M.Sc. Thesis

Master of Science in Engineering

**Link Embeddings in Complex Networks**

Unai Villanueva Errazquin

S161072

Supervisor:

Sune Lehmann

Kongens Lygnby 2018

 Imagen que contiene imágenes prediseñadas, objeto

Descripción generada con confianza alta

DTU Compute

Department of Applied Mathematics and Computer Science

Technical University of Denmark

Matematiktorvet

Building 303B

2800 Kongens Lyngby, Denmark

Phone +45 4525 3031

compute@compute.dtu.dk

www.compute.dtu.dk

# Summary

Understanding the social ties between individuals has been a widely analysed topic over the past years. With the increase and spread of social interaction media as well as mobile communication systems, it has been possible to gather data from these interactions to better understand the social relationships between the individuals that form these social networks. In this thesis, the Sensible DTU data has been used to analyse the social relationship between individuals using the technology and ideas behind Autoencoders. The Sensible DTU dataset contains information regarding the interactions between students in DTU through smartphones whereas, autoencoders are a type of neural network capable of encoding data and reducing the dimensionality of long strings of data to further analyse it. It is shown that using autoencoder it is possible to encode and cluster certain types of data. However, it has not been possible to replicate this results on the SensibleDTU dataset.

# Preface

This thesis was prepared at the Department of Applied Mathematics and Computer Science at the Technical University of Denmark in fulfilment of the requirements for acquiring an M.Sc. degree in Digital Media Engineering. The project was carried out from January 2th to June 2th of 2018 under the academic supervision of associate professor Sune Lehmann from DTU compute. The workload of this thesis corresponds to 30 ECTS points.

Kongens Lyngby, June 2, 2018

Unai Villanueva Errazquin

s161072

# Acknowledgements

Firstly, I would like to thank Sune Lehmann for the opportunity of working in this project, providing the best advice with an always present positive attitude towards my work. He has been an excellent supervisor and his comments on my work have been extremely helpful. I’d like to also mention Ulf Aslak, who’s view on the technical aspects of my work have been crucial for me, especially during the first months of the project.

I’d like to also mention my friends and family who supported me during this project, but also during these last two years that I have been living away from them. This project is the culmination of a two-year adventure that wouldn’t have been possible without their help and support.

Finally, I’d like to thank all my friends that I’ve met during these past years, who have made the experience of living, studying and working in this country one of the best experiences of my life.

# Contents

[Summary i](#_Toc515454073)

[Preface iii](#_Toc515454074)

[Acknowledgements v](#_Toc515454075)

[Contents 1](#_Toc515454076)

1 [Introduction 1](#_Toc515454077)

[1.2 Structure of the thesis 2](#_Toc515454078)

2 [Human activity patterns 4](#_Toc515454079)

[2.1 Research from mobile phone data 6](#_Toc515454080)

[2.2 SensibleDTU project 9](#_Toc515454081)

[2.3 Missing answers 10](#_Toc515454082)

3 [The dataset 12](#_Toc515454083)

[3.1 Sensible DTU 12](#_Toc515454084)

[3.2 Call interactions 12](#_Toc515454085)

[3.3 SMS interactions 13](#_Toc515454086)

[3.4 Bluetooth 14](#_Toc515454087)

4 [Autoencoders 16](#_Toc515454088)

[4.1 Artificial Neural Networks 16](#_Toc515454089)

[4.2 Autoencoders 20](#_Toc515454090)

[4.3 Denoising Autoencoders 22](#_Toc515454091)

[4.4 Sparse Autoencoders 25](#_Toc515454092)

[4.5 Variational Autoencoders 26](#_Toc515454093)

5 [Preliminary work 30](#_Toc515454094)

[5.1 MNIST database 30](#_Toc515454095)

[5.2 Encoding 31](#_Toc515454096)

[5.2.1 Fully connected Autoencoder 32](#_Toc515454097)

[5.2.1 Variational Autoencoder 40](#_Toc515454098)

[5.2.3 Conclusions 45](#_Toc515454099)

6 [Data pre-processing 46](#_Toc515454100)

[6.1 Call data 46](#_Toc515454101)

[6.2 Bluetooth and call data 49](#_Toc515454102)

[6.2.1 1h bins 49](#_Toc515454103)

[6.2.2 8h bins 51](#_Toc515454104)

7 [Analysis and results 52](#_Toc515454105)

[7.1 Input recreation 52](#_Toc515454106)

[7.2 Data Analysis 55](#_Toc515454107)

8 [Conclusion 59](#_Toc515454108)

[8.1 Outline 59](#_Toc515454109)

[8.2 Discussion and future work 61](#_Toc515454110)

[References 63](#_Toc515454111)

[Appendix 69](#_Toc515454112)

[Appendix 1 – Code 69](#_Toc515454113)

CHAPTER 1

# Introduction

We humans are highly social beings. We like to be surrounded by people, friends, family and to share our experiences and feelings [53]. In a way, we have evolved to what we are now thanks to this ability to communicate with others, to create social bonds and to navigate in the social network that is modern society [59]. However, the way we interact can change dramatically from person to person. We might have best friends from childhood that we only get to see a few times a year, co-workers that we don’t appreciate that much but that we see every day or romances that only last for a few months. Understanding how these relationships work has been a highly discussed topic in recent years. One of the most modern approaches to obtaining this understanding have been Computational Social Sciences (CSS).

Computational Social Science is a wide field that aims to understand social phenomena with computational tools [2]. It is a relatively new field, since only with the technologies developed in recent years it has been possible to obtain and analyse social data in large scale. An early example of large-scale digital data being used on a social science issue was a study in 2002 by Kleinberf and David Liben-Nowell, in which they aimed to prove a concept that had had been suspected for a long time: people tend to become friends with friends of their friends [1]. However, in the early days of CSS researchers were limited by surveys, which were retrospective, and lab experiments, which were almost always done on small numbers of college sophomores. Today, the digital data-streams are able to give researchers a portrait of individual and group behaviour at unprecedented scales and levels of detail [2]. A dataset including this kind of information regarding users’ digital traces is the SensibleDTU dataset, that will be described and analysed in this thesis. It includes records of interactions and mobile communications between around 1000 students from the Technical University of Denmark over a period of 2.5 years (from late 2013 to early 2016). The data was gathered using smartphones that were given to the participants, that obtained information on their locations, telecommunications via SMS or phone calls, face-to-face interactions, social media and other personal information. A dataset with this kind of information and detail allows for a wide range of studies on social interaction, one of them being understanding personal relationships from daily interaction.

The research project documented here is an attempt to use the technology behind autoencoders to analyse the relationships between subjects of the experiment. As such, this thesis presents the trajectory from the understanding of the field of study and previous related works, the exploration of the theory behind the methodology that is intended to be used, the logical conclusion of its usability, to the analysis of the dataset. In brief, the reader will be presented with the following narrative: people with different types of relationships interact differently with each other (i.e. I might call a friend on a Saturday night, but not my professor from university), and therefore studying interaction records between the participants of the experiment should show different interaction patterns. On the other hand, autoencoders, usually used for image analysis, have shown potential on grasping the fundamental structures of images, being able to cluster groups of images by analysing the low dimensional representation that they create. The research idea is that autoencoders might be able to identify hidden structures in the long chains of interactions in a more detailed way than current studies in human activity patterns, that aggregate data resulting on a loss of information that can be crucial. With this, interactions that on the surface seem dissimilar, might have hidden patterns that indicate their similarity. As it will be presented on this document, the result if this analysis has not been satisfactory due to several reasons that will be explained, however, this doesn’t compromise the potential of this methodology for future studies.

## 1.2 Structure of the thesis

The thesis will first include the relevant theory in which the fundamentals for the work done will be explained. Chapter 2 will introduce the topic of human activity patterns, and the previous works done in the field. Chapter 3 will introduce the dataset as well as the subset that is used during the project. Chapter 4 will show the theory behind autoencoders, types and uses. Chapter 5 will show the preliminary work done on using autoencoders to compress and analyse the MNIST dataset. In chapter 6 the pre-processing of the data will be shown. Chapter 7 is the main section in which the work on the dataset will be shown and finally in Chapter 8 the final conclusions and future work will be presented.

CHAPTER 2

# Human activity patterns

Humans are animals. We might be very sophisticated animals, but in the end, we are very similar other creatures in this planet, except for our brains. Our brains are incredibly sophisticated machines that govern basic processes such as breathing or eating, but also let us appreciate artistic expressions from other humans. As animals, we have needs, and as intelligent and social creatures, we like to talk to other people, to interact with them [53]. We have conversations, we take turns to express our ideas, and at the same time we perform many other actions that in some ways are automated for us; if they were not it would be just too much for us to handle [54]. This is known as cognitive load, which refers to the actions that are taking an effort in our working memory. When we are doing something that is habitual for us, that we do every day, the action falls into our unconscious mind to ease the cognitive load on our brain. And so, we fall into a series of patterns that we repeat almost every day, from waking up, to going to sleep, our daily lives could almost be explained with a series of patterns that we repeat over time. The aim of this thesis is to observe if this also applies to our social interactions, the patterns that govern them, and if they hold any predictive power within them.

In sociology, the most basic social group is a dyad; a group of two people that interact with each other [55]. As the book “*Sociological Analysis of the Dyad*” states, dyads can be linked by a number of interests, such as romantic, work, family relation, friends or even partners in crime, with varying strength that is usually measured by the time the individuals spend together, as well as on the emotional intensity of their relationship. Understanding and detecting how these relationships among individuals in a society work is one of the objectives of Social Sciences [56].

Intuitively, one might say that part of the time the individuals spend together falls into a series of activity patterns that we humans usually fall into. Our daily lives are commonly a series of activities that we perform in a certain pattern, from waking up, to getting ready, going to work, after-work activities, and socializing, to having dinner with our family or friends. These patterns play a key role in understanding how our social relationships are formed.

Understanding human activity patterns plays a fundamental role in Computational Social Science [2]. However, previous studies that tried to understand human behaviour were developed without access to the terabytes of data describing minute-by-minute human actions that nowadays are available. As an example, sociological network theory worked with one-time “snapshot” data with only dozens of people. This is not something that can be applied to massive longitudinal data sets of millions of people. For the longest time, data acquisition for these studies relied on questionnaires which lack depth and cannot be done on a large scale. As Michael Macy, social scientist at Cornell University states in [1], surveys are retrospective, meaning that they lack reliability. It can be shown statistically that where a relation exists between two variables measured with no or minimum error, this relation vanishes if either or both variables have sizable error components. Therefore, such data has little utility in assessing simple relations, let alone in making causal inferences [57].

In recent years, many advances have been made in the data acquisition aspect, making it increasingly feasible to build models of human activity patterns, in both individual level or in large populations. Most of the studies in human mobility have focused on exploring individual mobility patterns, but with a wide range of purposes. Some studies have focused on predicting the spreading of diseases [7, 8, 9] while others have tried to understand human mobility to improve urban areas or for healthcare purposes [51,10,11,12].

On the other hand, the number of mobile phone users has been increasing every year, being 2013 the first year where there were almost as many users as people in the world (Figure 2.1). This has been useful in a number of ways, for instance, gathering positional data of phone users. Since mobile phones need to register their position to a nearby cell tower, it is possible for mobile phone providers to gather data on location, volumes of calls and patterns from the users. The data lifts some concerns regarding the privacy of users, but in scientific randomized environments it provides huge opportunities to uncover the structure and dynamics of social networks. From small-scale individual perspective, to large-scale collective behaviour of masses, research can now be done with unprecedented degree of accuracy.

Imagen que contiene captura de pantalla

Descripción generada con confianza muy alta

**Figure 2.1:** Converging trends in mobile-cellular subscription and global population. Source: ITU World Telecommunication/ICT Indicators database

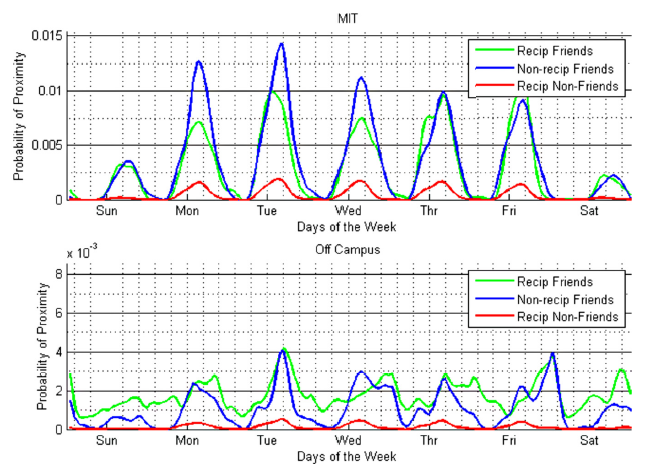
## 2.1 Research from mobile phone data

As smartphones have become more popular, Reality Mining has also become a relevant field of study that has attempted to utilize mobile phone data to understand human behaviour [58]. Reality Mining studies human interaction based on the usage of wireless devices such as mobile phones and GPS systems providing a more accurate picture of what people do, where they go and with whom they communicate with. For instance, [39] showed that user’s social network could be detected using calls, locations and Bluetooth proximity logs. Since then, many studies have tried to study different attributes of human behaviour, such as demographic information [40, 45, 46], predicting mobile phone interaction [41], personality [42] or predicting friendships [43]. The following, are examples that are especially relevant to the topic that is being discussed in this thesis, and will therefore be explained in greater detail.

**Calling patterns for extended periods of time hold predictive power.** Onnela et al [16] analysed the structure of weighted call graphs arising from reciprocal calls, being able to differentiate the nature of the relationship between dyads: “whereas a single call between two individuals for 18 weeks may not carry much information, reciprocal calls of long duration between two users serves as a signature of some work-, family-, leisure-, or service-based relationship”. In other words, calling or not calling for long periods of time does not necessarily mean that there is no relationship between two people (i.e. one might have childhood friends that only contacts one or twice a year), but long calling patterns are a strong indicator of some kind of relationship between them.

**It is possible to accurately infer 95% of friendships based on observational phone data**, where friend dyads demonstrate distinctive temporal and spatial patterns in their physical proximity and calling patterns [6]. In their research Eagle et. al. (2009) demonstrated the power of collecting not only communication information but also location and proximity data from mobile phones over an extended period of time. They were able to show that dyads that are self-reported as friends show distinctive behavioural signatures. Detecting friendship in a dyad is fundamentally a different challenge than observing whether the dyad is near to each other (i.e. two people can be good friends, but barely see each other for most part of the year). However, spatial and temporal context is an important indicator of types of relationships, where time spent between two people in a location far away from work on a Friday night is quite different from spending the same time in a location close to work on a Monday morning. The research was based on the concept of scripts used in cognitive science: mental constructs like schemas, but which consist of sequences of actions or events necessary to achieve a goal [17]. Specifically, they focused on examining whether proximity, location and time cluster together in a predictable fashion and can predict friendship. By capturing the average hour-by-hour levels of proximity for symmetric friend and nonfriend dyads, as well asymmetric dyads, it can be seen that proximity is in general much higher for friends, but time and location are also important factors (Figure 2.2).

