



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



Escola Tècnica
Superior d'Enginyeria
Informàtica

Dept. of Information Technology and Electrical Engineering
ETH Zürich
Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Machine Learning for Decoding Neuronal Activity

DEGREE FINAL WORK

Degree in Computer Engineering

Author: López Andrango, Kevin Danilo

Tutor: Juan Císcar, Alfonso

Course 2019-2020

*A mis padres, que con su apoyo
incondicional han hecho esto posible.*

Abstract

Calcium imaging has been proved to be an effective technique to record in-vivo neural activity. In this work we present a probabilistic generative model that attempts to capture the relationship between the stimulus and the recorded neural activity. However instead of working with inferred spikes from the recorded activity, we work with raw calcium image data. Working with raw calcium image data as well as with visual stimuli presents the challenge of dealing with high dimensional spaces, in our work we address this problem by means of variational autoencoders. The parameter estimation of a conditional probability distribution from which we can sample neural activity given a stimulus was done using variational inference. We also present a method to validate the quality of the generated samples.

Contents

Contents	iii
0 Prologue	1
1 Introduction	3
1.1 Motivation	3
1.2 Related Work	4
1.3 Setup	4
2 Representation Learning	7
2.1 The Learning Problem	7
2.2 Neural Networks	10
3 MLE with Neural Networks	13
3.1 Neural Encoding	13
3.2 Maximum Likelihood Estimation	13
4 Latent Representations	19
4.1 Disentangled Representation	19
4.1.1 Graphical Models	19
4.1.2 Evidence Lower Bound	21
4.1.3 Autoencoding Variational Bayes	24
4.1.4 β Variational Autoencoder	26
5 Probabilistic Neural Encoding	31
5.1 Proposed Graphical Model	31
5.2 Results	34
5.3 Validation	37
5.3.1 Decoding	37
5.3.2 Method	37

CONTENTS

6 Conclusion	41
Bibliography	43

Chapter 0

Prologue

This work was the result of my exchange year at the ETH Swiss Federal Institute of Technology back in 2018/2019, where I had the chance to take the lecture *Neural Systems* which was the seed originating the following thesis. There I discovered the potential that a computer scientist can also have by working in the intersection of both fields, neuroscience and computer science.

The project was originally offered by the Institute of Neuroinformatics and supervised by Prof. Dr. Mehmet Fatih Yanik from the ETH Neurotechnology group. The thesis was already graded but since there was no defense, which at ETH is only required at graduate studies, I had to follow the UPV guidelines resubmit my thesis and do a defense in order to give academic validity to my work.

I would like to express my continued appreciation to my daily advisor, Dr. Markus Marks, for his support and excellent advises while I was working on the project.

Chapter 1

Introduction

In this introductory chapter we briefly present the motivation of the project and the setup used to collect the data.

1.1 Motivation

Although not biologically plausible, artificial neural networks [12] can be proved [2] to be powerful function approximators capable of discovering highly non-linear relationships between two, assumed to be related, data sources. In this work we make use of recently developed algorithms in the literature, that implicitly use the universal approximation capability of feed-forward neural networks to estimate intractable probability distributions.

D. H. Hubel and T. N. Wiesel proved in their seminal paper [6] that there exist neurons that selectively fire to a specific stimulus' feature such as degree of inclination in the case of a white light bar as showed in [6]. With this in mind one could hypothesize that the spatial topology of a set of activated neurons at certain point in time, encodes the response to a presented stimulus, and consequently if the encoding is unique a mapping of stimulus to neural data can be assumed to exist, but of course that reductionist assumption is prone to error, and the neural activity at certain point in time is not only due to a presented stimulus, more factors that are out of control of the experimenter are also present and affect the neural activity observed. This raises the question of how we can take only the valuable information from the signal (the neural activity) and discard the part of the signal that is not informative to our purposes. Is here where we believe that unsupervised machine learning techniques can help, and consequently they play a major role in our work.

Along the development of this project we wanted to investigate up to what amount we can employ machine learning techniques to address the problem of neural population encoding and decoding. Trying to follow a structural

organization similar to a research paper, chapters 2, 3, 4 can be seen as the methods part, in those chapters a detailed explanation of the required theory to understand our work is presented. In chapter 5 we present the main result of our work, and finally in chapter 6 we end with the conclusions.

1.2 Related Work

To the best of our knowledge not so many work has been done on using raw calcium image data to design and train generative models of neural activity, nevertheless the variational inference framework, that plays a major role in our work, has been successfully applied to the analysis of neural spike activity and different researchers in the community have worked within this framework. A recent example can be seen in [11] in which the authors propose a generative model of cortical responses using sequential variational auto encoders.

Neural networks in general have also been used to predict spiking activity in response to natural images [1], further interesting work with relation to neuroscience has also been done and the interested reader is referred to [8], [3], to see more examples. We believe that these examples demonstrate empirically that the understanding and correct use of artificial neural networks can help neuroscientific research, and even though there exists certain skepticism in the neuroscience community about their inclusion as a common method to employ while doing research, this skepticism is mostly argued based on the lack of interpretation that a neural networks offers, however we believe that when we are dealing with highly dimensional data with thousand of degrees of freedom interpretation cannot be well defined, and therefore a well designed experiment that at some stage makes use of neural networks, should allow to empirically test the results without a strong dependency on the interpretation of the trained artificial neural network.

1.3 Setup

In order to carry out our work first a recording experiment was designed in which a continuous random walk of stimulus with the shape of a ball was presented to mildly anesthetized rats. A glass window was implanted in the brain region of interest, and then the imaging was done using 1-photon widefield calcium imaging using the indicator GCaMP6f, this indicator was previously transfected to the rats via a virus. The presentation of the stimuli and the acquisition of the neural data was done using a Nikon AZ100 and custom python software. In figure 1.1 we can see a diagram of setup used. After the neural data was acquired standard motion correction algorithms as well as image enhancement techniques were applied to improve the quality of the recordings.

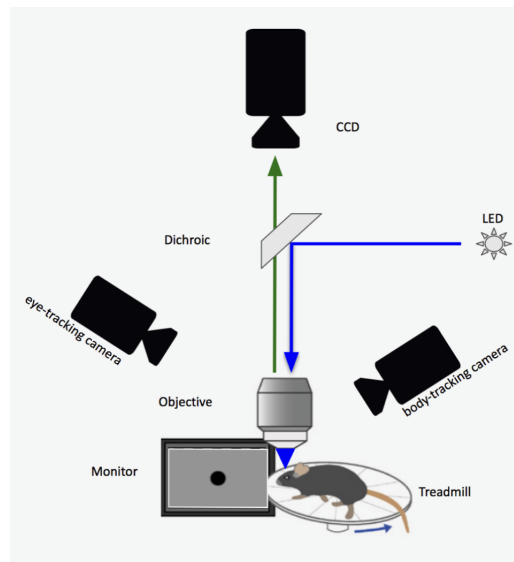


Figure 1.1: Setup of the recording.

Chapter 2

Representation Learning

In this chapter we introduce the principle of Empirical Risk Minimization [13] that proposes a framework to design algorithms with the objective of solving the learning problem, we follow this principle to perform neural population decoding in future chapters. In addition to it, the machine learning model that we use along this thesis will also be presented.

2.1 The Learning Problem

In the following we will consider the problem of object classification, in which we represent an object as a variable \mathbf{x} , and its corresponding class as a categorical or discrete variable c which takes values in $\{1, 2, \dots, k\}$, where k is the total number of classes ¹. We will assume that there exists a function that perfectly maps every object to its class

$$c = f^*(\mathbf{x}) \tag{2.1}$$

which is of course unknown for us. Therefore our objective will be to find a function \hat{f} within a set of functions such that

$$f^*(\mathbf{x}) \approx \hat{f}(\mathbf{x}) \tag{2.2}$$

for all \mathbf{x} in the object space. Having established our objective, remains the question of how we can design an algorithm that provides us with the proper function \hat{f} . A natural approach for obtaining our desired function \hat{f} , can be deviated inspired on how we learn to recognize objects. Usually a set of examples is presented to the learner and based in our inherent ability to generalize, we use the previous set of examples to further classify unseen objects. Consequently an algorithm \mathcal{A} that mimics that procedure will also need a set of examples. The previous defined procedure is what in the literature is also

¹We should keep in mind that no particular representation of the object x is assumed.

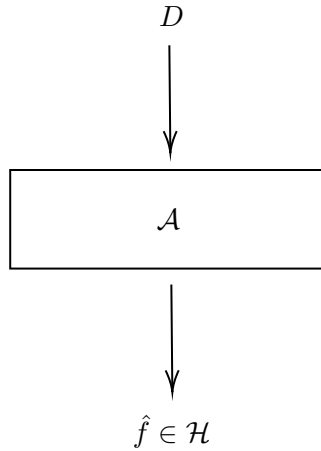


Figure 2.1: Abstract representation of the learning algorithm.

know as supervised learning, in this setting we usually represent a set of examples as a dataset $D = \{(\mathbf{x}^{(1)}, c^{(1)}), (\mathbf{x}^{(2)}, c^{(2)}), \dots, (\mathbf{x}^{(n)}, c^{(n)})\}$ in which each tuple in the set contains an object and its corresponding class, the diagram in figure 2.1 can serve as an illustration of how supervised learning works, an input D is passed to \mathcal{A} and as output we get a hypothesis \hat{f} .

Having defined an abstract procedure to mimic the process of learning, we can now move into the problem of how the algorithm will choose a proper function from a given set of functions that we call the hypothesis space \mathcal{H} . This is done by establishing a partial ordering in the hypothesis space such that for all the functions f_i within \mathcal{H} we have

$$\hat{f}_1 \leq \hat{f}_2 \leq \hat{f}_3 \leq \hat{f}_4 \dots \quad (2.3)$$

this partial ordering is induced by the algorithm \mathcal{A} and a function \mathcal{R} , usually called risk, that evaluates how well a particular hypothesis \hat{f} performs relative to the assigned task, \mathcal{R} can be more formally defined as

$$\mathcal{R} : \mathcal{H} \longrightarrow R \quad (2.4)$$

$$\hat{f} \longmapsto e \quad (2.5)$$

The reader should keep in mind that by selecting a risk function \mathcal{R} among others, we are imposing a bias towards some particular functions, and it can be the case that \mathcal{R} ranks $\hat{f}_i < \hat{f}_j$ but it's actually f_j who better approximates the true but unknown function f^* .

At this point we would like emphasize that it might be foolish to assume that an observation \mathbf{x} that comes from nature does not carry any inherent noise, therefore in the following we will define an observation \mathbf{x} as

$$\mathbf{x} = \text{Object} + \epsilon \quad (2.6)$$

this addition of uncertainty to our observations lead us to naturally work in the framework of probability theory and therefore we will assume the existence of an unknown probability distribution of the observations $p(\mathbf{x})$. Within this framework any function that we define on the random variable \mathbf{x} will implicitly define a new probability distribution, hence f^* will implicitly define a probability distribution $p(c | \mathbf{x})$, we make this point here in order to justify the following derivations.

