learning both these features, and are also promising in this research [11].

The mapping we want to create would allow us to understand which part of the neural activity, recorded by the calcium indicators, drives the hemodynamic activity. The main research question we would like to answer is:

How could artificial intelligence techniques be used to quantify a disconnect between the normal coupling, between hemodynamics and calcium, and abnormal/disease or age-related coupling?

The following sub-questions will support the main research questions:

1. *Which machine learning techniques can be used to interpret the data and learn the coupling?*
2. *What neural network architectures are potentially viable for learning the coupling?*
3. *What processing is needed for the data to be used in the network while maintaining its integrity?*
4. *How can the results generated by the network be validated?*
5. *Out of the HbO and HbR data, which is the better predictor for the neuronal activity?*

6. *How can the neurovascular coupling be quantified using the results generated by the learned models?*

The first two questions form the theoretical background for the design of the models and identify potential machine learning techniques to use. Question three answers how the data that is available can be adapted for use in the training of the models and how the integrity of the data can be maintained. The next two questions give insight how the results of the network can be validated and which input data is better for generating said results. Finally, we answer the question how the neurovascular coupling can be quantified from the results generated by the models.

Answering these questions would help to understand the impact of the afflictions on resting state networks and help confirm the assumption of conserved neuro-vascular coupling.

1.2 Scope

This research is conducted over 30 weeks at Polytechnique Montreal within Laboratoire d'imagerie optique et moléculaire (Laboratory of optical and molecular imaging, LIOM). The project focuses on the analysis of data, produced by other members of the laboratory, using machine learning. Due to time constraints, deep learning networks are created using known architectures which can generate examples based on the data mentioned before. No new data will be acquired, and existing data will be processed as little as possible to preserve the integrity.

1.3 Chapter Overview

This thesis is structured into two main sections. The first is the literature review which is covered by Chapter 2. This forms the theoretical background which is further expanded upon in the following chapters. The next section will be about the application of the theory discussed in Chapter 2 and will cover the remaining chapters. In Chapter 3, the data is discussed. This covers the acquisition of the data, the subsequent processing into the data files used by this research and the final preparation before it is used for training. Next, in Chapter 4, the neural network architectures and network training method are discussed which is followed by the analysis of the models (Chapter 5) the results they produce. Finally, the results are summarised in Chapter 6 which is followed by the discussion, potential future work and the answer to the main research question in Chapter 7.

2 Literature Review

The aim of this research is to use machine learning to build a model that learns the mapping between the hemodynamic and the neuronal activity. Due to the complexity and volume of the available resting state data, a machine learning approach is preferred. We have either the option to choose for conventional machine learning, where features are designed manually and the distribution is learned, or for a deep neural network, where through a series of layers with multiple nodes the features in the data are found automatically whereupon the distribution is learned.

2.1 Machine Learning

Conventional machine learning (i.e. machine learning without the use of deep artificial neural networks) has in the past been good a finding complex patterns in data. This has always required good feature extractors, which require expert knowledge and manual work before the machine learning techniques can be used.

With these features models like Linear Regression, Support Vector Machines and K Nearest Neighbours can be trained either for classification or for regression. Performance with classification models has traditionally been decent with well designed features [12]. Regression is very sensitive to the distribution of the original data. Linear models will not work on data which can not be linearly separated, requiring more complex models like SVMs [13].

Images add another level of complexity, as we are often not sure what features we should look for in data and the features themselves are more complex. This makes using traditional machine learning for processing images harder, as we would need to design features which would capture the data distribution well. Ideally we would want a way for the models to learn the features themselves.

Representation learning gives us the tools to create machine learning systems which are able to do that. These systems take the input data and build representations of them, preserving the important features. By stacking these features after each other, they can be combined to learn complicated features in the data [14].

Traditional machine learning does not have any model capable of this, however. Furthermore, there are no models which are able to generate complex data distributions [13].

2.2 Deep Learning

In recent years, deep learning has become a large field within artificial intelligence research. It has shown itself to be a good method to learn the underlying distribution of data without explicit labeling of the data, thus performing unsupervised learning

[15, 16]. One of the key differences of deep learning compared to conventional machine learning is that deep learning automatically learns the features present in the data [15]. This means that expert knowledge to design feature extractors is no longer needed and features that are too complicated to design manually or even to be noticed by humans can still be learned. Due to the multi-layer architecture in deep neural networks, representations are learned in each layer and build upon in subsequent layers, creating complex representations of the data [14].

Since the data used for this research will be recordings of brain activity in the form of a video, an application of deep learning that sparks immediate interest is that of video prediction [17, 18]. These networks predict the next frame in a video based on the frames that have come before, thus making use of both the spatial and temporal aspects of the data.

A distinction to the conventional approach of these models is that we are not interested in predicting the next frame however. We are interested in predicting what the neuronal activity will look like based on the hemodynamics. This therefore requires a multi-modal approach. Furthermore, this problem entails that we try to predict domain A (neuronal activity) based on domain B (hemodynamics). Image-to-image translation is an area which focuses specifically on this problem. Pang et al. [19] have done a survey on the latest development within the field of image-to-image translation, including multiple multi-modal approaches of a two domain problem, or even approaches for more than two domains.

Both these ideas are worth considering for further development and evaluation. Since the image-to-image methods do not have a temporal component, something which our data does have, we can look to the video prediction networks to try and combine architecture elements from both approaches. This leads to a few considerations of different types of architectures, namely the variational autoencoder (VAE) [20] and the generative adversarial network (GAN) [21]. Later sections will cover these in more detail and give arguments as to why they might be suited to this problem.

Based on the data we have, two deep learning techniques can be combined to take advantage of the temporal and spatial structure of the data. Convolutional neural networks have been shown to have great affinity for spatial data, due to their restricted local connections which allows them to learn local features in the data. They outperform humans on tasks like image recognition and can generate images that are almost impossible to distinguish from real images [15, 22, 23]. The second technique is the recurrent neural network. These networks have good performance in sequential data due to the input being handled in a sequence and nodes in this network have an internal state so that previous information can be reused to influence the next prediction [15]. The two techniques will be discussed in more detail in Section 2.2.1.

2.2.1 Deep Learning Techniques

Deep learning involves the creation of an artificial neural network with one or more 'hidden' layers, together with an input and an output layer. Each layer of the network has multiple nodes and these nodes are connected to each other using weights with non-linear activations, the most common of which is the rectified linear unit (ReLU) due to its performance in training speed [24, 15]. The weights in the network

are then tuned using stochastic gradient descent (SGD), making use of backpropagation of error to correct the weights based on the error obtained by comparing the difference of the prediction of the network with the actual ground truth [25]. By repeating this process with enough data we approach a set of weights which have the best generalised performance. Note that this might be not be the optimal set of weights for the training data as this can lead to the over fitting of the training data to the network.

This section will cover the three primary techniques under consideration to be implemented in the network to be designed. First convolutional neural networks will be covered, which are suited for spatial data and automatic feature learning. Next, deconvolutional networks are discussed which are used to generate new samples based on either noise or representations. Finally, we will cover recurrent neural networks, which have an affinity for sequential data.

Convolutional Neural Networks

Convolutional neural networks work by spatially restricting the possible connections in the network. Instead of a fully-connected network where every node is connected to every other node in the previous and consequent layer, the connections are restricted to the immediate neighbours of the unit of interest. Taking the example of an image, this works by having a filter, an $n \times n$ matrix, pass over the image pixel-by-pixel and multiplying this filter with the pixels it is passing over, see Figure 2.1. The resulting image representation is called a feature map. Mathematically speaking this operation is called a discrete convolution and it is where the technique gets its name from.

The filters are then trained through backpropagation, adjusting them based on the error obtained at the end of the network, just as one would do with a fully connected network [26]. The result of this operation is then passed through a non-linear activation, usually the ReLU. By having multiple convolutional layers in sequence, more complicated features (filters) can be learned. This hierarchy is inspired by how the visual cortex ventral path works [15]. After the generation of each feature map, a pooling operation can be performed. Since neighbouring pixels in images usually contain similar information, the feature map representation can be compressed by applying a pooling operation. This operation takes a subset the image of a specific dimension ($n \times n$) and takes the maximum of these values. This is called max-pooling [15]. In the context of this research, convolutional networks are a good fit due to the inherent spatial nature of the video data and the automatic feature extraction it can perform on this data. We do have to keep in mind that noise needs to be eliminated out of the training data as we do not want to learn features based on this noise.

Deconvolutional Neural Networks

While convolutional networks learn the features present in the data by creating smaller but deeper representations of the input data, deconvolutional networks attempt to do the opposite. They take an input which is small and deep and attempt to scale this up to a bigger representation which has less filters. The final layer of this network can then be used to create samples which resemble the original target data, for example images. These layers are thus essential for generative models, and have been used multiple generative deep learning architectures [20, 21, 15].

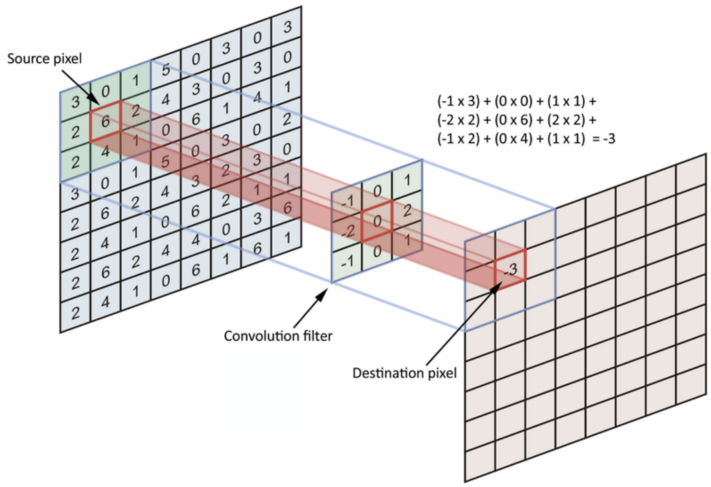


FIGURE 2.1: Visual representation of the convolution operation.

The training of these layers is similar to that of convolutional networks. Each layer has multiple filters that are used to generate the features present in the image. These filters are trained using backpropagation of the error gradient. Each subsequent deconvolutional layer thus builds up a hierarchy of features, where each layers can learn more complicated features based on the ones it has been presented from the previous layer, similar to convolutional layers. By continuing this process through multiple layers, the complex representations can be turned into a image at the final layers [27].

Recurrent Neural Networks

Due to the data used in this project being a sequence of recorded images, Recurrent Neural Networks (RNNs) can be used in the network design. RNNs have been shown to work well with sequential data, such as sentences, especially in machine-translation and text prediction [28, 29]. The network works by having nodes which have recurrent connections. These recurrent connections carry information about a previous time step into the current one, see Figure 2.2. This allows for information to be remembered and used in future activations. Each of these nodes can be seen as a self contained unit and every previous or subsequent unit can be viewed as its own network [15]. This is the main difference compared to fully-connected or convolutional networks, as these networks do not have the notion of state.

Advances in the training process and architecture of RNNs have lead to networks which have state of the art performance in a variety of problems [30, 31, 32]. Through these advances a new type of recurrent unit was created, the long short-term memory (LSTM) unit. This unit has better performance both for training and predictions, and has largely replaced traditional recurrent units. [33, 15]. The main issue with traditional RNNs is that they are not good at keeping track of long term dependencies. For example, if something at the start of a long sequence is important for the end of sequence, RNNs will lose this dependency. LSTMs are designed to keep this information persisting in the unit so that it might be used in later time steps. It does this by having an additional connection which is used to propagate the cell state throughout time, while having only minimal operations performed on it [33]. The cell state is visualised in Figure 2.3 as the horizontal line through the top.

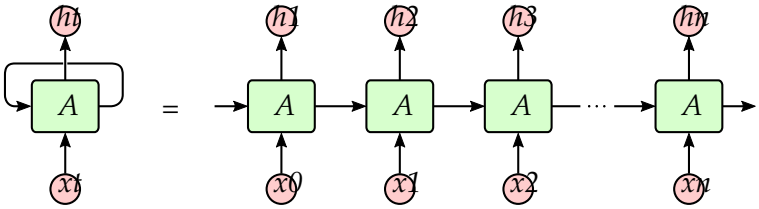


FIGURE 2.2: Illustration of a RNN, unrolled through time.

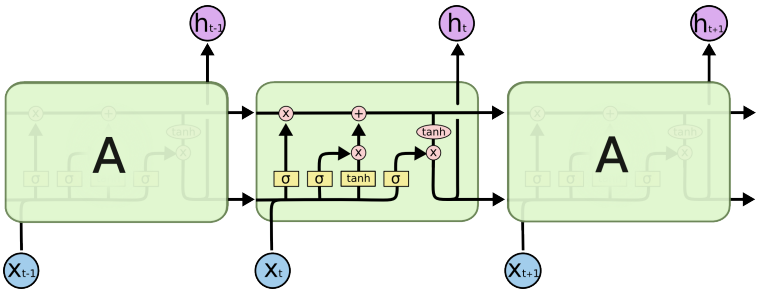


FIGURE 2.3: Illustration of an LSTM chain, unrolled through time.