**Modelling the temporal nature of human behaviour can accurately be used for prediction through several machine learning methodologies**. As shown in [3], certain assumptions such as the predictive power of the time of the week and time of the day of an interaction, the similarity of temporal patterns across week days, or the possibility to combine local patterns into predictive global features can be powerful tools. This allows to design data representations that can be used as input for deep learning models, expanding the possibilities of raw mobile phone meta-data. Felbo et. al. [3] created what they called a “week-matrix”, a 3-dimensional matrix with eight channels (number of unique contacts, calls, texts, and total duration of call for incoming and outgoing interactions) where the x-axis are the hours of the day and the y-axis the days of a week (Figure 2.3). Each cell in the matrix represents the amount of activity for each of the channels at any given time. This week-matrix can then be used as input for a Convolutional Neural Network to predict gender with state-of-the-art accuracy using only the temporal modality in mobile data, improving the previous state-of-the art approach that also exploited patterns related to mobility.



**Figure 2.2:** Proximity probabilities at work and off campus for symmetric friend, asymmetric friend, and nonfriend dyads. Probability of proximity is calculated for each hour in the week and is generally much higher for friends than nonfriends. However, it is also apparent that asymmetric and symmetric friend dyads have different temporal and spatial patterns in proximity, with symmetric friends spending more time together off campus in the evenings. Image from [6].

**It is possible to classify contacts as “family”, “work” or “social” using mobile SMS and call records** [38].Social networks nowadays classify contacts between users as “friend” or “friend of friend” but don’t give any more details on the type of the relationship. The truth is, people engage in different social roles depending on the person and the situation. In this research participants were asked through questionnaires to classify their relationships in 11 categories corresponding to the 3 facets mentioned with varying levels of intensity. Then 5 factors were defined to characterize the communication patterns in the context of the life facets: intensity, regularity, temporal tendency, channel selection and maintenance cost. These factors were justified by their proven capability to predict social interaction in previous similar studies. Finally, 153 features were extracted from the mobile phone data that were used in different combinations to train classification models such as SVM, Decision Trees and Naïve Bayes models. 90.5% accuracy was achieved classificating contacts a participant communicated with.



**Figure 2.3:** Visualization of a single channel, the number of unique outgoing contacts, in the week-matrix most predictive of the male gender (top) and of female gender (bottom). The week-matrix most predictive of male gender has a higher number of outgoing contacts during the hours from 7am to 4pm on workdays while the “female" week-matrix's outgoing contacts are spread across the day. Image from [3].

## 2.2 SensibleDTU project

As mentioned in the previous section, the SensibleDTU project includes records of interactions and mobile communications between around 1000 students from the Technical University of Denmark over a period of 2,5 years (from late 2013 to early 2016). With this project, several studies have been feasible to try to understand human behaviour and activity patterns across populations or individuals. For instance, Mollgaard et. al. [5] showed that it is possible to generalize human communication and mobility or physical proximity from populations to individuals with high accuracy, 71% and 85% respectively, while individual statistics only improve the results marginally.

Additionally, it has allowed to model human behaviour by time binning and aggregating weekly interactions between individuals for clustering [4]. This was done by binning by hours the face-to-face interactions in dyads and aggregating them weekly. Then, they were clustered in four groups, based on “lecture”, “weekend”, “evening”, and “night” and showed that for each group there is a distinctive example that represents it. Furthermore, the different groups show different means of interaction, being “lecture” the group that characterises by having almost all face-to-face interactions, “evening” and “weekend” most of the telecommunications and “night” a mix between the two.

Farman and Churchill [44] also found out that people tend to use specific communication mediums in different social roles, such as only using the phone to communicate with your mom, instead of talking through social media.

## 2.3 Missing answers

The studies cited above aggregate the data based on assumptions such as repeatability of the activity patterns across series of weeks or generalizations based on population groups, assuming general traits, or shared activities. However, the data used is in most cases richer and more detailed, and these assumptions, even though proven to be effective for modelling, lead to not taking into account details that can be extremely interesting.

The success of machine learning algorithms generally depends on data representation, since different representations can entangle and hide the different explanatory factors of variation behind the data [15]. As stated before, specific domain knowledge can be used to design representations making certain assumptions, but heavily affecting the performance of machine learning methods. This feature engineering accentuates the weaknesses of current machine learning algorithms; the ability to organize and discriminate data.

One of the aims of this thesis is to study the possibility of using data in a more raw form, trying to maintain the structures and patterns lying in the data without making such assumptions. Instead of creating representations assuming a certain structure, we want to let neural networks find those structures, since they might give a different view on how we think data is structured. In the following chapters the theory of the methodology that we want to explore will be presented, as well as the data that was used and the results from applying these methodologies.

CHAPTER 3

# The dataset

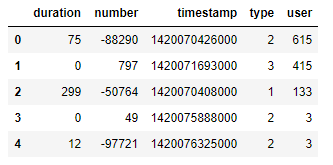
In this chapter a description of the dataset is given, along with the details of the specific parts of the data that have been used for the project.

## 3.1 Sensible DTU

The SensibleDTU dataset is a collection of interactions between students at the Technical University of Denmark. It was collected as part of an experiment that involved around 1000 newly started students that received a smartphone with an app that collected data on social interactions between them on relevant channels: face-to-face interactions, SMS, phone calls, emails, social media (Facebook), as well as GPS locations and other personal information. For this project only a subset of the data has been used, focusing on the interactions happening via calls, SMS or Bluetooth proximity.

## 3.2 Call interactions

Data regarding call interactions involved information on calls between participants as well as other people not belonging to the experiment. For each call a series of features were gathered that gave information about the interaction.



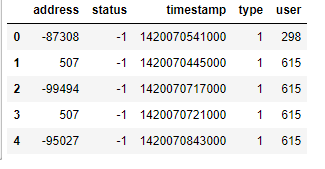
**Figure 3.1:** Example dataframe of raw call data

For each one of the calls the data shows the following information:

* **Duration:** duration of the call in seconds.
* **Number:** unique integer referencing the other person involved in the interaction.
* **Timestamp:** time of the interaction in Unix epoch timestamp.
* **Type:** integer referencing the type of interaction. 1 for incoming calls, 2 for made calls and 3 for missed calls.
* **User:** unique integer referencing the user/participant of the experiment.

## 3.3 SMS interactions

SMS data was gathered in a similar way to call data. It also involved interactions between participants of the experiments and interactions between participants and other agents outside of it.



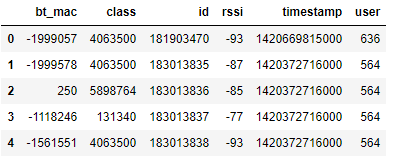
**Figure 3.2:** Example data frame with raw SMS data

For each SMS the data shows:

* **Address:** unique integer referencing the other person involved in the interaction.
* **Timestamp:** time of the interaction in Unix epoch timestamp.
* **Type:** integer referencing the type of interaction. 1 for received, 2 for sent.
* **User:** unique integer referencing the user/participant of the experiment.

## 3.4 Bluetooth

This category of the dataset is one of the most important ones for this thesis. Bluetooth proximity between devices can be used to determine if two people have been in face-to-face contact during a certain time.



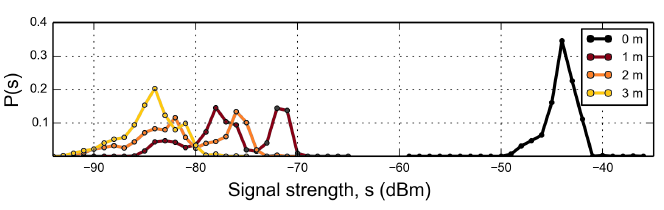
**Figure 3.3:** Example data frame with raw Bluetooth data

For this category the data shows:

* **Bt\_mac:** unique integer representing the MAC address of the device the user/participant is in contact with. This integer coincides with the user ID if the device corresponds to another participant.
* **Class:** unique integer representing the type of device the user is in contact with.
* **Id:** integer referencing the event.
* **Rssi:** received signal strength indicator. Represented in dBm (decibels/m).
* **Timestamp:** time of the interaction in Unix epoch timestamp.
* **User:** unique integer referencing the user/participant of the experiment.

The data was recorded every 5 minutes for each device that has been in contact. This means that for two people in contact, the approximate resolution is of 2.5 minutes.

Since the smartphones were able to gather the RSSI data in dBm, it is possible to infer the distance between de devices in the interaction. As explained in [47], the distance between to devices can be inferred from the received signal strength, since it is inversely proportional to the distance between them. Figure 3.4 shows the distribution of signal strength for two devices related to their distances.



**Figure 3.4:** Distributions of signal strength two devices collapsed into single distributions. Measurements for both phones are statistically indistinguishable, i.e. there is no difference between whether A observes B or vice versa. Image from [47].

Even though the dataset includes other channels of communication, those were not relevant to what it is studied in this thesis and will therefore not be explained. However, as mentioned they include information regarding locations and social media interactions between the participants.

CHAPTER 4

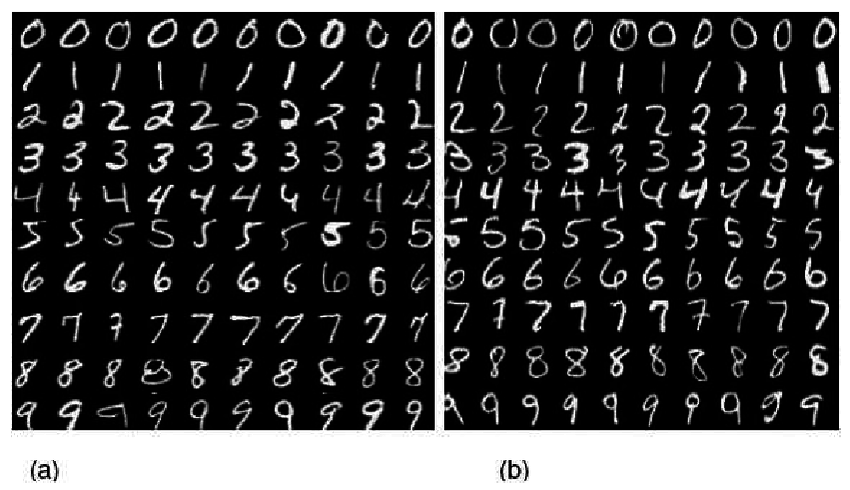
# Autoencoders

Usually, the underlying structure of high dimensional data can be much easier explained in a lower dimensional space that takes into account the relationships in the data. A lot of techniques in machine learning have tried to accomplish this by trying to compress the dimensionality of the data into a smaller space that represents the original data without losing much information. One of these methods are autoencoders [48]. In recent years, some very popular variations of the original autoencoders have emerged that expand the possibilities of these networks from dimensionality reduction to being generative models.

In their simplest form, autoencoders are three-layer artificial neural networks, with the middle layer being of a smaller dimensionality than the other two, and that are trained to take an input data, pass it through the middle layer, reconstruct it in the last layer. This will be further explained in this section, but for the sake of building from the ground up, it is necessary to first take a look at what an artificial neural network is and how it can be used as dimensionality reduction techniques.

## 4.1 Artificial Neural Networks

It is trivial for us humans to define what is shown in Figure 4.1 as a bunch of threes. Even though the characterization of each digit is quite different, for some reason our brain is capable of abstracting the most basic shapes of the image, comparing it with a premade idea of what the number three is and how it looks, and deciding that what is on from of our eyes is in fact the number three. If we tried to do the same thing with a computer program, this task can become incredibly complex.



**Figure 4.1:** Examples of digit three from MNIST dataset

Neural networks are inspired by the human brain, and as the name indicates they are made by a number of neurons connected with each other. A neuron, in its simplest form is a function that takes a number and spits out another number, usually but not necessarily ranging from zero to one. These neurons are organized by layers, where neurons in a layer are connected to all the neurons in the next layer, and where certain patterns of activation of the neurons in one layer will trigger other patterns of activation in the next layer (Figure 4.2).

Imagen que contiene mapa, texto

Descripción generada con confianza muy alta

**Figure 4.2:** Basic structure of an artificial neural network.

For data to go from one layer to the next, the value in each neuron of the first layer needs to be multiplied by a weight corresponding to the link between the neuron and a neuron in the next layer. Therefore, the value a neuron that is not in the first layer receives is given by:

(4.1)

where yi is the value the neuron receives, xi is the value of a neuron in the previous layer, wi is the value of the weight associated to the link between these two neurons and n is the number of links. Then, the value is taken by the activation function in the neuron, that yields the final value the neuron will give for the next layer. The activation function varies from network to network and is a choice that needs to be made during the design of the network. However, it can be generally expressed as:

(4.2)

The bias is an additional term that indicates how high the weighted sum needs to be before the neuron is active. It is in a sense a threshold for the output values from the neuron.

As seen by the expression above, the value of yj can be anything from -∞ to +∞ so the neuron does not know where are the limits of the value to decide whether it should be active or not. In order to solve that, several activation functions can be defined, depending on what the objective of the neural network is.

**Step function**

The step function is the simplest of activation functions. It will yield a one or a zero depending of the input value being higher than a certain threshold.

(4.3)

The downside of this function is in fact that it can only give values of zero or one. In a binary classification problem this is not an issue, but in more complex tasks it can fall short.

**Sigmoid function**

In this case, the activation function is given by the sigmoid function:

(4.4)

This function gives a range of values between zero and one and forces them to be very close to these. If the input values are in the middle ranges, small changes in the input values will force the function to incline more or less to the extremes, and therefore behaves in a similar way to the step function. However, if the input values are close to the extremes, the sigmoid function is very unsensitive. In any case, it is still one of the most used functions in classification problems.

**Tanh function**

The tanh function is a scaled sigmoid function defined as:

(4.5)

In this case the derivatives are steeper, being a bit better on the extremes than the sigmoid function.

**ReLu**

A ReLu function is given by , and it yields x if x is positive or 0 otherwise. In this case, the output of the function is not bound as in the sigmoid or tanh functions and goes from 0 to ∞, so it can scale really quickly. However, it offers an advantage that the other two do not. With the sigmoid or tanh functions almost all the neurons will activate in an analogue way, and in a big network this is extremely costly. With ReLu, the activations are sparse and efficient and the network is lighter. Nowadays, ReLu is the most used activation function, and is also the one that is most similar to how neurons and neural networks work in nature.

The process just explained above needs, or generates, a quite large amount of weights and biases that need to be tweaked to meet the objective of the network. In order to do that, an algorithm called backpropagation is used, but before explaining what backpropagation does, it is necessary to test how good that network is doing its job so it can “learn” to do it better. For a neural network, learning means finding the weights and biases that minimize the cost function. The cost for a given example is the difference between to output of the network, and what the output should be given the input, expressed as the sum of the square differences of each of the neurons in the final layer. Doing this for all of the training examples gives the average training cost. Therefore, it is possible to define a cost function that takes all the weights and biases as inputs and the average cost as output, and find a local minimum to the function through stochastic gradient descent.

Backpropagation takes two phases: propagation and weight update. The propagation phase takes the input and propagates it through the network to generate the output values, then calculates the cost as defined above, and finally propagates the output of activations back through the network using the training pattern in order to generate the deltas of all output and hidden neurons. In phase two, the weight’s output delta and the input activation are multiplied to find the gradient of the weight and then a ratio of the weight’s gradient is subtracted from the weight.