For now on assume that we have access to f^* and our algorithm \mathcal{A} has given us one particular \hat{f} . A common approach in the context of object classification is to define \mathcal{R} as

$$\mathcal{R}(f^*(\mathbf{x}), \hat{f}(\mathbf{x})) = \begin{cases} 1 & \text{if } f^*(\mathbf{x}) \neq \hat{f}(\mathbf{x}) \\ 0 & \text{otherwise} \end{cases} \quad (2.7)$$

once we defined a risk function a reasonable next step will be to minimize the following expectation

$$\mathbf{E}[\mathcal{R}(c, \hat{c})] = \sum_c \left(\int_R \mathcal{R}(c, \hat{c}) p(c, \mathbf{x}) d\mathbf{x} \right) \quad (2.8)$$

where $\hat{c} = \hat{f}(\mathbf{x})$ and where the joint probability distribution $p(c, \mathbf{x})$ is obtained by the definition of conditional probability

$$p(c, \mathbf{x}) = p(c | \mathbf{x})p(\mathbf{x}) \quad (2.9)$$

we can see equation 2.8 as a functional over the hypothesis space \mathcal{H} that maps a function to the expected risk of miss classifying a random sample. Therefore our objective can be stated as follows:

$$\arg \min_{\hat{f} \in \mathcal{H}} \mathcal{F}(\hat{f}) = \mathbf{E}[\mathcal{R}(c, \hat{c})] \quad (2.10)$$

however as it's obvious $p(c, \mathbf{x})$ is unknown to us, which implies that it's impossible to minimize the above equation. Nevertheless by the law of large numbers we know that we can estimate the expectation by:

$$\mathbf{E}[\mathcal{R}(c, \hat{c})] \approx \frac{1}{N} \sum_{i=1}^N \mathcal{R}(c^{(i)}, f(\mathbf{x}^{(i)})) \quad (2.11)$$

and in the limit the result will converge to the true expectation, the N samples that we use to approximate the expectation are the samples that belong to our dataset D , and we assume that each tuple inside D is sampled from the true joint distribution $p(c, \mathbf{x})$. We call the above equation the *empirical risk*, since it depends on empirical data.

What the principle of Empirical Risk Minimization states is that the learning algorithm \mathcal{A} should output hypothesis function \hat{f}^* such that

$$\hat{f}^* = \arg \min_{f \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^N \mathcal{R}(c^{(i)}, f(\mathbf{x}^{(i)})) \quad (2.12)$$

and we hope that although that learning algorithm doesn't have full access to the true distribution of the data, $p(c, \mathbf{x})$, the returned hypothesis will generalize to new unseen samples, machine learning models such as neural networks in conjunction with their training algorithms follow this principle. Theoretical guarantees of the ERM principle have been shown for the learning problem but will not be presented in this thesis, the interested reader can look for more information in [14].

2.2 Neural Networks

Back in the 80's David E. Rumelhart, Geoffrey E. Hinton & Ronald J. Williams published their seminal paper 'Learning representations by back-propagating errors', in which they successfully applied the backpropagation algorithm (already known in the control theory literature) to train a multi-layer perceptron also known as a feed-forward neural network. Neural networks have gained high popularity among the last years due to their outstanding results in fields such as computer vision or natural language processing.

Neural networks enter within the gradient based learning approach in which a *differentiable* risk function $\mathcal{L}(\theta)$, also known as loss or error, is used to guide the optimization method. We emphasize the importance of being differentiable since due to this constraint the 0-1 loss function cannot be used in the learning algorithm in case we are dealing with a classification problem. A diagram of the model can be seen in figure 2.2, for a more detailed description we refer the reader to [12].

At a glance a neural network can be defined as composite parametrized function that process the input X_0 by a set of feed-forward layers, this can be seen better in the following way

$$X_n = F(Y_n) \quad (2.13)$$

$$Y_n = W_n X_{n-1} \quad (2.14)$$

in which W_n represents the weights of the model at layer n , one mandatory requirement that this type of functions should have is a non-linear differentiable function in at least one layer. As described in [12] the training consists on recurrently updating the weights of the model by applying the following

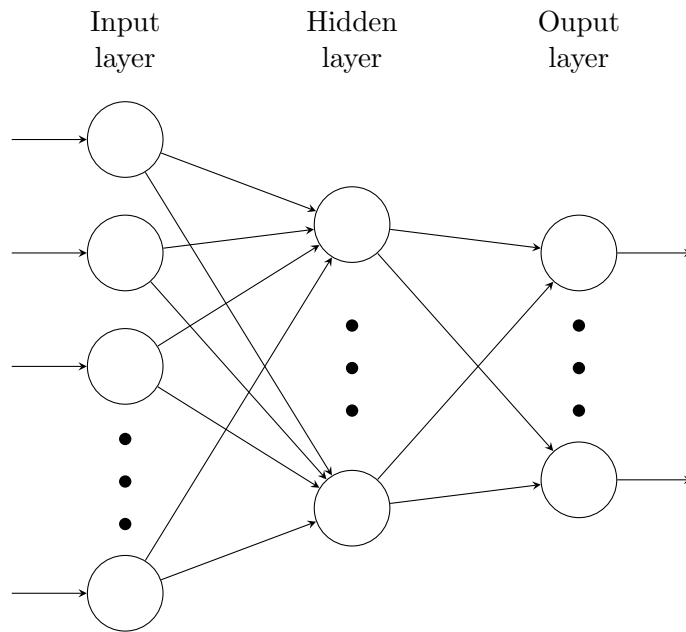


Figure 2.2: Diagram of a Neural Network with one hidden layer

equations [9]

$$\frac{\partial \mathcal{L}}{\partial Y_n} = \mathcal{F}'(Y_n) \frac{\partial \mathcal{L}}{\partial X_n} \quad (2.15)$$

$$\frac{\partial \mathcal{L}}{\partial W_n} = X_{n-1} \frac{\partial \mathcal{L}}{\partial Y_n} \quad (2.16)$$

$$\frac{\partial \mathcal{L}}{\partial X_{n-1}} = W_n^T \frac{\partial \mathcal{L}}{\partial Y_n} \quad (2.17)$$

gradient descent can be now applied to minimize the loss function \mathcal{L} .

Chapter 3

MLE with Neural Networks

In this chapter we briefly present the assumptions that we take in order to perform neural population encoding with the recorded data, we also review maximum likelihood estimation of probability distributions parametrized by artificial neural networks, which will help us in the encoding task.

3.1 Neural Encoding

In our approach we assume that the relation between the neural activity and the stimulus follows an invertible deterministic mapping

$$x = \mathcal{F}(s) \tag{3.1}$$

in which s represents the stimulus and x represents the neural activity. In our work we aim to find \mathcal{F} as well as \mathcal{F}^{-1} . However due to the nature of the data and the techniques we use to record the neural activity, it is reasonable to expect that an observation x carries some implicit noise, and therefore the mapping of our observations actually follow

$$x = \mathcal{F}(s) + \epsilon \tag{3.2}$$

and consequently instead of trying to find \mathcal{F} we should try to estimate $p(x|s)$.

Before addressing the problem of estimating $p(x|s)$ we will first introduce to the reader an easy example of how we can make use of feed-forward neural networks in the context of maximum likelihood estimation.

3.2 Maximum Likelihood Estimation

As explained in the introduction we used a random walk to generate the stimulus dataset $\mathcal{S} = \{s^{(1)}, s^{(2)}, \dots, s^{(n)}\}$, each $s^{(i)}$ is represented by a vector in B^M in which $B = \{0, 1\}$, this binary representation allows us to easily define

a probability distribution of a stimulus s , by assuming statistical independence between neighbors pixels so that the distribution of s can be factorized as

$$p(s; \theta) = \prod_{j \leq M} p(s_j; \theta) \quad (3.3)$$

in which each $p(s_j; \theta)$ follows a bernoulli distribution $\text{Bern}(\lambda)$. Although one can argue that the assumption of independence between pixel neighbors does not match reality, we will see later on that for the purpose of modeling simple visual stimulus does not suppose a big problem, nevertheless if more complicated stimulus are needed to be modeled one can establish a partial ordering over the set of all pixels and factorize the probability distribution as follows

$$p(s; \theta) = \prod_{j \leq M} p(s_j | s_{j-1}, s_{j-2}, \dots, s_1; \theta), \quad (3.4)$$

this type of models are called autoregressive models.

After fitting $p(s; \theta)$ we will be able to assign probabilities to any possible binary visual stimulus, and for instance, in a regression or classification setting in which s is the ground truth and \hat{s} is the estimate $\log p_\theta(s = \hat{s})$ can serve as metric to measure the quality of the estimation. However for future experiments we would like to not only be able to assign probabilities, but also to control how the stimulus is generated based on a set of representative features that are in control of the experimenter. These set of representative features should be as small as possible so that few degrees of freedom are involved in the experiment.

In the context of visual stimulus that lives in a high dimensionality space, the choice of which features are representative is highly non-trivial for instance when we deal with natural images, we believe that this choice should be made as objectively as possible, and for that purpose latent variable models play a major role as we will see in future chapters. Since in our project the stimulus is just a circular blob with fixed radius that moves around a square, we temporarily take the freedom to assume that the set of representative variables is formed by the x and y coordinates of the plane that are located in the middle of the blob, this assumption will be avoided in the next section and we take it here just to easily show the use of neural networks on fitting probability distributions. In figure 3.1 we can see a set visual stimulus with centroid coordinates in red.

The visual stimulus dependence on the centroid will be modeled as follows

$$p(s | c; \theta) = \prod_{j \leq M} p(s_j; \mathcal{F}(c)_j) \quad (3.5)$$

in which the vector of parameters θ is produced by the output of the function $\mathcal{F}(c)$, note that as can be seen in the above equation, \mathcal{F} provides the λ

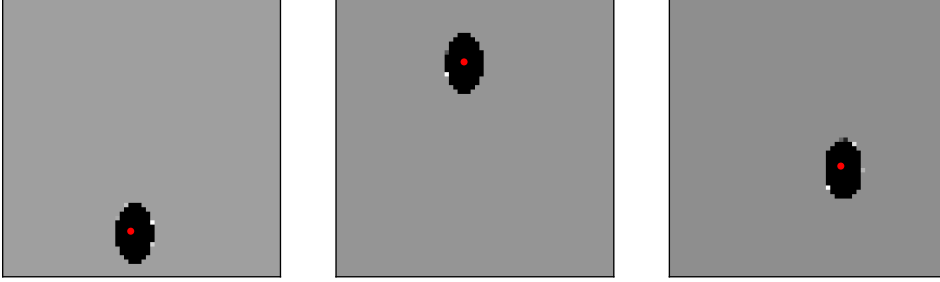


Figure 3.1: Visual stimulus with the computed centroid at different time steps of the random walk.

parameter for each bernoulli probability distribution that governs a pixel in the image as we defined in 3.3. A slightly more formal definition of \mathcal{F} is

$$\mathcal{F}: \mathcal{C} \longrightarrow R^M \quad (3.6)$$

$$c \longmapsto \boldsymbol{\theta} \quad (3.7)$$

The decision of modeling the dependence by a non linear deterministic function allows us to given input coordinates sample visual stimulus from $p(s | c; \boldsymbol{\theta})$ i.e we can control the generation of visual stimulus and therefore create patterns of stimuli with just a set of representative features, in our case the centroids.

As stated at the start of the section 3.1, we will perform maximum likelihood estimation to fit each of the $p(s_j | c; \boldsymbol{\theta})$, and we will use a feed-forward neural network to approximate $\mathcal{F}(c)$ with $\mathcal{F}(c, W)$ in which W represents the weights of the neural network. In order to work in a supervised setting, for each $s^{(i)}$ we computed it's respective centroid and we formed a new dataset $\mathcal{S}' = \{(s^{(1)}, c^{(1)}), \dots, (s^{(i)}, c^{(i)})\}$, in which $c^{(i)}$ represents the centroid, in figure 3.2 we can see a scatter plot of all the computed centroids, as expected, given that we used a random walk to generate the blobs, most of the space is filled.