Gated Recurrent Units (GRUs) have also been considered. The argument for using them is similar performance to LSTM while having less parameters [34]. This would be beneficial in a large network as it would potentially reduce the complexity of training the network. Further research into the GRUs has shown however that in tasks where deep context understanding is required LSTMs outperform the GRUs. Furthermore, the LSTMs also have a tendency for a higher true positive rate, which is desired in this research [35].

2.2.2 Deep Learning Architectures

As mentioned in Section 2.2, there are two deep neural network architectures that are possible candidates for this research. These architectures are the autoencoder and the generative adversarial networks, both of which are generative models. The next sections will cover these architectures in more detail.

Generative Models

Since the goal of the project is to produce new data based on existing data, the obvious choice for neural network models are the generative models. These models try to generate new synthetic data based on the learned posterior distribution of the data. Formally, this is done by estimating the posterior distribution through the Baysian rule: $P(Y|X) = \frac{P(Y)\times P(X|Y)}{P(X)}$, where $P(X)$ is the probability of of observing data X, $P(Y)$ is the prior and $P(X|Y)$ is the likelihood of observing the data given the prior. This prior can take on any form, but in the case of the variational autoencoder (VAE) and the generative adversarial network (GAN) this is a latent representation z, sampled from a multivariate Gaussian distribution. The likelihood then takes the form of $P(X|z)$.

By the Baysian rule, both models need to learn the likelihood $P(X|Y)$ to estimate the posterior distribution. However, the VAE and the GAN both approach the learning of the posterior differently. VAEs learn the likelihood distribution directly through the loss function (Equation 2.1). By minimising the reconstruction error, the lower

likelihood bound is estimated and used to estimate the posterior distribution. Due to the min max game used to train GANs (Equation 2.2), the generator minimises the difference between $P(X)$ and $P(X|\boldsymbol{z})$. It has thus directly learned how to create the posterior distribution, and implicitly learned the likelihood distribution.

Furthermore, both of these models are trained in an unsupervised manner, meaning they do not need labelled data to be trained. This is a very useful property when large unstructured datasets are used.

Autoencoders

Autoencoders are a type of network where it tries to generate reconstructions by compressing the input data into a latent space, and then using that latent space to recreate the input. The auto encoder consists of two parts: the encoder $E_\phi(x)$, used to generate the latent representation $z = E_\phi(x)$, and the decoder $D_\theta(z)$, which uses z to try and recreate x by $\hat{x} = D(z)$. Training is done by passing the data through the network and adjusting the parameters based on the reconstruction error between the ground truth samples and the reconstructions. This reconstruction is usually the mean squared error. The goal of an autoencoder is not to recreate perfect reconstructions however, this would be of limited use in the real world. What we are interested in is the properties of the latent space z. By creating a latent space which has less dimensions than the data, we force the network to capture the most important features of the data. This follows from that fact that if the latent dimension would be bigger than the data, it would be able to just copy over the data into the latent dimension, not learning from the the data, but rather remembering it. These features can then be used to generate new examples from data which has not been seen before by the network [14].

Variational Autoencoder Traditional autoencoders encode the data into a latent space, which is very sparse. Much of the latent representations sampled from this space will not be able to generate accurate reconstructions. This is due to the latent space not being regularised over all the examples used to train the network. To fix this issues, Kingma et al. [20] introduced the variational autoencoder. It constrains the latent space to conform to a normal distribution, regularising the latent space. Furthermore, instead of giving the latent representation for each input sample, it calculates a mean and log variance for it. This can then be used to create a distribution, and sample it to create the latent representation. Due to this random sampling however, it is difficult to propagate the gradient through the network for training. Kingma et al. [20] fixed this by reparameterising the latent space to be a function of the encoders output. This is known as the reparameterisation trick. Another benefit of sampling a distribution for creating the latent space, compared to them being given, is that the network becomes better for generative tasks. Since the latent space is restricted to a normal distribution, random samples drawn for a normal distribution can also generate meaningful examples.

To train a variational autoencoder a loss is calculated consisting of two parts representing the goals of the network. The first goal (first term in Equation 2.1) is that the latent space should be normally distributed. To achieve this, the Kullback-Leibler divergence between the approximate posterior distribution of $E_\phi(z|x)$ and an assumed prior normal distribution is calculated. Minimising this loss means that the posterior distribution is as close to a normal distribution is possible. A loss of zero at this term means that the posterior distribution is a perfect normal distribution.

The second goal (second term in Equation 2.1) is the same as with the regular autoencoders, namely that we want to create accurate reconstructions using the mean squared error. Because the prior is a standard Gaussian, the second term of the loss function reduces to the mean squared error between the input of the encoder and the output of the decoder. Both losses together form the total loss of the network [20].

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}^{(i)}) \simeq \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\right) + \frac{1}{L} \sum_{l=1}^{L} \log D_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)} | \boldsymbol{z}^{(i,l)})$$
$$\text{where } \boldsymbol{z}^{(i,l)} = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)} \text{ and } \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I}) \tag{2.1}$$

Autoencoders have been used with success in areas which require sequential data, like machine translation, and have made extensive use of LSTM networks [29, 10]. In Sirpal et al. [10], a similar problem to the one posed in this research was faced. They tried to predict the resting state hemodynamic parameters (HbO, HbR and Hb total) based on resting state EEG signal in humans. This problem also incorporates the neurovascular coupling, as the fNIRS data used is reliant on this phenomenon.

In the (VAE) architecture used, they integrate both LSTM as well as convolutional layers. This is done to learn the sequential structure of the EEG data as well as to automatically learn the features in the data [10]. They did not however, have the amount of spatial data available that the fcOIS in this research data has. With this data available, better features could be extracted by the convolutional layers of the architecture making an autoencoder a strong candidate as a solution to our problem.

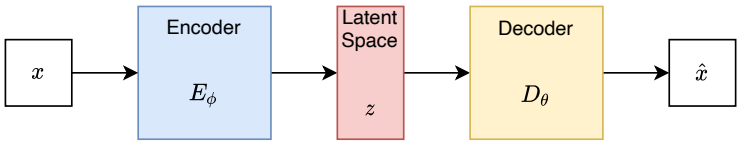


FIGURE 2.4: Illustration of the autoencoder architecture.

Generative Adversarial Networks

Generative adversarial networks (GANs) are, as the name implies, generative networks that have shown to have good performance in the image domain [21, 16]. In a GAN two neural networks are being trained, one generator network G and one discriminator network D. The generator is trying to capture the distribution of the training data, and generate plausible examples based on noise sampled from a Gaussian (the latent space, z in Fig. 2.5). The discriminator is being trained to recognize if a sample that it is given is from the real distribution or if it is created by the generator. This results in a min max game between the generator and the discriminator, making the network adversarial. The full value function is given in Equation 2.2. The first term is the accuracy of the discriminator for the real data where the second term is the accuracy of the discriminator on the generated data. This value function is used to calculate the loss of the network and to compute the gradient. Subsequently, the full loss is used to train the discriminator and the only the second term is used to train the generator of the network. Both the discriminator and the generator are trained separately, alternating between them. Because $\log(1 - D(G((\boldsymbol{z}))$

generates weak gradients at the start of training, the term can be reversed to maximise $\log(D(G(\boldsymbol{z}))$. In practise, this generates the same dynamics as the original term, but the gradients are stronger in early learning. The theoretical optimum for the network is reached when the discriminator reaches an accuracy of 0.5. At this point the discriminator can no longer tell which data is generated and which is real, assigning equal probability to both [21, 14].

$$\min_G \max_D V(G,D) = \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}))] \tag{2.2}$$

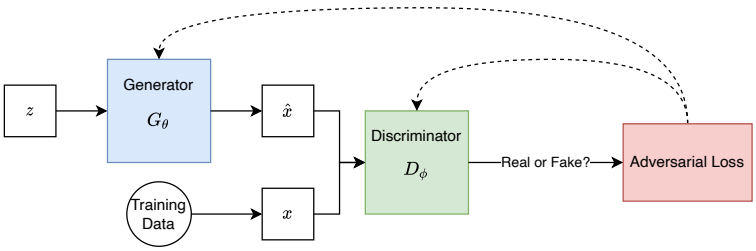


FIGURE 2.5: Overview of the GAN architecture.

In the introduction paper of GANs from Goodfellow et al. (2014) [21], it is mentioned that they were not yet aware of the approaches by Kingma et al. (2013) [20] in the development of the variational autoencoder, where both the neural networks in the architecture are trained through backpropagation. The approaches are similar however, in that they both have a differentiable (i.e. trainable through backpropagation) generative network. The difference is that the autoencoder builds its latent space through the encoder network whereas the GAN uses a fixed latent distribution, usually a standard Gaussian, which is sampled by the generator to generate its target [21].

Merging VAEs and GANs

More work has been done to try and merge the two architectures to attempt to create the best of both worlds. Where VAEs usually have a problem of generating blurry examples, traditional GAN architectures have inherent instabilities which could lead to mode-collapse, where the diversity of the original distribution is lost [21, 36].

Larsen et al. [37] propose a hybrid of the VAE en GAN architecture by adding an encoder to the GAN and collapsing the decoder and the generator into the same network as they fulfill the same role in their respective original architectures. The proposed strength of the architecture lies in that it uses the learned feature representations by the discriminator of the GAN for the reconstruction error used to train the encoder. This replaces the element-wise error of the usual mean squared error of VAEs with feature-wise errors. This allows the network to better capture the data distributions that it is trying to learn, as element-wise distance measure are inadequate for complex data distributions like images [38].

To train the network, three criteria are used (Equation 2.3). The first $\mathcal{L}_{prior}$ (Equation 2.4), is the KL-Divergence (D_{KL}) of the encoded latent space and a normal Gaussian. This is equivalent to the regularisation term of the VAE. The second term, $\mathcal{L}_{dislike_l}$ (Equation 2.5), is the difference between the features of the data and the features of the reconstructions as learned by the discriminator. This is used to replace the mean squared error between the original data and the reconstructions, and thus replaces

the reconstruction loss of the VAE. $\mathcal{L}_{GAN}$ (Equation 2.6) is the final loss and this is the summed binary cross entropy of the confidence of the discriminator between the real data, the reconstructions and the samples generated from a random latent space [37].

$$\mathcal{L} = \mathcal{L}_{prior} + \mathcal{L}_{dislike_l} + \mathcal{L}_{GAN} \tag{2.3}$$

$$\mathcal{L}_{prior} = D_{KL}\left(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\right) \tag{2.4}$$

$$\mathcal{L}_{dislike_l} = -\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(D_l(\boldsymbol{x})|\boldsymbol{z})] \tag{2.5}$$

$$\mathcal{L}_{GAN} = \log(D(\boldsymbol{x})) + \log(1 - D(G(\boldsymbol{z}))) + \log(1 - D(G(E(\boldsymbol{x})))) \tag{2.6}$$

where $\boldsymbol{x}$ = real data, $p(\boldsymbol{z}) \sim \mathcal{N}(0, \mathbf{I})$, $q(\boldsymbol{z}|\boldsymbol{x}) = z \sim E(\boldsymbol{x})$,
D is the discriminator, G is the generator, E is the encoder,
and D_{KL} is the Kullback-Leibler divergence

The combined loss is not used to update all the parameters in the network. Better performance was observed by not propagating $\mathcal{L}_{GAN}$ to the encoder and and $\mathcal{L}_{dislike}$ cannot be used to update the discriminator as would result in the collapse of the network. Furthermore, $\mathcal{L}_{dislike}$ is weighed using hyperparameter γ when calculating the loss for the generator. This is done to create a trade-off between the reconstruction abilities and the ability to fool the discriminator. Equation 2.7, 2.8 and 2.9 give the update rules of the gradients for the different networks.

$$\theta_E \overset{+}{\leftarrow} -\nabla_{\theta_E}(\mathcal{L}_{prior} + \mathcal{L}_{dislike_l}) \tag{2.7}$$

$$\psi_G \overset{+}{\leftarrow} -\nabla_{\psi_G}(\gamma\mathcal{L}_{dislike_l} - \mathcal{L}_{GAN}) \tag{2.8}$$

$$\phi_D \overset{+}{\leftarrow} -\nabla_{\phi_D}\mathcal{L}_{GAN} \tag{2.9}$$

Since the target we try to generate in this research is an image with limited spatial resolution, blurry images pose a problem for the accuracy of the result. By using a GAN and its adversarial training this could be prevented, and it is therefore a potential choice for use in this research. Since success with a VAE in comparable problems already has been shown [10], extending it with components of the GAN to increases image quality is a logical next step.