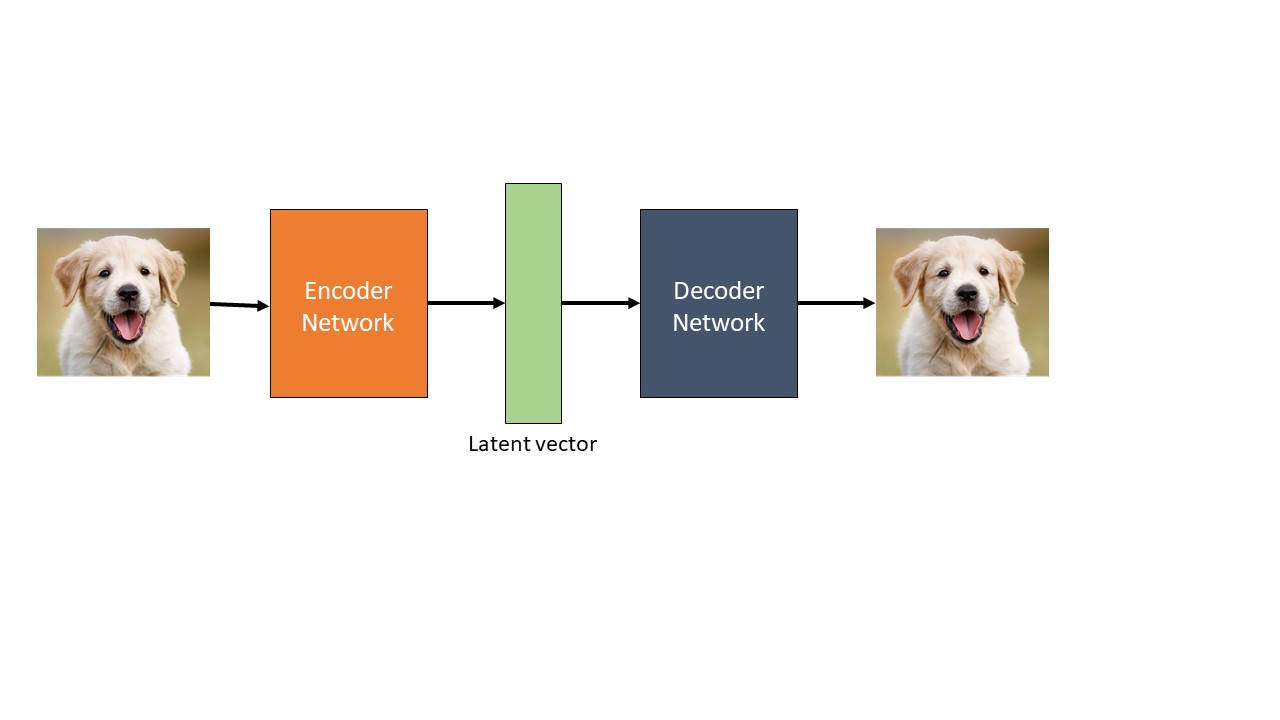
The ratio is called learning rate, which can be big for faster learning or smaller for slower more accurate learning. The sign of the gradient of the weight indicates the correlation with the weight and, therefore, the weight must be updated in the opposite direction to descend the gradient.

Repeating this process for all of the learning examples is incredibly time and resource consuming. To solve this, the training data is divided in mini-batches and for each one of those a step of the gradient descent is computed with backpropagation.

Autoencoders are variations on the architecture of neural networks, adjusted for the task of dimensionality reduction or feature extraction. However, they follow the same principles in their way of operating.

## 4.2 Autoencoders

Autoencoders are the simplest form of encoding neural networks; they are multilayer artificial neural networks used for unsupervised learning of efficient encodings. The aim of the network is to learn a representation of the input, that can be an image or a vector with a high dimensionality, in a low dimensional space by compressing the data into a short code, and uncompressing it back reconstructing the original input (Figure 4.3). The first step is done through an “encoder” network while the second phase with a “decoder” network. The middle layer that contains the low dimension representation of the input is usually called the “latent representation” or “latent vector”.



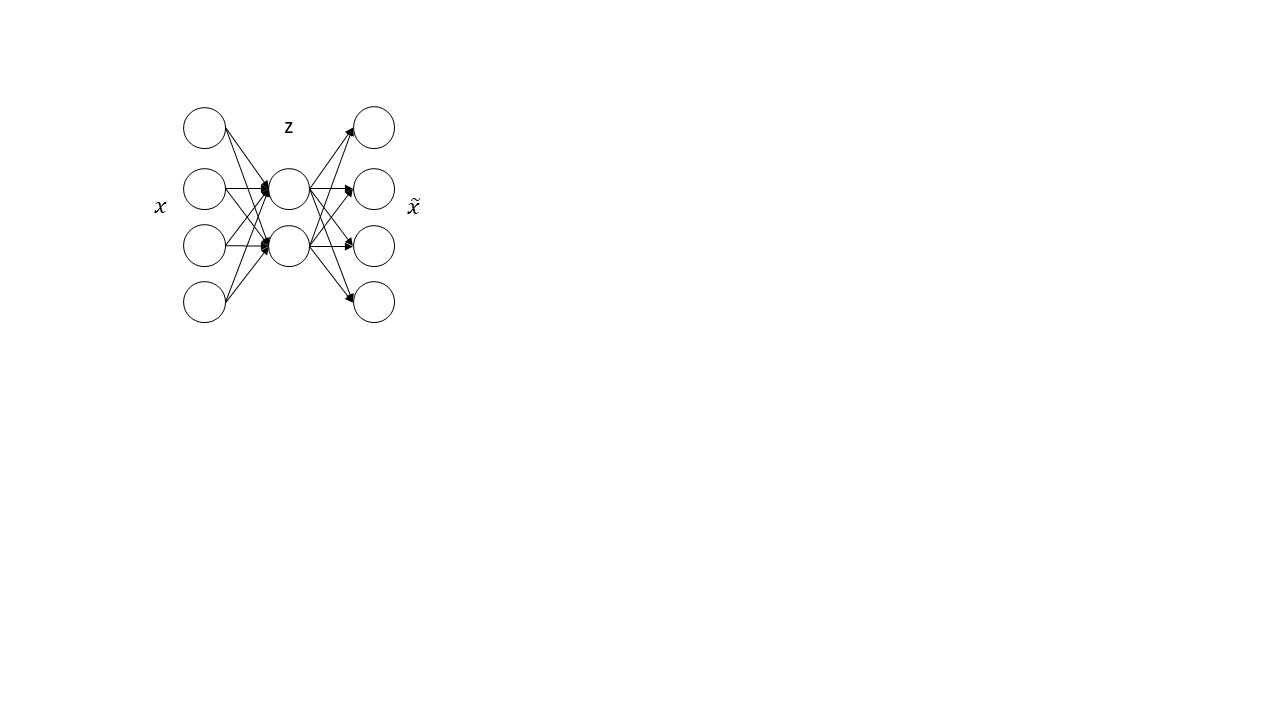
**Figure 4.3:** Basic structure of an autoencoder with an encoder, decoder and latent vector.

The most basic architecture of an autoencoder is a feedforward, non-recurrent neural network as can be seen in Figure 4.4, with the output layer having the same number of nodes as the input layer.

The encoder is simply a number of layers that can be fully connected or convolutional, that will take the original data and transform it to a lower dimensional representation. This creates a “bottleneck” that we call the latent representation.

The decoder on the other hand, takes whatever is in the bottleneck and tries to reconstruct the input data again in a similar way as the encoder using a series of fully connected layers.

Finally, the loss function of the network compares the reconstruction and the original data. By comparing pixel to pixel or node to node differences, the network is forced to learn a meaningful representation of the input data in a low dimensional code. In a sense, it can be said that autoencoders are a data specific learned compression that fits to a concrete case that is being worked on.



**Figure 4.4:** Example of a feed-forward, non-recurrent, fully connected autoencoder network.

Following the previous example, the latent representation and reconstruction can be expressed as functions of their respective inputs:

(4.6)

(4.7)

Where is the latent representation based on the original input, is the reconstruction, and are the weights matrices and and are the bias vectors. To compute the loss the following function can be set up, which is the sum of the squared errors between and

(4.8)

(4.9)

(4.10)

and that can be optimized using stochastic gradient descent.

By having a bottleneck layer, autoencoder networks are forced to learn the fundamental structures in the input data to be able to reconstruct it again. Autoencoders of this kind have been proven useful in many situations, from efficient compression of speech spectrograms [19], HIV classification [20], embedding of paragraphs in text generation [21], or cross-language learning that allows to build models for different languages [22].

## 4.3 Denoising Autoencoders

Denoising autoencoders are similar to regular autoencoders, but with an intentional corrupted input (Figure 4.5). Introduced in 2008 (Vincent et al., 2008)[49], to train a denoising autoencoder it is necessary to perform preliminary stochastic mapping in order to corrupt the data and use as input for the network. The loss however, must still be computed with the original input instead of the corrupted input . With this technique, the robustness of the model is increased by teaching the network to bypass errors on the input to reconstruct the data.

Imagen que contiene perro

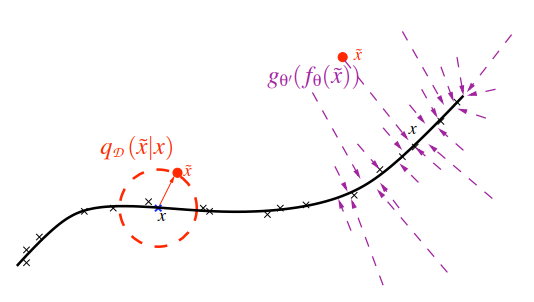
Descripción generada con confianza alta

**Figure 4.5:** Example of denoising autoencoder. The corrupted data is used as input and the network tried to reconstruct the original data.

The corruption can be introduced I many ways, where prior knowledge can be applied. However, three techniques are discussed by the authors above mentioned: additive isotropic Gaussian noise, masking noise and salt-and-pepper noise.

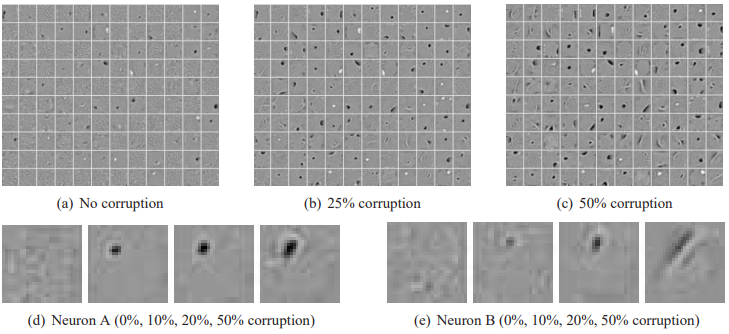
* Gaussian noise: noise is given by (GS):
* Masking noise: a fraction of the elements of x chosen randomly are forced to 0
* Salt-and-pepper noise: a fraction of the elements of x are set to their maximum or minimum values (usually 0 or 1).

As explained by Vincent et. al. [49], there is an intuitive way of understanding the denoising process that takes place in the autoencoder using the *manifold assumption* (Chapelle et al., 2006). The manifold assumption states that high dimensional data concentrates near a non-linear low-dimensional manifold. During the denoising, the network learns a stochastic operator that maps the corrupted back to . The corrupted examples are likely to be further away from the manifold than the uncorrupted ones and therefore the stochastic operator learns a map that tends to go from lower probability points to nearby high probability points on or near the manifold (Figure 4.6).



**Figure 4.6:** Manifold learning perspective. Suppose training data (x) concentrates near a low-dimensional manifold. Corrupted examples (•) obtained by applying corruption process qD(Xe|X) will generally lie farther from the manifold. The model learns with p(X|Xe) to “project them back” (via autoencoder g‘θ (fθ(·))) onto the manifold. Image from [49].

Denoising autoencoders produce better feature detecting filters than regular autoencoders. As it can be seen in Figure 4.7, as we increase the noise level, denoising training forces the filters to differentiate more, and to capture more distinct features (i.e. local blob detectors, stroke detectors, or character part detectors).



**Figure 4.7:** Filters learnt by denoising autoencoder on MNIST digits (see section 5.1) with zero-masking noise. a-c show some filters learnt by the autoencoder with various noise levels. d and e are zoom ins on the filters obtained for two neurons. Image from [49].

The use of these autoencoders does not differ much from the regular autoencoders, since they just provide a more robust example. For instance, they have been successful in speech emotion recognition [24], extraction and composing of robust features from images [23], or denoising and blind inpainting of images [25,26]. Qualitative experiments show that, contrary to ordinary autoencoders, denoising autoencoders are able to learn Gabor-like edge detectors from natural image patches and larger stroke detectors from digit images [49].

## 4.4 Sparse Autoencoders

Sparse autoencoders don’t impose the constraint of the latent layer to be much smaller than the original input. Instead, the latent layer can be of the same of even greater dimension than the input. In order to still be able to grasp the fundamental structure of the data, they impose a sparsity constraint in which only certain cells in the hidden layer can be activated at a time, being still able to discover interacting structures in the data. This is the case for k-Sparse Autoencoders (Makhzani et. al., 2013).

K-Sparse Autoencoders find the k highest activations in the latent vector z and zero out the rest. Then the error is backpropagated only through the k active nodes. For K-Sparse Autoencoders, reducing the number of k active neurons forces the network to learn more and more complete representations of the input data. In high k values, the representation becomes overcomplete and learns highly local features, whereas for low values of k the representation is so sparse that each node can only represent features from a single input data (Figure 4.8).



**Figure 4.8:** Filters of the k-sparse autoencoder for different sparsity levels k, learnt from MNIST (see section 5.1) with 1000 hidden units.

Sparse autoencoders have also been used for speech recognition or object detection on images [28,29,30].

## 4.5 Variational Autoencoders

Variational Autoencoders (Kingma et. al., 2013) are a variation on the autoencoder models to approach the generative modelling problem that GANs (Goodfellow et. al., 2014) and other similar models have tried to solve. The main use of these autoencoders is to take an input and to generate new data that is similar to it.

Contrary to regular autoencoders, instead of mapping the input to a fixed vector, they map it to a distribution. Therefore, instead of having a single vector as the latent representation, VAEs have two vectors representing the mean and deviation of the distribution (Figure 4.9). With this, anytime it’s required to feed the decoder network, only a sample from the distribution is needed to generate an output similar to the input.

In order to train these autoencoders, the loss function consists of two parts instead of one:

(4.11)

where the first term represents the reconstruction loss, similar to regular autoencoders with an expectation operator since we are sampling from a distribution. The second term is a KL divergence term that tries to make the distribution the autoencoder is learning similar to a normal distribution N(0,1). However, there is one last step that needs to be taken in order to train the model, which is called the reparameterization trick.

Imagen que contiene texto

Descripción generada con confianza alta

**Figure 4.9:** Example of variational autoencoder.

The random nature of the sampled latent representation makes it impossible to backpropagate through it. On the other hand, it can be represented as

(4.12)

where and are parameters that the network is trying to learn and an that is going to be forced to be a standard Gaussian distribution. Instead of having a full stochastic node that stops backpropagation through it, we split it up to a part where it is possible to do backpropagation and a stochastic part where it can’t. The advantage if this is that the stochastic part is fixed and we don’t need to backpropagate through it (Figure 4.10).