The sum of the individual likelihoods

$$\mathcal{L}(W) = \sum_i^N \log p(\mathbf{s}^{(i)} | \mathbf{c}^{(i)}; \mathcal{F}(c^{(i)}, W)) \quad (3.8)$$

$$= \sum_i^N \sum_j^M \log p(\mathbf{s}_j^{(i)}; \theta_j) \quad (3.9)$$

$$= \sum_i^N \sum_j^M \mathbf{s}_j^{(i)} \log \theta_j + (1 - \mathbf{s}_j^{(i)}) \log(1 - \theta_j) \quad (3.10)$$

will be used as the function to optimize. While implementing the above function one should take care of numerical stability so it's recommended to use

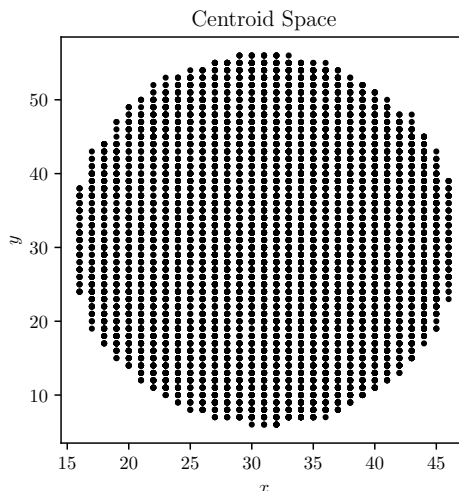


Figure 3.2: Centroids of generated by the random walk.

instead

$$\mathcal{L}(W) = \sum_i^N \sum_j^M \mathbf{s}_j^{(i)} \log(\theta_j + \epsilon) + (1 - \mathbf{s}_j^{(i)}) \log(1 - \theta_j + \epsilon) \quad (3.11)$$

in which epsilon is arbitrarily small. As a matter of routine we use mini-batch gradient ascend to find

$$W^* = \arg \max \mathcal{L}(W), \quad (3.12)$$

the reader should keep in mind that in the literature stochastic gradient descent is normally used as synonym to mini-batch gradient descent, we make this point since in future chapters we will use *SGD* as it's properly defined, using only one sample to update the weights, the details of the neural network architecture used for this model are omitted since just a fully connected model will do the job, in figure 3.3a we can see a plot of the loss function per evaluated batch, the plot clearly shows that the non-convex optimization process converges fast towards a local minimma.

Once we have found W^* we implicitly have fitted $p(s | c; \theta)$ for each $s \in \mathcal{S}$. As mentioned before we can make use of $p(s | c; \theta)$ as a metric or as an anomaly detector in case we are in working in a neural decoding experiment. For the purpose of illustration consider that for a fixed stimulus s_0 and it's centroid $c_0 = [x, y]$ we compute the respective θ_0 via $\mathcal{F}(c_0, W^*)$ which also provides us p_0 , if now we keep fixed x and we vary y uniformly to obtain a small set of of visual stimulus $\{s_i\}_{i=1}^k$ via sampling from their respective p_i , we would expect to have a curve

$$y = \log(p_0(s_i | c_0; \theta_0)) \quad (3.13)$$

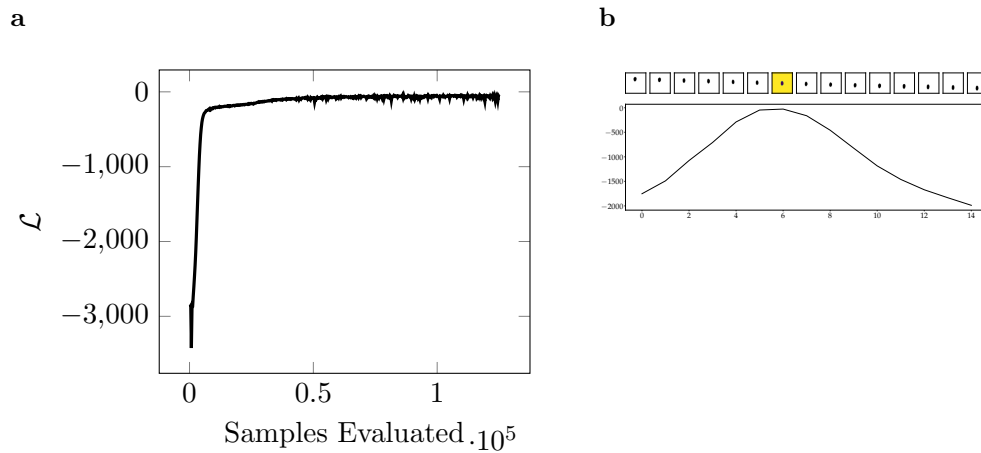


Figure 3.3: (a) Sum of the individual elbos per processed batch during the training. (b) Plot of the log of p_0 for each stimulus in $\{s_i\}_{i=1}^k$

with a peak in the vicinity of s_0 . Figure 3.3a shows the curve we got, in which the stimulus with the yellow background is s_0 , as expected the samples that fall far away of s_0 get much less probability from the ones that fall more near to s_0 .

We conclude this chapter by summarizing the important insights that will allow us to further progress in our attempt to estimate $p(x|s)$:

- Neural Networks are able to capture highly-non linear relationships between two sources of data.
- We can easily use neural networks to model the parametrization of a probability distribution.

Latent Representations

In the previous chapter we showed a simple example of how we can control the generation of a visual stimulus, however a necessary step was to first hand compute a set of features that we assumed control the generative process, whereas this can be done in the case of simple visual stimulus, when more complex images are involved, the choice of which features should be selected might lead to different interpretations when asked to different experimenters. In order to try to address this issue, in this chapter we will introduce the variational autoencoder [7] and later on a modification of it named β -variational autoencoder [5] which adds a structural bias in the latent factors that will suit our needs.

4.1 Disentangled Representation

Even though the community still lacks a proper theoretical definition of what disentanglement in the latent factors means, an intuitive description can be made if we consider an observation as a set of independent ground truth features, the variation of a disentangled latent factor of the observation will only affect one feature and no others. Before diving on how we can obtain a set of disentangled latent factors from one observation we first need to briefly introduce graphical models.

4.1.1 Graphical Models

We can decompose the underlying generation of a visual stimulus into two steps:

1. First a set of hidden latent variables is sampled from a probability distribution $p(\mathbf{z}; \theta)$. These latent variables might represent correlated or disentangled features of the visual stimulus such as shape, color, position and so on.

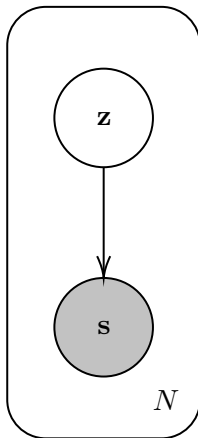


Figure 4.1: Graphical model of the generation of a stimulus.

2. We use the sampled latent variables to generate a visual stimulus by sampling from $p(\mathbf{s} | \mathbf{z}; \theta)$,

In figure 4.1 we can see a graphical representation of the described process, the shadow nodes represent observed variables and the white nodes represent hidden variables, the edges represent a conditional structure on the joint probability distribution which allows us to factorize it as

$$p(\mathbf{s}, \mathbf{z}; \theta) = p(\mathbf{s} | \mathbf{z}; \theta)p(\mathbf{z}; \theta), \quad (4.1)$$

the N at the bottom represents the number of repetitions or observed samples. The example showed in the previous section 3.2 can be described as a graphical model in which all the variables involved were observed.

Usually a group of observations assumed to be produced by the graphical model is collected in a dataset $D = \{\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \dots, \mathbf{s}^{(n)}\}$, in this scenario the latent factors \mathbf{z} corresponding to each observation in D are unknown to us as well as θ^* . In order to obtain θ^* we can attempt to compute the likelihood of the observations

$$\mathcal{L}(\theta) = \prod_{i \leq N} p(\mathbf{s}^{(i)}; \theta) \quad (4.2)$$

and find the θ^* by $\arg \max_{\theta} \mathcal{L}(\theta)$, however the marginal distribution

$$p(\mathbf{s}; \theta) = \int p(\mathbf{s} | \mathbf{z}; \theta)p(\mathbf{z}; \theta) d\mathbf{z} \quad (4.3)$$

depending of how the latent factors are defined might be intractable to compute, this implies that the posterior distribution of the latent variables

$$p(\mathbf{z} | \mathbf{s}; \theta) = \frac{p(\mathbf{s} | \mathbf{z}; \theta)p(\mathbf{z}; \theta)}{p(\mathbf{s}; \theta)} \quad (4.4)$$

is also intractable since it involves the computation of the marginal $p(\mathbf{s}; \theta)$ in the denominator, this fact supposes us a problem since we want to discover what represents each latent factor and how a variation affects the observed stimulus. In order to bypass this issue we will introduce in the next subsection a lower bound of the marginal distribution 4.3, that will allow us to approximate the intractable posterior distribution 4.4.

4.1.2 Evidence Lower Bound

The lower bound of the marginal distribution $p(\mathbf{s}; \theta)$ can be derived in two different ways, the key in both derivations is the introduction of a parametrized variational posterior $q(\mathbf{z} | \mathbf{s}; \phi)$ that we will approximate towards $p(\mathbf{z} | \mathbf{s}; \theta)$ so that in a local minimum ϕ^* of an optimization criterion we will have that

$$q(\mathbf{z} | \mathbf{s}; \phi^*) \approx p(\mathbf{z} | \mathbf{s}; \theta^*), \quad (4.5)$$

this variational posterior is usually selected from a family of probability distributions that are more tractable to compute than the true posterior $p(\mathbf{z} | \mathbf{s}; \theta)$. A common choice is to define $q(\mathbf{z} | \mathbf{s}; \phi)$ as a factorized probability distribution over the latent factors

$$q(\mathbf{z} | \mathbf{s}; \phi) = \prod_{j \leq M} q(\mathbf{z}_j | \mathbf{s}; \phi_j) \quad (4.6)$$

but in some cases this factorized assumption is too strong, and normally the true posterior does not live in the space of factorized probability distributions, we will see along this section how we can relax this assumption.