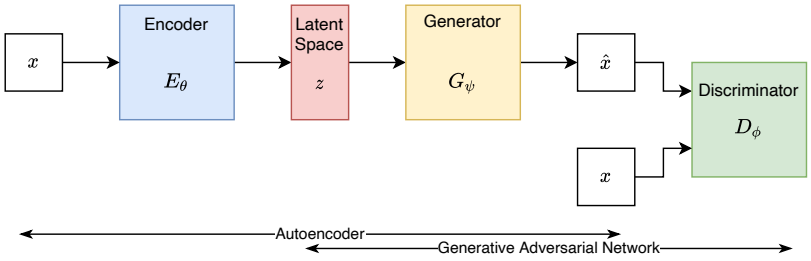


FIGURE 2.6: Overview of the VAE-GAN architecture.

2.3 Discussion

In this chapter, both machine learning techniques and architectures have been discussed. Due to lack of representation learning and generative models, traditional machine learning cannot be used for this research, and deep learning needs to be used. Convolutional neural networks are the most apt to use in this project as they can automatically learn the features present in the data which is being used to train net network. Recurrent connections are also useful as they can capture the temporal element of the resting state recording used in this research. Previous work has already shown that this is an important element and to this end they will be implemented in the architectures used for this project. Furthermore, the two architectures that will be used in this research have been identified: the Variational Autoencoder and the Variational Autoencoding Generative Adversarial Network. These architectures will be compared en evaluated to see which performs best on the given data, and if they can be used to generate accurate recreations of the original data.

2.4 Intermediate Conclusion

After reviewing the relevant literature we can give answers to the research questions 1 and 2:

1. *Which machine learning techniques can be used to interpret the data and learn the coupling?*
2. *What neural network architectures are potentially viable for learning the coupling?*

The answer to the first question is that convolutional networks and recurrent neural networks can be used to learn the features present in the data. Followed by deconvolutional layers, which do the reverse of the convolutional layers, new samples based on representations of the data can then be generated.

Due to the generative capabilities and success in previous research, the variational autoencoder en the variational autoencoding generative adversarial network are the most suited for this task, hereby answering the second question.

3 Data

3.1 Data Acquisition

The data used in the project is collected during a different project which focused on the effects of hypoxia on the resting state networks of neuronal activation and hemodynamics in mice. Neuronal activity is measured in 7 mice with the help of GCaMP/Calcium imaging, and hemodynamics are measure in 8 mice with intrinsic optical imaging. The gathering of the data spanned multiple weeks, with one acquisition of 40 minutes per week, alternating between normoxia recordings (atmospheric oxygen, 21%) and hypoxia (either 12%, 10% or 8% oxygen) recordings. This resulted in a dataset consisting of 28 normoxia and 28 hypoxia recordings for neuronal activity, and 32 normoxia and 32 hypoxia recordings for hemodynamic activity. This resulted in a dataset consisting of 24 normoxia recordings and 24 hypoxia recordings. For training the model for normal resting state, only the first normoxia recording is used, as subsequent hypoxia recordings could have potential influence on the resting state. Furthermore, only minute 10-17.5 of each recording is taken. This window is chosen as the first 10 minutes of the recording might be stressful for the mice, influencing the recording. Choosing too big of window also leads to having too much data to train the network effectively due to over fitting. The resulting dataset for network training for healthy mice consists of 8 normoxia recordings (4 female, 4 male) with 10 minutes worth of recording per mouse [39].

3.1.1 Ethics Statement

All surgical procedures were approved by the Animal Research Ethics Committee of the Montreal Heart Institute and were performed according to the recommendation of the Canadian Council on Animal Care.

3.1.2 Acquisition Process

The acquisition process consists of two major parts. The first is the preparation of the mice and the other being the recording setup.

Mice Preparation

A total of eight mice were used in the acquisition (4 female, 4 male). The sex is not thought to have influence on the resting state and thus no distinction is made between for the purpose of this research.

Mice were injected with a virus to spread the GCaMP6s indicator throughout the brain. After this a minimum of 5 days passed before proceeding to the next step of implementing the surgical window. The surgical window needs to be implemented on top of the skull so the optical recording can be made. This is done by anaesthetising the mouse, cutting away the skin in the area where the window needs to

be implemented and fixing the window to the skull using dental cement. A titanium head bar is also glued to the head so that the mice can be fixed in place during the recordings [39].

Recording Setup

The imaging equipment used consists of the LightTrack OiS200 system (Labeotech inc.) with multicolour interlacing leds for recording the different datasets (GCaMP and intrinsic optical imaging). Red, green and amber LEDs (535 nm, 590 nm, 620 nm), are used to illuminate the brain for the recording by CMOS sensors to create the data used for the hemodynamic calculation and a blue led (475 nm) is used to excite the GCaMP indicators and record the calcium activity. CMOS sensors are used to capture the data at 80 hz total, giving 20 hz per colour channel for the 4 colours. The final recording is made over a 10x10 mm area of the mouse brain, captured at a resolution of 192x192 pixels [39]. The processing of this data is described in next section.

3.2 Data Processing

After capturing the raw recordings the four colour channels need to be processed into the hemodynamic and calcium activity files. For this, a pipeline was developed which processes the recordings to the final files used in this project freely available at https://github.com/marl1bakker/Hypoxia).

The first step is the co-registration of the recording. This corrects the recording for any movement of the mice by lining up the frames of the recording so that the brain is in the same position in every image.

Using the same techniques as described in Valley et al. [40], the hemodynamic signals are calculated. Because oxygenated (HbO) and deoxygenated (HbR) blood have different absorption rates from the LEDs, the difference can be calculated when assuming the baseline values of the blood present in the mouse brain (60 μM HbO, 40 μM HbR). This produces the HbO and HbR recordings used in training the models.

Next, the calcium data is corrected for hemodynamic fluctuations by using the data obtained from the red, green and amber LEDs. The modified Beer-Lambert law is used to calculate the estimate of changes of hemoglobin which is then regressed onto the calcium data. This eliminates most of the hemodynamic influence on the calcium data [40]. The data is then normalised to obtain the the percentage of fluorescence change ($\Delta F/F$) across the recording. Finally, to correct for non-physiological artefacts like illumination or camera noise, global signal regression is applied to the data [41].

3.3 Data Preparation

To use the data for training, a custom interface is written so the binary files are converted to an array, and this array is cut to five minutes required for training. This is done to prevent the network overfitting on too much data and to keep some data for validation purposes. A mask corresponding to the recording is then loaded to mark which area of the image is the actual mouse brain. This mask is applied to the array to remove the noise present in the image around the brain. Afterwards the array is normalised between 0 and 1.

The different data files (HbO, HbR and Ca) are loaded into a custom Dataset class of Pytorch where the data is kept for the duration of the training. Due to this, the data can easily be integrated with the Pytorch framework and its data transformations. Finally, custom indexing is implemented to account for the hemodynamic delay in the brain. When a certain index of the calcium data is picked, representing a specific frame at a specific time, the corresponding HbO and HbR data are delayed. This delay is controlled by a hyperparameter for the dataset.

3.4 Intermediate Conclusion

After reviewing the data and determining how it is structured an answer can be given to research question 3.

3. *What processing is needed for the data to be used in the network while maintaining its integrity?*

Since the data has already thoroughly been processed for different research, little further processing is needed to prepare the data for training. The only additional step added is re-scaling all the values present in the matrices of the data between 0 and 1 so they are inline with what is expected by the deep learning frameworks.

4 Deep Learning

Due to the complexion and large size of the dataset, hand engineering features to learn from the the data is too costly. To learn these features, representation learning using a deep neural network is needed. Because of this, two network architectures of generative models have been chosen, which are able to create recreations of the original dataset: the variational autoencoder (VAE) and the variational autoencoding generative adversarial network (VAE-GAN). The implementation of these architectures will be discussed in the next section of this chapter, followed by the training strategy and finally the use of the network to attempt and produce the desired reconstructions of the data.

4.1 Architectures

The neural network architectures chosen are the variational autoencoder and the variational autoencoding generative adversarial network, both discussed in Section 2.2.2. Both architectures make use of deep convolutional networks to learn the features present in the data, compressing these features to a latent space, which is in turn used by the decoder/generator to create recreations of the original images. These architectures are chosen so that unseen hemodynamic data can be encoded and used to generate the corresponding neural activity.

The goal of the architecture is to reproduce the distribution of the calcium activity P_{ca}. Samples from this distribution are recreated by reconstruction from a sample for the latent distribution Pz. This latent distribution is generated by encoding the hemodynamic data. This hemodynamic data is represented by either P_{HbO} or P_{HbR}; oxygenated blood or the deoxygenated blood data respectively. This turns the architectures used into multi modal networks where they translate between the hemodynamic modality into the neuronal activity modality. Since the architectures used train by measuring the losses over the relationship between P_z and P_{ca} the input to the encoder network does not need to be the target that is attempted to be recreated. The goal of this setup is for the encoder to learn and try to encode the relevant features of the hemodynamic such that it can be used to learn the neuronal activity, thus learning the hemodynamic coupling.

4.1.1 Variational Autoencoder

The variational autoencoder follows the standard design as first presented by Kingma et al. [20], shown in Figure 2.4. The architecture works by encoding data samples from either $\boldsymbol{x} = P_{\text{HbO}}$ or $\boldsymbol{x} = P_{\text{HbR}}$ into the latent distribution by $P(\boldsymbol{z}|\boldsymbol{x}) = E_\phi(\boldsymbol{x})$. The latent distribution is then sampled to create the latent representation $\boldsymbol{z}$, which is used in the recreation of the calcium data by $\hat{\boldsymbol{x}} = D_\theta(\boldsymbol{z})$. Instead of using the input data from P_{HbO} or P_{HbR} as the ground truth for the loss calculation in the network, we use the data from P_{ca}. This now means that the features learned from the hemodynamic

data are used to recreate the calcium data. This slightly alters the loss function from Equation 2.1, modifying the second term. Instead of the reconstruction loss using the input data, it uses data from the calcium dataset where the sample is taken that is d amount of frames earlier than the hemodynamic data. The modified equation is given in Equation 4.1.

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}^{(i)}) \simeq \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\right) + \frac{1}{L} \sum_{l=1}^{L} \log D_{\boldsymbol{\theta}}(\boldsymbol{k}^{(i-d)} | \boldsymbol{z}^{(i,l)})$$
$$\text{where } \boldsymbol{z}^{(i,l)} = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)} \text{ , } \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I}) \text{ ,}$$
$$\boldsymbol{k} = \text{calcium data and } d = \text{ hemodynamic delay} \tag{4.1}$$

Multiple networks are trained to see which data is relevant for generating the neural activity. One network is trained with the HbO data, another with the HbR data and an extended architecture is created (Fig. 4.1) which uses both the HbR and the HbO data to make the reconstructions. This architecture uses two encoders which both produce half the latent space representation used in that network. These representations are then concatenated and used by the decoder to create the reconstructions. This means that for these networks the latent space size is doubled compared to the networks which use a single encoder. This also modifies the loss function as we now have to account for the two encoders. While the reconstruction loss is still the same for both encoders, a new term is added to regularise the latent distribution of the second encoder, shown in Equation 4.2.

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}; \boldsymbol{x}^{(i)}, \boldsymbol{k}^{(i-d)}) \simeq \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log((\sigma_{\phi j^{(i)}})^2) - (\mu_{\phi j^{(i)}})^2 - (\sigma_{\phi j^{(i)}})^2\right)$$
$$+ \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log((\sigma_{\psi j^{(i)}})^2) - (\mu_{\psi j^{(i)}})^2 - (\sigma_{\psi j^{(i)}})^2\right) + \frac{1}{L} \sum_{l=1}^{L} \log D_{\boldsymbol{\theta}}(\boldsymbol{k}^{(i-d)} | \boldsymbol{z}^{(i,l)}) \tag{4.2}$$
$$\text{where } \boldsymbol{z}^{(i,l)} = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)} \text{ , } \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I}) \text{ ,}$$
$$\boldsymbol{k} = \text{calcium data and } d = \text{ hemodynamic delay}$$

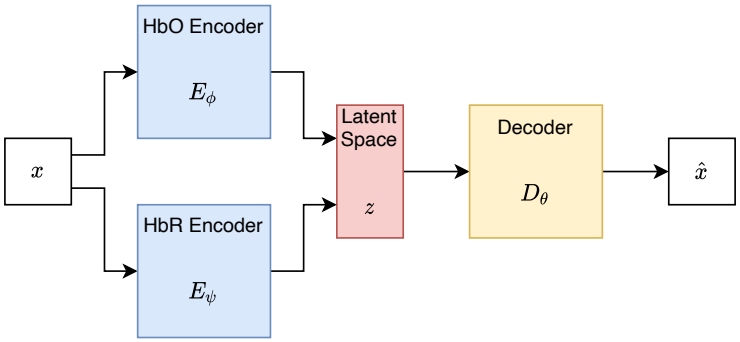


FIGURE 4.1: Overview of the expanded autoencoder architecture.