Imagen que contiene texto, mapa

Descripción generada con confianza muy alta

**Figure 4.10:** Representation of the reparameterization trick. Image from [50].

In Figure 4.11 image generations from a VAE trained on the MNIST (see section 5.1) dataset can be seen. After training the model with a 2-dimensional latent space, linearly spaced coordinates on the unit square can be transformed through the inverse CDF of the Gaussian to produce values of the latent variables. Each of these can then be plotted with the trained generative model.

A variation on the Variational Autoencoders are Disentangled Variational Autoencoders [33]. This autoencoders force the different neurons in the latent representation to be uncorrelated so that they try to learn something different about the input data. In order to implement this, it is necessary to modify the loss function seen previously by adding a new β hyperparameter that evaluates how the KL divergence is present in the loss function.

(4.13)

Imagen que contiene rojo

Descripción generada con confianza alta

**Figure 4.11:** Visualizations of learned data manifold for VAE

CHAPTER 5

# Preliminary work

Before starting to work on the SensibleDTU dataset some preliminary work has been done on the MNIST database. The purpose of this section is to validate the idea of using autoencoders to encode data into different types of representations that can be clustered in a meaningful way.

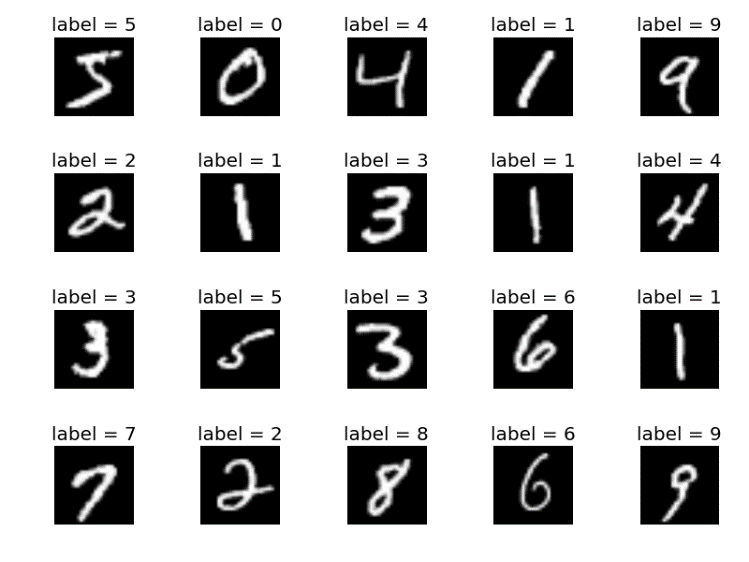
The work involved training an autoencoder and encoding several unlabelled handwritten digits to analyse the generated latent space. This work is able to give an idea of whether the latent representations in a case similar to the SensibleDTU dataset can be clustered in different groups for each kind of digit.

In this section, the work on the MNIST database will be presented, using regular autoencoders and variational autoencoders with varying sizes of the latent representation.

## 5.1 MNIST database

The MNIST database (Modified National Institute of Standards and Technology database) is a widely used database in the training of image processing systems and for the training and testing of machine learning models. It was originally created by mixing samples from the NIST dataset, since at the time the training and testing samples came from different sources, the American Census Bureau’s employees and the American high school students, which wasn’t suitable for machine learning experiments.

The database consists of 60,000 training images and 10,000 testing images normalized to 28x28 pixel images in grayscale levels. Half of each set, training and testing, were obtained from the two sources cited above. The data is labelled with the corresponding digit value; however, it can be used for both supervised and unsupervised learning.



**Figure 5.1:** Example of digits from 0 to 9 on the MNIST database with their corresponding labels.

## 5.2 Encoding

To encode and analyse the database two methods have been used and compared: a regular autoencoder and a variational autoencoder. Both methods have been compared to the purpose of this thesis to evaluate which one would be able to give better results.

The encoding for both methods takes two parts. First, the network is trained with a training sample of 60,000 images, as explained in section 4, fine tuning the variables of the network in batches. Then, the test sample is parsed through the network once without backpropagation to obtain the latent representations for each of the images. However, there are a couple of considerations that need to be taken into account before obtaining the latent representation.

The idea behind this work is to be able to visualize the representation, which depends greatly on the dimensionality of the latent representation that is chosen. For a 2-dimension latent representation, visualizing it is trivial, but for larger representations some post-processing is needed. As it will be shown in this section, several dimensionality reduction and visualization techniques have been tested to evaluate their viability: PCA, t-SNE, and UMAP.

### 5.2.1 Fully connected Autoencoder

A fully connected Autoencoder has been tested with two latent representations: a 2-dimensional and a 20-dimensional representation. Both methods have consisted in a similar architecture for the hidden layers, but with different latent vector dimensionalities.

**- 2-dimension latent space representation:**

In order to be able to easily visualize the latent space generated by the autoencoder, the first method used is an autoencoder with the following architecture:

Imagen que contiene captura de pantalla

Descripción generada con confianza alta

**Figure 5.2:** Architecture for the regular autoencoder using the MNIST database. The architecture consists of a 500-dimension hidden layer and a 2-dimension latent representation.

The first layer of the network is a 784-dimension vector corresponding to the 28x28 pixel image of the MNIST database. This vector is simply obtained by unravelling the image matrix to obtain a single vector. The second layer is a 500-dimension layer and finally the 2-dimension latent representation is generated.

The steps in this example have been selected arbitrarily, but changing them doesn’t change the results in a noticeable way. This model allows to easily plot the latent space in a 2D graph and gives an idea of how the values for the different digits change from one to another.

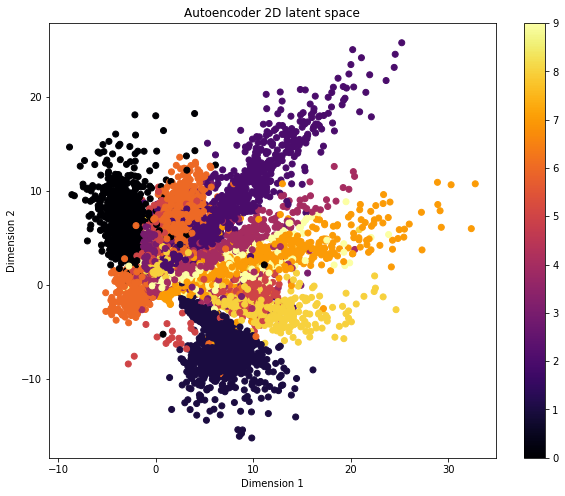
The training of the model has been done over 100 epochs. The following are the results for the image reconstruction for epoch 10 and epoch 100. In order to make the network as lightweight as possible, the activation function in the nodes is a ReLu. This is not necessarily an issue at this point since these networks can easily be run in a regular laptop, but when working with 7000+ dimension vectors of the SensibleDTU dataset it can become a problem.

|  |  |
| --- | --- |
| Imagen que contiene teclado, electrónica, ordenador, máquina de escribir  Descripción generada con confianza muy alta | Imagen que contiene teclado, electrónica  Descripción generada con confianza muy alta |

**Figure 5.3:** Reconstruction of image subset in epoch 10 (left) and epoch 100 (right).

Subsequent epochs improve significantly the resolution of the reconstruction making it look more similar ot the original input (Figure 5.3). The reconstruction accuracy serves as a first step to measure how well the network is working: if the reconstructions do not look like the input data at all, looking at the latent representation does not make much sense.

In Figure 5.4, the 2-dimension latent space can be observed, with each data point labeled to their corresponding digit. The different digits generate latent representations that are in a way similar to each other. However, even though each digit seems to be grouped, the clusters would not be clear without labeling, which would be crucial to do in the case of the SensibleDTU dataset.



**Figure 5.4:** Plot of the 2D latent representation for MNIST database, coloured by label.

**- 20-dimension latent space representation:**

The same process has been repeated but using a 20-dimension latent representation instead. The architecture of model used for this method can be seen in Figure 5.5.

The training if this network has followed a similar procedure as the 2-dimension latent representation network, so the explanations will be omitted to avoid repetitions.

To visualize this representation is not a direct task, and it is necessary to post process each data point into a lower dimension space. There are several techniques that have been proven to be useful in reducing the dimensionality of data for visualization tasks. For this project three have been chosen and tested to prove their viability: Principal Component Analysis, t-SNE UMAP.

Imagen que contiene captura de pantalla

Descripción generada con confianza alta

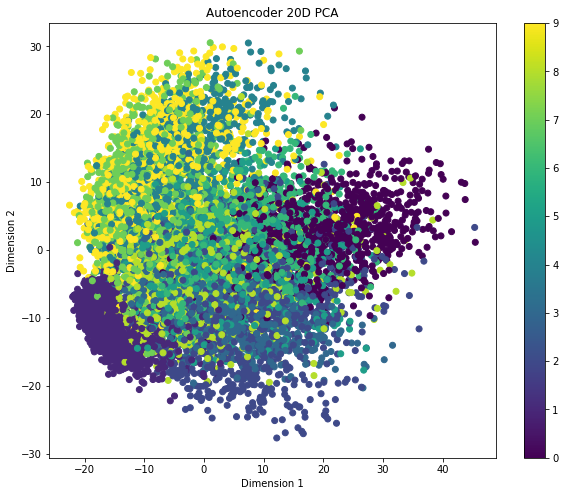
**Figure 5.5:** Architecture of a regular autoencoder with a 500-dimension hidden layer and a 20-dimension latent representation.

**PCA**

PCA, or Principal Component Analysis, (PCA; Hotelling, 1933) is a statistical procedure that uses orthogonal transformations to convert a set of high dimensional observations with highly correlated variables, into a set of low dimensional uncorrelated variables. This is done by finding the principal components that account for the maximum variance in the data. The principal components create an uncorrelated orthogonal basis set of vectors that represent the original data in a lower dimensional space.

With a PCA it is possible to reduce the 20-dimension space into a 2-dimension space that we can visualize.

The results for the PCA (Figure 5.6) are not satisfactory, since they do not yield a result that is better than the 2-dimension encoding shown previously. As it can be seen in Figure X, plotting both axes against each other does not give any good idea of where the representations for each digit are, and without labels, clustering the representations into groups to further analyse them would be highly difficult.



**Figure 5.6:** Visualization of resulting principal components for PCA

**t-SNE**

t-SNE (T-distributed Stochastic Neighbour Embedding) is a technique developed by Laurens van der Maaten and Geoffrey Hinton that visualizes high-dimensional data by giving each data point a location in a two or three-dimensional map [35]. It models each point in a way that similar points are modelled nearby while dissimilar ones are modelled far away from each other and it is often used to visualize high dimension representations learned by artificial neural networks. As the author explains, linear dimensionality reduction techniques such as PCA focus on keeping the low dimensional representations of dissimilar points far apart, but for high dimensional data points that lie in a low dimensional map, it is more important to keep similar points close to each other, which is not possible with linear mapping.

The t-SNE algorithm works in two phases. First, it constructs probability distributions over pairs of high-dimensional objects in a way that similar objects have a high chance of being picked. In a second phase, it defines a similar probability distribution over the points in the low dimensional space and minimizes the KL divergence (Kullback-Lieber divergence) between these distributions with respect to the points in the map. It is derived from the SNE method presented by Hinton and Roweis (2002), with the difference that instead of using Gaussian distributions for both the high dimensional space and the low dimensional space, it uses a Student-t distribution in the low dimensional space. This is done with the objective of improving the crowding problem in the optimization of the method.

The algorithm defines the following similarities for the high and low dimensional space. In the high dimensional space similarity of data point xj to xi is the conditional probability (pji) (5.1) that xi would pick xj as its neighbour if neighbours were picked by their probability density under a Gaussian distribution centred around xi. This means that for nearby points the probability will be high whereas for separated points it will be infinitesimal.

(5.1)

where is the variance of the distribution centred around xi.

The value of is given by doing a binary search of thethat reduces the probability Pi (5.2) with a fixed perplexity selected by the user. The reason for this is because there isn’t a single value of that is optimal for all data points in the dataset, since the density of the points might change from region to region. In dense regions a lower value of would be advisable, while in sparse regions a higher value of is more appropriate.

(5.2)

where H(Pi) is the Shannon entropy of Pi measured in bits

(5.3)

In the low dimensional space, the probabilities are given by a Student-t distribution with one degree of freedom.

(5.4)

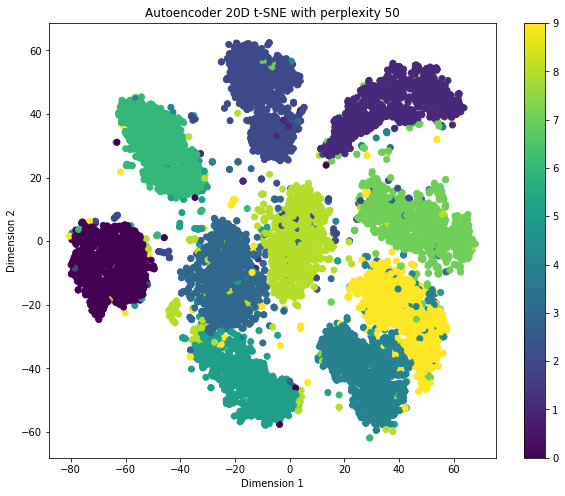
The use of the Student-t distribution with one degree of freedom is justified by the property that approaches an inverse square law for large pairwise distances in the low dimensional map. Finally, in order to compare the faithfulness in which qij models pji, the Kullback-Liebler divergence (5.5) is used, minimizing the sum of KL divergences over all data points with gradient descent.

(5.5)

This method has been applied to the 20-dimensional space generated by the autoencoder obtaining the following results for several runs of perplexity. The best results are obtained from perplexities 30 and 50 (Figure 5.7).

The negative aspect of using the t-SNE algorithm is that it is not particularly fast. The computing time the t-SNE with perplexity=30 is 4min and 52 seconds.

Imagen que contiene texto

Descripción generada con confianza alta

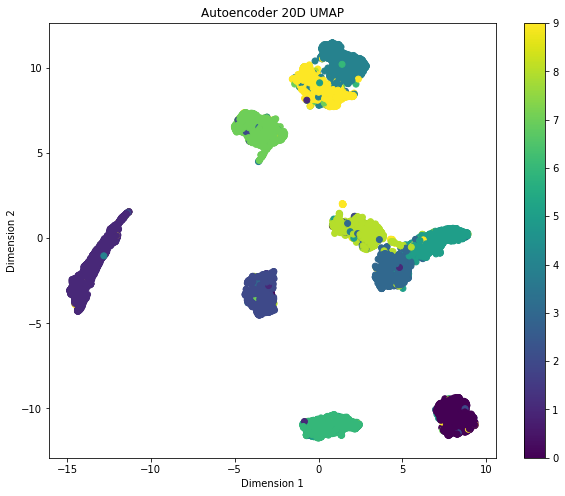
**Figure 5.7:** Visualization of resulting spaces from t-SNE with different values of perplexity. Perplexity=30 (top) and perplexity=50 (bottom) show similar results.