Before starting with the first derivation of the lower bound we recall to the reader the Jensen's inequality which in the context of probability theory states that for any concave function the following inequality holds

$$\varphi(\mathbb{E}[X]) \geq \mathbb{E}[\varphi(X)]. \quad (4.7)$$

Note that equation 4.3 can also be seen as

$$p(\mathbf{s}; \theta) = \mathbb{E}_{\mathbf{z} \sim p_\theta(\mathbf{z})} [p(\mathbf{s} | \mathbf{z}; \theta)] \quad (4.8)$$

and we can change under which probability distribution the expectation is taken by introducing $q(\mathbf{z} | \mathbf{s}; \phi)$ as follows

$$\mathbb{E}_{\mathbf{z} \sim p_\theta(\mathbf{z})} [p(\mathbf{s} | \mathbf{z}; \theta)] = \int p(\mathbf{s} | \mathbf{z}; \theta) p(\mathbf{z}; \theta) d\mathbf{z} \quad (4.9)$$

$$= \int p(\mathbf{s} | \mathbf{z}; \theta) p(\mathbf{z}; \theta) \frac{q(\mathbf{z} | \mathbf{s}; \phi)}{q(\mathbf{z} | \mathbf{s}; \phi)} d\mathbf{z} \quad (4.10)$$

$$= \int p(\mathbf{s}, \mathbf{z}; \theta) \frac{q(\mathbf{z} | \mathbf{s}; \phi)}{q(\mathbf{z} | \mathbf{s}; \phi)} d\mathbf{z} \quad (4.11)$$

$$= \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z} | \mathbf{s}; \phi)} \left[\frac{p(\mathbf{s}, \mathbf{z}; \theta)}{q(\mathbf{z} | \mathbf{s}; \phi)} \right] \quad (4.12)$$

by taking logarithm on both of sides of 4.11 and applying Jensen's inequality we obtain what in the literature is called the evidence lower bound or elbo

$$\begin{aligned}\log p(\mathbf{s} ; \theta) &= \log \left(\int p(\mathbf{s}, \mathbf{z} ; \theta) \frac{q(\mathbf{z} | \mathbf{s} ; \phi)}{q(\mathbf{z} | \mathbf{s} ; \phi)} d\mathbf{z} \right) \\ &\geq \int \log \left(\frac{p(\mathbf{s}, \mathbf{z} ; \theta)}{q(\mathbf{z} | \mathbf{s} ; \phi)} \right) q(\mathbf{z} | \mathbf{s} ; \phi) d\mathbf{z} \\ &= \mathbb{E}_{z \sim q(\mathbf{z} | \mathbf{s}; \phi)} [\log p(\mathbf{s}, \mathbf{z} ; \theta) - \log q(\mathbf{z} | \mathbf{s} ; \phi)]\end{aligned}\tag{4.13}$$

if we re-arrange the terms inside the expectation we can conclude that

$$\begin{aligned}\log p(\mathbf{s} ; \theta) &\geq \mathbb{E}_{z \sim q(\mathbf{z} | \mathbf{s}; \phi)} [\log p(\mathbf{s} | \mathbf{z} ; \theta)] + \mathbb{E}_{z \sim q(\mathbf{z} | \mathbf{s}; \phi)} \left[\log \frac{p(\mathbf{z} ; \theta)}{q(\mathbf{z} | \mathbf{s} ; \phi)} \right] \\ &= \mathcal{L}(\mathbf{s}, \phi, \theta)\end{aligned}\tag{4.14}$$

in which \mathcal{L} stands for the evidence lower bound. From the above inequality it might not seem very clear when the equality in the upper bound for the elbo is achieved, the second way of the deriving the elbo will make this point more clear.

In order to proceed to the second derivation we first need to introduce to the reader one non-symmetric dissimilarity measure between two probability distributions which is well known in information theory. This dissimilarity measure is called the Kullback-Leibler divergence and is defined for continuous random variables as

$$D_{\text{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log \left(\frac{p(x)}{q(x)} \right) dx\tag{4.15}$$

this dissimilarity measure is going to be the optimization criterion we mentioned before to make $q(\mathbf{z} | \mathbf{s} ; \phi) \approx p(\mathbf{z} | \mathbf{s} ; \theta)$ as close as possible, therefore our objective becomes now to find ϕ^* such that

$$\phi^* = \arg \min_{\phi} D_{\text{KL}}(q(\mathbf{z} | \mathbf{s} ; \phi) \parallel p(\mathbf{z} | \mathbf{s} ; \theta)).\tag{4.16}$$

The divergence in 4.16 can also be expressed it the following way

$$D_{\text{KL}}(q(\mathbf{z} | \mathbf{s}; \phi) \| p(\mathbf{z} | \mathbf{s}; \theta)) = \quad (4.17)$$

$$= \int p(\mathbf{z} | \mathbf{s}; \phi) \log \left(\frac{q(\mathbf{z} | \mathbf{s}; \phi)}{p(\mathbf{z} | \mathbf{s}; \theta)} \right) d\mathbf{z} \quad (4.18)$$

$$= \int q(\mathbf{z} | \mathbf{s}; \phi) (\log q(\mathbf{z} | \mathbf{s}; \phi) - \log p(\mathbf{z} | \mathbf{s}; \theta)) d\mathbf{z} \quad (4.19)$$

$$= \int q(\mathbf{z} | \mathbf{s}; \phi) \log q(\mathbf{z} | \mathbf{s}; \phi) - \int q(\mathbf{z} | \mathbf{s}; \phi) \log p(\mathbf{z} | \mathbf{s}; \theta) d\mathbf{z} \quad (4.20)$$

$$= \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z} | \mathbf{s}; \phi)} [\log q(\mathbf{z} | \mathbf{s}; \phi)] - \int q(\mathbf{z} | \mathbf{s}; \phi) \log \left(\frac{p(\mathbf{z}, \mathbf{s}; \theta)}{p(\mathbf{s}; \theta)} \right) d\mathbf{z} \quad (4.21)$$

$$= \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z} | \mathbf{s}; \phi)} [\log q(\mathbf{z} | \mathbf{s}; \phi)] - \int q(\mathbf{z} | \mathbf{s}; \phi) \log p(\mathbf{s}, \mathbf{z}; \theta) d\mathbf{z} \\ + \log p(\mathbf{s}; \theta) \int q(\mathbf{z} | \mathbf{s}; \phi) d\mathbf{z} \quad (4.22)$$

$$= \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z} | \mathbf{s}; \phi)} [\log q(\mathbf{z} | \mathbf{s}; \phi)] - \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z} | \mathbf{s}; \phi)} [\log p(\mathbf{s}, \mathbf{z}; \theta)] + \log p(\mathbf{s}; \theta) \quad (4.23)$$

note the dependence on the intractable $p(\mathbf{s}; \theta)$ on 4.23 which implies that we cannot compute the $D_{\text{KL}}(q(\mathbf{z} | \mathbf{s}; \phi) \| p(\mathbf{z} | \mathbf{s}; \theta))$ explicitly. By rearranging terms in the equality we can observe that

$$\log p(\mathbf{s}; \theta) = D_{\text{KL}}(q(\mathbf{z} | \mathbf{s}; \phi) \| p(\mathbf{z} | \mathbf{s}; \theta)) + \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z} | \mathbf{s}; \phi)} [\log p(\mathbf{s}, \mathbf{z}; \theta)] \\ - \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z} | \mathbf{s}; \phi)} [\log q(\mathbf{z} | \mathbf{s}; \phi)] \quad (4.24)$$

and since it is well know that $D_{\text{KL}} \geq 0$ we arrive again, to the same evidence lower bound as we derived before in 4.14

$$\log p(\mathbf{s}; \theta) = D_{\text{KL}}(q(\mathbf{z} | \mathbf{s}; \phi) \| p(\mathbf{z} | \mathbf{s}; \theta)) + \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z} | \mathbf{s}; \phi)} [\log p(\mathbf{s}, \mathbf{z}; \theta)] - \\ \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z} | \mathbf{s}; \phi)} [\log q(\mathbf{z} | \mathbf{s}; \phi)] \\ \geq \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z} | \mathbf{s}; \phi)} [\log p(\mathbf{s}, \mathbf{z}; \theta)] - \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z} | \mathbf{s}; \phi)} [\log q(\mathbf{z} | \mathbf{s}; \phi)] \\ = \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z} | \mathbf{s}; \phi)} [\log p(\mathbf{z} | \mathbf{s}; \theta)] + \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z} | \mathbf{s}; \phi)} \left[\log \frac{p(\mathbf{z}; \theta)}{q(\mathbf{z} | \mathbf{s}; \phi)} \right] \\ = \mathcal{L}(\mathbf{s}, \phi, \theta) \quad (4.25)$$

however with this way of deriving the elbo we can see that equality in the upper bound is achieved when $D_{\text{KL}}(q(\mathbf{z} | \mathbf{s}; \phi) \| p(\mathbf{z} | \mathbf{s}; \theta)) = 0$ which happens when the variational posterior and the true posterior are equal. This fact give us the theoretical guarantee that by maximizing the elbo we are making progress on the attempt to make $q(\mathbf{z} | \mathbf{s}; \phi) \approx p(\mathbf{z} | \mathbf{s}; \theta)$.

The experimented statistician will note that the second term of equation 4.25 is in fact the negative Kullback Leibler divergence between the prior $p(\mathbf{z}; \theta)$

and the variational posterior $q(\mathbf{z} | \mathbf{s}; \phi)$, therefore from now on we will define the elbo as

$$\mathcal{L}(\mathbf{s}, \phi, \theta) = \mathbb{E}_{z \sim q(\mathbf{z} | \mathbf{s}; \phi)} [\log p(\mathbf{z} | \mathbf{s}; \theta)] - D_{\text{KL}}(q(\mathbf{z} | \mathbf{s}; \phi) \| p(\mathbf{z}; \theta)) \quad (4.26)$$

4.1.3 Autoencoding Variational Bayes

Once we have derived the evidence lower bound we can return to our initial objective and try to find θ^* by lower bounding equation 4.2 as follows

$$\begin{aligned} \arg \max_{\theta} \mathcal{L}(\theta) &= \arg \max_{\theta} \log \mathcal{L}(\theta) = \arg \max_{\theta} \sum_{i \leq N} \log p_{\theta}(\mathbf{s}^{(i)}) \\ &\geq \arg \max_{\phi, \theta} \sum \mathcal{L}(\mathbf{s}^{(i)}, \phi, \theta) \end{aligned} \quad (4.27)$$

ideally we would like to maximize the sum of individual elbos by gradient ascent, for that purpose we need to be able to compute the gradients of the above equation with respect to ϕ as well as for θ

$$\nabla_{\phi, \theta} \sum \mathcal{L}(\mathbf{s}^{(i)}, \phi, \theta) = \sum \nabla_{\phi, \theta} \mathcal{L}(\mathbf{s}^{(i)}, \phi, \theta) \quad (4.28)$$

to declutter notation we will avoid the index i of each respective stimulus $\mathbf{s}^{(i)}$. If there is no closed form solution to the expectations one can resort to Monte Carlo approximation to compute an approximation, since by the law of large numbers we have that

$$\int f(x)p(x) dx = \mathbb{E}[f(x)] \approx \frac{1}{N} \sum_{i \leq N} f(x_i) \quad (4.29)$$

where $x_i \sim p(x)$, the requirement to use this method is that the terms inside the integral must factorize as product of a function and a probability distribution, with this in mind we can easily approximate the gradient with respect to θ as we can see below

$$\begin{aligned} \nabla_{\theta} \mathcal{L}(\mathbf{s}, \phi, \theta) &= \nabla_{\theta} (\mathbb{E}_{z \sim q(\mathbf{z} | \mathbf{s}; \phi)} [\log p(\mathbf{s}, \mathbf{z}; \theta)] - \mathbb{E}_{z \sim q(\mathbf{z} | \mathbf{s}; \phi)} [\log q(\mathbf{z} | \mathbf{s}; \phi)]) \\ &= \mathbb{E}_{z \sim q(\mathbf{z} | \mathbf{s}; \phi)} [\nabla_{\theta} \log p(\mathbf{s}, \mathbf{z}; \theta)] \approx \sum_{i \leq N} \nabla_{\theta} \log p(\mathbf{s}^{(i)}, \mathbf{z}^{(i)}; \theta) \end{aligned} \quad (4.30)$$

the problem comes with gradient with respect to the variational parameters ϕ since

$$\begin{aligned} \nabla_{\phi} \mathcal{L}(\mathbf{s}, \phi, \theta) &= \nabla_{\phi} \mathbb{E}_{z \sim q_{\phi}(\mathbf{z} | \mathbf{s})} [\log p(\mathbf{s}, \mathbf{z}; \theta)] - \nabla_{\phi} \mathbb{E}_{z \sim q_{\phi}(\mathbf{z} | \mathbf{s})} [\log q_{\phi}(\mathbf{z} | \mathbf{s})] \\ &= \int \log p(\mathbf{s}, \mathbf{z}; \theta) \nabla_{\phi} q(\mathbf{z} | \mathbf{s}; \phi) d\mathbf{z} \\ &\quad - \int \nabla_{\phi} (\log(q(\mathbf{z} | \mathbf{s}; \phi)q(\mathbf{z} | \mathbf{s}; \phi))) d\mathbf{z} \end{aligned} \quad (4.31)$$