To see how much the latent dimension influences the generation and to find which is suitable for this task, every network is trained with dimension sizes, 256 and 512.

The values of 256 was chosen as this was used in previous research [10]. The data used here, however, has more dimensions and thus might need more room for encoding the latent representation. Because of this the value of 512 has been chosen.

Network Structure

Encoder The encoder is a deep convolutional network with two flat outputs. It takes the input data en passes it down through the convolutional layers, learning the features in the data. Afterwards, the data is flattened and through two fully connected layers it is fed into the two output layers. These output layers represent the mean and log variance of the latent distribution and are together are used to create the latent distribution z and to calculate the KL-Divergence loss of the latent distribution compared to a normal Gaussian distribution. The full structure of the network is shown in Table 4.1. In the extended architecture, the encoder structures are identical.

Decoder The decoder is made up of the reverse convolutional structure as the encoder. Before this, however, two LSTM layers take the input of the network, a latent representation z. For the expanded architecture this is a concatenated latent representation from both encoders. After the LSTM layers the representation of the input is fed trough a fully connected layer and reshaped to match the input expected by the first deconvolutional layer. The deconvolutional layers then try and reconstruct the image in the likeness of the calcium data. Due to the deconvolutional layers being prone to generating noise pattern over the data, a final layer with a stride of one has been added to try and mitigate these effects [42]. A full overview of the design of the decoder is shown in Table 4.1.

Encoder	Decoder
3×3 32 conv. ↓, BNorm, ReLU	Latent dim LSTM
3×3 64 conv. ↓, BNorm, ReLU	Latent dim LSTM
3×3 128 conv. ↓, BNorm, ReLU	36864 FC, BNorm, ReLU
3×3 256 conv. ↓, BNorm, ReLU	3×3 128 conv. ↑, BNorm, ReLU
4× latent dim FC., BNorm, ReLU	3×3 64 conv. ↑, BNorm, ReLU
2× latent dim FC., BNorm, ReLU	3×3 32 conv. ↑, BNorm, ReLU
Latent dim FC., mean	3×3 32 conv. ↑, BNorm, ReLU
Latent dim FC., log variance	3×3 1 conv. ↑, Sigmoid

TABLE 4.1: Layers of the networks in the VAE.

Network Training

VAE are traditionally jointly trained using one optimiser [43] and this will also be the case in this research. For the optimiser, the Adaptive Movement Estimation (ADAM) optimiser is used as according to its creator it is well suited to problems which have many parameters and big datasets [44]. Furthermore, ADAMGrad is also used to preempt any non-convergence issues which may arise in ADAM [45]. The learning rate α was set to 0.001 as recommended by Kingma et al (2017) [44]. Further training details relevant for both networks are discussed in Section 4.2.

4.1.2 VAE Generative Adversarial Network

The variational autoencoding generative adversarial network follows the design by its creators, Larsen et al. [37], shown in Figure 2.6. Similar to the VAE, the input data gets encoded into the latent distribution $P(\boldsymbol{z}|\boldsymbol{x}) = E_{\phi}(\boldsymbol{x})$ and samples get reconstructed using the generator $\hat{\boldsymbol{x}}_{rec} = G_{\theta}(\boldsymbol{z})$. After this however, the the reconstructions are used for the adversarial training process of the GAN. The discriminator D_{η} is used to decide if the sample that the network is given is real or not. This is also done for samples generated for $\hat{\boldsymbol{x}}_{gen} = G_{\theta}(\hat{\boldsymbol{z}})$ where $\hat{\boldsymbol{z}} \sim \mathcal{N}(0, \mathbf{I})$, the latent representation randomly drawn from a Gaussian. Combined with the discrimination of the calcium data, this results in the adversarial loss of the network. This does mean that, similar to the VAE, the input data is not used as the ground truth as we want to reconstruct the calcium data. We are not interested in the generations of the network, as we want to control what the output will be using the encoding, but they are used to improve the generator. Since the latent space generated by the encoder is regularised using a standard Gaussian, the random distribution used for $\hat{\boldsymbol{x}}_{rec}$ and the encoded latent space are similar and should result in similar samples. This is useful for training the discriminator. Due to this changes the loss functions are modified slightly and the modified ones are given in Equation 4.5.

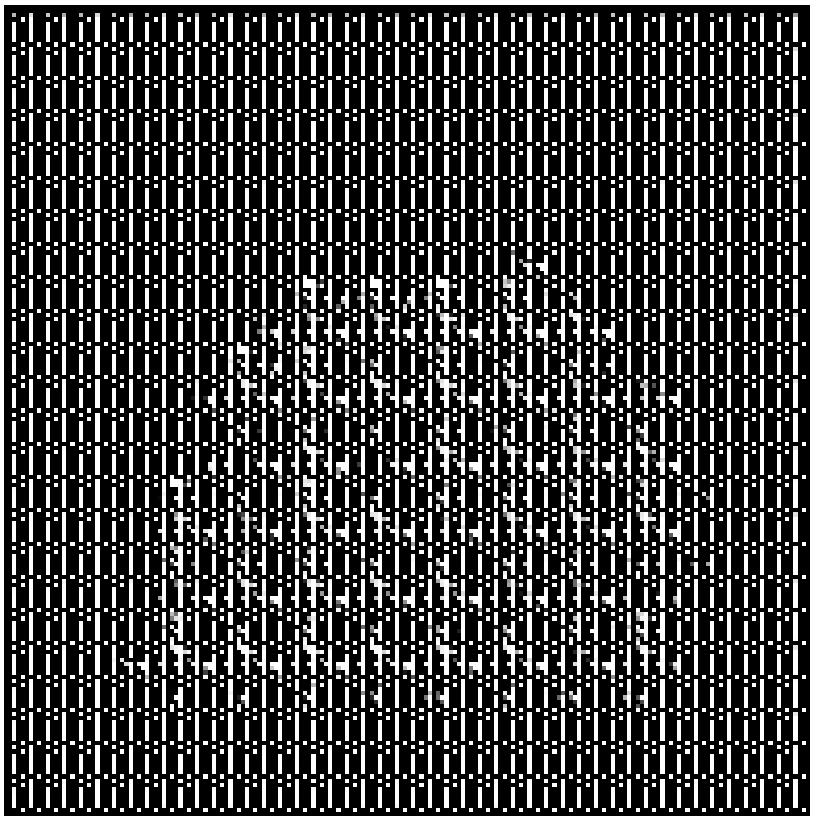


FIGURE 4.2: Failed reconstruction using the dislike loss with learned features.

During the initial training of the network it was found that using the dislike loss (Equation 2.5) with the learned representations resulted in unstable training. We have two hypotheses as to why this happened with the data we had. The first is that the discriminator was under trained. If this is the case, the discriminator has not learned the features well enough to distinguish between the real data and the generated samples. Since these learned features are also used to train the encoder

and the generator, this might result in a feedback loop which would ultimately result in the bad samples generated (see Figure 4.2). The other hypothesis is that since the data is noisy itself, it could lead to the discriminator learning the noise and then over fitting in it. While we have made attempts to try and remedy the issue with the original loss calculation in mind, unfortunately, due to time constraints, we were not able to solve it. The results improved however, when instead of using the learned features to calculate the dislike loss calculation, we swapped to a more traditional mean squared error reconstruction loss between the generated samples $\hat{\boldsymbol{x}}_{rec}$ and the ground truth. This resulted in more stable training and qualitatively better results. This resulted in another modification of the loss formulas, shown in full in Equation 4.4.

$$\mathcal{L} = \mathcal{L}^{\phi}_{prior} + \mathcal{L}^{\psi}_{prior} + \mathcal{L}_{dislike} + \mathcal{L}_{GAN} \tag{4.3}$$

$$\mathcal{L}_{dislike} = \log G_{\boldsymbol{\theta}}(\boldsymbol{k}|\boldsymbol{z}) \tag{4.4}$$

$$\mathcal{L}_{GAN} = \log(D(\boldsymbol{k})) + \log(1 - D(G(\boldsymbol{z}))) + \log(1 - D(G(E(\boldsymbol{x})))) \tag{4.5}$$

where $\boldsymbol{x}$ = hemodynamic data, $p(\boldsymbol{z}) \sim \mathcal{N}(0, \mathbf{I})$, $\boldsymbol{k}$ = calcium data ,
D is the discriminator, G is the generator, and E is the encoder

Analogous to the VAE, the VAE-GAN is also trained using the three different datasets: The HbO, the HbR and the combined dataset. For the combined dataset the VAE-GAN architecture was also extended using an additional encoder to create half the latent space representation, see Figure 4.3. This was then concatenated from both encoders to create the full latent representation. Because of this, the loss formula changes slightly, adding another prior loss to the full loss calculation to regularise the latent space of the second encoder, as shown in Equation 4.3.

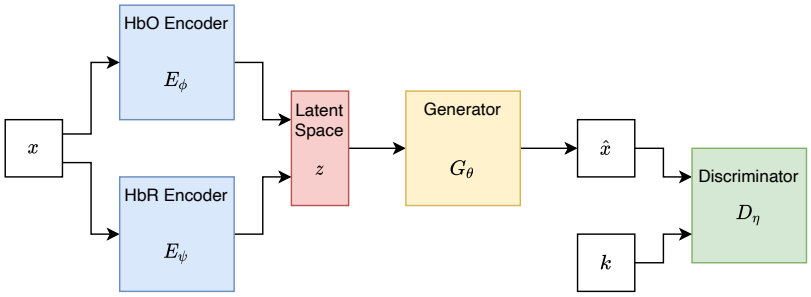


FIGURE 4.3: Overview of the VAE-GAN architecture.

Network Structure

Encoder The encoder(s) follows the same structure as the encoder from the VAE. It uses a combination of convolutional and fully connected layers to output the mean and log variance of the latent representations corresponding to the input data. This is then turned into the latent representation used for the generator and to calculate $\mathcal{L}_{prior}$.

Generator Since the generator has the same function as the decoder in the VAE, this structure is also the same. The generator takes the encoded or the randomly

sampled latent representation, and generates the reconstructions. The only difference is that is uses the hyperbolic tangent as output as this is deemed more stable in GANs [16]. Finally, this network has also added a final deconvolutional layer with a stride of one to combat artifact creation in the images [42].

Discriminator The discriminator is made up of multiple convolutional layers which try and learn the features of the input that it is trying to discriminate. This is followed by two fully connected layers, the final one of which has a one neuron output for discrimination. In contrast to the other networks, the discriminator uses LeakyReLU for the activation as the results in more stable GAN training [16]. The final layer has no activation as the loss function used (binary cross entropy) is more efficient with logits as input. The full structure of the networks are shown in Table 4.2.

Encoder	Generator
3×3 32 conv. $\downarrow$, BNorm, ReLU	Latent dim LSTM
3×3 64 conv. $\downarrow$, BNorm, ReLU	Latent dim LSTM
3×3 128 conv. $\downarrow$, BNorm, ReLU	36864 FC, BNorm, ReLU
3×3 256 conv. $\downarrow$, BNorm, ReLU	3×3 128 conv. $\uparrow$, BNorm, ReLU
$4 \times$ latent dim FC., BNorm, ReLU	3×3 64 conv. $\uparrow$, BNorm, ReLU
$2 \times$ latent dim FC., BNorm, ReLU	3×3 32 conv. $\uparrow$, BNorm, ReLU
Latent dim FC., mean	3×3 32 conv. $\uparrow$, BNorm, ReLU
Latent dim FC., log variance	3×3 1 conv. $\uparrow$, Tanh

Discriminator
3×3 32 conv. $\downarrow$, BNorm, LeakyReLU
3×3 64 conv. $\downarrow$, BNorm, LeakyReLU
3×3 128 conv. $\downarrow$, BNorm, LeakyReLU
3×3 256 conv. $\downarrow$, BNorm, LeakyReLU
2048 FC., BNorm, LeakyReLU
1 FC.

TABLE 4.2: Layers of the networks in the VAE-GAN.

VAE-GAN Training

The VAE-GAN is trained by an optimiser for each network, for a total of 3 or 4 optimisers, in the case of the extended architecture. Each of these optimisers is the ADAM optimiser [44] with a learning rate α of 0.001. ADAMGrad [45] was also enabled to preempt the non-convergence issues. The training algorithm from the original paper is used where the losses for each network are calculated according to Equations 2.7, 2.8 and 2.9 [37]. The different networks are trained in sequence, first the encoder parameters get updated, followed by the generator and finally the discriminator. Further training details will be discussed in the next section.