**UMAP**

Uniform Manifold Approximation and Projection for Dimension Reduction [37] is a recently published technique for dimensionality reduction constructed from a theoretical framework based in Riemannian geometry and algebraic topology. As described by its creators, it competes with t-SNE in performance but with much shorter computation times.

UMAP uses local manifold approximations and patches together their fuzzy local simplicial set representations, to construct a topological representation of the high dimensional data. Then, given a low dimensional representation of the data, a similar process is used to construct an equivalent topological representation, by optimizing the layout of the data representation in the low dimensional space minimizing the cross-entropy between the two. The construction of the fuzzy topological representations takes two steps. First it approximates a manifold in which the data is assumed to lie and then constructs a fuzzy simplicial set representation of the approximated manifold.

The results from this algorithm are quite similar to the results obtained from the t-SNE algorithm (Figure 5.8). The only difference is the computation time for both methods where for UMAP is 21.8 seconds.



**Figure 5.8:** Visualization of resulting space from UMAP

### 5.2.1 Variational Autoencoder

As it has been explained in section 4, the main objective of Variational Autoencoders is not encoding but image generation. The reason why they are called autoencoders is because of the similarities with regular autoencoders in which there is an encoder and a decoder network that resembles a traditional autoencoder. However, they still create a latent representation that can be useful for this thesis. In the following sections the tests done on variational autoencoder will be presented, in a similar way the what it has been done for the autoencoder. First a 2-dimensional latent space has been analysed, followed by a 20-dimensional latent space.

**- 2-dimension latent space representation:**

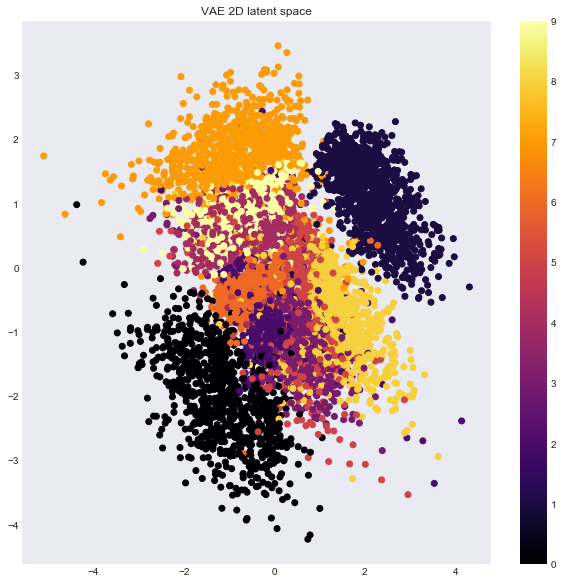
Before going straight to the latent representation of the Variational Autoencoder, a consideration needs to be taken. In section 4 it was shows that Variational Autoencoders need to sample from the latent representation to be able to generate images that resemble the input data. To solve this, and to be able to train the network two latent vectors are created, one representing the mean of the distribution and another one representing the standard deviation. In order to be able to visualize the latent space, the mean vector of the distribution has been chosen, and a similar process to what it’s been done with the autoencoder has been repeated. The architecture of the variational autoencoder trained can be seen in Figure 5.9.

Plotting the two dimensions on the latent mean vector for digits with their corresponding labels gives the result in Figure 5.10. As it can be observed, the results are slightly better than for the regular autoencoder trained in similar conditions. However, without labels, most of the digit clusters in the middle region of the plot would be difficult to obtain.

Imagen que contiene captura de pantalla

Descripción generada con confianza alta

**Figure 5.9:** Architecture for the 2-dimension Variational Autoencoder



**Figure 5.10:** Visualization of 2-dimension latent mean vector labelled by digit.

**- 20-dimension latent space representation:**

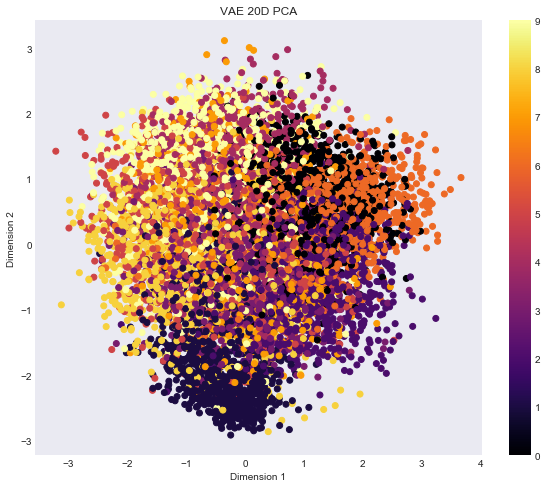
Like the previous tests, the architecture of the model used in this case is the following:

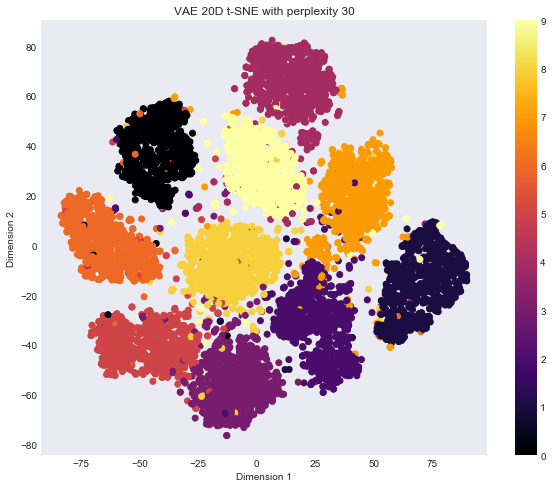
Imagen que contiene captura de pantalla

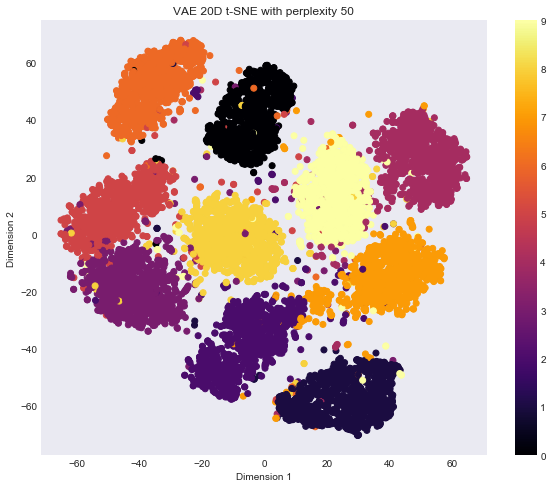
Descripción generada con confianza alta

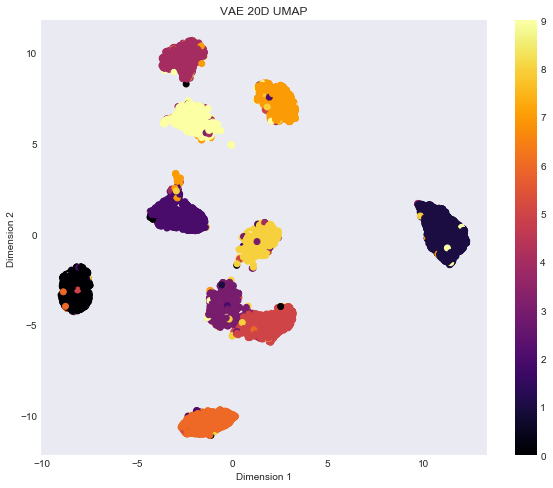
**Figure 5.11:** Architecture for Variational Autoencoder on MNIST database

The results for the PCA, t-SNE and UMAP for the dimensionality reduction can be seen in Figure 5.12. As in the case of the autoencoder, the PCA does not give satisfactory results and it is even worse than a variational autoencoder with a 2d latent representation. For both the t-SNE and UMAP methods, the results are similar, being the clustering in the UMAP slightly better, with much faster computing times. The time difference between the two methods can be seen in Figure 5.13.



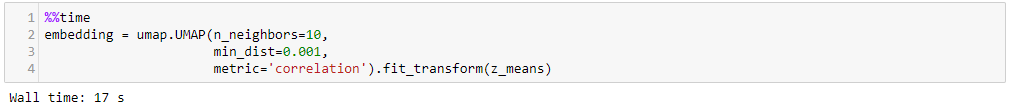






**Figure 5.12:** Results for dimensionality reduction and visualization algorithm. In order from top to bottom: PCA, t-SNE with perplexity 30, t-SNE with perplexity 50 and UMAP.





**Figure 5.13:** Computing ties for t-SNE (top) and UMAP (bottom)

### 5.2.3 Conclusions

The intent of this section was to use an example with a dataset that is readily available to analyse and understand how autoencoders and variational autoencoders are able to create a latent representation, and to see if it would be possible to obtain different types of relationships from the latent space when using the SensibleDTU dataset.

Using a 2-dimension latent representation did not give any satisfactory results for any of the models, so it is hard to think that it would work for the SensibleDTU dataset. This is an expected result, since having only two variables to explain the differences between 10 digits seems too complex. Additionally, for the SensibleDTU case one might think that there must be more than two variables needed to explain personal interactions for a long period of time like the one in the dataset. Even if it was possible, the loss of information would be significant, and not something that would be interested in.

On the other hand, the 20-dimension latent representation for both models together with a t-SNE or UMAP post processing gave some interesting results, being able to cluster all the digits on their own groups. As it has been shown, the UMAP clustering was slightly better and way faster so it would be the ideal choice.

When it comes to deciding between a regular autoencoder and a variational autoencoder, the results are very similar. However, the computing times for the training of both models as seen in Table 5.14 are shorter for the autoencoder. Additionally, the main purpose of a variational autoencoder is not exactly encoding, but generative modelling. The reason why it was tested during this thesis was to see if the variational component improved in any way the encoding process, which did not seem to be the case.

In conclusion, the most suitable method to proceed with the SensibleDTU dataset seems to be a regular autoencoder with the possibility or postprocessing the data with a UMAP or even a t-SNE.

|  |  |
| --- | --- |
| **Model** | **Training time** |
| Autoencoder | 1 min 9 seconds |
| VAE | 4 min 5 seconds |

**Table 5.14:** Computing times for training of Autoencoder and VAE

CHAPTER 6

# Data pre-processing

To be able to translate the results from the previous work using the MNIST database, it is necessary to transform the interactions in the SensibleDTU dataset to “images” that can be fed to the network. This section will focus on the pre-processing of the data and the creation of image like inputs.

As seen in previous sections, the SensibleDTU dataset contains records of interactions between individuals over two years. In order to create the interaction images, all the interactions between individuals in the dataset have been collected, from the 1st of January 2014 to the 31st of December of 2014. Even though the data was gathered from late 2013 to early 2016, the number of users and the volume of interactions changed throughout the project, with some students dropping the experiment. For this project it is necessary that the time span of the data that is used is long enough, but also that it contains as much information as possible. Figure 6.1 shows the amount of activity tracked over the duration of the experiment. The year 2014 has been chosen since it provides the most stable number of users across a period of 1 year.

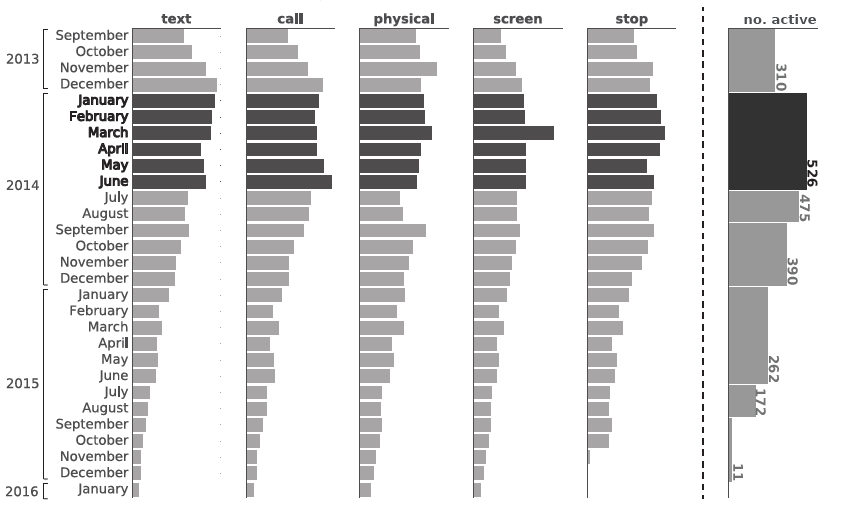
## 6.1 Call data

The call interactions have been then ordered by timestamp and binned every hour, creating a string of 8736 elements, one for each hour in a year. If there has been any call between the users at some point during any of those hours the string would have a 1 in that position, or a 0 if no interaction has happened (Figure 6.2).

[0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, …, 0, 0, 0, 0, 1, 0]

Figure 6.2: Example of interaction string

These strings can then be plotted as a 52x168 matrix, corresponding to the 52 weeks of a year and the 168 hours in a week.



**Figure 6.1:** Overview of the amount of activity tracked in the respective months of the experiment. The far-right bar plot show the number of students in each semester and summer holiday, which are active on more than 75% of days through the period. Image from [52].

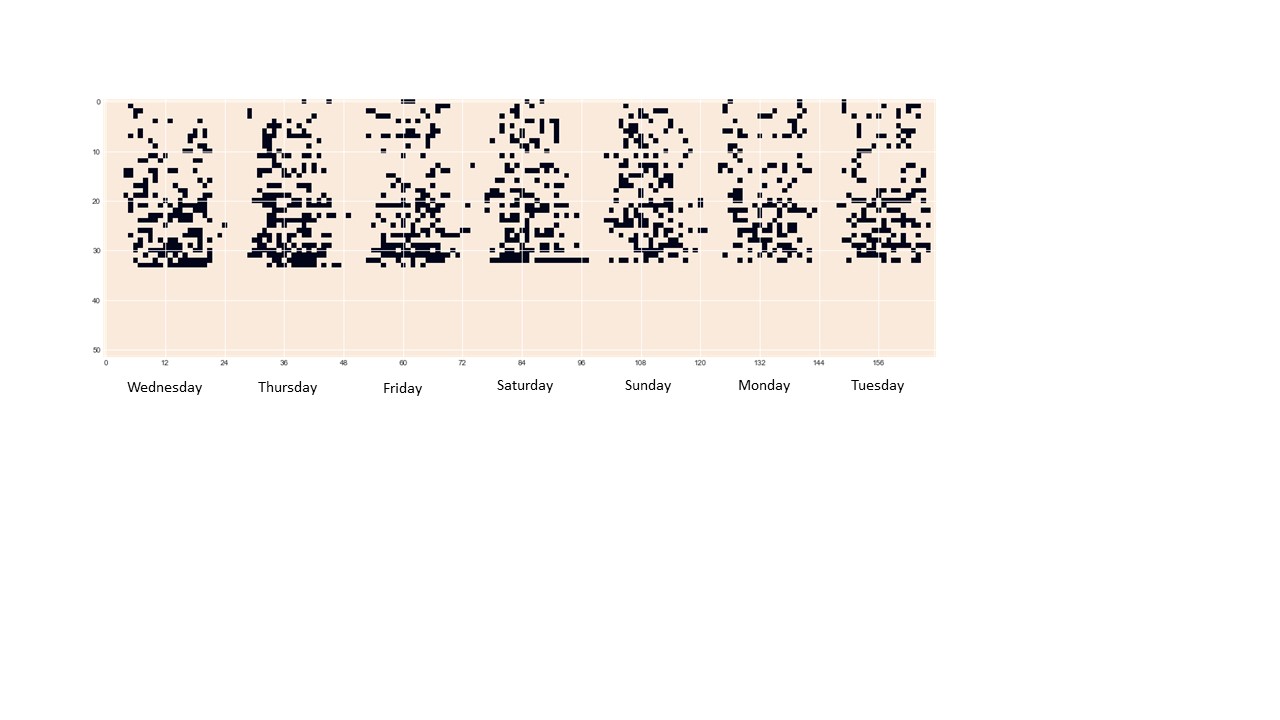
As seen in section 2, the research by Felbo et. al. [3] showed that is possible to assume a certain level of repeatability in the patterns of interaction between people, specially across weeks, and therefore, creating the image matrixes as described, should show some kind of pattern in the interactions. These patterns could show higher intensities during similar timeframes in similar days like weekdays or weekends. In Figure 6.3, examples of users with the greatest number of interaction during the year can be seen. While some dyads show more activity only during the hours around mid-day, for others this activity is more spread out.