and therefore we cannot estimate the gradient with respect to ϕ using a Monte Carlo approach. In order to solve this issue two families of estimators have been proposed in the literature [15]. One family uses the fact that

$$\nabla f(x) = f(x) \nabla \log f(x) \quad (4.32)$$

which allows us to obtain the following estimator

$$\nabla_{\theta} \mathbb{E}_{z \sim p_{\theta}(z)} [f(z)] = \mathbb{E}_{z \sim p_{\theta}(z)} [f(z) \nabla_{\theta} \log p_{\theta}(z)] \approx \sum_{i \leq N} f(z_i) \nabla_{\theta} \log p_{\theta}(z_i) \quad (4.33)$$

the other family is the one introduced by Kingma and Welling in their seminal paper [7], they propose a trick known as the reparametrization trick, which allows us to easily compute an estimator of the gradient of the expectation. The reparametrization trick comes from the fact that given a random variable \mathbf{z} following a conditional distribution $q(\mathbf{z} | \mathbf{x})$, a random variable ϵ following $p(\epsilon)$, and a deterministic mapping $z = g_{\phi}(\epsilon, \mathbf{x})$, in the limit we have

$$q(\mathbf{z} | \mathbf{x}) d\mathbf{z} = p(\epsilon) d\epsilon \quad (4.34)$$

which allow us to state that [7]

$$\begin{aligned} \int f(\mathbf{z}) q(\mathbf{z} | \mathbf{x}) d\mathbf{z} &= \int f(\mathbf{z}) p(\epsilon) d\epsilon \\ &= \int f(g_{\phi}(\epsilon, \mathbf{x})) p(\epsilon) d\epsilon \end{aligned} \quad (4.35)$$

and therefore the gradient of 4.31 can now be computed as

$$\begin{aligned} \nabla_{\phi} \mathcal{L}(\mathbf{s}, \phi, \theta) &= \nabla_{\phi} \mathbb{E}_{z \sim q(\mathbf{z} | \mathbf{s}; \phi)} [\log p(\mathbf{s}, \mathbf{z}; \theta)] - \nabla_{\phi} \mathbb{E}_{z \sim q(\mathbf{z} | \mathbf{s}; \phi)} [\log q(\mathbf{z} | \mathbf{s}; \phi)] \\ &= \nabla_{\phi} \int \log p_{\theta}(\mathbf{s}, \mathbf{z}) q_{\phi}(\mathbf{z} | \mathbf{s}) d\mathbf{z} - \nabla_{\phi} \int \log q_{\phi}(\mathbf{z} | \mathbf{s}) q_{\phi}(\mathbf{z} | \mathbf{s}) d\mathbf{z} \\ &= \nabla_{\phi} \int \log p_{\theta}(\mathbf{s}, \mathbf{z}) p(\epsilon) d\epsilon - \nabla_{\phi} \int \log q_{\phi}(\mathbf{z} | \mathbf{s}) p(\epsilon) d\epsilon \\ &= \int \nabla_{\phi} \log p_{\theta}(\mathbf{s}, \mathbf{z}) p(\epsilon) d\epsilon - \int \nabla_{\phi} \log q_{\phi}(\mathbf{z} | \mathbf{s}) p(\epsilon) d\epsilon \\ &= \mathbb{E}_{\epsilon} [\nabla_{\phi} \log p_{\theta}(\mathbf{s}, \mathbf{z})] - \mathbb{E}_{\epsilon} [\nabla_{\phi} \log q_{\phi}(\mathbf{z} | \mathbf{s})] \\ &\approx \sum_{i \leq N} \nabla_{\phi} \log p_{\theta}(\mathbf{s}, g_{\phi}(\epsilon_i, \mathbf{s})) - \nabla_{\phi} \log q(g_{\phi}(\epsilon_i, \mathbf{s}) | \mathbf{s}) \end{aligned} \quad (4.36)$$

this remarkable fact will be our workhorse for the rest of the thesis. Note that in case there is an analytical solution for the Kullback-Leibler divergence in the elbo the gradient can be estimated as

$$\nabla_{\phi} \mathcal{L}(\mathbf{s}, \phi, \theta) \approx \sum_{i \leq N} \nabla_{\phi} \log p_{\theta}(\mathbf{s} | g_{\phi}(\epsilon_i, \mathbf{s})) - \nabla_{\phi} D_{\text{KL}}(q(\mathbf{z} | \mathbf{s}; \phi) \| p(\mathbf{z}; \theta)) \quad (4.37)$$

which shows less variance than the previous estimator [7]. Now that we can compute the gradients of the elbo we can maximize equation 4.27 by gradient ascent and at the end of optimization process $q(\mathbf{z} | \mathbf{s}; \phi)$ will serve as a proxy for the true posterior and therefore we can estimate the latent variables \mathbf{z} for a given stimulus \mathbf{s} .

Nothing has been said about where feed-forward neural networks come into play in this process, this can be easily seen if we remember the example showed in section 3.2 in which a feed-forward neural network modeled the dependence on the parameters of a probability distribution, in this case

$$q(g_\phi(\mathbf{s}, \epsilon) | \mathbf{s}) = \mathcal{F}_z(\mathbf{s}, \epsilon; W) \quad (4.38)$$

in which \mathcal{F} represents a neural network, and the same for

$$p_\theta(\mathbf{s} | \mathbf{z}) = \mathcal{F}_s(\mathbf{z}; W) \quad (4.39)$$

this way of modeling allow us to capture complicated non-linear relationships between the latent factors and the stimulus and plays a major role in our work. The readers with some experience in autoencoders will recognize the autoencoding structure that the elbo reveals in which the latent factors can be considered as the lower dimensionality representation of a sample, this gives name to how this type of latent models are called in the deep learning literature in which are usually named as variational autoencoders.

As a last comment in this subsection we would like to emphasize that along all the derivations we did not make the assumption of working with a factorized variational posterior probability distribution, which greatly increases the modeling capacity.

4.1.4 β Variational Autoencoder

In the previous section we introduced the variational autoencoder which allows us to compute $q_\phi(\mathbf{z} | \mathbf{s})$ and it was shown that the function to optimize 4.26 contains the negative Kullback-Leibler divergence between the posterior and the prior

$$D_{\text{KL}}(q_\phi(\mathbf{z} | \mathbf{s}) \parallel p_\theta(\mathbf{z})) \quad (4.40)$$

this term forces the $q_\phi(\mathbf{z} | \mathbf{s})$ to be as close as possible to $p_\theta(\mathbf{z})$ and therefore the choice of the prior imposes some bias towards a family of probability distributions. It is common in the literature to see $p_\theta(\mathbf{z})$ as

$$p_\theta = \mathcal{N}(\mathbf{0}, I) \quad (4.41)$$

this choice of prior forces the posterior latent variables to be independent, however this does not happen in practice because when the unsupervised optimization process has finished the D_{KL} term is usually not zero, and if that is

the case we should be concerned since it implies that $q_\phi(\mathbf{z} | \mathbf{s})$ is not using any possible information contained in \mathbf{s} and it just outputs random latent factors.

In [5] the authors proposed a method to make the latent factors independent and still use the information contained in \mathbf{s} , they assume that the set of latent factors is divided into two disjoint groups, one group \mathbf{v} of conditional independent factors such that the true posterior of \mathbf{v} factorizes as

$$p(\mathbf{v} | \mathbf{s}) = \prod_{j \leq M} p(v_j | \mathbf{s}) \quad (4.42)$$

and other group \mathbf{m} of dependent factors, with this assumption the observations are generated according to $p(\mathbf{s} | \mathbf{v}, \mathbf{m})$. They argue that in order to encourage disentanglement in the variational posterior, the following function should be optimized

$$\mathcal{L}(\theta, \phi, \beta) = \mathbb{E}_{z \sim q(\mathbf{z} | \mathbf{s}; \phi)} [\log p(\mathbf{z} | \mathbf{s}; \theta)] - \beta D_{\text{KL}}(q(\mathbf{z} | \mathbf{s}; \phi) \| p(\mathbf{z}; \theta)) \quad (4.43)$$

in which $\beta \geq 0$. Depending on the value of β we vary the pressure applied to variational posterior to match the unit gaussian and therefore we are at the same time encouraging the disentanglement. This β variational formulation comes from a Lagrangian and the interested reader can see the derivation in [5].

In our work we make use of the presented formulation to obtain the independent latent factors of the stimulus, the reader should notice that the selection of the number of the latent factors as well as the selection of the β is not trivial and should be taken with care, different sizes and values of β can be tested and one can select proper values based on the tradeoff between the quality of the samples or the disentanglement and the size of the latent factors. In figure 4.2 and 4.3 we can see the results after the training has finished, the architecture of the neural network is summarized in.

In future chapters we will make use of the obtained latent factors of the stimulus to control the generation of neural activity.

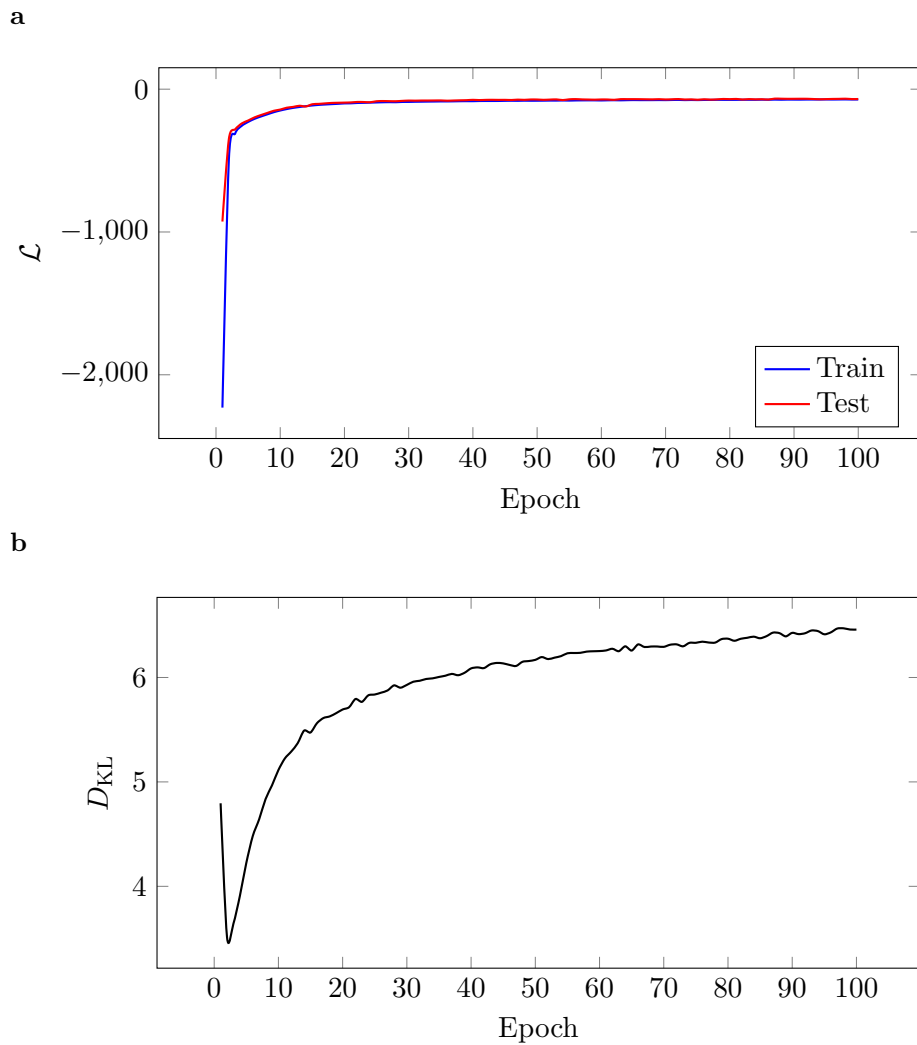


Figure 4.2: (a) The training and test values of the elbo per epoch obtained during the optimization process. (b) Kullback-Leibler term with a coefficient $\beta = 5$, the start of the training clearly reflects how the algorithm balances the tradeoff between a higher value of D_{KL} and a better quality of the reconstructions.