4.2 Network training

As mentioned in the previous sections, the networks are trained by using either the HbO data, the HbR data or both. This is done because some of the data might be

better indicator for the calcium activation, as HbO might be influenced by cardiovascular activity, creating confounds, whereas HbR is a direct effect of the neuro vascular coupling. In loading the dataset the entire hemodynamic datasets are delayed by 44 frames, to account for the hemodynamic delay that follows after neural activation. The value of 44 is chosen as 2.2 seconds is the average delay in hemodynamic activity in awake mice. As the data is recorded in 20hz per channel, the resulting delay would be 44 frames.

The data used for training is 5 minutes (minute 12.5-17.5) of the first normoxia recording of a mouse. When trying to train the network with data from multiple mice, the network training became very unstable, unable to produces sensible results. This is due to the different mice not having the same brain size and position in the images, making it hard to capture the fine details of the neuronal activation. For the validation data, 2.5 minutes (minute 10-12.5) were used.

The networks are trained for a total of 200 epochs. This much training likely results in over fitting of the network however, so every 10 epochs the networks are saved. After training we use the validation data recorded during training to find out when the validation losses start to diverge from the training losses. The model closest to the epoch where the diversion gets noticeably worse is loaded to generate the results of the models. This is the model which has the best generalisation performance to the evaluation set. For the monitoring of the training, every epoch a sample for $\hat{\boldsymbol{x}}_{rec}$ is saved for both the training an validation data to qualitatively evaluate the training process.

The VAE-GAN paper notes it uses a batch size of 64 for training [37]. We have limited the batch size to 32 however, as an increased batch size lead to computational issues, especially in memory usage.

5 Model Analysis

Based on the architecture and training process of the previous chapter, multiple models were trained to attempt to reconstruct the neural activity. The reconstruction of this neural activity, visualised through the GCAMP indicators, is done using the HbO, HbR data or both combined. The layers of the network were fixed, and between the models the latent dimension and the training data type was switched. These options were varied for both architectures.

5.1 Model Selection

Before evaluation, a model first needs to be chosen. For every 10 of the 200 epochs for training, the network was saved for future use. To select one of these models we need to evaluate the training performance. As mentioned in Section 4.2, a validation set was used during training to keep track of the generalisation performance. After training all the networks, this data was used to determine which of the saved models is the best performing one. For every network design a model was chosen, and the choices are shown in tables 5.1a and 5.1b. This choice is based on the minimum validation loss and evaluation of the training graphs, see Appendix A and B. Most of the networks have epoch 200 as their optimal epoch. This indicates that 200 epochs was not enough to find the best parameters and training should have progressed longer. Due to limited time and computational resources this was not possible.

VAE	Both	HbO	HbR
256	200	200	200
512	200	200	200

(A) Optimal epochs of the VAEs.

VAE-GAN	Both	HbO	HbR
256	200	180	200
512	200	190	200

(B) Optimal epochs of the VAE-GANs.

TABLE 5.1: Optimal epochs of the different network designs.

5.2 Evaluation Criteria

To evaluate the results of the network, two evaluation criteria were used: The mean squared error (MSE) and the structural similarity index (SSIM). Each of these criteria measure image quality compared to the ground truth.

The mean squared error has been discussed before in Section 2.2.2, where it is the main training target of the autoencoders. Indeed, the autoencoders use the MSE to guide their reconstructions in the right direction. This is a very basic measure, but it gives a good indication and is widely used in a variety of fields [38]. Its goal is to compare two signals, in this case images, and give a quantitative comparison between the two. This comparison is usually made between the ground truth and a generated sample. This will also be the case here, the original ground truth images

will be compared to generated samples. Equation 5.1 shows the details of the MSE computation.

$$\text{MSE}(x,y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \tag{5.1}$$

The structural similarity index was conceived to create a metric which can measure the structural consistency of an image compared to another one. This means that nonstrucutral distortions, like a change in luminance or contrast, do not impact the similarity of an image. This is in line with what we humans consider to be similar, as an object stays the same object even if the object is more bright. By measuring just the structure of the image, we can see if it is the same in the samples even though the network generating the sample might have introduced some nonstructural distortions. SSIM is used widely in the image processing domain, including in infrared imaging and MRI imaging [38].

SSIM works by sliding a window over the image which computes three local elements of the images patches. These elements are the similarity of luminance $l(x,y)$, the similarity of the contrasts $c(x,y)$ and the similarity of the local structures $s(x,y)$. The final score is then computed by averaging all the values across the image, giving a score between 0 and 1 where 1 is the highest score, being the most similar. The full details of the SSIM computation is given in Equation 5.2. μ_x and μ_y are the local sample means of x and y, σ_x and σ_y are the standard deviations of the local sample x and y and σ_{xy} is the local cross correlation of x and y without their means. $C1$, $C2$ and $C3$ are small positive constants for numerical stability.

$$\begin{gathered} S(x,y) = l(x,y)\cdot c(x,y)\cdot s(x,y) = \\ \left(\frac{2\mu_x\mu_y + C1}{\mu_x^2 + \mu_y^2 + C1}\right)\cdot\left(\frac{2\sigma_x\sigma_y + C2}{\sigma_x^2 + \sigma_y^2 + C2}\right)\cdot\left(\frac{\sigma_{xy} + C3}{\sigma_x\sigma_y + C3}\right) \end{gathered} \tag{5.2}$$

5.3 Comparing the samples

To compare the samples, the optimal model was loaded and used to generate a batch of samples. This batch has the same batch size used in training, 32 images. These images are generated by feeding the HbO and/or HbR data into the network to obtain the reconstructions. For the VAE-GANs, only the encoded latent dimension was used as we want more control of the output. The same batch of images was used for all the networks to try and make a direct comparison between the networks. This batch is shown in Figure 5.1.

The batch of images is then used to compute the MSE and SSIM scores. The scores are computed per image and averaged over the batch. This results in a more consistent score and it prevents any outliers from generating results not representative of the entire dataset. This also prevents any cherry picking of the data to present misleading results.

5.4 Discussion

This chapter outlined two methods to evaluate the results presented in the next chapter, the mean squared error and the structural similarity index. The optimal

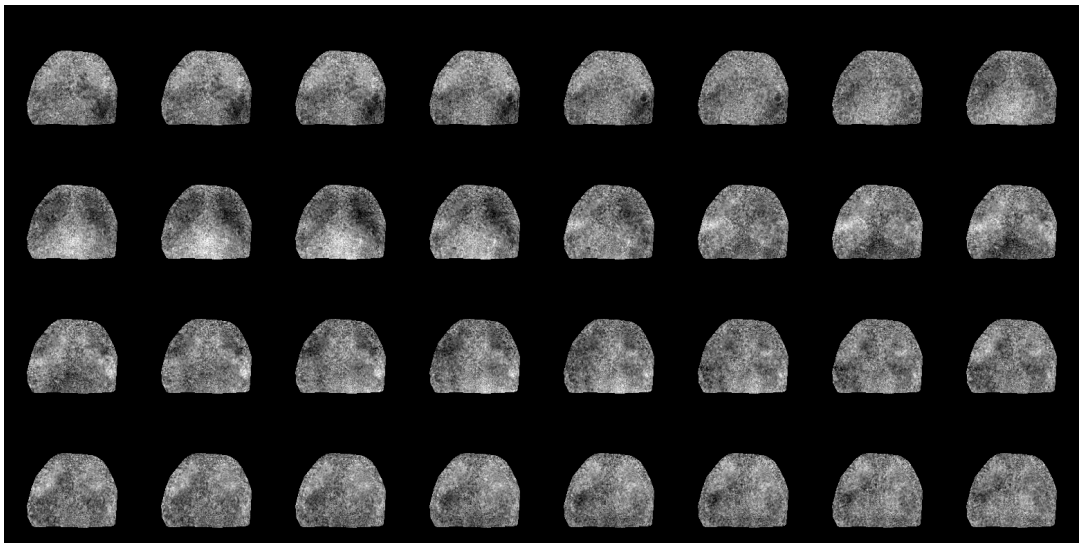


FIGURE 5.1: The fixed batch of samples used for the results.

models have been chosen by looking at the difference between the training and validation loss. For the evaluation of the results the peak signal to noise ratio was also considered, but considering that the original and the generated samples have the same dynamic range, this would have yielded no new information compared to the MSE [38].

5.5 Intermediate Conclusion

After this chapter we can answer the following sub question:

4. *How can the results generated by the network be validated?*

For the validation of the results of the networks we have chosen to use the mean squared error and the structural similarity index. This will allow us to compare the results of the different networks and architectures quantitatively and see which of them produced the best samples.

6 Results

After choosing the methods with which to validate the output of the trained networks we can now present the results of the different networks. This chapter will show the results of the different networks, compare them between each other and try to draw insight for the presented results.

6.1 Variational Autoencoder

As explained in Section 4.1.1, the VAE architecture was varied between two latent space dimensions, 256 and 512, and the input was varied between HbO data, HbR data and both combined. The resulting MSE and SSIM scores are thus generated for each of these networks generating six values in total. The values for the MSE and SSIM scores are shown in Tables 6.1a and 6.1b. The results show that the differences in the scores is minimal. Furthermore, the generated images, while sharp, are very similar, not close to the original which was supposed to be reconstructed. An example of the images generated can be seen in Figure 6.1, these are the samples generated by the VAE with a latent dimension size of 512 and using both the hemodynamic datasets. More results can be seen in Appendix C.

	Both	HbO	HbR
256	0.00766	0.00765	0.00765
512	0.00767	0.00766	0.00766

(A) MSE Scores of the VAE

	Both	HbO	HbR
256	0.76117	0.76146	0.76146
512	0.76110	0.76148	0.76150

(B) SSIM Scores of the VAE

TABLE 6.1: Resulting scores of the VAE

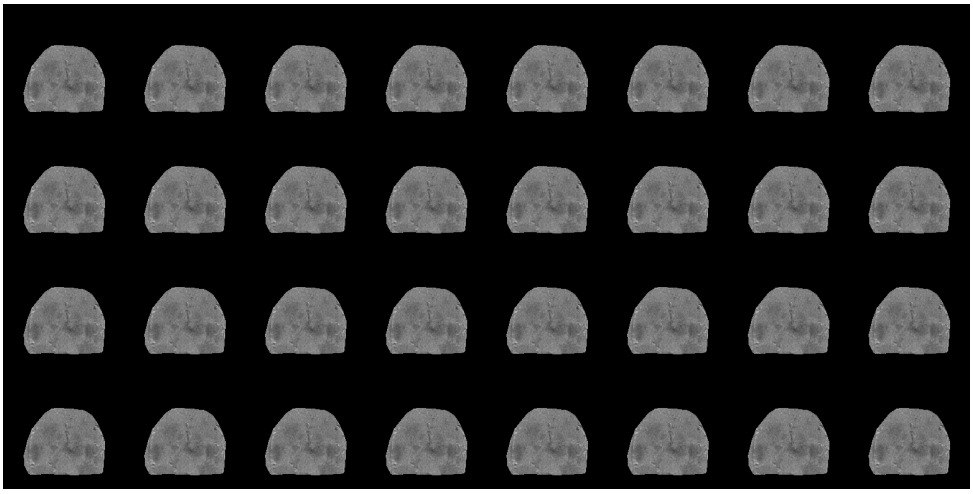


FIGURE 6.1: Example of the VAE results.

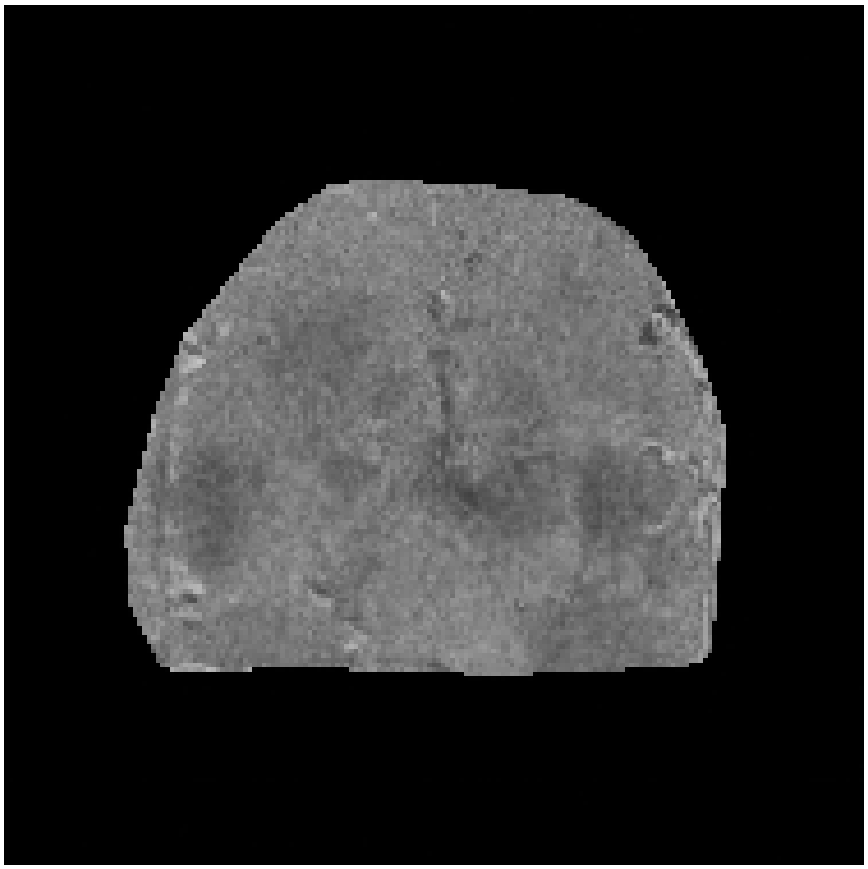


FIGURE 6.2: Detail view of a single image.