One of the common issues that will be found from now on is the sparsity of the data. These images correspond to the 3 dyads with most interactions during the year but most of the dyads have less than 100 interactions, out of the possible 7836 (Figure 6.4). This sparsity comes from the methodology used to model the data, since it almost does not aggregate it and puts it in a mostly raw form. The initial idea behind the project did not consider that the data was not as rich as it could be for this kind of modelling. The sparsity makes it almost impossible for the autoencoder network to learn the structure of the data, as there is practically no data to work with. Training the autoencoder network with this images results on reconstructions that are the same for all the images, which means that it interprets all of them equally.

Imagen que contiene cielo, foto, negro

Descripción generada con confianza muy altaImagen que contiene cielo, foto

Descripción generada con confianza muy alta



**Figure 6.3:** Examples of call interactions for 3 dyads. Each of them contains 52 rows and 168 columns, corresponding to 52 weeks/year and 168h/week.

Imagen que contiene captura de pantalla

Descripción generada con confianza muy alta

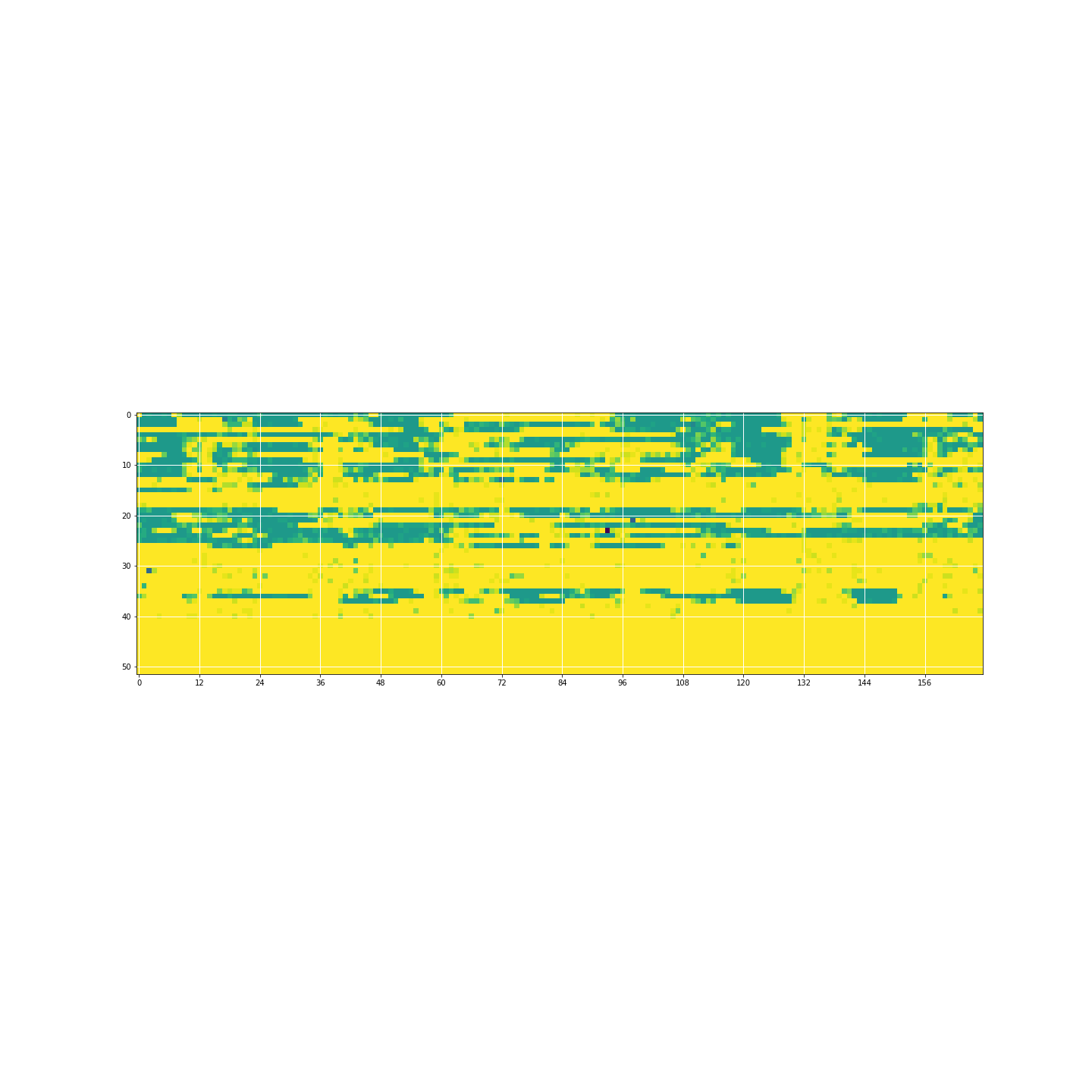
**Figure 6.4:** Distribution of number of call interactions for all dyads.

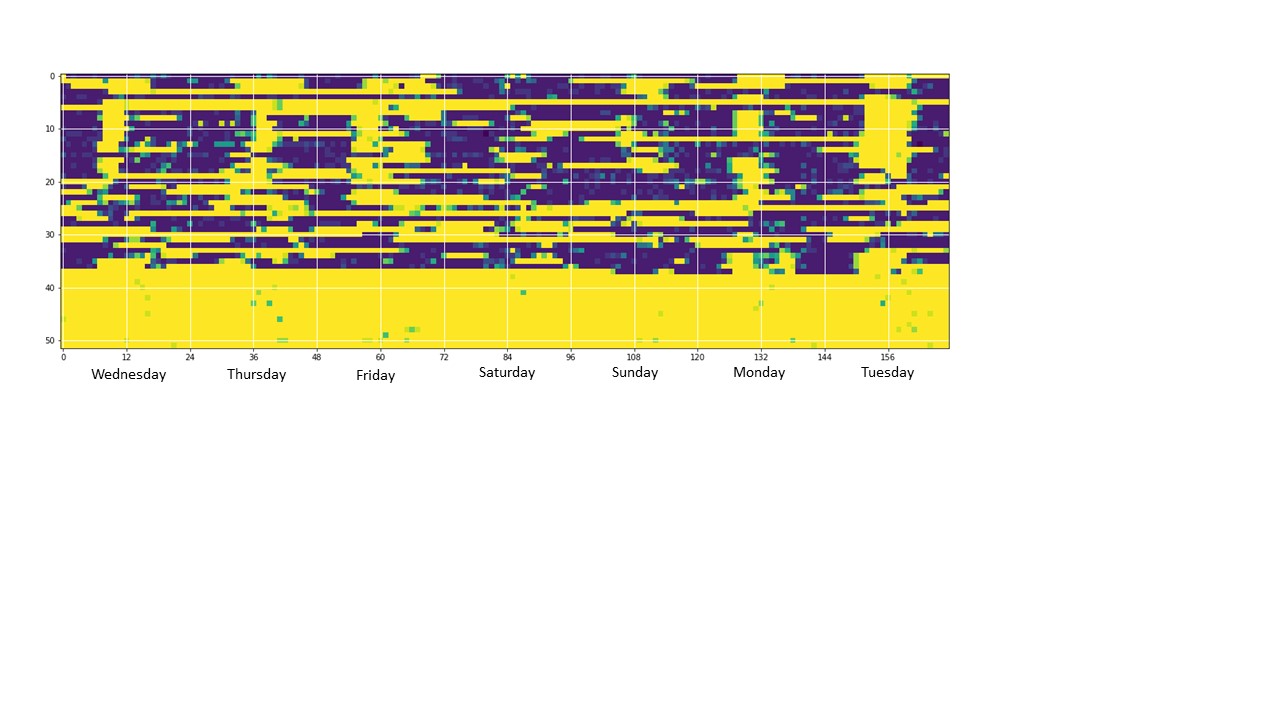
## 6.2 Bluetooth and call data

We can obtain richer and more detailed data if Bluetooth interactions and call data are combined. As stated in [6], two people being close to each other does not necessarily mean that there is some kind of friendship or relationship between them, but it is a good indicator, and it can be taken as a basis to estimate the relationship in a dyad. Moreover, as shown in section 3, Bluetooth data serves as a proxy to calculate the distance between two devices by looking at the intensity of the signal received, and therefore, we can threshold the intensity to make sure that only times when the users are close enough to each other are used. In this section, the call and Bluetooth data have been combined following a similar methodology used for call data.

### 6.2.1 1h bins

The same process has been repeated but this time combining Bluetooth and call data, with the aim of solving the sparsity problem. For each dyad the daily interactions have been binned by hours obtaining 52x168 images in a similar manner to call data. Additionally, the interactions have been added for each hour and all the images have been standardized by the maximum value so that all the pixels range from 0 to 1. Examples of these images can be seen in figure 6.5. Even though the images are fuller, the structures that seemed to be more obvious in the call images are harder to see now.





**Figure 6.5:** Examples of Bluetooth and call data interactions for 2 dyads. The data has been binned and summer every hour. After standardizing the images it can be seen how some dyads have more intense (bottom image) interaction patterns than others (top image).

An issue that seems obvious after obtaining these images is that most of the data seems to be empty after half of the year. This could happen because there was some problem during the gathering of the data or because the user dropped the experiment after some time. Anyway, it is advisable to remove this part for all the dyads, obtaining images of half a year instead of the entire period (Figure 6.6).

Imagen que contiene captura de pantalla

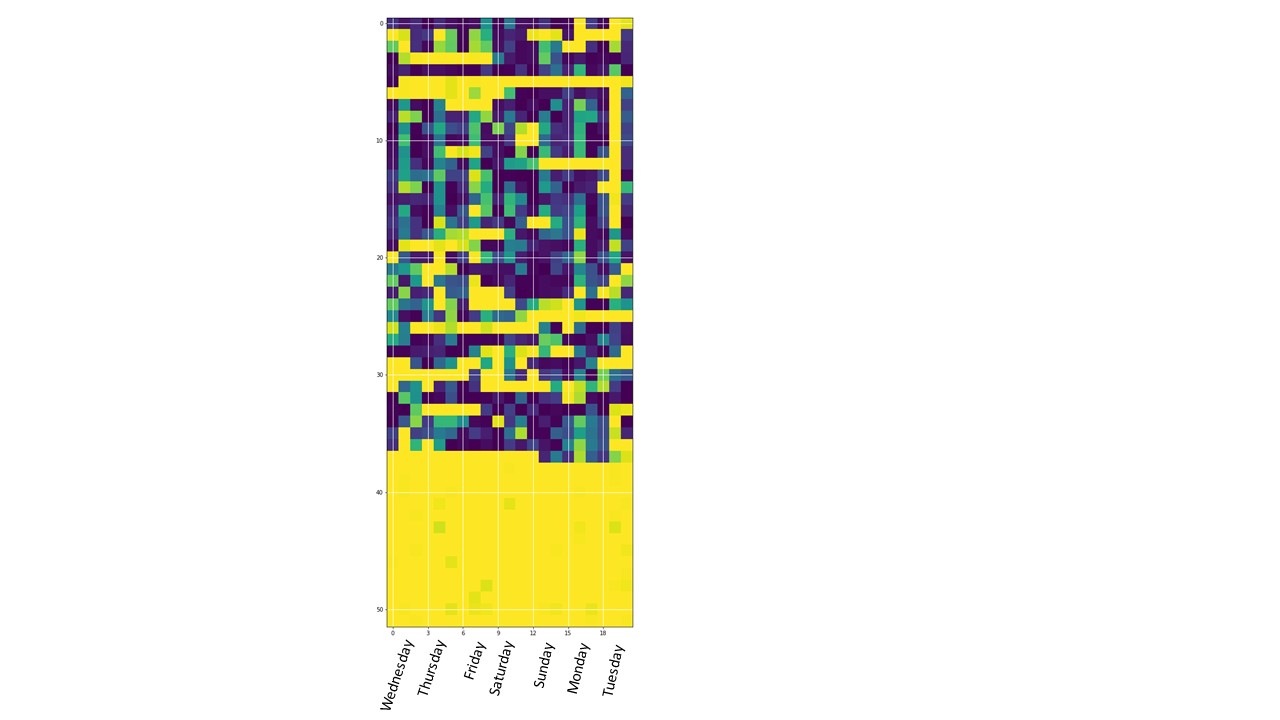
Descripción generada con confianza muy alta

**Figure 6.6:** Example of image for half year period.

### 6.2.2 8h bins

The process of creating these images revealed that the way they are created can improve or worsen significantly the results. One of the problems that seems to be more apparent with the 1h binned images is that the structure or possible patterns were not visible at plain sight. This gave the idea that maybe the autoencoder was not going to be able to learn common patterns between the images.

A similar procedure was repeated using 8h bins instead of 1h (Figure 6.7). Images of size 21x52 with values from 0 to 1 for each pixel where created, using both call and Bluetooth interactions. Again, the interactions were summed to for the period of the bins. These images did not show significant improvements in pattern visualization.