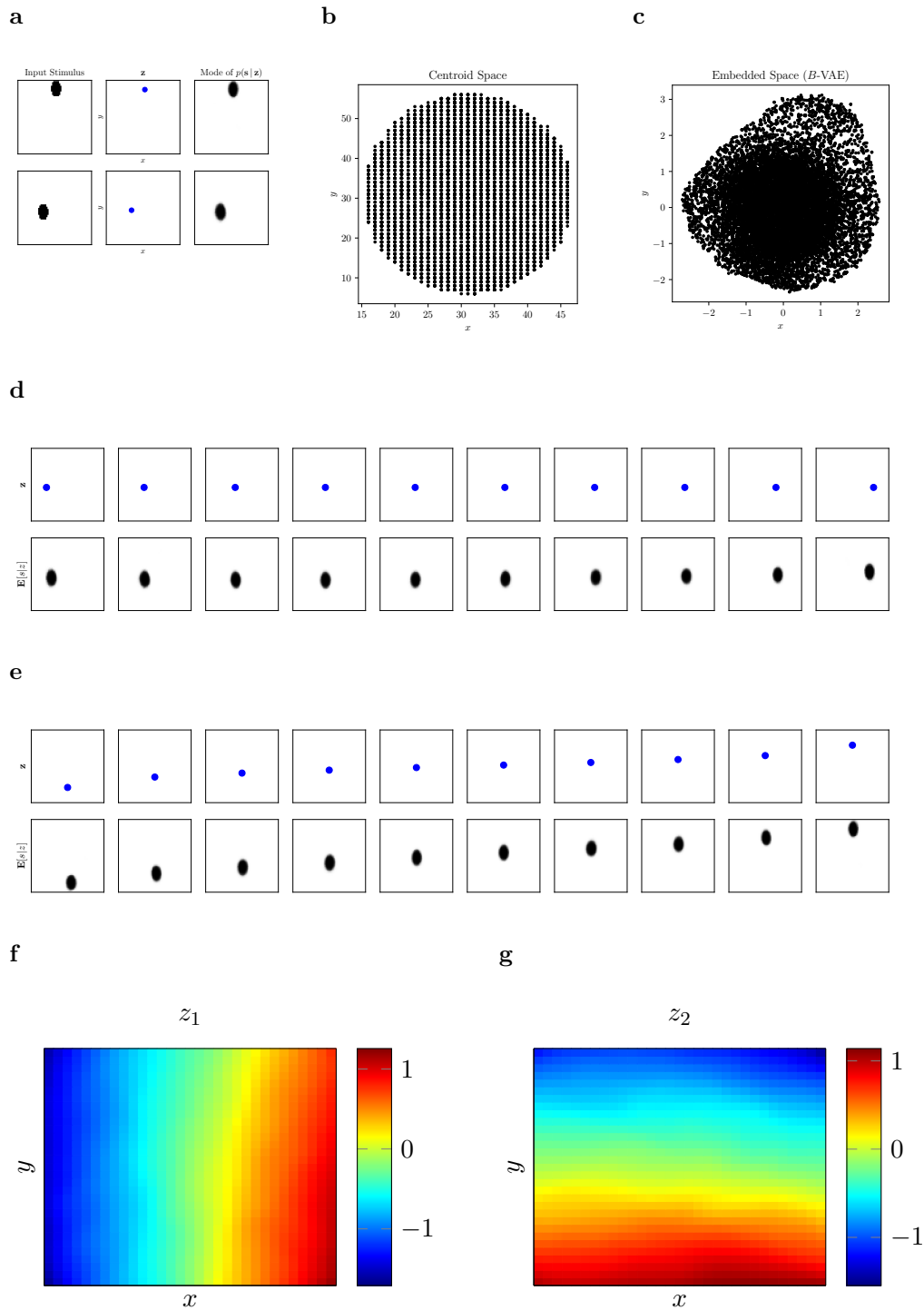


Figure 4.3: (a) Latent factors \mathbf{z} and the reconstructions obtained by plotting the mode of $p_{\theta^*}(\mathbf{s}|\mathbf{z})$ for two random input stimulus. (b)(c) Comparison of the centroid space obtained by the β -vae and the hand computed centroid space. (d)(e) Samples from $p_{\theta^*}(\mathbf{s}|\mathbf{z} = [x, y])$ keeping x fixed and varying y according to the inverse cdf of $p(\mathbf{z})$ and viceversa. (f)(g) Value of the respective variable z_i for a grid of blob positions uniformly sampled from the stimulus space.

Probabilistic Neural Encoding

In this chapter we make use of the theory presented in the previous chapters, to propose a graphical model that attempts to capture the relationship between the stimulus presented to mice and the corresponding generated neural activity.

5.1 Proposed Graphical Model

We recall to the reader our initial objective which was to obtain $p_\theta(\mathbf{x} | \mathbf{s})$ in which \mathbf{x} represents the neural activity and \mathbf{s} represents the stimulus, so that given a stimulus we can sample neural activity. To make the inference more tractable we will assume that there is a set of latent variables \mathbf{z} that depends on the input stimulus \mathbf{s} , the neural activity is then generated according to $p_\theta(\mathbf{x} | \mathbf{z})$, this process can be summarized in the following generative model

$$\mathbf{s} \sim \prod_{i=1}^N \text{Bern}(\lambda_i) \quad (5.1)$$

$$\mathbf{z} | \mathbf{s} \sim \mathcal{N}(\mu_z, \Sigma_z) \quad (5.2)$$

$$\mathbf{x} | \mathbf{z} \sim \mathcal{N}(\mu_x, \Sigma_x) \quad (5.3)$$

The joint probability distribution of the generative model can be factorized as follows

$$p_\theta(\mathbf{x}, \mathbf{z}, \mathbf{s}) = p_\theta(\mathbf{x} | \mathbf{z}) p_\theta(\mathbf{z} | \mathbf{s}) p_\theta(\mathbf{s}) \quad (5.4)$$

under this factorization we are going to make an important assumption about the dependence of the latent factors \mathbf{z} in \mathbf{s} , we will assume that

$$p_\theta(\mathbf{x}, \mathbf{z}, \mathbf{s}) = p_\theta(\mathbf{x} | \mathbf{z}) p_\theta(\mathbf{z} | \mathbf{s}) p_\theta(\mathbf{s}) \quad (5.5)$$

$$= p_\theta(\mathbf{x} | \mathbf{z}) p_\theta(\mathbf{z} | \mathbf{z}_s) q_\phi(\mathbf{z}_s | \mathbf{s}) \quad (5.6)$$

in which $q_\phi(\mathbf{z}_s | \mathbf{s})$ is the variational posterior obtained in the previous chapter with the β -variational autoencoder, this allows us to generate neural activity

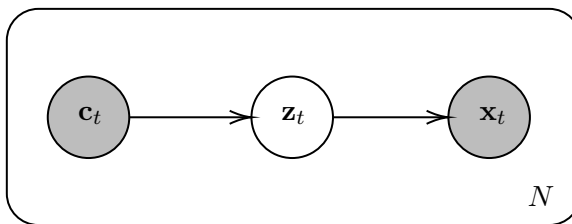


Figure 5.1: Proposed graphical model of the relationship between the neural activity and the centroid of the stimulus in which \mathbf{x}_t represents the neural activity, \mathbf{z} represents the latent factors of the neural activity and \mathbf{c}_t represents the centroid coordinates.

with just a set of latent factors of the stimulus and if those latent factors are disentangled we have even more control on the generative process, as an example we can take one latent variable and vary it while keeping the others fixed to see how the generated neural activity responds and therefore study the influence of the varied latent factor, this would not be possible if the latent factors are entangled because the change of one factor affects the others.

In the following and to make the notation more easy we will refer to $q_\phi(\mathbf{z}_s | \mathbf{s})$ as $q_\phi(\mathbf{c} | \mathbf{s})$ in which with \mathbf{c} we refer to the centroid of the stimulus \mathbf{s} i.e $\mathbf{c} := \mathbf{v}_s$. In figure 3.1 we can see a graphical model of the observed variables $\{\mathbf{x}, \mathbf{c}\}$ and the hidden latent factors \mathbf{z} of the neural activity.

In order to make inference on the hidden latent factors \mathbf{z} of the neural activity we can make use of the observed variables $\{\mathbf{x}, \mathbf{c}\}$ and therefore compute the following posterior

$$p_\theta(\mathbf{z} | \mathbf{x}, \mathbf{c}) = \frac{p_\theta(\mathbf{z}, \mathbf{x}, \mathbf{c})}{p_\theta(\mathbf{x}, \mathbf{c})} \quad (5.7)$$

$$= \frac{p_\theta(\mathbf{x} | \mathbf{z}) p_\theta(\mathbf{z} | \mathbf{c}) q_\phi(\mathbf{c} | \mathbf{s})}{\int p_\theta(\mathbf{x}, \mathbf{z}, \mathbf{c}) d\mathbf{z}} \quad (5.8)$$

$$= \frac{p_\theta(\mathbf{x} | \mathbf{z}) p_\theta(\mathbf{z} | \mathbf{c})}{\int p_\theta(\mathbf{x} | \mathbf{z}) p_\theta(\mathbf{z} | \mathbf{c}) d\mathbf{z}} \quad (5.9)$$

to further simplify the posterior distribution we will make use of the following simple observation

$$p(\mathbf{x} | \mathbf{z}, \mathbf{c}) = \frac{p_\theta(\mathbf{x} | \mathbf{z}) p_\theta(\mathbf{z} | \mathbf{c}) q_\phi(\mathbf{c} | \mathbf{s})}{p_\theta(\mathbf{z} | \mathbf{c}) q_\phi(\mathbf{c} | \mathbf{s}) \int p_\theta(\mathbf{x} | \mathbf{z}) d\mathbf{x}} \quad (5.10)$$

$$= p_\theta(\mathbf{x} | \mathbf{z}) \quad (5.11)$$

which allows us to simplify the denominator in 5.9 by applying bayes rule as follows

$$\int p_\theta(\mathbf{x} | \mathbf{z}) p_\theta(\mathbf{z} | \mathbf{c}) d\mathbf{z} = \int p_\theta(\mathbf{x} | \mathbf{z}, \mathbf{c}) p_\theta(\mathbf{z} | \mathbf{c}) d\mathbf{z} \quad (5.12)$$

$$= \int \frac{p_\theta(\mathbf{x}, \mathbf{z}, \mathbf{c})}{p_\theta(\mathbf{z}, \mathbf{c})} \frac{p_\theta(\mathbf{z}, \mathbf{c})}{p_\theta(\mathbf{c})} d\mathbf{z} \quad (5.13)$$

$$= p_\theta(\mathbf{c})^{-1} \int p_\theta(\mathbf{x}, \mathbf{z}, \mathbf{c}) d\mathbf{z} \quad (5.14)$$

$$= \frac{p_\theta(\mathbf{x}, \mathbf{c})}{p_\theta(\mathbf{c})} \quad (5.15)$$

$$= p_\theta(\mathbf{x} | \mathbf{c}) \quad (5.16)$$

and therefore by plugging 5.16 in 5.7 we see that the posterior distribution can be written as

$$p_\theta(\mathbf{z} | \mathbf{x}, \mathbf{c}) = \frac{p_\theta(\mathbf{x} | \mathbf{z}) p_\theta(\mathbf{z} | \mathbf{c})}{p_\theta(\mathbf{x} | \mathbf{c})} \quad (5.17)$$

We use of the AEVB algorithm to approximate the intractable posterior $p_\theta(\mathbf{z} | \mathbf{x}, \mathbf{c})$ as well as to find the parameters θ of the proposed generative model. For that purpose we need to derive the evidence lower bound between a variational posterior and the true posterior. As a variational posterior we propose

$$q_\phi(\mathbf{z} | \mathbf{x}, \mathbf{c}) = \mathcal{N}(\mu(\mathbf{x}, \mathbf{c}), \sigma^2(\mathbf{x}, \mathbf{c})I) \quad (5.18)$$

since under the principle of maximum entropy the normal distribution is the one that maximizes the entropy for a finite mean and variance. We proceed now to compute the evidence lower bound by computing the D_{KL} divergence between the both posteriors

$$\begin{aligned} D_{\text{KL}}(q_\phi(\mathbf{z} | \mathbf{x}, \mathbf{c}) || p_\theta(\mathbf{z} | \mathbf{x}, \mathbf{c})) &= \int q_\phi(\mathbf{z} | \mathbf{x}, \mathbf{c}) \log \frac{q_\phi(\mathbf{z} | \mathbf{x}, \mathbf{c})}{p_\theta(\mathbf{z} | \mathbf{x}, \mathbf{c})} d\mathbf{z} \\ &= \int q_\phi(\mathbf{z} | \mathbf{x}, \mathbf{c}) \log q_\phi(\mathbf{z} | \mathbf{x}, \mathbf{c}) d\mathbf{z} - \int q_\phi(\mathbf{z} | \mathbf{x}, \mathbf{c}) \log p_\theta(\mathbf{z} | \mathbf{x}, \mathbf{c}) d\mathbf{z} \\ &= \int q_\phi(\mathbf{z} | \mathbf{x}, \mathbf{c}) \log q_\phi(\mathbf{z} | \mathbf{x}, \mathbf{c}) d\mathbf{z} - \int q_\phi(\mathbf{z} | \mathbf{x}, \mathbf{c}) \log p_\theta(\mathbf{x} | \mathbf{z}) p_\theta(\mathbf{z} | \mathbf{c}) d\mathbf{z} \\ &\quad + \int q_\phi(\mathbf{z} | \mathbf{x}, \mathbf{c}) \log p_\theta(\mathbf{x} | \mathbf{c}) d\mathbf{z} \\ &= \mathbb{E}[\log q_\phi(\mathbf{z} | \mathbf{x}, \mathbf{c}) - \log p_\theta(\mathbf{x} | \mathbf{z}) p_\theta(\mathbf{z} | \mathbf{c})] + \log p_\theta(\mathbf{x} | \mathbf{c}) \end{aligned} \quad (5.19)$$

in which the expectation is taken over $q_\phi(\mathbf{z} | \mathbf{x}, \mathbf{c})$, the evidence lower bound for the proposed generative model is therefore

$$\begin{aligned} \log p_\theta(\mathbf{x} | \mathbf{c}) &\geq \mathcal{L}(\phi, \theta, \mathbf{x}, \mathbf{c}) \\ &= \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z} | \mathbf{x}, \mathbf{c})} [\log p_\theta(\mathbf{x} | \mathbf{z}) p_\theta(\mathbf{z} | \mathbf{c}) - \log q_\phi(\mathbf{z} | \mathbf{x}, \mathbf{c})] \\ &= \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z} | \mathbf{x}, \mathbf{c})} [\log p_\theta(\mathbf{x} | \mathbf{z})] - D_{\text{KL}}(q_\phi(\mathbf{z} | \mathbf{x}, \mathbf{c}) || p_\theta(\mathbf{z} | \mathbf{c})) \end{aligned} \quad (5.20)$$

the Kullback-Leibler divergence term in the above equation has an analytical solution which allows us to write equation 5.20 as

$$\begin{aligned} \mathcal{L}(\phi, \theta, \mathbf{x}, \mathbf{c}) &\approx \log p_\theta(\mathbf{x} | g_\phi(\boldsymbol{\epsilon})) \\ &- \sum_{j \leq |\mathbf{z}|} (\log \sigma_{p_\phi, j} - \log \sigma_{q_\phi, j}) + \frac{\sigma_{q_\phi, j}^2 + (\mu_{q_\phi, j} - \mu_{p_\theta, j})^2}{2\sigma_{p_\theta, j}^2} - \frac{1}{2} \end{aligned} \quad (5.21)$$

in which we used a monte carlo estimate of the expectation by sampling one $\boldsymbol{\epsilon}$ from $p(\boldsymbol{\epsilon}) = \mathcal{N}(\mathbf{0}, I)$ and applying the reparametrization trick.

5.2 Results

As routine feed-forward neural networks were used to parametrize the different probability distributions present in the generative model, however instead of using conventional layers that attempt to approximate the assumed unknown hierarchy of transformations

$$Y_l = \mathcal{F}(X_{l-1}) \quad (5.22)$$

we used a hierarchy of residual layers first introduced in [4], the authors of this paper assume that there exists a function \mathcal{H} that computes the residual difference between Y_l and X_{l-1} such that

$$Y_l - X_{l-1} = \mathcal{H}(X_{l-1}) \implies \mathcal{F}(X_{l-1}) = \mathcal{H}(X_{l-1}) + X_{l-1} \quad (5.23)$$

and is this function \mathcal{H} that during the training will be approximated at each layer of the network

$$\mathcal{F}(X_{l-1}) \approx \mathcal{H}(X_{l-1}; W) + X_{l-1} \quad (5.24)$$

empirical advantages of using residual layers for very deep learning architectures have been shown over the last years in the literature [10]

Figure 5.2 shows a table with the details of the designed architecture as well as the evolution of the elbo during the optimization process. In figure 5.3 we can see the results after the training has finished.

a

Neural Network Architecture		
$q_\phi(z x, c)$		$p_\theta(x z)$
x	c	z
Conv3-20	·	FC-200
Residual block 20	·	FC-500
Residual projection 30	·	FC-5000
Residual block 30	·	Transpose residual projection 50
Residual projection 40	·	Transpose residual projection 40
Residual block 40	·	Transpose residual block 40
Residual projection 50	·	Transpose residual projection 30
Residual projection 50	·	Transpose residual block 30
FC-500	·	Transpose residual projection 20
FC-200	c	Transpose residual block 20
FC-202		ConvTranspose-1
FC-60		·
FC-60		·
$\hat{\mu}, \hat{\sigma}$		\hat{x}

b

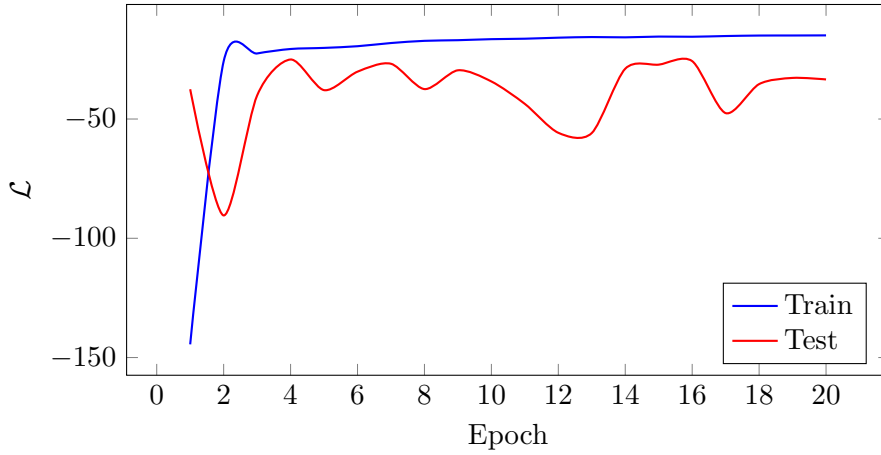


Figure 5.2: (a) Details of the architecture used to perform variational inference. (b) Sum of individual elbos per epoch during the optimization process of the proposed generative model.

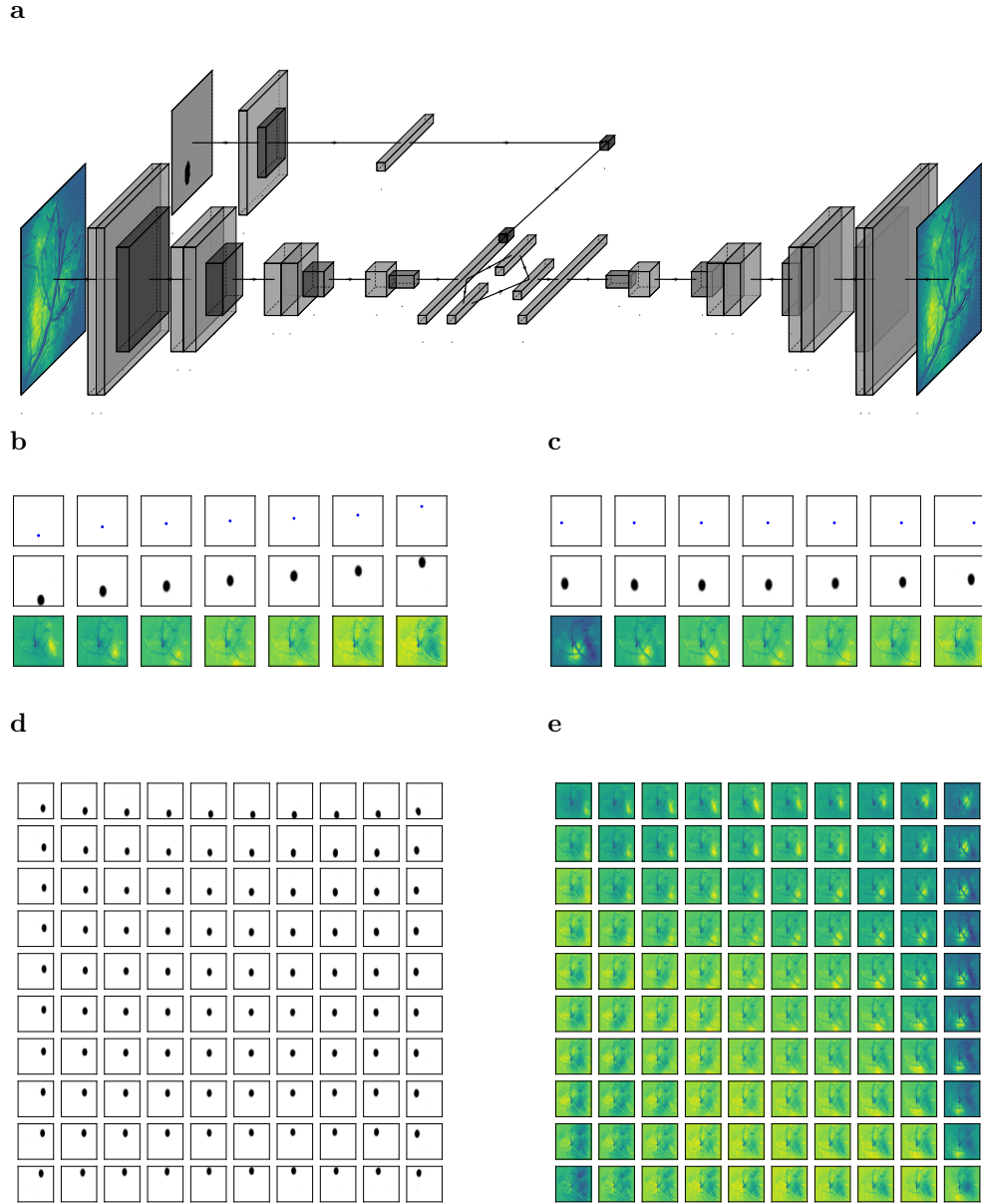


Figure 5.3: Samples of conditional generated neural activity: (a) Diagram of the architecture used to perform the variational inference. (b)(c) Samples generated by computing $\mathbb{E}_{\mathbf{x} \sim p_{\theta^*}(\cdot)}[\mathbf{x} | \mathbf{z}]$ in which \mathbf{z} is produced by keeping one centroid coordinate fixed in \mathbf{c} and varying the other independently and then computing the mean of $p_{\theta^*}(\mathbf{z} | \mathbf{c})$. (d)(e) Stimulus samples and their corresponding neural activity produced in the same way as explained previously but for linearly spaced values of the CDF of $q_{\phi}(\mathbf{c} | \mathbf{s})$.

Now that we have seen how we can generate neural activity given an input stimulus, in the next chapter we will address the problem of validating the quality and the feasibility of the generated samples.

5.3 Validation

In the previous section we showed how we can obtain a conditional probability distribution of neural activity given a stimulus. In this section we will address the problem of validating our proposed generative model.

5.3.1 Decoding

From now on we will refer to a conditional generated sample as \mathbf{x}_c in which \mathbf{c} represents the centroid that was used to compute the latent factors $\mathbf{z} \sim p_\theta(\mathbf{z} | \mathbf{c})$ used to generate \mathbf{x}_c through $p_\theta(\mathbf{x} | \mathbf{z})$.

To address the problem of validating the generated samples we propose a method that depends on the following assumption. There exists a function \mathcal{F} that captures the relationship

$$\mathbf{c}_t = \mathcal{F}(\mathbf{x}_t) \quad (5.25)$$

in which as in previous sections \mathbf{x}_t represents the neural activity recorded at time step t and \mathbf{c}_t represents the centroid of the stimulus that we assume has elicited the neural activity. Under this assumption we would expect that if we provide a conditional generated sample \mathbf{x}_c to \mathcal{F} , it should return us \mathbf{c} up to a small deviation, however \mathcal{F} is unknown to us, and we will try to approximate it by means of feed-forward neural networks, for that purpose we have created a dataset \mathcal{D} with one to one time correspondence such that for each time step t of the recorded neural activity we have the centroid of the stimulus that was projected to the mice, \mathcal{D} therefore contains T tuples $(\mathbf{x}_t, \mathbf{c}_t)$ i.e

$$\mathcal{D} = \{(\mathbf{x}_1, \mathbf{c}_1), (\mathbf{x}_2, \mathbf{c}_2), \dots, (\mathbf{x}_T, \mathbf{c}_T)\} \quad (5.26)$$

5.3.2 Method

Our proposed method to validate the generated samples can be summarized in four steps

1. We create a new dataset $\mathcal{D}' \subset \mathcal{D}$ and we define

$$\mathcal{D}_{\text{test}} = \mathcal{D} \setminus \mathcal{D}' \quad (5.27)$$

as the test dataset.

2. A regression model $\hat{\mathcal{F}}$ is then trained with \mathcal{D} to estimate the centroid \mathbf{c} given a true sample of the recorded neural activity \mathbf{x} . The objective in this step is to make

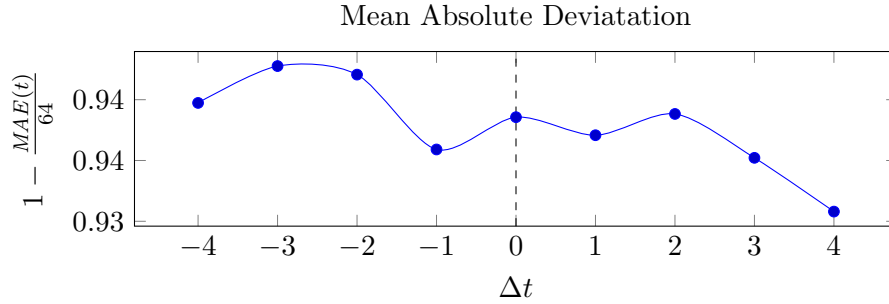
$$\mathcal{F}(\mathbf{x}) \approx \mathcal{F}(\mathbf{x}; W) \tag{5.28}$$

as close as possible.

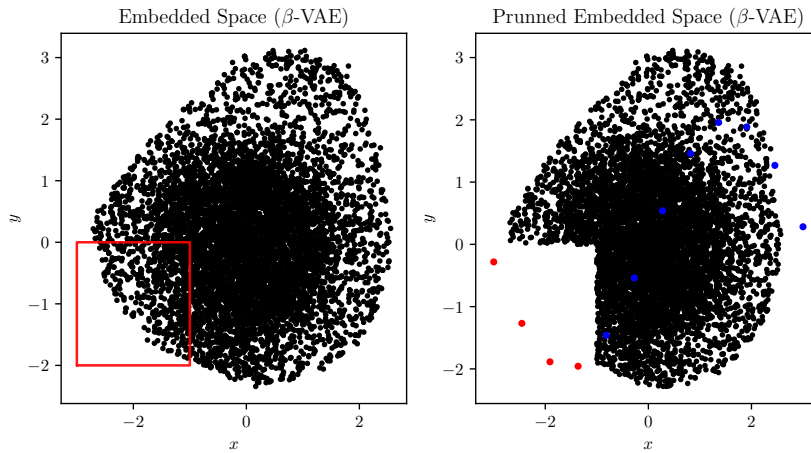
3. We train the generative model with \mathcal{D}' until convergence. Once the training has finished we will have estimated $p_\theta(\mathbf{x} | \mathbf{c})$ as well as $p_\theta(\mathbf{z} | \mathbf{c})$. Note that in this step the generative model has never seen the tuples in D_{test} , nevertheless since the estimated probability distributions are continuous distributions we are still able to generate samples conditioned on any coordinate \mathbf{c} within the centroid space and within $\mathcal{D}_{\text{test}}$.
4. As a last step we use the centroids in D_{test} to generate samples of neural activity $\mathbf{x}_{\mathbf{c}}$ and we feed those samples to the already trained regression model, we then compare the regressed values with the ones used to generate samples.

In figure 5.4 we can see the results of the proposed validation method.

a



b



c

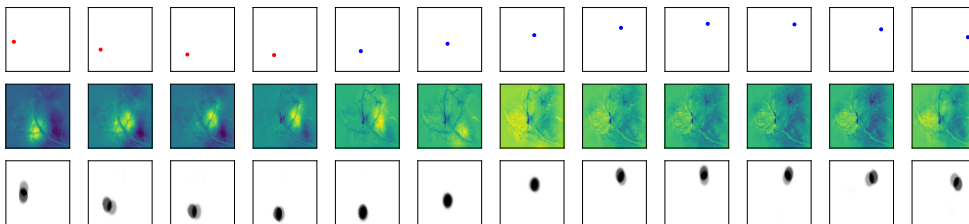


Figure 5.4: (a) Mean absolute deviation of trained the regression model per time deviation such that the regressed values are computed according to $c_{t+\Delta t} = \hat{\mathcal{F}}(x_t; W)$. (b) Left: The original embedded space used to train $\hat{\mathcal{F}}$, the points within the red box correspond to $\mathcal{D}_{\text{test}}$. Right: The pruned dataset \mathcal{D}' used to train the generative model, in red the points that fall within the red box these points will be used to test the quality of the generated samples.. (c) Top row: In red the points that belong to $\mathcal{D}_{\text{test}}$. Middle row: The conditional generated samples $\mathbf{x}_{\mathbf{c}}$, the centroids \mathbf{c} in which they are conditioned are the ones in the top row. Bottom row: In black the expected regressed stimulus, and in gray the actual regressed stimulus produced by $\hat{\mathcal{F}}(\mathbf{x}_{\mathbf{c}}; W)$.

Chapter 6

Conclusion

As criticism, the author of this thesis is concerned about the regression model, and strongly believes that even that model it's able to generalize up to a small expected error we need a method to estimate how uncertain is the model about one regressed value. Gaussian processes were tried as a proof of concept assuming that

$$p(\mathbf{c} | \mathbf{x}) = p(c_1 | \mathbf{x})p(c_2 | \mathbf{x}) \quad (6.1)$$

but this assumption it's not biologically plausible. For further work in the regression model a suitable multivariate probability distribution with a non diagonal covariance matrix can be hypothesized and estimated, but this presents the challenge of how to estimate the covariance matrix while fulfilling the constraints of symmetry and positive semidefiniteness. Differentiable factorization methods such as Cholesky decomposition can be used to parametrize the covariance matrix and make the estimation easier.

For future work in the direction of the generative model, more complex graphical models that take into account time dependence can be designed and tested.

Bibliography

- [1] S. A. Cadena, G. H. Denfield, E. Y. Walker, L. A. Gatys, A. S. Tolias, M. Bethge, and A. S. Ecker. Deep convolutional models improve predictions of macaque v1 responses to natural images. *PLOS Computational Biology*, 15(4):1–27, 04 2019.
- [2] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2, 1989.
- [3] A. S. Ecker, F. H. Sinz, E. Froudarakis, P. G. Fahey, S. A. Cadena, E. Y. Walker, E. Cobos, J. Reimer, A. S. Tolias, and M. Bethge. A rotation-equivariant convolutional neural network model of primary visual cortex. *arXiv preprint arXiv:1809.10504*, 2018.
- [4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.
- [5] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.
- [6] D. H. Hubel and T. N. Wiesel. Receptive fields of single neurons in the cat’s striate cortex. *Journal of Physiology*, 148:574–591, 1959.
- [7] D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2013.
- [8] D. Klindt, A. S. Ecker, T. Euler, and M. Bethge. Neural system identification for large populations separating “what” and “where”. In *Advances in Neural Information Processing Systems*, pages 3506–3516, 2017.

- [9] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, pages 9–50, London, UK, UK, 1998. Springer-Verlag.
- [10] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. Visualizing the loss landscape of neural nets. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6389–6399. Curran Associates, Inc., 2018.
- [11] C. Pandarinath, D. J. O’Shea, J. Collins, R. Jozefowicz, S. D. Stavisky, J. C. Kao, E. M. Trautmann, M. T. Kaufman, S. I. Ryu, L. R. Hochberg, J. M. Henderson, K. V. Shenoy, L. F. Abbott, and D. Sussillo. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature Methods*, 15(10):805–815, 2018.
- [12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.
- [13] V. Vapnik. Principles of risk minimization for learning theory. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 831–838. Morgan-Kaufmann, 1992.
- [14] V. N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.
- [15] C. J. Walder, R. Nock, C. S. Ong, and M. Sugiyama. New tricks for estimating gradients of expectations. *CoRR*, abs/1901.11311, 2019.



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

PROBABILISTIC GENERATIVE MODEL
OF CALCIUM IMAGING

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):

KEVIN DANILLO

First name(s):

LOPEZ ANDRANGO

With my signature I confirm that

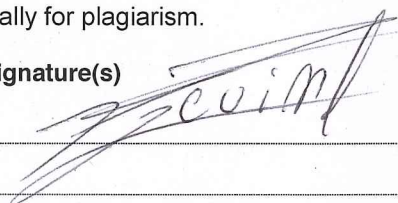
- I have committed none of the forms of plagiarism described in the ['Citation etiquette'](#) information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date

Zurich, 31 August 2019

Signature(s)



For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.