6.2 Variational Autoencoding Generative Adversarial Network

Analogous to the VAE, the VAE-GAN has six different networks with the same variations. The resulting MSE and SSIM scores are listed in Tables 6.2a and 6.2b. These scores are generated using the same image batch as before, seen in Figure 5.1. While these scores show more variety then with the VAE, they are still very similar. The generated results are blurry, which is not what we expected from GANs. Figure 6.3 shows an example of the images generated by the VAE-GAN, in this case the network with a latent dimension size of 256 and both the HbO and HbR data. More results can be seen in Appendix D.

	Both	HbO	HbR
256	0.00805	0.00896	0.00866
512	0.00858	0.00816	0.00875

(A) MSE Scores of the VAE-GAN.

	Both	HbO	HbR
256	0.73754	0.70463	0.71459
512	0.74243	0.73765	0.72000

(B) SSIM Scores of the VAE-GAN.

TABLE 6.2: Resulting scores of the VAE-GAN

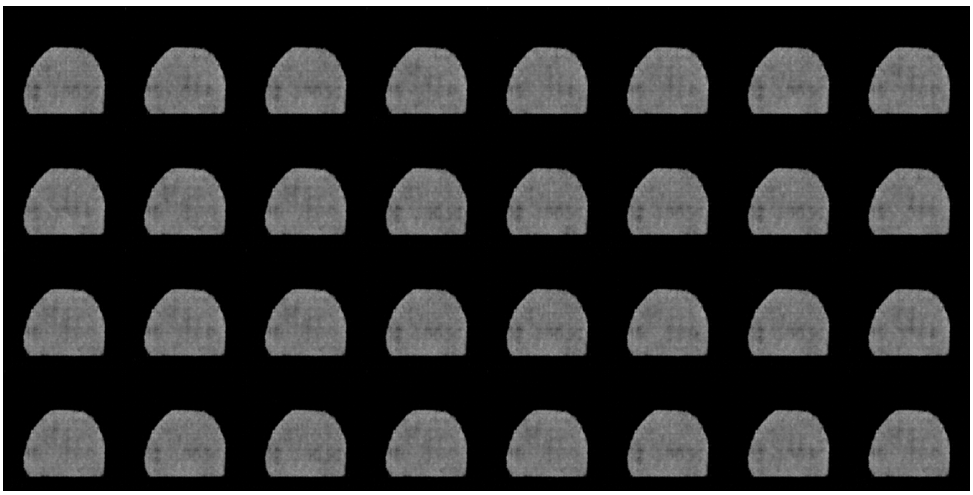


FIGURE 6.3: Example of the VAE results.

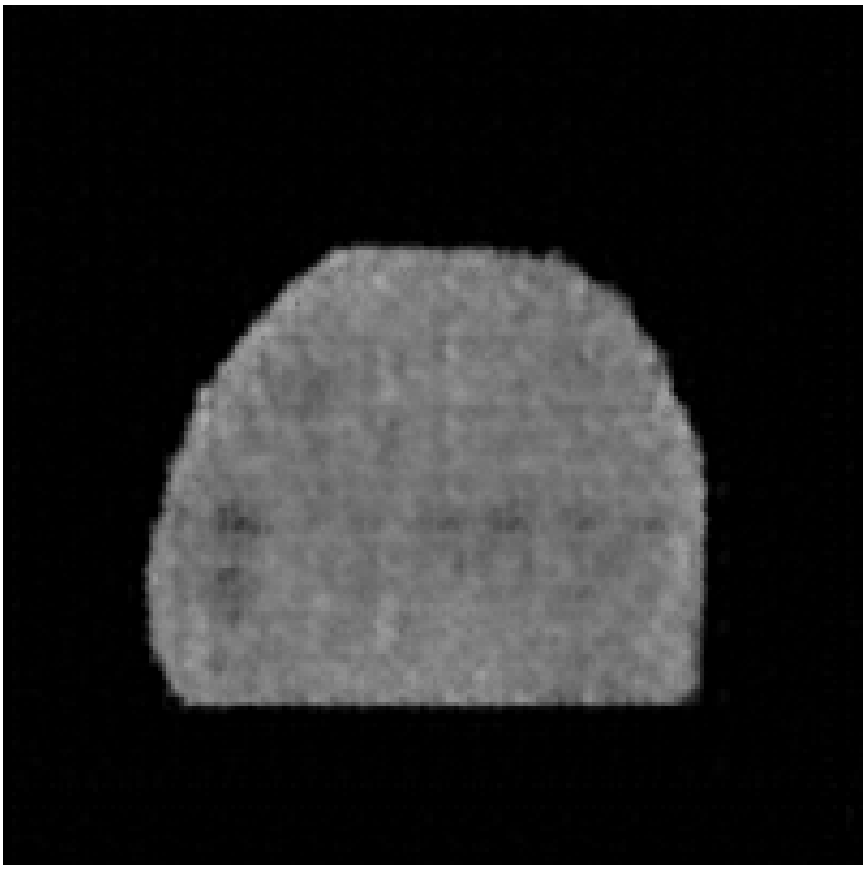


FIGURE 6.4: Detail view of a single image.

7 Discussion

Now that the results of the networks are obtained, we can make a quantitative compression between them and try to answer the the main research question:
How could artificial intelligence techniques be used to quantify a disconnect between the normal coupling, between hemodynamics and calcium, and abnormal/disease or age-related coupling?

7.1 Discussion of Results

Based on the results of the previous chapter, we can see that the variational autoencoders perform better overall, compared to the variational autoencoding generative adversarial networks. The difference in the quantitative scores is low however, with the average MSE difference only being 0.00086 and the SSIM varying by around 2%. This indicates that both architectures perform the same when applied to this problem. This performance is not very good, however. The networks failed to produce qualitatively satisfying examples of the calcium activity. This would indicate that the networks have not learned the neurovascular coupling between the hemodynamics and neuronal activity.

When looking at the qualitative results however, it would seem that the VAE is performing better. The image is sharper and more details of the brain seem the have been learned. The problem however, is that the all the generated images seem to be (close to) the same, whereas the images that are supposed to be reconstructed show a lot more variation. These results are interesting as we would have expected the reverse to happen. GANs are usually better for generating sharper images but suffer from mode collapse. It seems here that the VAE has suffered mode collapse but generates sharper images. We theorize that the VAE tries to average the images too much due to the use of the mean squared error as the target and a reduction of batch size might alleviate the problem somewhat. Regarding the lack of sharpness of the GAN, it would seem that the discriminator required more training before it could effectively be used to guide the training of the GAN towards creating accurate samples. The earlier results in Section 4.1.2 and Figure 4.2 also support this conclusion as with better learned features, the original method of using the learned similarity metric would likely also perform better.

For the VAEs there seem to be no difference in using HbO and/or HbR data. The scores of all the networks are very similar. For the VAE-GAN however, the SSIM scores of the HbO and HbR networks were slightly lower than those of the combined networks. This would support the conclusion that using more data in the network provides better results, which has traditionally been true for machine learning [13].

7.2 Intermediate Conclusion

Based on the discussion in the previous section we can now answer the final two sub questions:

5. *Out of the HbO and HbR data, which is the better predictor for the neuronal activity?*
6. *How can the neurovascular coupling be quantified using the results generated by the learned models?*

Since the scores for the VAEs regarding the different networks do not seem to differ their seem to be no difference in using the HbO or HbR data. For the VAE-GAN however, just using the HbO and HbR resulted in worse performing networks, while using both the datasets resulted in better performing networks. This leads to the conclusion that using both the datasets is the best solution and neither one is a better predictor.

Unfortunately, due to time constraints we were not able to train a network which can convincingly recreate neuronal activity from the hemodynamic data. We can see that the brain itself can be recreated, but the details of the activity are missing. This means that we cannot give a satisfying answer the final sub question. We have considered several ideas, like co-registering the brain to the Allen Atlas to see if the activations would match a certain brain region or creating a synthetic dataset which would demonstrate certain known principles in the neurovascular coupling like symmetry. Due to the lack of quality samples these however, these ideas did not come to fruition.

7.3 Conclusion

With the results discussed and all the sub-questions answered we can now formulate an answer to the main research question of this project:
How could artificial intelligence techniques be used to quantify a disconnect between the normal coupling, between hemodynamics and calcium, and abnormal/disease or age-related coupling?

We have shown that the variational autoencoder and the variational autoencoding generative adversarial network with convolutional and recurrent layers are potentially suited to producing recreations of the data. The results are not of the quality so that they could be used for verifying the neurovascular coupling, however, they do show potential. More work is needed to refine the networks and produce results that would be satisfactory.

7.4 Future Work

Based on the insights gained during this projects there are multiple aspects that warrant further investigation to find an answer to the research question. The first of these is knowing the exact hemodynamic delay of the mouse. In this research the average of 2.2 seconds was used, but this value varies between different mice. More accurate predictions could be made if the delay was measured exactly during the acquisitions. This was not relevant for the project for which this data was acquired however, so it was not measured.

As we have seen with the generated samples from the VAE-GAN in Figure 6.3 and the failed reconstructions using the learned similarity measure, the GAN could perform better if the discriminator was pre-trained on the calcium data. The instabilities in the network seem to stem from the fact that the discriminator was not performing adequately, and this might stabilise the network leading to better results.

References

[1] M. E. Raichle and M. A. Mintun, "Brain work and brain imaging," *Annual Review of Neuroscience*, vol. 29, no. 1, pp. 449–476, Jul. 21, 2006.

[2] B. R. White, A. Q. Bauer, A. Z. Snyder, B. L. Schlaggar, J.-M. Lee, and J. P. Culver, "Imaging of functional connectivity in the mouse brain," *PLoS ONE*, vol. 6, no. 1, e16322, Jan. 20, 2011.

[3] A. Grinvald, E. Lieke, R. D. Frostig, C. D. Gilbert, and T. N. Wiesel, "Functional architecture of cortex revealed by optical imaging of intrinsic signals," *Nature*, vol. 324, no. 6095, pp. 361–364, Nov. 1986.

[4] Y. Ma *et al.*, "Wide-field optical mapping of neural activity and brain haemodynamics: Considerations and novel approaches," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 371, no. 1705, p. 20 150 360, Oct. 5, 2016.

[5] C. Stosiek, O. Garaschuk, K. Holthoff, and A. Konnerth, "In vivo two-photon calcium imaging of neuronal networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 100, no. 12, pp. 7319–7324, Jun. 10, 2003.

[6] M. P. Vanni and T. H. Murphy, "Mesoscale transcranial spontaneous activity mapping in GCaMP3 transgenic mice reveals extensive reciprocal connections between areas of somatomotor cortex," *The Journal of Neuroscience*, vol. 34, no. 48, pp. 15 931–15 946, Nov. 26, 2014.

[7] G. Silasi, D. Xiao, M. P. Vanni, A. C. N. Chen, and T. H. Murphy, "Intact skull chronic windows for mesoscopic wide-field imaging in awake mice," *Journal of neuroscience methods*, vol. 267, pp. 141–149, Jul. 15, 2016.

[8] A. Badhwar, A. Tam, C. Dansereau, P. Orban, F. Hoffstaedter, and P. Bellec, "Resting-state network dysfunction in alzheimer's disease: A systematic review and meta-analysis," *Alzheimer's & Dementia : Diagnosis, Assessment & Disease Monitoring*, vol. 8, pp. 73–85, Apr. 18, 2017.

[9] M. Gorges, F. Roselli, H.-P. Müller, A. C. Ludolph, V. Rasche, and J. Kassubek, "Functional connectivity mapping in the animal model: Principles and applications of resting-state fMRI," *Frontiers in Neurology*, vol. 8, p. 200, May 10, 2017.

[10] P. Sirpal, R. Damseh, K. Peng, D. K. Nguyen, and F. Lesage, "Multimodal autoencoder predicts fNIRS resting state from EEG signals," *Neuroinformatics*, Aug. 10, 2021.

[11] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile: IEEE, Dec. 2015, pp. 4489–4497.