**Figure 6.7:** Example of image for 8h bins

CHAPTER 7

# Analysis and results

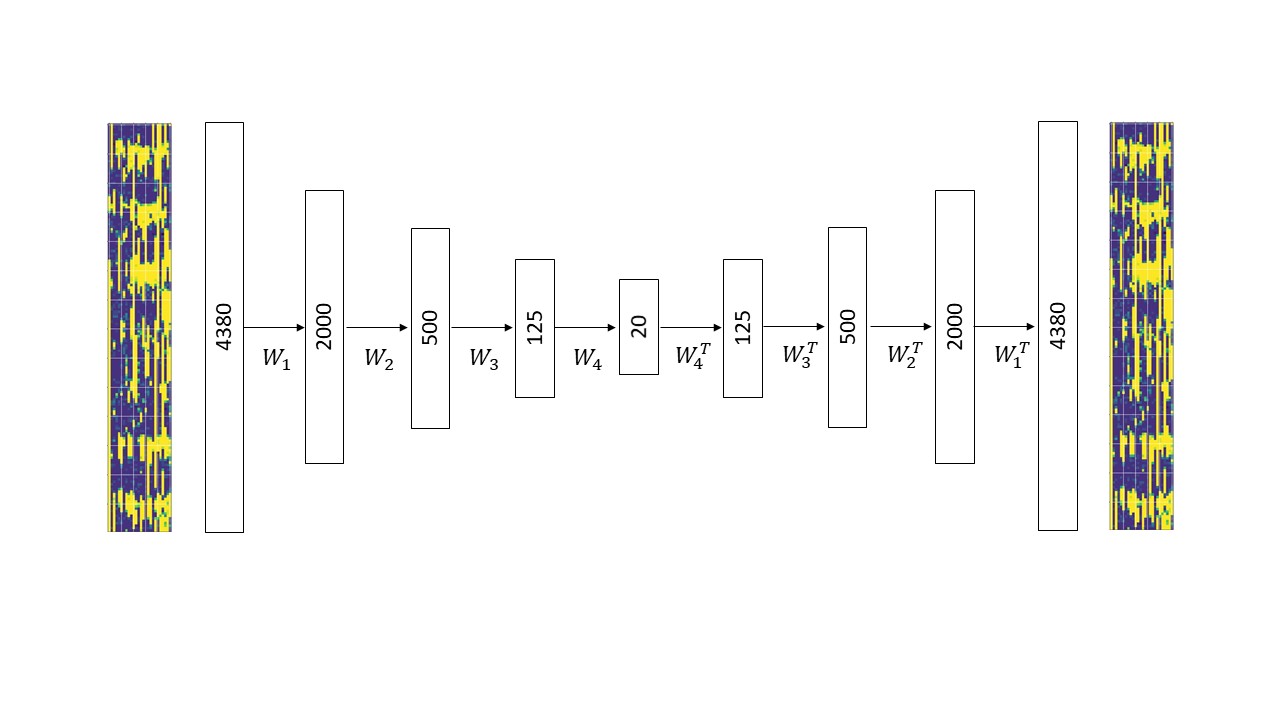
The research objective is to find similar patterns across daily interactions, modelled in a way that can be fed to an autoencoder network, in the encoded latent representation created by the network. As stated in the introduction, this thesis documents the attempts to test this idea, and therefore this goal is unknown to be feasible or not.

In section 5 it has been shown that this procedure is useful to cluster data from the MNIST dataset, being able to create distinct latent representations for each type of digit that could be analysed further. In this section, this same methodology will be tested using the modelled input data shown in sections 6.1 and 6.2.1.

The analysis is designed to take the following steps. First the autoencoder network needs to be trained to recreate the input data with high accuracy. From there, the latent representations can be analysed in a similar way shown in chapter 5 to observe if any groups of similar representations have been clustered together. With this, the information can be taken to the SensibleDTU dataset for further analysis of the dyads that form the clusters. As such, the analysis is separated into parts with the objectives mentioned, and will be presented in this section separately.

## 7.1 Input recreation

To test the repeatability of the method in the SensibleDTU dataset, an autoencoder (see section 4.2) was trained and tested. The architecture of the autoencoder model that was used for this process can be seen in Figure 7.1. This autoencoder follows a similar pattern to the ones explained in section 5 of the thesis. The first layer has 4380 nodes corresponding to each of the pixels in the input images, and subsequent layers reduce the dimensionality of the input data. The bottleneck layer is a 20-dimension vector, that will need to be further reduced in dimensionality for visualization.



**Figure 7.1:** Architecture for autoencoder network used in call and Bluetooth data.

The cost function that needs to be reduced in the training of an autoencoder is given by , being the input data point and the recreation generated after that data point goes through the autoencoder network. As such, the network needs to learn to reconstruct the input data with a high degree of accuracy.

As it has been shown in chapter 6, for call data, the images modelled after the data were too sparse to be fed to the autoencoder network. In fact, after testing the reconstruction for every image was the same, generating just values close to zero in each cell (Figure 7.2). Considering that the data is so sparse, it is not surprising that the network learns to model all nodes as zeroes, since this gives a low reconstruction error.

Imagen que contiene interior, pared

Descripción generada con confianza alta

Imagen que contiene interior, pared

Descripción generada con confianza alta

**Figure 7.2:** Examples of image reconstructions obtained from autoencoder using call data

A similar result can be seen with the call and Bluetooth data. The network, trained with the images generated in section 6.2.1 that aggregated all call and face-to-face interactions of dyads hourly, is not able to learn the reconstruction of each image. Instead, it generalizes by generating the same reconstruction for all of the training and test examples (Figure 7.3). It does however, generate the column-like shapes that characterize the weekly patterns in which people interact with each other. Additionally, the network seems to be able to differentiate between dyads with more or less intense interaction patterns. As shown in Figure 6.5, some dyads show a larger volume of interactions than others, that can be seen by the lighter colour of the images. This can still be seen in the reconstructions generated by the autoencoder.

Even though several tests were made, the results could not be improved in a significant way; the autoencoder was never able to reconstruct the input data. To understand why this was happening, an analysis of the images was done to see the characteristics of the images that made this impossible.





**Figure 7.3:** Examples of image reconstructions obtained from autoencoder using call and Bluetooth data from section 6.2.1

## 7.2 Data Analysis

With the aim of understanding why the network was not able to reconstruct and learn about the data, an analysis of the input images was carried out. There is not a specific answer to how a training dataset for an autoencoder must look like, since it changes for case to case. However, there some general guidelines for neural networks that can be taken as a basis to evaluate how good our data is. As it will be presented, there are two main factors that is believed they contribute to the inability of the network to learn the representations of the data: the quality of the data and the number of examples that is being given.

**Data quality**

The quality of the data can be expressed as the richness of the data, or in other words, how many daily interactions are present in each of the images that we are trying to analyse. For the autoencoder to learn patterns in the images, these need to be rich enough, but if the images are half empty there is not much to learn.

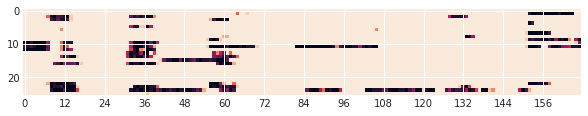
It has been mentioned that the call data was too sparse, and therefore it made sense that the network wasn’t able to learn anything from it. On the other hand, the combination of call and face-to-face data seemed to be fuller and richer. However, looking at the distribution of number of interactions across all the dyads gives another idea (Figure 7.4). The dyads with less than 50 interactions take up to 96.89% of all the dyads, with only 1.72% being above 100 interactions. In Figure 7.5 some examples are shown of the average input that has been fed to the autoencoder network, where it can be seen that the data is probably not good enough to learn anything from it. In most of the images, there is a column-like shape that is present for all of them. This is something to be expected since the images were created to reflect the weekly patterns of communication between dyads, and the network is able to understand these patterns in some way; but it can be concluded that there is not enough in the data to learn from.

Imagen que contiene captura de pantalla

Descripción generada con confianza muy alta

**Figure 7.4:** Histogram of number of interactions for call and Bluetooth data

Imagen que contiene captura de pantalla

Descripción generada con confianza alta

**Figure 7.5:** Examples of average images for Bluetooth and call data.

**Number of examples and similarity**

A second factor that needs to be taken into account is the number of training examples and their similarity.

Following the example from the MNIST dataset in section 5, the data that is fed to the autoencoder network needs to follow certain patterns for the network to be able to learn. Firstly, there needs to be a sufficient number of examples and classes from where the network can abstract the intricacies of the data. In the MNIST database we had 60,000 training examples for 10 different digits. This makes roughly 6,000 examples for each digit, assuming that the dataset is balanced. For the set of images used in this thesis it is a very different case.

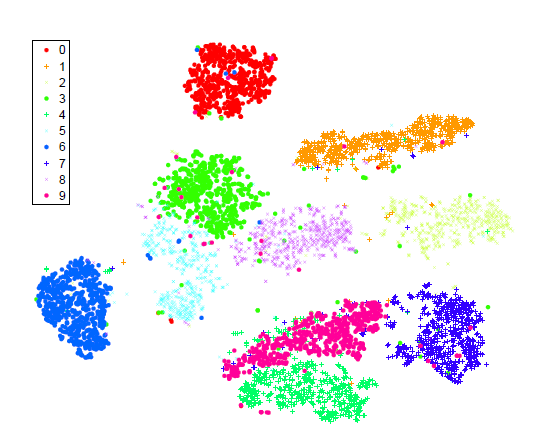
After modelling each of the dyads, 2027 examples are obtained, from which as mentioned above, only around 2% of them constitute what can be called as “good” data (i.e. data rich and full enough to contain any relevant information to what is being studied). Considering that this data still needs to be divided into train and test splits, the examples left to train the network are very reduced. Initially the data was split into 80% train and 20% test splits. This makes 1621 train samples to learn from. It can be argued that this is definitely not enough to train a network of this size.

**t-SNE analysis of the input**

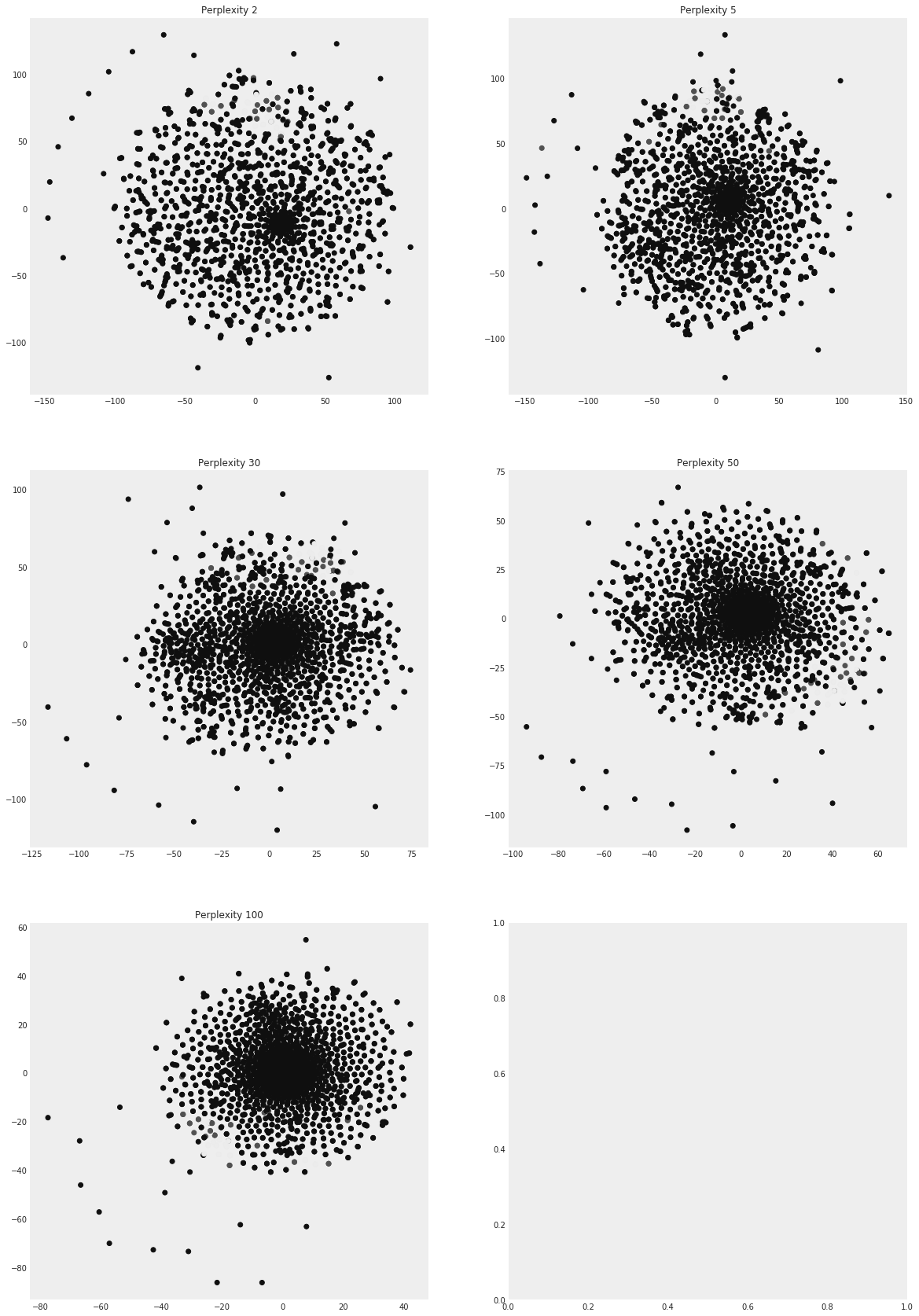
The t-SNE algorithm explained in previous sections has proven to be useful to reduce the dimensionality of the latent representations generated on the MNIST dataset and to visualize them in a 2-dimension space. However, in its conception this algorithm had been tested directly on the MNIST dataset, giving good results in clustering the data into clusters for each digit (Figure 7.6) [35]. As explained by the authors, the performance of t-SNE compared to other techniques such as Sammon mapping, Isomap and LLE on the MNIST dataset a vastly superior. The map produced by t-SNE reveals much more of the local structure of the data, and even though it contains some points that are clustered with the wrong class most of these points correspond to distorted digits which are difficult to identify.

The same methodology has been followed to finally estimate the similarity of the input data (Figure 7.7). As it can be seen, even applying the t-SNE algorithm directly on the input data does not show any significant clustering, meaning that all the images are too similar in their structure.

In conclusion, it is reasonable to argue that the amount of data and the ratio of possible classes versus the number of examples per class is very low.



**Figure 7.6:** Clustering of MNIST data by t-SNE. Image from [35]



**Figure 7.7:** Visualization of input data by t-SNE.

CHAPTER 8

# Conclusion

The purpose of this thesis was an attempt to use the technology behind autoencoders to find structure and similarities in the social interactions between the participants of the SensibleDTU dataset. Taking a look at the process that has been shown here, we started with a look at the previous attempts to understand social interactions of large populations, and the possibilities that it offered. From there, we have seen the potential of using autoencoders to reduce the dimensionality, visualize and cluster data in an example database that worked as a benchmark for our intentions. At last an explorative analysis has been done on the SensibleDTU dataset where these technologies have been tested. In this section the concluding remarks will be presented, along with some ideas for future work.

## 8.1 Outline

An overview of the literature in understanding social interactions has been presented, setting the background for the work of this thesis. Even though the literature is not extensive in the exact same topic that this thesis is about, relevant studies have been shown and explained that aimed to use similar datasets to understand how human interactions and interaction patterns work. These studies used different methodologies that served as a basis to establish several assumptions that were useful for the purpose of this thesis. Specifically, the research by Felbo et al. [3] helped to stablish the repetitive nature of human behaviour and how taking this nature as a basis can help model data in a meaningful way. Additionally, it proved that the models generated with this methodology can be used as input for deep learning techniques that can be used for classification.