[12] Y. LeCun, C. Cortes, and C. Burges. "MNIST handwritten digit database, yann LeCun, corinna cortes and chris burges," MNIST handwritten digit database. (), [Online]. Available: `http://yann.lecun.com/exdb/mnist/` (visited on 06/23/2022).

[13] C. M. Bishop, *Pattern recognition and machine learning*, ser. Information science and statistics. New York: Springer, 2006, 738 pp.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, ser. Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2016, 775 pp.

[15] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 28, 2015.

[16] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv:1511.06434 [cs]*, Jan. 7, 2016.

[17] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using LSTMs," Feb. 16, 2015.

[18] Z. Hu, T. Turki, and J. T. L. Wang, "Generative adversarial networks for stochastic video prediction with action control," *IEEE Access*, vol. 8, pp. 63 336–63 348, 2020, Conference Name: IEEE Access.

[19] Y. Pang, J. Lin, T. Qin, and Z. Chen, "Image-to-image translation: Methods and applications," *arXiv:2101.08629 [cs]*, Jul. 3, 2021.

[20] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," Dec. 20, 2013.

[21] I. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, vol. 27, Curran Associates, Inc., 2014.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile: IEEE, Dec. 2015, pp. 1026–1034.

[23] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of StyleGAN," *arXiv:1912.04958 [cs, eess, stat]*, Mar. 23, 2020.

[24] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ISSN: 1938-7228, JMLR Workshop and Conference Proceedings, Jun. 14, 2011, pp. 315–323.

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[26] Y. LeCun *et al.*, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems*, vol. 2, Morgan-Kaufmann, 1990.

[27] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, USA: IEEE, Jun. 2010, pp. 2528–2535.

[28] I. Sutskever, J. Martens, and G. Hinton, "Generating text with recurrent neural networks," p. 8, Jun. 28, 2011.

[29] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *arXiv:1409.3215 [cs]*, Dec. 14, 2014.

[30] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[31] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," *arXiv:1211.5063 [cs]*, Feb. 15, 2013.

[32] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, BC, Canada: IEEE, May 2013, pp. 6645–6649.

[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1, 1997.

[34] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," arXiv, arXiv:1412.3555, Dec. 11, 2014, type: article.

[35] N. Gruber and A. Jockisch, "Are GRU cells more specific and LSTM cells more sensitive in motive classification of text?" *Frontiers in Artificial Intelligence*, vol. 3, 2020.

[36] M. Rosca, B. Lakshminarayanan, D. Warde-Farley, and S. Mohamed, "Variational approaches for auto-encoding generative adversarial networks," Jun. 15, 2017.

[37] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," *arXiv:1512.09300 [cs, stat]*, Feb. 10, 2016.

[38] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? a new look at signal fidelity measures," *IEEE Signal Processing Magazine*, vol. 26, no. 1, pp. 98–117, Jan. 2009, Conference Name: IEEE Signal Processing Magazine.

[39] M. E. Bakker, I. Djerourou, S. Belager, F. Lesage, and M. Vanni, "Alteration of functional connectivity despite the preservation of cerebral oxygenation during acute hypoxia," Unpublished, Unpublished, 2022.

[40] M. T. Valley *et al.*, "Separation of hemodynamic signals from GCaMP fluorescence measured with wide-field imaging," *Journal of Neurophysiology*, vol. 123, no. 1, pp. 356–366, Jan. 1, 2020.

[41] S. Bélanger, B. Oliveira Ferreira de Souza, C. Casanova, and F. Lesage, "Correlation of hemodynamic and fluorescence signals under resting state conditions in mice's barrel field cortex," *Neuroscience Letters*, vol. 616, pp. 177–181, Mar. 2016.

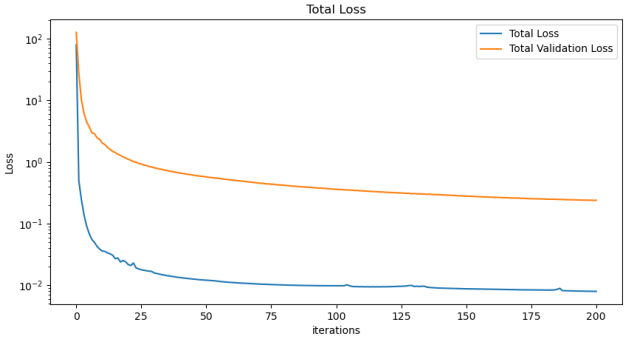
[42] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," arXiv, arXiv:1606.03498, Jun. 10, 2016, type: article.

[43] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improving variational inference with inverse autoregressive flow," Jun. 15, 2016.

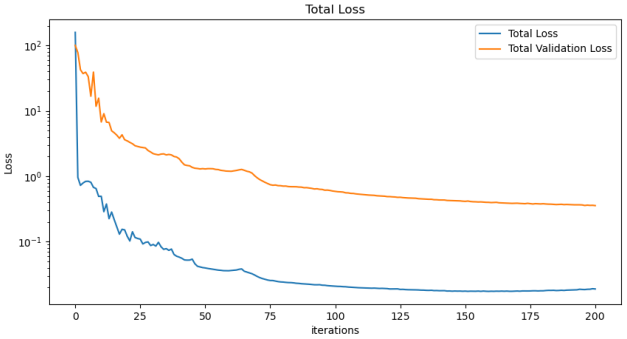
[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv, arXiv:1412.6980, Jan. 29, 2017, type: article.

[45] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of ADAM and beyond," p. 23, 2018.

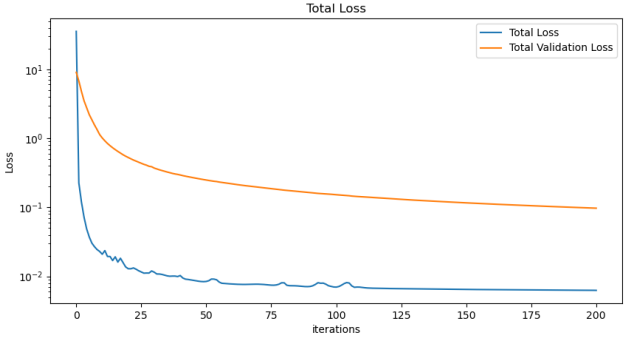
A VAE Training Graphs



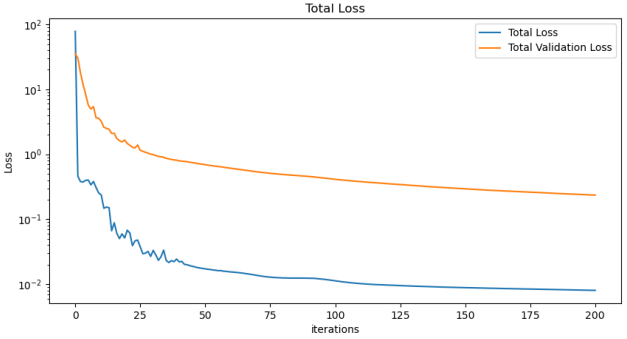
(A) Both datasets, 256 latent dimension size.



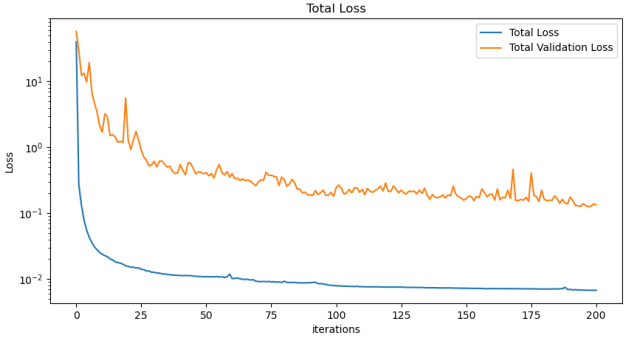
(B) Both datasets, 512 latent dimension size.



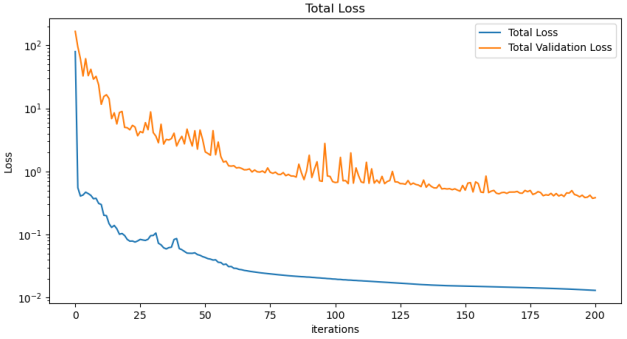
(C) HbO dataset, 256 latent dimension size.



(D) HbO dataset, 512 latent dimension size.



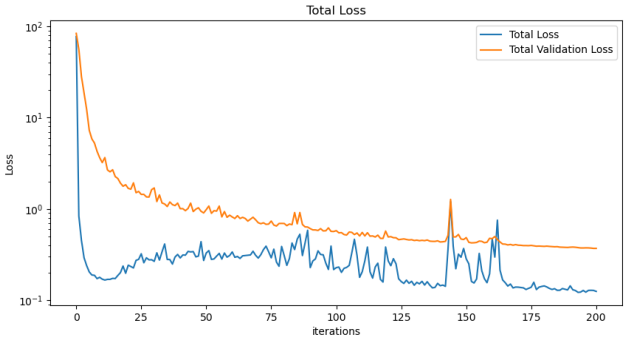
(E) HbR dataset, 256 latent dimension size.



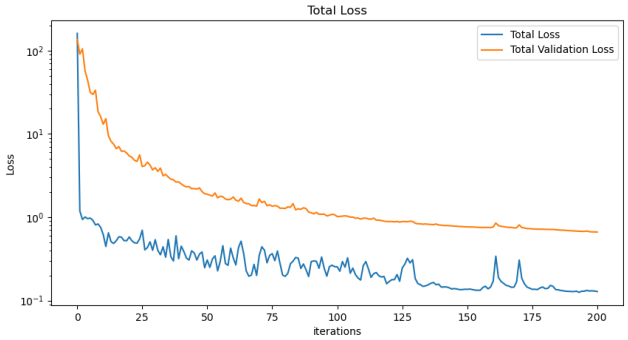
(F) HbR dataset, 512 latent dimension size.

FIGURE A.1: Plots of the training and validation losses of VAE training. Captions provide information about the used data and latent dimension size.

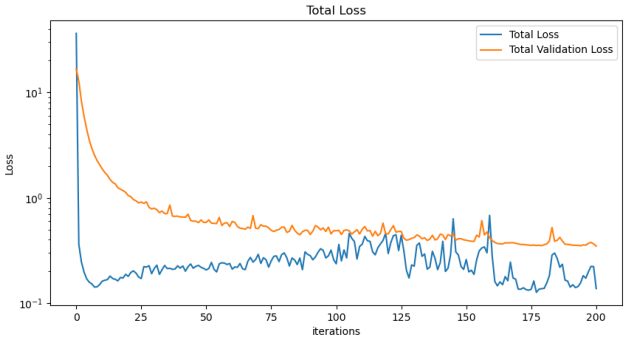
B VAE-GAN Training Graphs



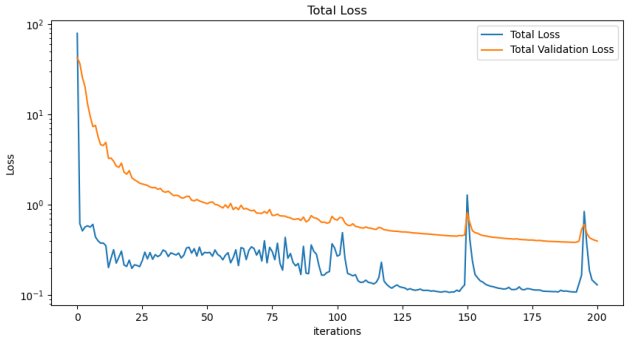
(A) Both datasets, 256 latent dimension size.



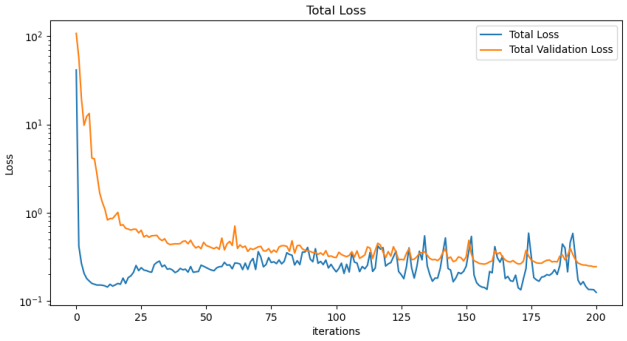
(B) Both datasets, 512 latent dimension size.



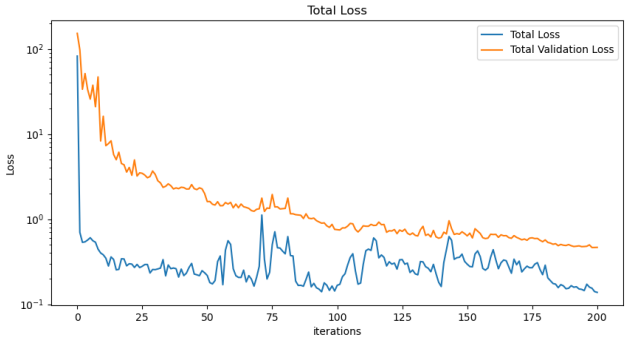
(C) HbO dataset, 256 latent dimension size.



(D) HbO dataset, 512 latent dimension size.



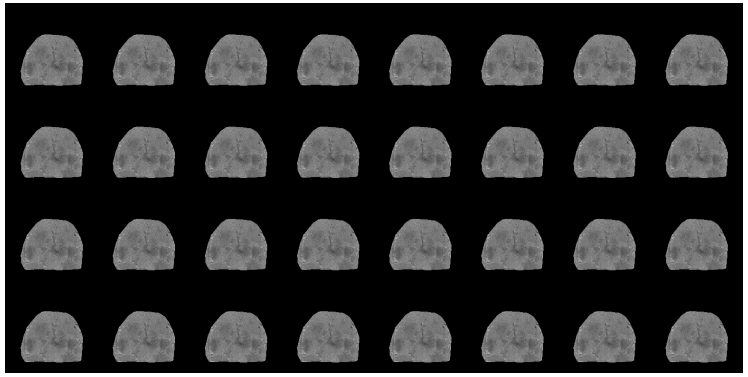
(E) HbR dataset, 256 latent dimension size.



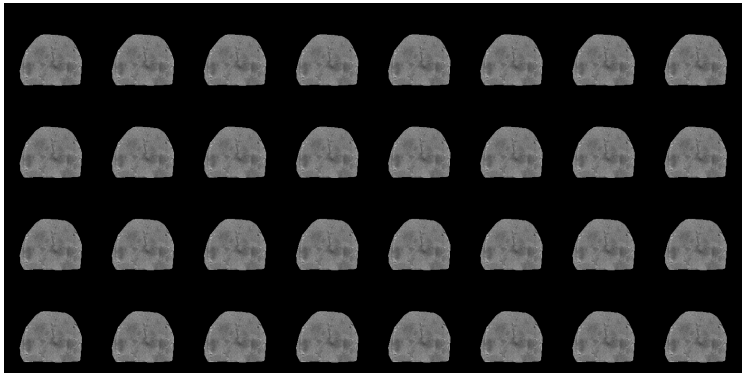
(F) HbR dataset, 512 latent dimension size.

FIGURE B.1: Plots of the training and validation losses of VAE-GAN training. Captions provide information about the used data and latent dimension size.

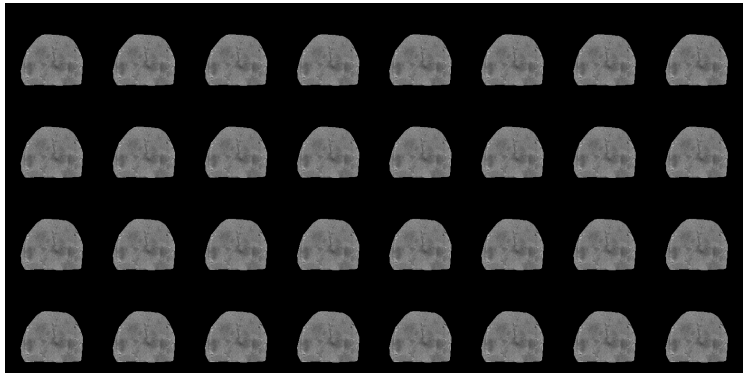
C VAE Results



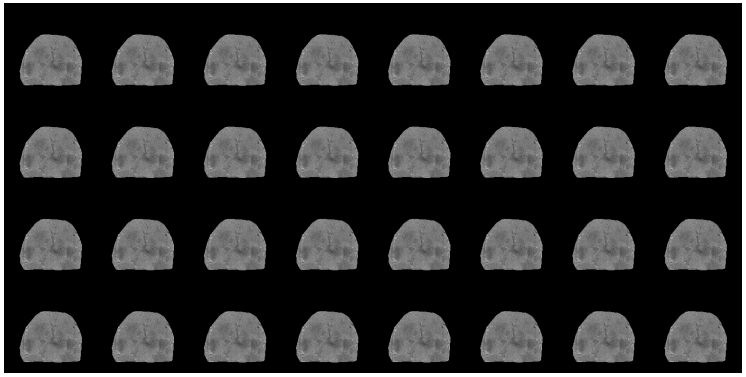
(A) Both datasets, 256 latent dimension size.



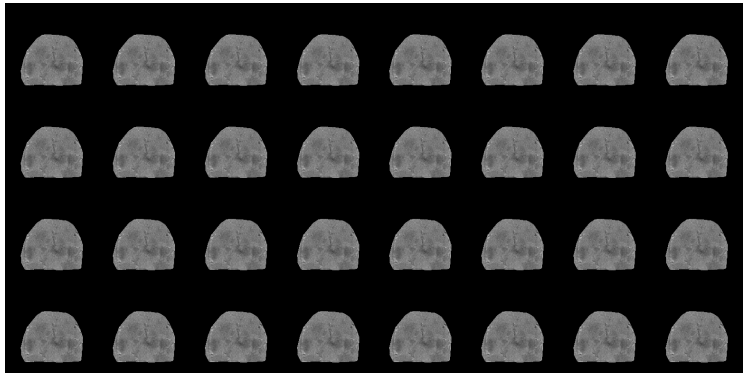
(B) Both datasets, 512 latent dimension size.



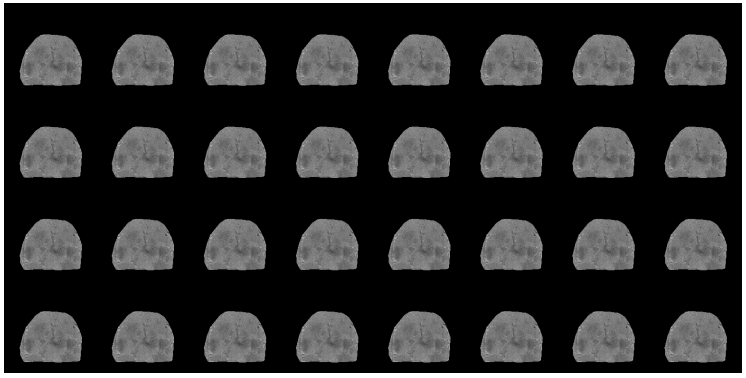
(C) HbO dataset, 256 latent dimension size.



(D) HbO dataset, 512 latent dimension size.



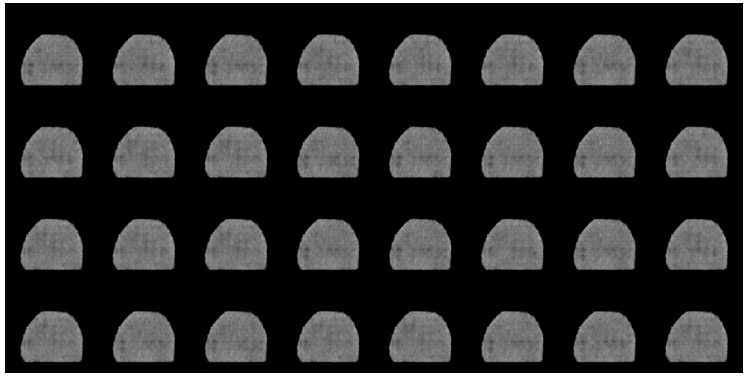
(E) HbR dataset, 256 latent dimension size.



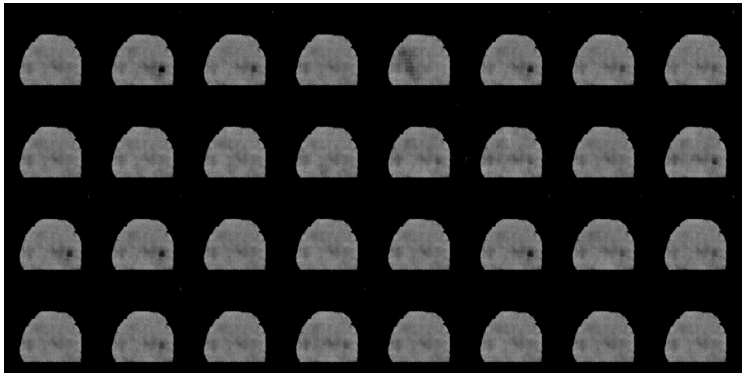
(F) HbR dataset, 512 latent dimension size.

FIGURE C.1: Examples of the results generated by the VAE. Captions provide information about the used data and latent dimension size.

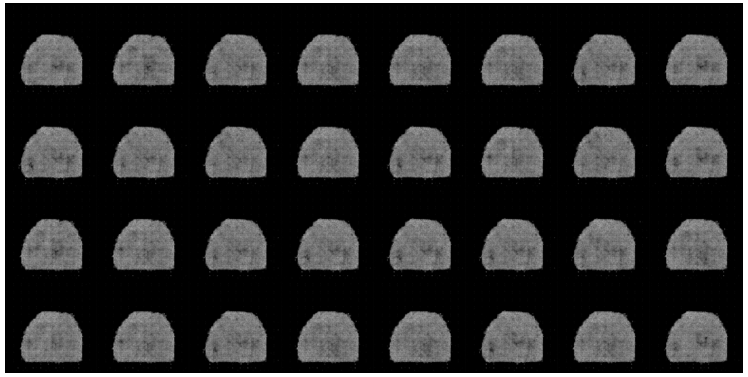
D VAE-GAN Results



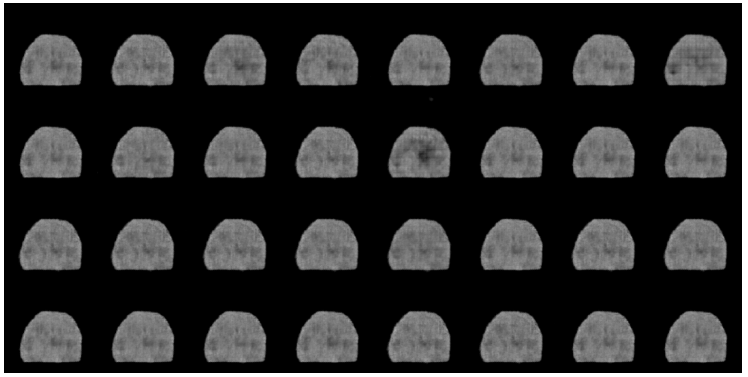
(A) Both datasets, 256 latent dimension size.



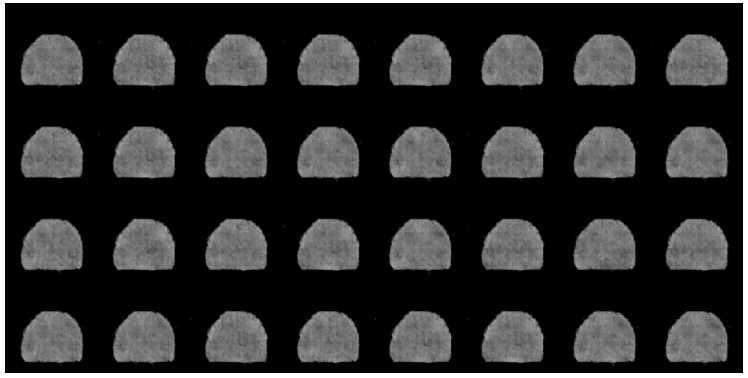
(B) Both datasets, 512 latent dimension size.



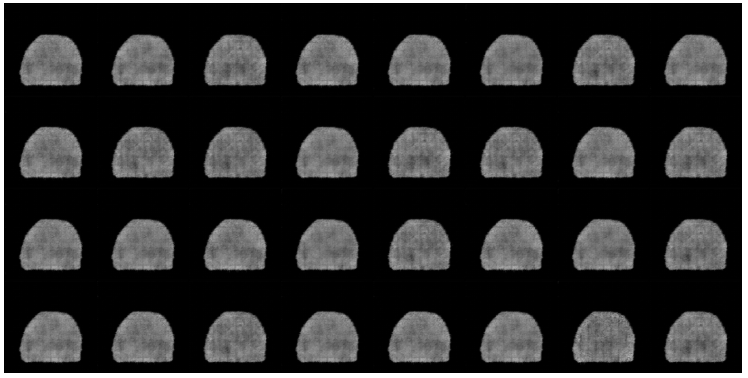
(C) HbO dataset, 256 latent dimension size.



(D) HbO dataset, 512 latent dimension size.



(E) HbR dataset, 256 latent dimension size.



(F) HbR dataset, 512 latent dimension size.

FIGURE D.1: Examples of the results generated by the VAE-GAN. Captions provide information about the used data and latent dimension size.