The theory behind autoencoders was presented, along with the different types and uses. It was shown that autoencoders can go from extremely simple 3-layer neural networks to increasingly complex systems depending on the purpose of the network and the complexity of the data. Additionally, it was presented how making tweaks to the architecture of the network can transform it from a simple dimensionality reduction algorithm to a powerful data generation tool

Then, we worked on a test dataset, the MNIST database to test the ideas behind this thesis. The idea was to use this database, that has been widely used for deep learning research, as a benchmark for the possibilities of autoencoders as dimensionality reduction and data visualization tools. We saw that using autoencoders to reduce the dimensionality of the data to a 2-dimension space that was easy to visualize was not satisfactory enough. However, generating a higher dimensional space that cold later on be post processed proved to be a powerful methodology. With the use of t-SNE and UMAP algorithms it was possible to visualize the latent space generated by the autoencoder network in a 2-dimensional space in which all the input data was clustered in groups that corresponded to the digit they represented. This proved that, in the case of the SensibleDTU dataset, in case that the personal relationships between individuals where different enough, showing distinct patterns that were repeated across a number of groups, the autoencoder network could be able to generate latent representations that where distinct enough as well. Therefore, the repetition of the process done on the MNIST dataset should generate a number of clusters corresponding to different types of relationships or communication pattern types. Even though ideally, these clusters would correspond to different relationships types, this was something that would need further analysis later.

Finally, the SensibleDTU dataset was tested. Before using the data as inputs for the autoencoder network, it was necessary to model it and present it in a way that was suitable for this kind of network. By obtaining all the dyads that composed the dataset their interactions were collected and aggregated, first using only call data and then combining calls and face-to-face interactions. The final version of the interaction images consisted on collecting all the call and face-to-face interactions for the year 2014 and binning and aggregating them by the hour, counting how many times the dyad interacted during each hour of the year. Then this data was presented in a 52x168 matrix, generating an overview of the interactions and patterns across the 52 weeks of the year. The images showed interesting patterns that were repeated in consecutive weeks, with some dyads interacting more on week days, while others did it on weekends. Some dyads had stronger interactions during “work hours”, and others showed strong interactions all day long. These images were then cut in half, since some dyads had data issues after half of the year. The testing of this data on the autoencoder network however was not successful; the network was not able to learn how to reconstruct the images. The reason for this, as it was concluded, is that the data was not rich enough and numerous enough. With 2027 dyad examples, only a small fraction of them (less than 2%) were composed of more than 100 interactions in the 4368 hours of half a year. This showed that there were not enough examples to train a deep learning network.

Finally, as an attempt to see if the data was at least distinct enough to create different groups, the t-SNE algorithm that was proven successful previously, was directly used in the data. This showed that there were not any visible clusters of similar images.

## 8.2 Discussion and future work

There were two main reasons for the inability to obtain satisfactory results from the autoencoder network on the SensibleDTU dataset. First, there was not enough data to train a deep learning model. Second, the data was not rich and distinct enough to generate any kind of different groups.

There are some steps that could be taken in order to solve this. First, as it was shown in section 2, the way data is modelled affects greatly the results that can be obtained. The idea behind modelling the interactions as it was done was to aggregate as little data as possible to keep the maximum amount of detail possible. However, the data showed not to be rich enough to model with this methodology.

A way of modelling the data that can be useful is the “punch card” modelling that was used by Felbo et al. [3] in their research. This method aggregated the interactions of users by obtaining a mean value of the interactions between the individuals across several weeks, generating and interaction pattern on itself directly. This leaves less space to the autoencoder to find out those patterns on its own, but might help with the data sparsity problem. Additionally, even though the number of example dyads would still be quite low, if the examples are distinct enough across a number of groups, it might be possible that this is enough to train a deep learning network like this.

Another aspect that hasn’t been tested with depth in this thesis is the use of Convolutional Neural Networks, instead of fully connected networks. CNNs have been widely used in image recognition tasks and have given state of the art performance in image classification. With further work it is possible they could be useful in generating the latent representations.

In conclusion, it cannot be said that the methodology used in this thesis is not suitable for the SensibleDTU dataset, but further exploration is necessary. As it has been proven, the methodology shows promising results in similar cases, which indicates that in the right conditions the results should be repeatable. With a correct modelling of the data images, the low data example issues could be solved to obtain the desired results.

# References

|  |  |
| --- | --- |
| [1] | Giles, J. (2012). Making the links. *Nature*, *488*(7412), 448-450. |
|  |  |
| [2] | Lazer, D., Pentland, A. S., Adamic, L., Aral, S., Barabasi, A. L., Brewer, D., ... & Jebara, T. (2009). Life in the network: the coming age of computational social science. *Science (New York, NY)*, *323*(5915), 721. |
|  |  |
| [3] | Felbo, B., Sundsøy, P., Lehmann, S., & de Montjoye, Y. A. (2017, September). Modeling the Temporal Nature of Human Behavior for Demographics Prediction. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 140-152). Springer, Cham. |
| [4] | Linards Kalnis, (2016). Activity Patterns for Links in Multiplex Social Networks (Master's thesis). Technical University of Denmark, Kongens Lyngby, Denmark. |
|  |  |
| [5] | Mollgaard, A., Lehmann, S., & Mathiesen, J. (2017). Correlations between human mobility and social interaction reveal general activity patterns. *PloS one*, *12*(12), e0188973. |
|  |  |
| [6] | Eagle, N., Pentland, A. S., & Lazer, D. (2009). Inferring friendship network structure by using mobile phone data. *Proceedings of the national academy of sciences*, *106*(36), 15274-15278. |
|  |  |
| [7] | Stehlé, J., Voirin, N., Barrat, A., Cattuto, C., Isella, L., Pinton, J. F., ... & Vanhems, P. (2011). High-resolution measurements of face-to-face contact patterns in a primary school. *PloS one*, *6*(8), e23176. |
|  |  |
| [8] | Min, B., Goh, K. I., & Vazquez, A. (2011). Spreading dynamics following bursty human activity patterns. *Physical Review E*, *83*(3), 036102. |
|  |  |
| [9] | Fairchild, G., Hickmann, K. S., Mniszewski, S. M., Del Valle, S. Y., & Hyman, J. M. (2014). Optimizing human activity patterns using global sensitivity analysis. *Computational and mathematical organization theory*, *20*(4), 394-416. |
|  |  |
| [10] | Yassine, A., Singh, S., & Alamri, A. (2017). Mining human activity patterns from smart home big data for health care applications. *IEEE Access*, *5*, 13131-13141. |
|  |  |
| [11] | Huang, W., & Li, S. (2016). Understanding human activity patterns based on space-time-semantics. *ISPRS Journal of Photogrammetry and Remote Sensing*, *121*, 1-10. |
|  |  |
| [12] | Phithakkitnukoon, S., Horanont, T., Di Lorenzo, G., Shibasaki, R., & Ratti, C. (2010, August). Activity-aware map: Identifying human daily activity pattern using mobile phone data. In *International Workshop on Human Behavior Understanding* (pp. 14-25). Springer, Berlin, Heidelberg. |
|  |  |
| [13] | Bengtsson, L., Lu, X., Thorson, A., Garfield, R., & Von Schreeb, J. (2011). Improved response to disasters and outbreaks by tracking population movements with mobile phone network data: a post-earthquake geospatial study in Haiti. *PLoS medicine*, *8*(8), e1001083. |
|  |  |
| [14] | Wesolowski, A., Qureshi, T., Boni, M. F., Sundsøy, P. R., Johansson, M. A., Rasheed, S. B., ... & Buckee, C. O. (2015). Impact of human mobility on the emergence of dengue epidemics in Pakistan. *Proceedings of the National Academy of Sciences*, *112*(38), 11887-11892. |
|  |  |
| [15] | Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1798-1828. |
|  |  |
| [16] | Onnela, J. P., Saramäki, J., Hyvönen, J., Szabó, G., Lazer, D., Kaski, K., ... & Barabási, A. L. (2007). Structure and tie strengths in mobile communication networks. *Proceedings of the national academy of sciences*, *104*(18), 7332-7336. |
|  |  |
| [17] | Abelson, R. P. (1981). Psychological status of the script concept. *American psychologist*, *36*(7), 715. |
|  |  |
| [18] | Sekara, V., Stopczynski, A., & Lehmann, S. (2016). Fundamental structures of dynamic social networks. *Proceedings of the national academy of sciences*, *113*(36), 9977-9982. |
|  |  |
| [19] | Deng, L., Seltzer, M. L., Yu, D., Acero, A., Mohamed, A. R., & Hinton, G. (2010). Binary coding of speech spectrograms using a deep auto-encoder. In *Eleventh Annual Conference of the International Speech Communication Association*. |
|  |  |
| [20] | Betechuoh, B. L., Marwala, T., & Tettey, T. (2006). Autoencoder networks for HIV classification. *Current Science*, 1467-1473. |
|  |  |
| [21] | Li, J., Luong, M. T., & Jurafsky, D. (2015). A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*. |
|  |  |
| [22] | AP, S. C., Lauly, S., Larochelle, H., Khapra, M., Ravindran, B., Raykar, V. C., & Saha, A. (2014). An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems* (pp. 1853-1861). |
|  |  |
| [23] | Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008, July). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning* (pp. 1096-1103). ACM. |
|  |  |
| [24] | Deng, J., Zhang, Z., Eyben, F., & Schuller, B. (2014). Autoencoder-based unsupervised domain adaptation for speech emotion recognition. *IEEE Signal Processing Letters*, *21*(9), 1068-1072. |
|  |  |
| [25] | Xie, J., Xu, L., & Chen, E. (2012). Image denoising and inpainting with deep neural networks. In *Advances in neural information processing systems* (pp. 341-349). |
|  |  |
| [26] | Cho, K. (2013). Boltzmann machines and denoising autoencoders for image denoising. *arXiv preprint arXiv:1301.3468*. |
|  |  |
| [27] | Makhzani, A., & Frey, B. (2013). K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*. |
|  |  |
| [28] | Deng, J., Zhang, Z., Marchi, E., & Schuller, B. (2013, September). Sparse autoencoder-based feature transfer learning for speech emotion recognition. In *Affective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on* (pp. 511-516). IEEE. |
|  |  |
| [29] | Xu, J., Xiang, L., Liu, Q., Gilmore, H., Wu, J., Tang, J., & Madabhushi, A. (2016). Stacked sparse autoencoder (SSAE) for nuclei detection on breast cancer histopathology images. *IEEE transactions on medical imaging*, *35*(1), 119-130. |
|  |  |
| [30] | Sankaran, A., Pandey, P., Vatsa, M., & Singh, R. (2014, September). On latent fingerprint minutiae extraction using stacked denoising sparse autoencoders. In *Biometrics (IJCB), 2014 IEEE International Joint Conference on* (pp. 1-7). IEEE. |
|  |  |
| [31] | Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*. |
|  |  |
| [32] | Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672-2680). |
|  |  |
| [33] | Varano, C. Disentangling Variational Autoencoders for Image Classification. |
|  |  |
| [34] | Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, *24*(6), 417. |
|  |  |
| [35] | Maaten, L. V. D., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, *9*(Nov), 2579-2605. |
|  |  |
| [36] | Roweis, S. T., Saul, L. K., & Hinton, G. E. (2002). Global coordination of local linear models. In *Advances in neural information processing systems* (pp. 889-896). |
|  |  |
| [37] | McInnes, L., & Healy, J. (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv preprint arXiv:1802.03426*. |
| [38] | Min, J. K., Wiese, J., Hong, J. I., & Zimmerman, J. (2013, February). Mining smartphone data to classify life-facets of social relationships. In *Proceedings of the 2013 conference on Computer supported cooperative work* (pp. 285-294). ACM. |
|  |  |
| [39] | Eagle, N., & Pentland, A. S. (2006). Reality mining: sensing complex social systems. *Personal and ubiquitous computing*, *10*(4), 255-268. |
| [40] | Altshuler, Y., Fire, M., Aharony, N., Elovici, Y., & Pentland, A. S. (2012, April). How many makes a crowd? On the evolution of learning as a factor of community coverage. In *International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction* (pp. 43-52). Springer, Berlin, Heidelberg. |
|  |  |
| [41] | Min, J. K., & Cho, S. B. (2011). Mobile human network management and recommendation by probabilistic social mining. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *41*(3), 761-771. |
| [42] | Chittaranjan, G., Blom, J., & Gatica-Perez, D. (2013). Mining large-scale smartphone data for personality studies. *Personal and Ubiquitous Computing*, *17*(3), 433-450. |
| [43] | Cranshaw, J., Toch, E., Hong, J., Kittur, A., & Sadeh, N. (2010, September). Bridging the gap between physical location and online social networks. In *Proceedings of the 12th ACM international conference on Ubiquitous computing* (pp. 119-128). ACM. |
| [44] | Farnham, S. D., & Churchill, E. F. (2011, March). Faceted identity, faceted lives: social and technical issues with being yourself online. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work* (pp. 359-368). ACM. |
|  |  |
| [45] | Dong, Y., Yang, Y., Tang, J., Yang, Y., & Chawla, N. V. (2014, August). Inferring user demographics and social strategies in mobile social networks. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 15-24). ACM. |
|  |  |
| [46] | Frias-Martinez, V., Frias-Martinez, E., & Oliver, N. (2010, March). A Gender-Centric Analysis of Calling Behavior in a Developing Economy Using Call Detail Records. In *AAAI spring symposium: artificial intelligence for development*. |
|  |  |
| [47] | Sekara, V., & Lehmann, S. (2014). The strength of friendship ties in proximity sensor data. *PloS one*, *9*(7), e100915. |
|  |  |
| [48] | Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, *313*(5786), 504-507. |
|  |  |
| [49] | Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P. A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, *11*(Dec), 3371-3408. |
|  |  |
| [50] | Kingma, D. P. (2017). Variational inference & deep learning: A new synthesis. |
|  |  |
| [51] | Cuttone, A., Lehmann, S., & Larsen, J. E. (2014, September). Inferring human mobility from sparse low accuracy mobile sensing data. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication* (pp. 995-1004). ACM. |
| [52] | Ulf Aslak, (2016). Personality Archetypes Support Evolutionarily Important Behavioural Strategies (Master's thesis). Weizmann Institute of Science, Technical University of Denmark, Kongens Lyngby, Denmark. |
|  |  |
| [53] | Hinde, R. A. (1974). *Biological bases of human social behaviour*. McGraw-Hill. |
|  |  |
| [54] | Quiroga, L. M., Crosby, M. E., & Iding, M. K. (2004, January). Reducing cognitive load. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on* (pp. 9-pp). IEEE. |
|  |  |
| [55] | Becker, H., & Useem, R. H. (1942). Sociological analysis of the dyad. *American Sociological Review*, *7*(1), 13-26. |
|  |  |
| [56] | Seligman, E. R., & Johnson, A. E. (2014). Encyclopaedia Of The Social Sciences Vol. 1-2. |
|  |  |
| [57] | Chen, M. K. (1986). Retrospective studies with survey data: Problems in social science and epidemiology research. *Socio-Economic Planning Sciences*, *20*(3), 131-133. |
|  |  |
| [58] | Pentland, A. (2009). Reality mining of mobile communications: Toward a new deal on data. *The Global Information Technology Report 2008–2009*, *1981*. |
|  |  |
| [59] | De Waal, F. B. (Ed.). (2009). *Tree of origin: What primate behavior can tell us about human social evolution*. Harvard University Press. |

# Appendix

## Appendix 1 – Code

The code used in this thesis can be found in the following repository: