The School of Mathematics

THE UNIVERSITY
of EDINBURGH

Neumann Series for Autoencoders with Missing Data

by

Evangelos Antypas

Dissertation Presented for the Degree of
MSc in Statistics with Data Science

August 2023

Supervised by
Professor C.K.I. Williams

Abstract

It is often the case, that we are given a data vector $x = (x_v, x_m)$, where x_v denotes data that is observed/visible, and x_m denotes data that is missing. We then wish to perform imputation based on x_v. A natural approach for this problem is to model x with a probabilistic latent variable model, such as a Variational Autoencoder (VAE), and then condition on x_v. The VAE uses an encoder network that predicts the latent variables given the data. However, it is not clear how to handle the missing variables in this encoder network. Williams et al. [WNN18] have shown that even for the linear VAE (i.e. Factor Analysis) the exact solution is quite complex, since it involves a matrix inversion that depends on the specific missingness pattern. In this project, our principal aim is to explore the use of an unrolled Neumann series to tackle this matrix inversion. We present a novel algorithm that solves this problem, which we term NSA (Neumann Series Algorithm). We test the imputation performance of NSA on an image in-painting task and compare it to the exact solution and various approximations found in [WNN18]. Our results suggest that our method is an improvement in terms of imputation quality. Additionally, we show that our algorithm can be extended for a mixture of Factor Analysers in the presence of missing data, we term this extension NSA-M (Neumann Series Algorithm-Mixture), and we test its performance, again, on an image in-painting task. Our experiments, show promising results.

Acknowledgments

First and foremost, I feel inclined to thank my supervisor Professor C.K.I. Williams for the conversations we had throughout this project, and for the guidance he offered. His comments were always insightful and led me to do a better work. I would also like to thank my friends and colleagues Christos Giannikos and Yannis Stylianou for reading early drafts of this work and giving me their suggestions. Lastly, I would like to thank my family for their patience and support, without them none of this would have been possible.

Own Work Declaration

I declare that the following work is my own except where otherwise noted.

Contents

List of Tables

List of Figures

1 Introduction

In this chapter, we introduce our research problem. We focus on clarifying the main objective of this project and we describe how we have expanded upon it. Subsequently, we explain the motivation behind the research direction we have taken and we indicate what our novel contributions are, as well as how they build on the existing scientific background. Finally, we conclude this discussion with an outline of the structure of the rest of the project, giving an overview of the following chapters.

1.1 Motivation

Missing data are ubiquitous in real world datasets [LR19, VB18]. In order to work effectively with such datasets, we must first deal with the problem of missing values. A naive approach is to throw away all the data that contain missing values and work only with fully observed cases. This approach is termed list-wise or case-wise deletion. However, this approach might severely limit the amount of training data we have at our disposal and, more importantly, it may bias the results of any subsequent analysis. Additionally, it doesn't deal with cases where filling in the missing values is a cause in itself. For example, let us consider an instance of image in-painting, where we have an image and a part of it is missing. In this case, our goal is to create a complete image, in this context case-wise deletion goes against the goal of the problem. A more principled way to deal with this problem is to try filling in the missing values based on the data we have observed, i.e. *missing data imputation*. Over the last years, numerous statistical and computational methods have been proposed in order to impute datasets with missing values, either by single imputation, where missing data values are replaced by a single set of imputations, or multiple imputation, where several sets of imputed values are created, and therefore allow for better quantification of the uncertainty of the imputed values (for a comprehensive study see e.g. [LR19, Sch97, vBGOR+15, VB18]). We look at the problem of missing data imputation from a generative modelling point of view.

A generative model can model the probability distribution of the input data [Bis06]. Therefore, generative modelling offers a very natural way to approach the problem of missing data imputation. More specifically, let us assume that we are given a data vector $x = (x_v, x_m)$, where x_v denotes the visible part of the data vector, while x_m denotes the part that is missing. In this case, imputing the missing values essentially amounts to filling in x_m based on x_v. To achieve this, we can build a generative latent variable model of x, and then condition on x_v to impute x_m. The Variational Autoencoder (VAE) [KW13, RMW14] is a very popular generative latent variable model that has been used in this context. However, inference in such a model can be quite complex. The VAE uses a *recognition* or *encoder* network to predict a latent variable z given the data and then feeds that latent variable to a *generative* or *decoder* network to reconstruct the input. In this case, however, it is not clear how to obtain a valid latent representation in the presence of missing data. This gives rise to the research question of how can the missingness of the data x_m be handled effectively in the encoder network. Williams et al. [WNN18] have shown that in the presence of missing-at-random data (MAR) (i.e. the probability of missingness is independent of the underlying missing values, for a detailed analysis see [Rub76]), even for the simple linear VAE case, i.e. Factor Analysis (FA), see e.g. [BKM11], the exact solution can be quite complex, involving a matrix inversion that depends on the specific missingness pattern, this leads to prohibitive computational complexity for high dimensional data in the worst case.

In recent work, Le Morvan et al. [LMJM+20] have shown that it is possible to use a Neumann series, see e.g. [MS23], to tackle a similar matrix inversion in the context of obtaining the Bayes predictor for linear regression in the presence of missing data. Additionally, Gilton et al. [GOW19] have shown that Neumann series can be used to learn a regularised solution for ill-posed inverse linear problems, e.g. image in-painting. Motivated by this work, in this project, following mainly [WNN18], we assume that we have a pre-trained generative model on complete data such as the decoder in FA/VAE, and we develop a novel algorithm that uses Neumann series to tackle the matrix inversion required for Factor Analysis in the presence of missing data [WNN18]. Our method demonstrates considerable computational gains. In this project, we are mainly interested in obtaining an effective encoding when there is missing data. The issue of training in the presence of missing data [GJ94] is different problem and is not dealt with here, for Principal Component Analysis (PCA) an excellent summary can be found in [IR10].

Additionally, we show that the exact solution of Williams et al. [WNN18], can be extended for a mixture of Factor Analysers, see [HDR97], in the presence of missing data. Subsequently, we show that our algorithm can be extended to fit this case as well. This can be viewed as a piece-wise linear extension to Factor Analysis with missing data. The effectiveness of our methods is tested on an image in-painting task with various patterns of MAR missingness. We compare our methods with [WNN18] and find that our approach produces superior imputation and computational performance.

1.2 Outline of Thesis

The remainder of this project is organised as follows. Chapter 2 is dedicated to presenting background material. We present the required theory regarding Variational Autoencoders and Factor Analysis and we clarify the connection between them. More specifically we explain how FA can regarded as a linear Gaussian VAE. Subsequently, we present a thorough review of the exact solution and the approximations in [WNN18]. These are a major part of this project, since they serve as building stone for our solution. Finally, we introduce the required theory for Neumann series and discuss their applications in [GOW19, LMJM+20] in more detail. Chapter 3 includes our novel contributions, and it is organised in two parts. In the first part we present the Neumann Series Algorithm (NSA) and prove its convergence. Additionally, we present our experiments for NSA and discuss the results. The second part is a logical continuation of the first. In this part, we introduce the Neumann Series Algorithm for Mixtures of Factor Analysers (NSA-M) and we present experiments and results. Chapter 4 concludes this project. In this chapter, we review our contributions and we give some possible avenues for future research based on unsolved problems and extensions to our project.

2 Background

In this chapter we present the required theoretical background to this project. The work is organised in four different sections. We devote the first section to explaining the project title *Neumann Series for Autoencoders with Missing data*. The project is titled after Autoencoders but for the most part we are working with Factor Analysis and its special case Probabilistic PCA [TB99b]. In this section we focus on clarifying the connection. In the second section, we discuss the bibliography on VAEs with missing data, and more specifically the attempts that have been made to adapt the VAE architecture to work in the presence of missing data. This discussion gives rise to our motivation to focus on the linear case (i.e. FA). Building on that, we discuss the exact solution and the approximations in [WNN18]. Subsequently, in the third section we introduce Neumann series and discuss their appearances in the bibliography so far, and more importantly we make the connection to our problem. Finally, in the fourth section we introduce the mixture of Factor Analysers model and show how the exact solution of Williams et al. [WNN18] can be extended for this model in the presence of missing data.

2.1 VAEs and Factor Analysis

The VAE model [KW13, RMW14] is a very popular unsupervised learning algorithm, that combines ideas both from autoencoders and probabilistic latent variable models. Let x denote a data vector and z a latent variable associated with x. The latent z is often assumed to be lower dimensional than x, offering a more parsimonious description of the data. In this case, the VAE architecture consists of two parts, a probabilistic encoder or recognition network, which we denote with $q(z|x)$ and a probabilistic decoder or generative network, which we denote with $p(x|z)$. The main assumption underlying the VAE is that the data are generated by a random process involving the lower dimensional continuous latent variable z. In that context, the recognition network is learning the parameters of an approximation to the true posterior $p(z|x)$, which might be intractable. The approximation distribution $q(z|x)$ is chosen from a known distribution family so that its parameters can be optimised to best fit the data. The generative network $p(x|z)$ reconstructs the original input by mapping the latent representation back to the data space. The generative process described by the VAE is really a two-step process. First, we generate a value z from some prior distribution $p(z)$ and then we generate a value x from some conditional distribution $p(x|z)$. From the perspective of coding theory , the latent variable z can be interpreted as a latent representation or code, see e.g. [Mac03]. Both the encoder and the decoder are implemented with neural networks and so, they are nonlinear functions of the input, for a detailed introduction to VAEs and their connections to latent variable probabilistic models see [KW+19]. A representation of the VAE architecture is given in Figure (1).

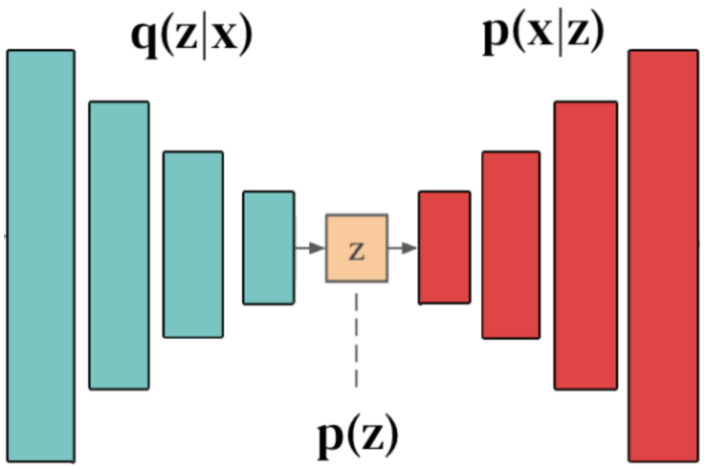


Figure 1: Representation of the VAE Architecture.

For VAEs the standard assumption is that the dataset being modelled consists of N fully observed data vectors $\{x^{(1)}, x^{(2)}, ..., x^{(N)}\}$. However, as we have already mentioned that is rarely the case in practical applications. This means that each data vector $x^{(j)}$ might consist of a visible part $x_v^{(j)}$ and a missing part $x_m^{(j)}$, which is not observed. Then, we are faced with the problem of how to make effective use of the encoder network in the presence of missing values. One possible fix is to replace the missing value with a constant such as zero, see e.g. [PKD+16]. Similarly in [NOGV20], the authors propose HI-VAE an input-dropout method where the missing part of the data is imputed with zeros. This creates the problem that the network cannot distinguish between imputed zeros and real zeros. Another technique is *mean imputation*, i.e. filling in the missing value with the unconditional mean of the corresponding variable. However, these are not very principled solutions and lead to systematic underestimation of the uncertainty of the missing values. The authors in [RMW14] have shown that one can construct a Markov chain Monte Carlo (MCMC) method to sample from the posterior over the missing data for a VAE, but this is an iterative technique that has the disadvantage of being computationally burdensome and taking a long time to achieve convergence. A more sophisticated approach is the indicator variable method proposed in [CNW20]. In this approach the encoder network is given direct access to the missingness pattern, thus it can distinguish between an observed data value and a missing value. However, the nonlinear nature of the encoder and decoder networks of the VAE makes an exact analysis of missing data inference techniques rather challenging, so instead of trying to tackle the problem directly, we focus our analysis on the simpler linear case of Factor Analysis as a starting point.

The connection with Factor Analysis becomes apparent when we make the following observation. If all the distributions of interest, i.e. $q(z|x), p(z), p(x|z)$ are chosen to be Gaussians, then we can obtain the FA model when the encoder and the decoder are just linear functions of the input and there is a single hidden layer. In that sense the FA model is a linear Gaussian VAE (we are calling the VAE Gaussian after the distribution $p(x|z)$ which is a Gaussian). Of course in the simple, linear case of FA, the distribution $q(z|x)$ really is the true posterior $p(z|x)$, and so we are in the tractable case, where the posterior can be computed analytically. It is useful to present the FA model in more detail since there is a strong connection between the fully observed case and the missing data case.

2.1.1 Factor Analysis

The standard FA model, see e.g. [BKM11], is a classical statistical model that is essentially a probabilistic generalisation of PCA. Such a model can be used to form simple low-dimensional generative models of data, see e.g. [Bar12]. Assume we have a dataset $\mathcal{D} = \{x^{(1)}, x^{(2)}, ..., x^{(N)}\}$, where each data vector x is D-dimensional. We are interested in obtaining a lower-dimensional probabilistic representation of the data. If the data is close to a lower-dimensional linear subspace of dimension K with $K < D$, then we may be able to approximately capture each data point by a K-dimensional representation plus some Gaussian noise to model the fact that the data will not lie exactly on the subspace. This is the intuition behind FA.

Mathematically, we can formulate the model by introducing an explicit latent variable z corresponding to the latent space representation. Next, we define a standard Gaussian prior over the latent variable

$$p(z) = N(z|0, I_K), \tag{2.1}$$

where the subscript K in I_K signifies that we are working with the identity matrix of $K \times K$ dimensions. Similarly, the conditional distribution of the data variables x given the latents is also a Gaussian of the form

$$p(x|z) = N(x|Wz + \mu, \Psi), \tag{2.2}$$

where W is the $D \times K$ factor loadings matrix, μ is the mean (offset) vector in the data space, and Ψ is a D-dimensional diagonal covariance matrix. The columns of W form a basis of the K-dimensional subspace. In such a model the posterior $p(z|x)$ will also be a Gaussian $N(z|\mu_{z|x}, \Sigma_{z|x})$, where

$$\Sigma_{z|x} = (I_K + W^T \Psi^{-1} W)^{-1}, \tag{2.3}$$

$$\mu_{z|x} = \Sigma_{z|x} W^T \Psi^{-1} (x - \mu), \tag{2.4}$$

the analytical derivation can be found in [Bis06]. In the autoencoder framework the posterior $p(z|x)$ can be regarded as the encoder, while the conditional distribution $p(x|z)$ can be regarded as the decoder. From a generative view point the FA model generates an observation by first sampling a value for z and then sampling a value for x conditional to z, i.e. the D-dimensional x is defined by linearly transforming the K-dimensional z plus some Gaussian noise, so that

$$x = Wz + \mu + \epsilon, \quad \epsilon \sim N(0, \Psi), \tag{2.5}$$

notice the correspondence with generative process of the VAE. The formulation in Equation (2.5) corresponds to a linear Gaussian model [Bis06], and so the marginal distribution $p(x)$ is also a Gaussian of the form

$$p(x) = N(x|\mu_x, \Sigma_x), \tag{2.6}$$

where

$$\mu_x = \mathbb{E}[x] = \mathbb{E}[Wz + \mu + \epsilon] = \mu, \tag{2.7}$$

$$\Sigma_x = cov[x] = \mathbb{E}[(Wz + \epsilon)(Wz + \epsilon)^T] = WW^T + \Psi, \tag{2.8}$$

where we used the fact that z and ϵ are independent and therefore uncorrelated. The form of this distribution implies that the model is subject to rotational ambiguity since for any $K \times K$ rotation matrix R we have

$$WRR^T W^T = WW^T, \tag{2.9}$$

so the covariance remains the same under rotations. This means that the solution space for W is not unique.

Now that we have presented the FA model, it is also useful to point out that Probabilistic PCA (PPCA) [TB99b] can be obtained through FA by setting $\Psi = \sigma^2 I$. The parameters of the model are W, Ψ, μ which can be fit with maximum likelihood. For FA the likelihood solutions cannot be obtained in closed form and thus an Expectation-Maximisation (EM) [DLR77] approach is required, for the sub-case of PPCA it is possible to obtain the maximum likelihood solutions in closed form [TB99b]. We take advantage of this fact and use PPCA in our experiments.

To conclude this presentation of FA we describe exactly how the encoding-decoding procedure works. Given an input x the encoding is achieved via the mapping

$$enc(x) = \Sigma_{z|x} W^T \Psi^{-1}(x - \mu) = \mu_{z|x}, \tag{2.10}$$

so the latent representation of an input is given by the posterior mean. The decoding is essentially a mapping back to the data space and is given by

$$\hat{x} = dec(enc(x)) = W enc(x) + \mu = W \mu_{z|x} + \mu. \tag{2.11}$$

This means that for FA, obtaining an effective encoding amounts to finding the posterior mean $\mu_{z|x}$, which can be done through solving a linear system, as per Equation (2.4). For more details on the connection between FA and autoencoders see [Row97, HZ93].

2.2 Factor Analysis with Missing Data

This section mainly follows [WNN18]. We review the exact solution and the related approximations, so that we can use them later on. We have presented the standard FA model, now let us consider the case where there is MAR missing data. In that case each data vector x can be split into a visible part x_v and a missing part x_m. An effective way of describing the missingness is to consider an extra random variable m, which serves as a missingness indicator function, i.e. $m_j = 1$, if x_j has been observed, and $m_j = 0$, if x_j is missing, where x_j is the j^{th} coordinate of x and $j = 1, ..., D$. By doing this, we can perform inference on the posterior of the latents, while also considering the missingness, i.e. given a pair (x_v, m) we want to obtain $p(z|x_v, m)$. In the context of autoencoders being able to determine the distribution $p(z|x_v, m)$ amounts to obtaining an effective encoding in the presence of missing data. The posterior $p(z|x_v, m)$ can be obtained by marginalising out the missing variables in Equations (2.3), (2.4). In this manner, we can obtain $p(z|x_v, m) = N(z|\mu_{z|x_v}, \Sigma_{z|x_v})$ with

$$\Sigma_{z|x_v} = (I_K + W_v^T \Psi_v^{-1} W_v)^{-1}, \tag{2.12}$$

$$\mu_{z|x_v} = \Sigma_{z|x_v} W_v^T \Psi_v^{-1}(x_v - \mu_v), \tag{2.13}$$

where W_v denotes the submatrix of W that corresponds to the visible variables and similarly for Ψ_v. Equations (2.12),(2.13) are essentially the standard FA model but only for the visible part of the data. To achieve a better understanding of what this actually means, let us consider a simple artificial example. Let $D = 3$ and $K = 2$, in that case x is a 3-dimensional vector, z is a 2-dimensional vector, W is a 3×2 matrix and finally Ψ matrix is a 3×3 matrix . Additionally, let us denote the dimension of the visible part of x with D_v. If x contains missing values, e.g.

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \star \end{bmatrix},$$

where we denote the missing values with $\star$, then $D_v = 2$ and the *visible* factor loadings matrix W_v will be a $D_v \times K$ matrix, which in this case is 2×2,

$$W_v = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix},$$

notice that in general W_v will have variable dimensionality depending on the missing values. In a similar manner Ψ_v is a $D_v \times D_v$ matrix, which again is 2×2

$$\Psi_v = \begin{bmatrix} \psi_{11} & 0 \\ 0 & \psi_{22} \end{bmatrix}.$$

In this case we can write,

$$\Sigma_{z|x_v}^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix} \begin{bmatrix} \psi_{11}^{-1} & 0 \\ 0 & \psi_{22}^{-1} \end{bmatrix} \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}. \tag{2.14}$$

An interesting observation now is that if we consider a mask matrix M to be a diagonal matrix where the main diagonal is equal to m, i.e. the missingness indicator,

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Then Equation (2.14) can be written equivalently

$$\Sigma_{z|x_v}^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{21} & w_{22} & w_{32} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \psi_{11}^{-1} & 0 & \\ 0 & \psi_{22}^{-1} & \\ 0 & 0 & \psi_{33}^{-1} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix},$$

and this deals with the varying dimensionality problem. In general, taking into account the fact that M is idempotent i.e. $M^2 = M$, Equations (2.12),(2.13) can be written equivalently as

$$\Sigma_{z|x_v} = (I_K + W^T M \Psi^{-1} M W)^{-1}, \tag{2.15}$$

$$\mu_{z|x_v} = \Sigma_{z|x_v} W^T M \Psi^{-1} M (x - \mu). \tag{2.16}$$

For more details, see [WNN18]. For the remainder of this project we shall refer to $\mu_{z|x_v}$ given by Equation (2.16) as the exact solution. As we have seen in the example, the diagonal elements in $M\Psi^{-1}M$ will be $m_j \psi_{jj}^{-1} m_j = m_j \psi_{jj}^{-1}$, which implies that when $m_j = 1$, i.e. x_j is observed the variance will be ψ_{jj}. In contrast, for missing data dimensions $m_j = 0$ implies an infinite variance for ψ_{jj}, meaning that in essence any data value for the missing coordinate will be ignored.

Now, let $\nu_j^T = (w_{j1}, ..., w_{jK})$ be the j^{th} row of W. Then, we can re-write Equation (2.12) as,

$$\Sigma_{z|x_v}^{-1} = I_K + W^T M \Psi^{-1} M W = I_K + \sum_{j=1}^{D} m_j \psi_{jj}^{-1} \nu_j \nu_j^T, \tag{2.17}$$

an interesting observation here is to look at the dimensions of the quantity being summed over in Equation (2.17), m is a $D-$dimensional vector, ψ_{jj}^{-1} is a scalar and the product $\nu_j \nu_j^T$ represents a $K \times K$ matrix of rank-1, for each $j = 1, ..., D$. This means that this quantity can be thought of as a 3 dimensional tensor, essentially stacking $K \times K$ matrices D times, where we can sum over the missingness pattern to obtain the effective weights. Similarly, for $\mu_{z|x_v}$ we can write,

$$\mu_{z|x_v} = \Sigma_{z|x_v} \sum_{j=1}^{D} m_j \psi_{jj}^{-1} (x_j - \mu_j) \nu_j. \tag{2.18}$$

If the data vector x does not contain missing values, we have $x = x_v$, and so it is rather straightforward to obtain an encoding, using Equation (2.4). If x contains missing values, however, exact inference cannot be carried out efficiently. The reason for that is that the matrix $\Sigma_{z|x_v}$ depends on the missingness pattern m, which means that there is a different $\Sigma_{z|x_v}$ for each of the 2^D non-trivial patterns of missingness, so as per Equation (2.16) we have to solve a different linear system per missingness pattern.

Note 1: To see why the number of different missingness patterns is 2^D consider the number of $D-dimensional$ binary vectors m.

2.2.1 Approximations

In this section we discuss two different approximations to the exact solution of Equation (2.16), both are taken from [WNN18]. These approximations are used as a baseline in our experiments in Chapter 3.

Full Covariance Approximation (FCA) The idea behind FCA is to reduce the complexity of the problem by doing mean imputation as pre-processing step. Notice that, if we replace the missing values x_m with their unconditional mean μ_m, then Equation (2.16) can be written equivalently as

$$\mu_{z|x_v} = \Sigma_{z|x_v} W^T \Psi^{-1}(x^{mi} - \mu), \tag{2.19}$$

where mi stands for mean imputation. This happens because for all missing data coordinates j we have $x_j^{mi} - \mu_j = 0$, and so missing data slots have no effect on the calculation. The covariance matrix $\Sigma_{z|x_v}$, however, still depends on the missingness, and so to complete the approximation we replace it with $\Sigma_{z|x}$ that corresponds to a fully observed x, i.e. when there is no missing data. In this case we obtain

$$\mu_{z|x_v}^{FCA} = \Sigma_{z|x} W^T \Psi^{-1}(x^{mi} - \mu), \tag{2.20}$$

By using the fully observed covariance FCA essentially ignores the specific missing patterns and so it uses a fixed encoder for all patterns.

Scaled Covariance Approximation (SCA) If we observe Equation (2.17) we can see that if no data is observed the posterior covariance will be equal to the prior covariance which is just the identity matrix. This corresponds to the case where $m_j = 0$, for each $j \in \{1, ..., D\}$. On the other hand if there are no missing data the posterior covariance is equal to the full covariance, this corresponds to the case where $m_j = 1$, for each $j \in \{1, ..., D\}$. Let us now define

$$\Sigma_{z|x}^{-1} = P_{z|x} = I_K + \sum_{j=1}^{D} P_j, \tag{2.21}$$

where $P_j = \psi_{jj}^{-1} \nu_j \nu_j^T$. In general, we have that $P_{j_1} \neq P_{j_2}$ for $j_1 \neq j_2$, so the idea behind SCA is to assume that all the P_j's are equal. In this manner we can rearrange Equation (2.21) to obtain

$$\Sigma_{z|x_v}^{-1} = I_K + \frac{\sum_{j=1}^{D} m_j}{D}(P_{z|x} - I_K) = \frac{D_m}{D} I_K + \frac{D_v}{D} P_{z|x}, \tag{2.22}$$

which essentially is a linear interpolation between the two extreme cases of all the data being missing and no data being missing. SCA does not use a fixed encoder network for all patterns of missingness but the adjustment can be achieved by a matrix multiplication with very low computational cost.

2.3 Neumann Series

In this section we introduce the definition and some relevant theory for Neumann series, see e.g. [MS23, Wer06, DS88]. Furthermore, we discuss how they can be used as a means of approximating the solution of a linear system. This is the connection to our problem, since an effective encoding for FA in the presence of missing data can be obtained by solving a linear system. We propose using a Neumann series to approximate the solution. We conclude this section with a discussion of the bibliography on the use of Neumann series in problems similar to ours.

2.3.1 Theory

Definition 1 Let $A \in \mathbf{R}^{n\times n}$ be a linear map (i.e. a matrix), then the infinite series $\sum_{j=0}^{+\infty} A^j$ is called the *Neumann Series* of A.

Definition 2 Let $A \in \mathbf{R}^{n\times n}$ a matrix, then the spectral radius of A is defined as

$$\rho(A) = \max\{|\lambda_1|, ..., |\lambda_n|\},$$

where $\lambda_1, ..., \lambda_n$ are the eigenvalues of A.

The following theorem (and its proof), which can be found in Chapter 7 of [MS23], gives some conditions on the convergence of the Neumann series.

Theorem 1. *For $A \in \mathbf{R}^{n\times n}$ the following statements are equivalent:*

- *The Neumann series $I_n + A + A^2 + ...$ converges.*
- $\rho(A) < 1$
- $\lim_{j\to\infty} A^j = 0$

In which case $(I_n - A)^{-1}$ exists and

$$(I_n - A)^{-1} = \sum_{j=0}^{+\infty} A^j. \tag{2.23}$$

To understand Equation (2.23) consider that

$$\sum_{j=0}^{+\infty} A^j(I_n - A) = (I_n - A)\sum_{j=0}^{+\infty} A^j = \sum_{j=0}^{+\infty}(A^j - A^{j+1}) = I_n - \lim_{j\to\infty} A^j = I_n. \tag{2.24}$$

Notice that the Neumann series essentially generalises the geometric series for matrices, see [MS23]. As per Equation (2.23), we see that if we make the change of variables $B = I_n - A$, we obtain

$$B^{-1} = \sum_{j=0}^{+\infty}(I_n - B)^j, \quad \text{for} \quad \rho(I_n - B) < 1 \tag{2.25}$$

which gives us a way of estimating the matrix inverse.

Note 2: The Neumann series definition can be extended for general infinite-dimensional Banach spaces, e.g. functional vector spaces, and bounded linear operators. However for our purpose the definition we have given is sufficient. For a detailed treatise on Neumann series defined on Banach spaces we refer the reader to [Wer06, DS88].

A very useful result connecting Neumann series to linear systems of equations is the following:

Theorem 2. *Let $A \in \mathbf{R}^{n\times n}$ with $\rho(A) < 1$, then given $g, u \in \mathbf{R}^n$ the equation*

$$(I_n - A)u = g,$$

has a unique solution, with

$$u = (I_n - A)^{-1}g = \sum_{j=0}^{+\infty} A^j g. \tag{2.26}$$

Furthermore, the solution u can be approximated by

$$u_l = g + Ag + A^2 g + ... + A^l g = \sum_{j=0}^{l} A^j g, \tag{2.27}$$

and the approximation error is given by

$$||u - u_l|| \le \frac{||A||^l}{1 - ||A||} ||g||, \quad \textit{for all} \quad l \in \mathbf{N}. \tag{2.28}$$

The proof can be found in Appendix A. Two important points arising from Theorem (2) are the following. First, by truncating the Neumann series at the l^{th} step we have a natural approximating process for the solution of a linear system. Second, when the Neumann series is convergent it turns the problem of solving a linear system into a problem of computing a repeated matrix-vector multiplication. This is the intuition behind the research direction we took with this project, if not too many iterations are required for convergence, we can estimate the solution to Equation (2.18) with Theorem (2) to save up a lot of computation. To put this into perspective for a $K \times K$ linear system the computational cost is generally $O(K^3)$, while a matrix-vector multiplication of a $K \times K$ with a $K \times 1$ vector will cost $O(K^2)$, see e.g. [GVL13], and so if not too many matrix-vectors multiplications are required the speed-up can be significant.

Note 3: In Equation (2.28) we have given a bound on the approximation error that depends on the norm $||A||$. We would like to give this bound in terms of the spectral radius $\rho(A)$, for reasons related to computational convenience. First, notice that in finite dimensional spaces all norms are equivalent, see e.g. [DS88] and so the specific choice of norm does not make a difference. Without loss of generality, we can work with the 2-norm, which we denote with $|| \cdot ||_2$. The important point now, is that if $A \in \mathbf{R}^{n \times n}$ is symmetric, i.e. $A^T = A$, then $\rho(A) = ||A||_2$. The proof of this result can be found in [HJ12]. This implies that for a symmetric matrix A, the upper bound in Equation (2.28) can be equivalently written as

$$||u - u_l||_2 \le \frac{[\rho(A)]^l}{1 - \rho(A)} ||g||_2, \quad \text{for all} \quad l \in \mathbf{N}. \tag{2.29}$$

This is relevant for us because we are working with covariance matrices, which are by definition symmetric. We will come back to this upper bound when we discuss the convergence of NSA in Chapter 3.

2.3.2 Related Work

Gilton et. al. [GOW19] have used Neumann Series in the context of ill posed linear inverse problems in imaging, where they propose a neural network architecture based on Neumann series. Their goal is to solve a regularised variant of a linear inverse problem. The main difference between our project and their work is that their model does not have a quadratic prior on the latent representation z, contrary to our FA model, which uses a Gaussian prior. Additionally, even though they discuss generative models they do not take an autoencoder view of their work and they are more interested in the context of inverse problems. We have drawn some inspiration from their work, e.g. the mixture of factor analysers model we discuss in Chapter 3 can be viewed as a probabilistic variant their *Union of Subspaces* model.

Le Morvan et. al. [LMJM+20], have employed Neumann series in the context of supervised learning and more specifically to tackle a similar matrix inversion to ours to compute the Bayes predictor for linear regression in the presence of MAR missingness (they also cover Missing-Not-At-Random (MNAR) in their work, which means that the probability of missingness depends on the underlying missing values, see e.g. [Rub76]). Their iterative formula for the Neumann series update (equation 6 in their paper) is very similar to what we are using (up to a scaling constant), with the main difference being that they are using it to invert a matrix, while we are solving a system. Additionally, with their approach they are dealing with a varying dimensionality of x_v and x_m, while we are working with a $K \times K$ matrix that varies according to M.

2.4 Mixture of Factor Analysers

Standard Factor Analysis assumes global linearity, i.e. a single linear model models the entire data space. However, a more flexible approach is to allow for different regions of the data space to be modelled by different sub-models. The probabilistic nature of FA lends itself naturally for an extension to a mixture model. For this purpose, Hinton et. al. [HDR97] proposed the mixture of Factor Analysers model. This model is essentially a hybrid of a linear dimensionality reduction method (FA) and a clustering method (Gaussian mixture models, see e.g. [Bis06]). In this context the mixture of FAs model can be understood as a mixture of Gaussians where the covariance matrix of each component is forced to have a certain low-rank structure.

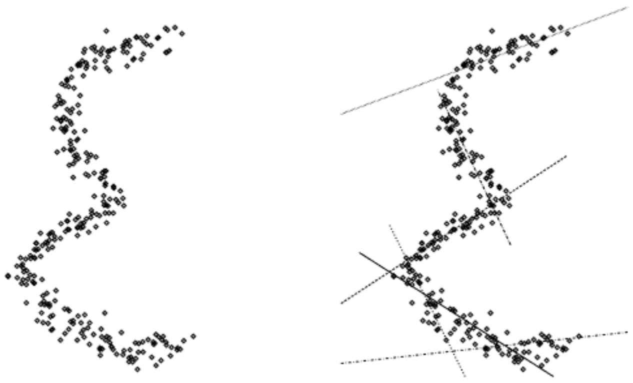


Figure 2: Illustration of how a highly nonlinear one-dimensional surface can be captured by 6 locally linear model. Figure taken from [HDR97].

Assuming that we are using M mixture components, the density of a data vector x under this model is given by

$$p(x) = \sum_{i=1}^{M} \pi_i p(x|i), \tag{2.30}$$

where $p(x|i), \quad i = 1, ..., M$, is a single standard FA model (see the definitions in Section (2.1.1)), and $\pi_i \geq 0$ is the corresponding mixture coefficient with $\sum_{i=1}^{M} \pi_i = 1$. Separate parameters μ_i, Ψ_i, W_i are associated with each different mixture component. The generative procedure for the mixture model corresponds to sampling a mixture component at random according to the proportions π_i and then proceeding as in the standard Factor Analysis case to generate a sample for x. An interesting observation now is that for a given data vector x there will be a posterior distribution associated with every mixture component. Considering what we have seen for the standard FA model, the posterior mean for component i is given by

$$\mu_{z|x,i} = \Sigma_{z|x,i} W_i^T \Psi_i^{-1}(x - \mu_i), \tag{2.31}$$

which is exactly the encoding predicted by each different mixture component. The parameters of each component can be obtained through EM, for more details see [TB99a, HDR97]. More specifically, we are interested in computing a quantity we call the posterior responsibility, following Tipping et al. [TB99a]. On a more intuitive level the posterior responsibilities show how *strongly* a data vector belongs to each component. Let us assume that we are working with a dataset $\mathcal{D} = \{x^{(1)}, x^{(2)}, ..., x^{(N)}\}$. In this case, the posterior responsibility of mixture component i for generating the data vector $x^{(n)}$, is defined as

$$r_{ni} = p(i|x^{(n)}) = \frac{p(x^{(n)}|i)\pi_i}{p(x^{(n)})}. \tag{2.32}$$

In terms of clusters, the posterior responsibility encodes the probability of membership for cluster/ component i, after observing the data vector $x^{(n)}$.

As per Equation (2.32), the denominator term, which is given by

$$p(x^{(n)}) = \sum_{i=1}^{M} \pi_i p(x^{(n)}|i), \tag{2.33}$$

is a normalising factor and thus it is the same for every mixture component. This implies that the quantity of interest in Equation (2.32), is really just the numerator. Taking advantage of this, we can write

$$r_{ni} \propto p(x^{(n)}|i)\pi_i, \tag{2.34}$$

and without loss of generality we can work with the log-responsibilities, because the log is a strictly monotonic function. This implies

$$\log r_{ni} \propto \log p(x^{(n)}|i) + \log \pi_i, \tag{2.35}$$

where the quantity of interest is

$$\log p(x^{(n)}|i) = \frac{1}{2}(x^{(n)} - \mu_i)^T \Sigma_i^{-1}(x^{(n)} - \mu_i) - \frac{1}{2}\log|\Sigma_i| - \frac{D}{2}\log(2\pi), \tag{2.36}$$

where D is the dimension of $x^{(n)}$, and $|\Sigma_i|$ denotes the determinant, where

$$\Sigma_i = \Psi_i + W_i W_i^T. \tag{2.37}$$

An interesting observation now is that we can use the Woodbury formula (see, e.g. Chapter 2 in [GVL13]) to tackle the matrix inversion required in Equation (2.36).

Woodbury Formula: For matrices $A \in \mathbf{R}^{n\times n}$, $C \in \mathbf{R}^{k\times k}$, $U \in \mathbf{R}^{n\times k}$, and $V \in \mathbf{R}^{k\times n}$ we have

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)VA^{-1}. \tag{2.38}$$

So, based on Equation (2.38) for $\Psi = A$, $U = W$, $V = W^T$, $C = I_K$ and by matching the dimensions to our case, i.e. $n = D$ and $k = K$ (for a detailed discussion relating the Woodbury formula to Probabilistic PCA, see [TB99a]), we obtain

$$\Sigma_i^{-1} = (\Psi_i + W_i W_i^T)^{-1} = \Psi_i^{-1} - \Psi_i^{-1}W_i(I_K + W_i^T\Psi_i^{-1}W_i)^{-1}W_i^T\Psi_i^{-1}, \tag{2.39}$$

so instead of inverting the original covariance matrix which is a $D \times D$ matrix, we can do the matrix inversion in the latent space by inverting the posterior covariance for each cluster

$$\Sigma_{z|x,i} = I_K + W_i^T\Psi_i^{-1}W_i, \tag{2.40}$$

which is a $K \times K$ matrix. An important thing to observe, is that Equation (2.36) can be equivalently written as

$$\log p(x^{(n)}|i) = \frac{1}{2}(x^{(n)}-\mu_i)^T(\Psi_i^{-1}-\Psi_i^{-1}W_i(I_K+W_i^T\Psi_i^{-1}W_i)^{-1}W_i^T\Psi_i^{-1})(x^{(n)}-\mu_i)-\frac{1}{2}\log|\Sigma_i|-\frac{D}{2}\log(2\pi), \tag{2.41}$$

which implies that

$$\log p(x^{(n)}|i) = \frac{1}{2}(x^{(n)}-\mu_i)^T\Psi_i^{-1}(x^{(n)}-\mu_i)-\frac{1}{2}(x^{(n)}-\mu_i)^T\Psi_i^{-1}W_i\mu_{z|x^{(n)},i}-\frac{1}{2}\log|\Sigma_i|-\frac{D}{2}\log(2\pi). \tag{2.42}$$

So far, we have shown that we can perform the required computations more efficiently by inverting the covariance matrix in the latent space, which has a lower dimension than the data space. An additional point worth considering is how to compute the determinant $|\Sigma_i|$, for $i = 1, ..., M$. An elegant and computationally efficient solution is to use the generalised matrix determinant lemma, for a proof see e.g. [ZZ09]. This allows us to reduce the computational complexity of our calculation even further.

Generalised Matrix Determinant Lemma: Let A be an invertible $n \times n$ matrix and U, V be $n \times m$, then we write

$$\left|A + UV^T\right| = \left|I_m + V^T A^{-1} U\right| |A|, \tag{2.43}$$

where $|\cdot|$ denotes the determinant.

In this case we can take $A = \Psi_i$, $U = W_i$, $V^T = W_i^T$ and $n = D$, $m = K$ to obtain

$$\left|\Psi_i + W_i W_i^T\right| = \left|I_K + W_i^T \Psi^{-1} W_i\right| |\Psi_i|, \tag{2.44}$$

The main benefit of Equation (2.44) is that we have reduced the computation of the determinant of a $D \times D$ matrix to the computation of the determinant of a $K \times K$ matrix multiplied by the determinant of a diagonal matrix, which can be computed with low computational cost. To summarise, we have deduced

$$\log r_{ni} \propto \frac{1}{2}(x^{(n)} - \mu_i)^T \Psi_i^{-1} (x^{(n)} - \mu_i) - \frac{1}{2}(x^{(n)} - \mu_i)^T \Psi_i^{-1} W_i \mu_{z|x^{(n)},i} - \\ -\frac{1}{2}\log\left(\left|I_K + W_i^T \Psi^{-1} W_i\right| |\Psi_i|\right) - \frac{D}{2}\log(2\pi) + \log \pi_i, \tag{2.45}$$

We have dedicated most of this section to analysing how to reason about the posterior responsibilities. This is motivated by the fact that, they become important in the presence of missing data. To see why let us consider the following scenario. Let us assume that, we have pre-trained the generative model parameters $\pi_i, \mu_i, \Psi_i, W_i$, for each cluster, using EM, and we would like to reconstruct a test vector with our model. In this case the posterior responsibilities demonstrate what the model *believes* about cluster membership. In this case, it is interesting to think how might the presence of missingness affect belief about cluster membership. In the next section, we show how the equations we have derived in this section can be extended to compute the posterior responsibilities in the presence of missing values.

2.4.1 Mixtures of Factor Analysers with Missing Data

In this section we consider the mixture of FAs in the presence of missing data. Following the same line of reasoning as in [WNN18], let us consider Equation (2.42) in the presence of MAR missing data. In this case marginalising over the missing values will result in working with the *visible* part of the n^{th} data vector, i.e. $x_v^{(n)}$ and so based on Equation (2.42) we obtain

$$\log p(x_v^{(n)}|i) = \frac{1}{2}(x_v^{(n)} - \mu_i)^T \Psi_{v,i}^{-1} (x_v^{(n)} - \mu_i) - \frac{1}{2}(x_v^{(n)} - \mu_i)^T \Psi_{v,i}^{-1} W_{v,i} \mu_{z|x_v^{(n)},i} - \\ -\frac{1}{2}\log|\Sigma_{v,i}| - \frac{D}{2}\log(2\pi). \tag{2.46}$$

If we now use a mask matrix M as in the standard FA case, Equation (2.46) can be written in an equivalent manner as

$$\log p(x_v^{(n)}|i) = \frac{1}{2}(x^{(n)} - \mu_i)^T M \Psi_i^{-1} M (x^{(n)} - \mu_i) - \frac{1}{2} M (x^{(n)} - \mu_i)^T M \Psi_i^{-1} M W_i \mu_{z|x_v^{(n)},i} - \\ -\frac{1}{2}\log|\Sigma_{v,i}| - \frac{D}{2}\log(2\pi), \tag{2.47}$$

where $\Sigma_{v,i} = \Psi_{i,v} + W_{i,v} W_{i,v}^T$.

As per Equation (2.44) we can write

$$|\Sigma_{i,v}| = \left|\Psi_{i,v} + W_{i,v} W_{i,v}^T\right| = \left|I_K + W_{i,v}^T \Psi_{i,v}^{-1} W_{i,v}\right| |\Psi_{i,v}| = \left|I_K + W_i^T M \Psi_i^{-1} M W_i\right| |\Psi_{i,v}|. \tag{2.48}$$

In this case, the visible posterior responsibility will be

$$\log r_{v,ni} = \frac{1}{2}(x^{(n)} - \mu_i)^T M \Psi_i^{-1} M (x^{(n)} - \mu_i) - \frac{1}{2} M (x^{(n)} - \mu_i)^T M \Psi_i^{-1} M W_i \mu_{z|x_v^{(n)},i} - \\ -\frac{1}{2}\log(\left|I_K + W_i^T M \Psi_i^{-1} M W_i\right| |\Psi_{i,v}|) - \frac{D}{2}\log(2\pi) + \log \pi_i, \tag{2.49}$$

An interesting point regarding the term $|\Psi_{i,v}|$, is that since we are only interested in the determinant, it is preferable to compute this term directly, rather than computing it as a multiplication with a mask matrix, i.e. $M\Psi_i$. The merit of this is that, we avoid introducing zeros to the diagonal and therefore making the determinant zero. In essence, we see that to compute the responsibilities for each component, we have to compute the posterior mean for each component and so to solve the problem of obtaining an effective encoding we are not required to do extra work in the computational sense, because the posterior mean/encoding can be computed only once and then stored. We will come back to this idea when we develop the Neumann series algorithm for the mixture of FAs model (NSA-M) in Chapter 3. A interesting idea relating the posterior responsibilities to missing data is to understand how the missingness affects model uncertainty about cluster membership. For example if part of an image is missing then this might introduce uncertainty not otherwise present in the complete case. For example, if we remove a specific part of an image, the model might change its belief about the most probable cluster. We discuss this further in Section 3.2, where we examine experimental results related to the reconstruction performance of NSA-M.

3 Methods

In this chapter we present our novel contributions. The work is mainly divided into two subsections. In the first section we derive the iterative formula required for the Neumann series algorithm for the FA model (NSA) in the presence of MAR missing data and prove its convergence. In addition, we show how our algorithm can be thought of as the encoder or recognition network of an autoencoder, which we call the *Neumann Autoencoder*. The first section is concluded with some experiments on an image in-painting task demonstrating the performance of our algorithm, and a discussion of the results. The second section is a logical continuation of the first and is structured much in the same way. At first we derive the iterative formula required for the Neumann series algorithm for the mixture of FAs model (NSA-M) extending the work we presented in Section (2.4.1) and we conclude with some experiments and discuss the results we obtain.

3.1 Neumann Series for FA with Missing Data

As we have previously described, the problem of obtaining an effective encoding when there is missing data, in essence amounts to solving a linear system that depends on the missingness pattern, see Equation (2.16). Furthermore, we have seen that performing mean imputation as a pre-processing step has no effect on the computation, because for all missing data coordinates j we have $x_j^{mi} - \mu_j = 0$ (see the Full Covariance Approximation in Section (2.2.1)). This implies that working with Equation (2.20) will produce equivalent results to working with Equation (2.16), for computational convenience we proceed with the former. More specifically, let us assume

$$u = W^T \Psi^{-1}(x^{mi} - \mu), \tag{3.1}$$

and

$$A = I_K + W^T M \Psi^{-1} M W, \tag{3.2}$$

then

$$\Sigma_{z|x_v} = A^{-1}. \tag{3.3}$$

This means that the posterior mean, i.e. the encoding is the solution to the linear system

$$A\mu_{z|x_v} = u \implies \mu_{z|x_v} = A^{-1}u. \tag{3.4}$$

As per the discussion in Chapter 2, Section (2.3.1), we propose using the Neumann series of $I_K - A$ to approximate the solution of the system (3.4). More specifically, by applying Theorem (2) for $I_K - A$, we can obtain

$$\mu_{z|x_v} = A^{-1}u = \sum_{j=0}^{+\infty}(I_K - A)^j u, \tag{3.5}$$

under the requirement that $\rho(I_K - A) < 1$. We support that, by taking advantage of the structure of our matrix A, the condition for convergence can be achieved with a simple pre-processing step. To be more specific, let us define the scaling factor

$$s := \rho(A) + \epsilon \quad \text{for some} \quad \epsilon > 0, \tag{3.6}$$

and the scaled matrix

$$A_s := \frac{1}{s}A. \tag{3.7}$$

In this case, if λ, v are an eigenvalue-eigenvector pair for A we can write

$$Av = \lambda v \implies \frac{1}{s}Av = \frac{1}{s}\lambda v \implies A_s v = \left(\frac{1}{s}\lambda\right) v, \tag{3.8}$$

and so if λ is an eigenvalue of A, then $\frac{1}{s}\lambda$ is an eigenvalue of A_s. This implies

$$\rho\left(A_s\right) = \frac{1}{s}\rho(A), \tag{3.9}$$

which is guaranteed to be less than 1, because of the scaling $\frac{1}{s}$. We will prove that this extra step also implies $\rho(I_K - A_s) < 1$, which will guarantee the convergence of a slightly modified Neumann series that allows us to obtain a solution. To see why this works observe that the pre-processing step does not affect the solution of Equation (3.4), by multiplying both sides with $\frac{1}{s}$, we obtain

$$\frac{1}{s}A\mu_{z|x_v} = \frac{1}{s}u \implies \mu_{z|x_v} = \left(\frac{1}{s}A\right)^{-1}\frac{1}{s}u = A^{-1}u. \tag{3.10}$$

In addition, if we consider the Neumann series for the matrix $I_K - A_s$, we can write

$$A_s^{-1} = \sum_{j=0}^{\infty}(I_K - A_s)^j, \tag{3.11}$$

and so the Neumann series for the initial (not scaled) matrix $I_K - A$ will be

$$A^{-1} = \frac{1}{s}A_s^{-1} = \frac{1}{s}\sum_{j=0}^{\infty}(I_K - A_s)^j. \tag{3.12}$$

So the linear system in Equation (3.10) can be written in an equivalent manner as

$$\mu_{z|x_v} = \frac{1}{s}\left(\frac{1}{s}A\right)^{-1}u = \frac{1}{s}\left[\sum_{j=0}^{\infty}(I_K - A_s)^j u\right]. \tag{3.13}$$

By truncating the infinite series at the l first terms we have a very natural approximating process, i.e.

$$\mu_{z|x_v}^{(l)} = \frac{1}{s}\left[\sum_{j=0}^{l}(I_K - A_s)^j u\right]. \tag{3.14}$$

We propose to compute the approximation with the following iterative formula

$$y^{(l)} = (I_K - A_s)y^{(l-1)} + u, \tag{3.15}$$

with $y^{(0)} = u$. In order to obtain the approximation $\mu^{(l)}_{z|x_v}$ we multiply $y^{(l)}$ with the inverse scaling factor after concluding the iterations, i.e.

$$\mu^{(l)}_{z|x_v} = \frac{1}{s} y^{(l)}. \tag{3.16}$$

To see why this works, consider that

$$((I_K - A_s)A_s^{-1} - I_K)u = A_s^{-1}u, \tag{3.17}$$

so our iterative scheme has the desired fixed point (for more details see Fixed Point Theorem, see e.g. [BFB15]). Additionally, note that A is the inverse of the posterior covariance and so it is a symmetric matrix, and therefore $I_K - A$ is also symmetric, which implies that $I_K - A_s$ is symmetric.
This means that, by applying Theorem (2), the error of the approximation is given by

$$\left|\left|y - y^{(l)}\right|\right| \leq \frac{||I_K - A_s||^l}{1 - ||I_K - A_s||}||u|| = \frac{[\rho(I_K - A_s)]^l}{1 - \rho(I_K - A_s)}||u|| \quad \text{for} \quad l \in \mathbf{N}, \tag{3.18}$$

which implies

$$\left|\left|\mu_{z|x_v} - \mu^{(l)}_{z|x_v}\right|\right| = \left|\left|\frac{1}{s}y - \frac{1}{s}y^{(l)}\right|\right| = \frac{1}{s}\left|\left|y - y^{(l)}\right|\right| \leq \frac{1}{s}\left[\frac{[\rho(I_K - A_s)]^l}{1 - \rho(I_K - A_s)}||u||\right] \quad \text{for} \quad l \in \mathbf{N} \tag{3.19}$$

So far, we have relied on using a scaling factor to make the algorithm converge. However, the scaling factor still depends on the missingness pattern, and computing it amounts to solving an expensive eigenvalue problem per missingness pattern. We would like to do something more efficient. Fortunately, there is an elegant solution. More specifically, let us define

$$B := W^T \Psi^{-1} W, \tag{3.20}$$

and

$$B_M := W^T M \Psi^{-1} M W, \tag{3.21}$$

for some missingness mask M. We can prove that

$$\rho(B_M) \leq \rho(B), \quad \text{for every missingness mask} \quad M, \tag{3.22}$$

which implies

$$\rho(A) = \rho(I_K + B_M) \leq \rho(I_K + B), \quad \text{for every missingness mask} \quad M. \tag{3.23}$$

A proof of this result is presented in the next section. The great advantage now is that we have found an upper bound for every possible M, so by setting

$$s = \rho(I_K + B) + \epsilon \quad \text{for some} \quad \epsilon > 0, \tag{3.24}$$

we only need to compute the scaling factor once and we can use it for every missingness pattern.

3.1.1 Algorithm and Proof of Convergence

In this section, we combine everything to present NSA (see Algorithm (1)), which is the main algorithm of this project. The parameters μ, W, Ψ are assumed to be pre-trained and so they can be given as inputs. Additionally, we show that NSA can be regarded as being the encoder part of an autoencoder.

Algorithm 1 NSA

```
    Input: $\mu, W, \Psi, l, M, x^{mi}$
    Output: $\hat{x}$

  compute $s$                                              ▷ scaling
  $u \leftarrow W^T \Psi^{-1}(x^{mi} - \mu)$
  $A_s \leftarrow \frac{1}{s}(I_K + W^T M \Psi^{-1} M W)$
  $z \leftarrow u$
  while count$< l$ do
      $z \leftarrow (I - A_s)z + u$
      count $\leftarrow$ count $+1$
  end while
  $\mu_{z|x_v} \leftarrow \frac{1}{s} z$
  $\hat{x} \leftarrow W \mu_{z|x_v} + \mu$
  return $\hat{x}$
```

Proof of Convergence To prove the convergence of our algorithm, we have to prove the convergence of the Neumann series of $I_K - A_s$ which is guaranteed if

$$\rho(I_K - A_s) < 1, \tag{3.25}$$

for

$$s = \rho(I_K + B) + \epsilon \quad \text{for some} \quad \epsilon > 0, \tag{3.26}$$

for B, B_M as defined in Equations (3.20) and (3.21) respectively. First we shall prove that

$$\rho(A_s) < 1 \tag{3.27}$$

and subsequently that

$$\rho(A_s) < 1 \implies \rho(I_K - A_s) < 1, \tag{3.28}$$

in which case, our convergence is guaranteed. We start by proving that B_M is positive semi-definite. Indeed, let us consider a vector $y \in \mathbf{R}^K$ with $y \neq 0$ then

$$y^T B_M y = y^T W^T M \Psi^{-1} M W y = (Wy)^T M \Psi^{-1} M (Wy), \tag{3.29}$$

then for $z = Wy$, we can write

$$y^T B_M y = \sum_{i=1}^{D} m_i \psi_{ii}^{-1} z_i^2 \geq 0, \tag{3.30}$$

because $m_i, \psi_{ii}^{-1} \geq 0$ for every $i = 1, .., D$, and so by the definition of positive semi-definiteness, B_M is positive semi-definite. This implies that all of its eigenvalues (denoted by $\lambda(\cdot)$) are bounded below by zero, i.e.

$$0 \leq \lambda_{min}(B_M) \leq ... \leq \lambda_{max}(B_M). \tag{3.31}$$

It also implies

$$\rho(B_M) = \lambda_{max}(B_M), \tag{3.32}$$

because all the eigenvalues are non-negative. Next, we are interested in comparing the eigenvalues of B_M with the eigenvalues of B. Let us first consider the case where only one coordinate is missing. Without loss of generality, let us assume that we are missing the t^{th} coordinate. In this case B_M, as per Equation (2.17), can be written as

$$B_M = B - \psi_{tt}^{-1} v_t v_t^T, \tag{3.33}$$

which means that to obtain B_M, we subtract a rank 1 matrix from B, we call this a rank 1 down-date. We can connect the eigenvalues of B and B_M through the Cauchy Interlacing Theorem for rank 1 updates, for a proof see Appendix A, for more details see e.g. Chapter 7 of [MS23].

Theorem 3. *Let $A \in \mathbf{R}^{n\times n}$ be symmetric and $v \in \mathbf{R}^n$, then if*

$$\alpha_n \leq ... \leq \alpha_1,$$

are the eigenvalues of $A + vv^T$, and

$$\beta_n \leq ... \leq \beta_1,$$

are the eigenvalues of A, we have

$$\beta_n \leq \alpha_n \leq ... \leq \beta_1 \leq \alpha_1.$$

This is called the interlacing property.

So, from Equation (3.33) we can write

$$B = B_M + \psi_{tt}^{-1} v_t v_t^T, \tag{3.34}$$

and by applying Theorem (3) for B_M we have that

$$\lambda_{max}(B_M) \leq \lambda_{max}(B_M + \psi_{tt}^{-1} v_t v_t^T) = \lambda_{max}(B), \tag{3.35}$$

where $\lambda_{max}(\cdot)$ denotes the largest eigenvalue, so taking into account the non-negativity of the eigenvalues as per Equation (3.31), we have

$$\rho(B_M) \leq \rho(B). \tag{3.36}$$

Note that we assumed that we had only one missing dimension. However, this argument can be easily be extended to accommodate any number of possible missing dimensions. To see why, consider that for more than one missing dimensions, we can apply consecutive rank 1 down-dates and so because the resulting matrix will remain symmetric we can inductively apply this argument to show that for any number of missing coordinates, the the spectral radius of B_M is always bounded above by the spectral radius of B. To summarise, so far we have shown, that the eigenvalues of B_M are bounded below by zero and bounded above by $\rho(B)$, i.e.

$$0 \leq \lambda_{min}(B_M) \leq ... \leq \lambda_{max}(B_M) \leq \rho(B), \tag{3.37}$$

which implies that

$$1 + \lambda_{max}(B_M) \leq 1 + \rho(B), \tag{3.38}$$

which implies that

$$\rho(A) = \rho(I_K + B_M) \leq \rho(I_K + B), \tag{3.39}$$

where we have made use of the fact that if λ is an eigenvalue of an arbitrary matrix C then $1 + \lambda$ is an eigenvalue of $I + C$.

This implies

$$\lambda_{min}(A) \geq 1 \implies \lambda_{min}(A_s) > 0, \tag{3.40}$$

and

$$\rho(A_s) = \frac{1}{s}\rho(A) \leq \frac{1}{s}\rho(I_K + B) < 1, \quad \text{for a suitable} \quad \epsilon > 0. \tag{3.41}$$

Finally, from the non-negativity of eigenvalues, we can write

$$\lambda_{max}(I - A_s) = 1 - \lambda_{min}(A_s), \tag{3.42}$$

but $\lambda_{min}(A_s) \in (0, 1)$ and so

$$\rho(I - A_s) = \lambda_{max}(I - A_s) < 1 \tag{3.43}$$

which concludes the proof. $\square$

3.1.2 Neumann Autoencoder

Finally, we present the *Neumann Autoencoder*, an architecture that encompasses the encoding and the decoding processes we have described. The Neumann blocks are a pictorial representation of equation (3.15), where each block gets fed into its subsequent block. The initial block $u = z^{(0)}$ gets multiplied with the next block and gets added to u and this process repeats. Notice how the connections in the encoder network resemble the residual connections.

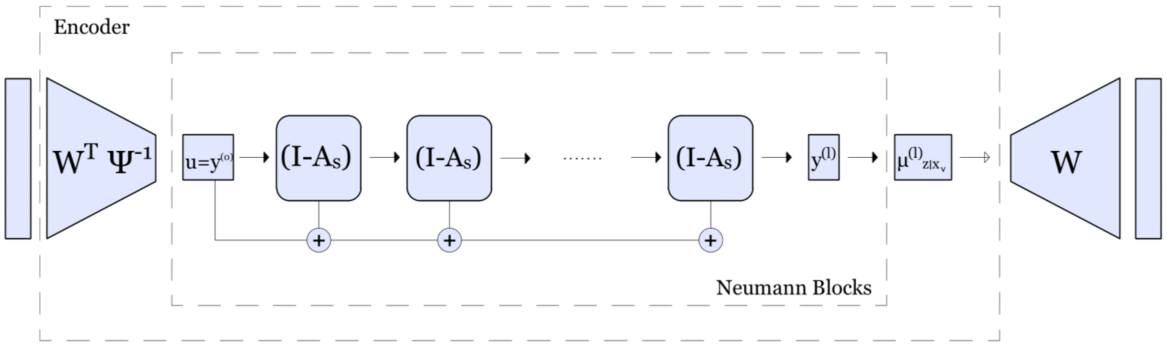


Figure 3: Neumann Autoencoder.

3.2 Experiments for NSA

We have trained a Probabilistic PCA (PPCA) model, as an instance of Factor Analysis. We have made that choice because the maximum likelihood estimates for the model parameters exist in closed form [TB99b]. We are using those closed form solutions to estimate W, μ, σ^2. We are using the Frey faces dataset, which consists 1956 grey-scaled images of Brendan Frey's face taken from sequential frames of a video with the dimension of 28×20. The data can be obtained here (`http://www.cs.nyu.edu/~roweis/data/frey_rawface.mat`). We have scaled pixel intensities, so that they lie in between -1 and 1. We have randomly selected 80% of the data as a training set, while the rest of the data is held out as a test set. The task we examine is in essence image in-painting. Given the trained PPCA model we impose some missingness pattern on the training and the test set and we use the model to perform missing data imputation. For the missingness patterns we impose we follow [WNN18] and we consider two missingness patterns. For the first pattern we just set each pixel to missing with probability 0.5, we call that pattern *random missingness*. For the second pattern, we sample a random integer in the range $1, \ldots, 4$ and based on the outcome we remove one of the four quarters of the image, e.g. if the outcome is 1 we remove the top left quarter. We call this pattern *quarters missingness*. Notice that for both patterns the probability of missingness is independent of the underlying missing values and so we are in the MAR case.

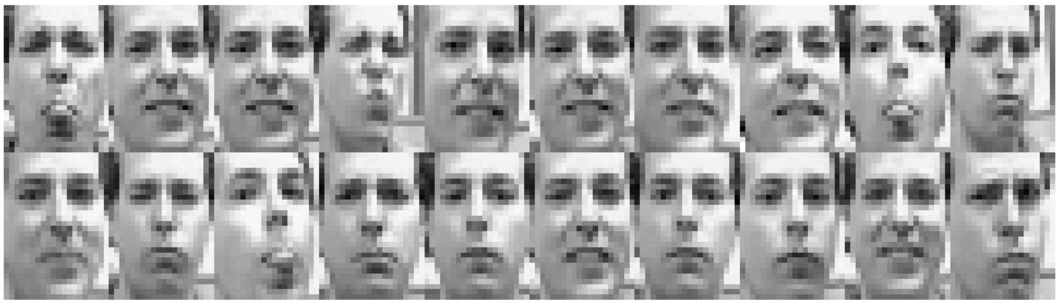


Figure 4: Examples from the Frey faces dataset.

We consider five different methods for data imputation, mean imputation where we impute the missing data dimensions with their unconditional mean, FCA, SCA and the exact method of Williams et. al. [WNN18] and of course the Neumann series algorithm. We are interested in testing which method provides the best reconstruction of the original image, and we are using the mean squared error (MSE) as a metric for that objective. For the dimension of the latent space we are using $q = 43$ latent components explaining 90% of the observed variance in the data. Finally, we are comparing how much time it takes for the exact solution to run versus our solution. The experiments are performed on a machine with an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz processor and 16GB RAM (the code for the experiments can be found `https://github.com/AntypasEvangelos/Dissertation`).

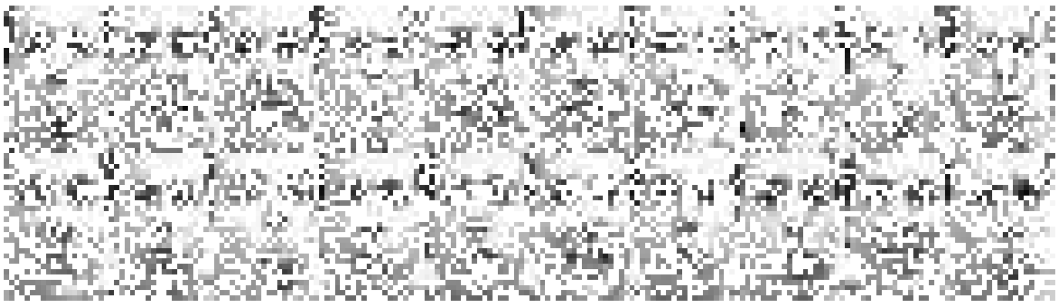


Figure 5: Examples of random missingness.

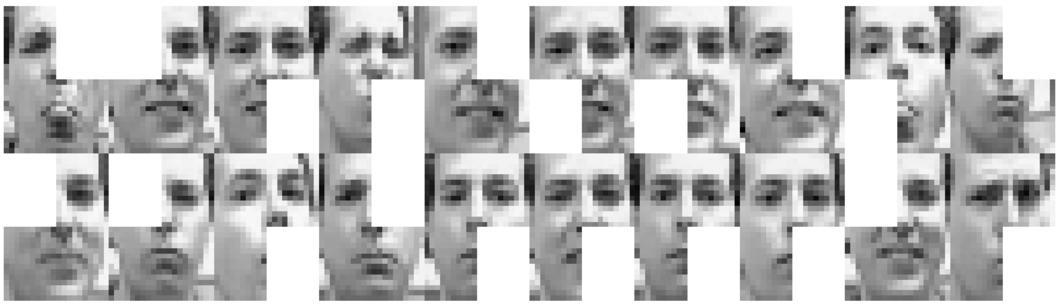


Figure 6: Examples of the quarters missingness.

Note 4: The white parts in images (see e.g. Figures (5),(6)) signify missing pixels. We follow this convention throughout the remainder of this project.

Note 5: Once we have obtained the encoding $\mu_{z|x_v}$, we are using the operation $W\mu_{z|x_v} + \mu$ to perform the decoding. During the development of the code for this project we became aware of a different solution. More specifically, Tipping and Bishop [TB99b] suggest using $W(W^TW)^{-1}(\sigma^2 I + W^TW)^{-1}W\mu_{z|x_v} + \mu$ as a decoder, the justification being that $W\mu_{z|x_v} + \mu$ is not an orthogonal projection because of the Gaussian prior over the latent variable z, which causes the posterior mean projection to become skewed towards the origin. This is a valid point and it was taken into consideration. However, the change in the experimental results was minuscule and so the original implementation was kept. This decision was made to keep logical continuity between this project and [WNN18], which we largely follow, as these authors have used $W\mu_{z|x_v} + \mu$ for the decoding.

3.2.1 Results

We are interested in studying how NSA performs in terms of the imputation/reconstruction quality as we take more steps, we refer to the number of steps as the *unrolling length*. Based on the theoretical results of Section (3.1.1) we expect that as the unrolling length increases the NSA reconstruction error should match the exact solution error. To see why that is, consider that if convergence in the latent space is achieved, then NSA and the exact solution produce approximately the same encoding. To study the behaviour of the reconstruction error, in Figures (7) and (8), we show the reconstruction error of the Neumann series algorithm for random missingness as a function of the unrolling length plotted against the reconstruction error of the exact method for the same task, for the training and the test set respectively. In a similar manner, in Figures (9) and (10), we show the reconstruction error as a function of the unrolling length for the quarters missingness pattern.

There is a number of very interesting observations to be made in Figures (7),(8),(9),(10). First and foremost, the plots indicate that the Neumann series method does indeed converge to the exact solution for both missingness patterns. However, this happens in a rather surprising way. For the random missingness pattern, the behaviour we observe, matches our prior expectation that the exact solution should outperform the NSA, until convergence is achieved, in which case the errors should be approximately the same. On the other hand, for the quarters missingness the NSA **outperforms** the exact solution for a range of unrolling lengths, in terms of the reconstruction error. To understand, how this can appear, consider that the NSA estimate approaches the exact solution as the unrolling length increases, this convergence is happening in the latent space. However, the reconstruction error is measured in the data space, which means that the encodings by NSA produced, before convergence is achieved, lead to reconstructions that are closer to the original/complete image than those produced by the exact solution.

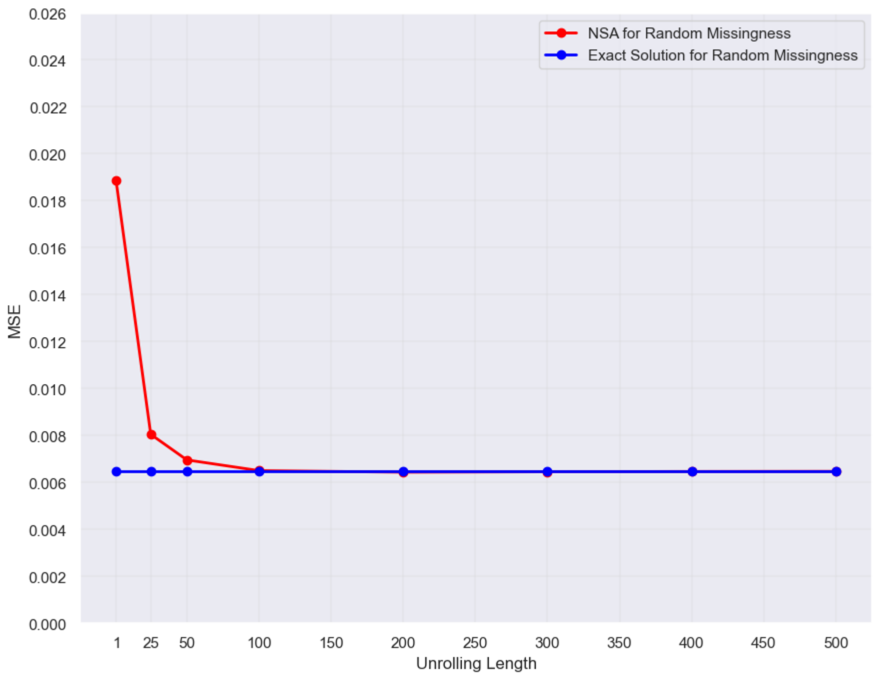


Figure 7: Reconstruction error as a function of the unrolling length of NSA vs the reconstruction error of the exact solution for random missingness on the training set.

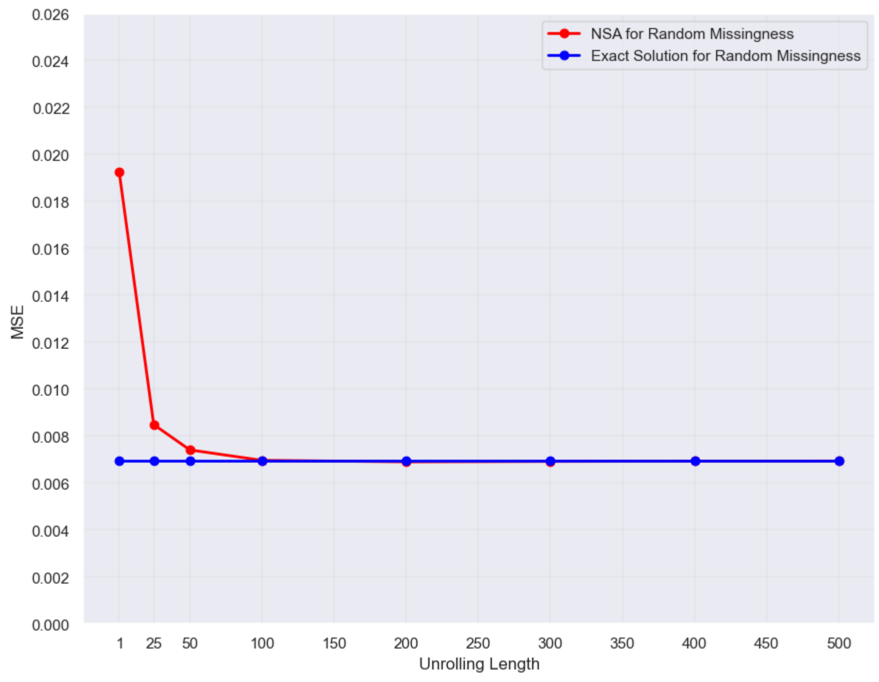


Figure 8: Reconstruction error as a function of the unrolling length of NSA vs the reconstruction error of the exact solution for random missingness on the test set.

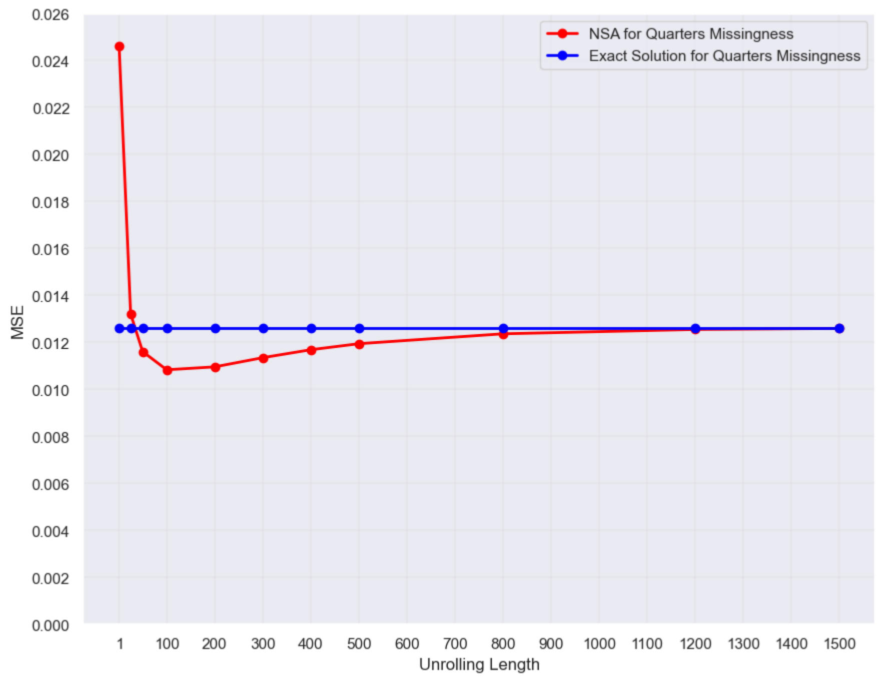


Figure 9: Reconstruction error as a function of the unrolling length of NSA vs the reconstruction error of the exact solution for the quarters missingness on the training set.

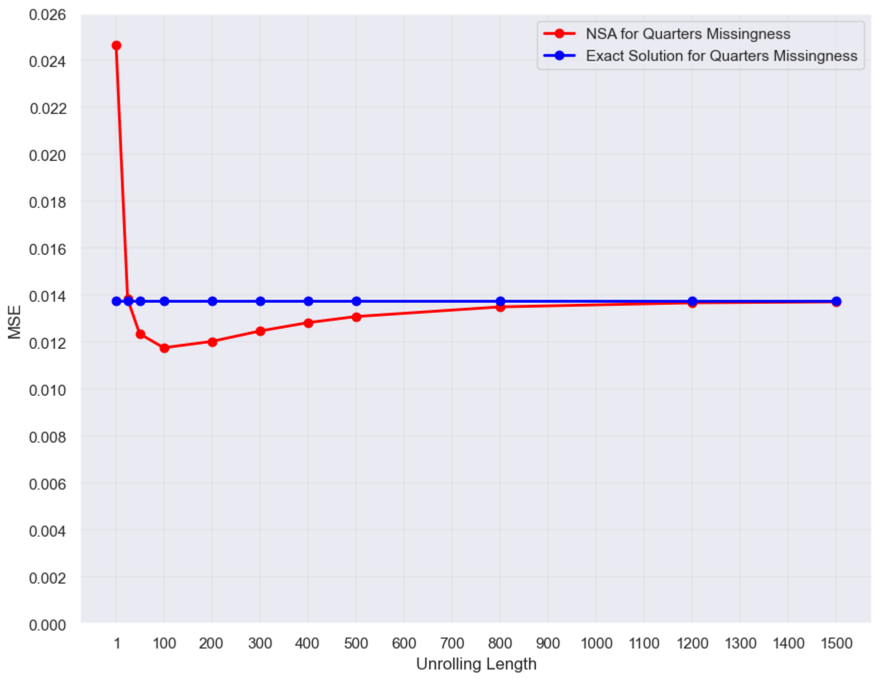


Figure 10: Reconstruction error as a function of the unrolling length of NSA vs the reconstruction error of the exact solution for the quarters missingness on the test set.

An additional thing to notice is that, even though convergence is achieved for both patterns of missingness, it appears that convergence for the quarters missingness takes substantially more steps. A way to interpret this is by studying the rate of convergence of the Neumann series. As per Equation (3.19), the rate of convergence is determined by the term $[\rho(I_K - A_s)]^l$, where l is the unrolling length. This implies that, for smaller spectral radii $\rho(I_K - A_s)$, fewer steps are required to achieve convergence. Based on this, we expect that random missingness produces matrices $I_K - A_s$ with smaller spectral radii than the quarters missingness, and therefore convergence is achieved faster. Indeed, experiments confirm this. To gain an intuitive understanding of how the spectral radii behave, in Figures (11),(12),(13) we look at $[\rho(I_K - A_s)]^l$ over the entire training set for $l = 1, 100, 1000$.

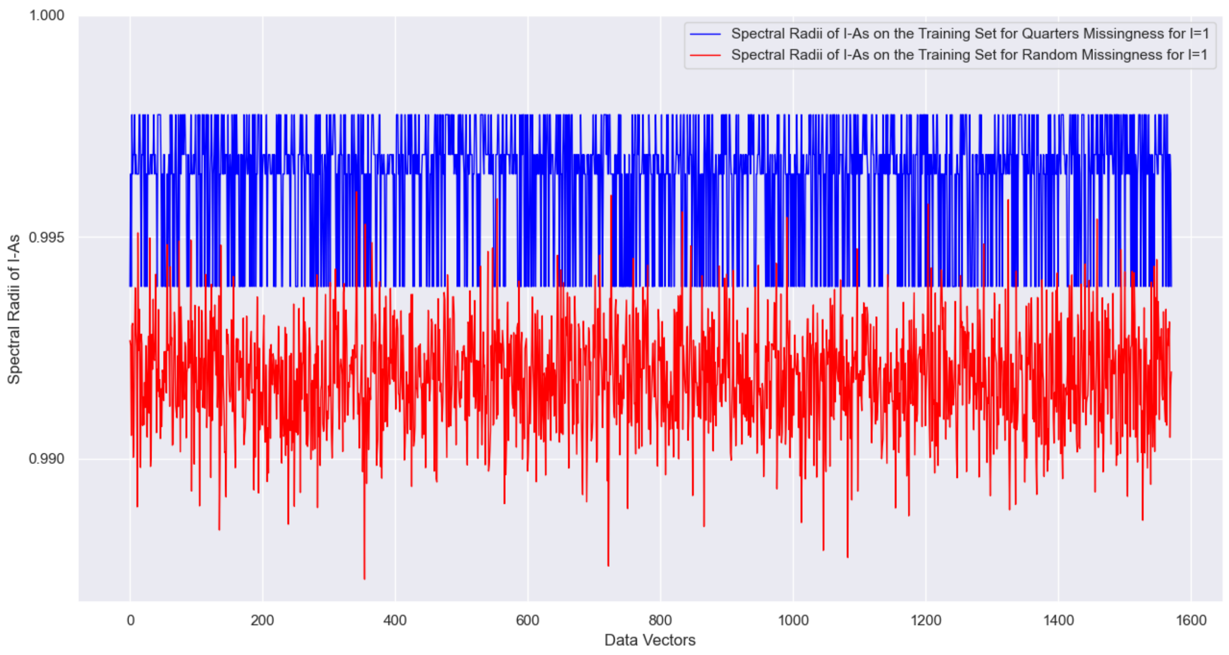


Figure 11: Behaviour of $[\rho(I_K - A_s)]^l$ for $l = 1$ for every image in the training set.

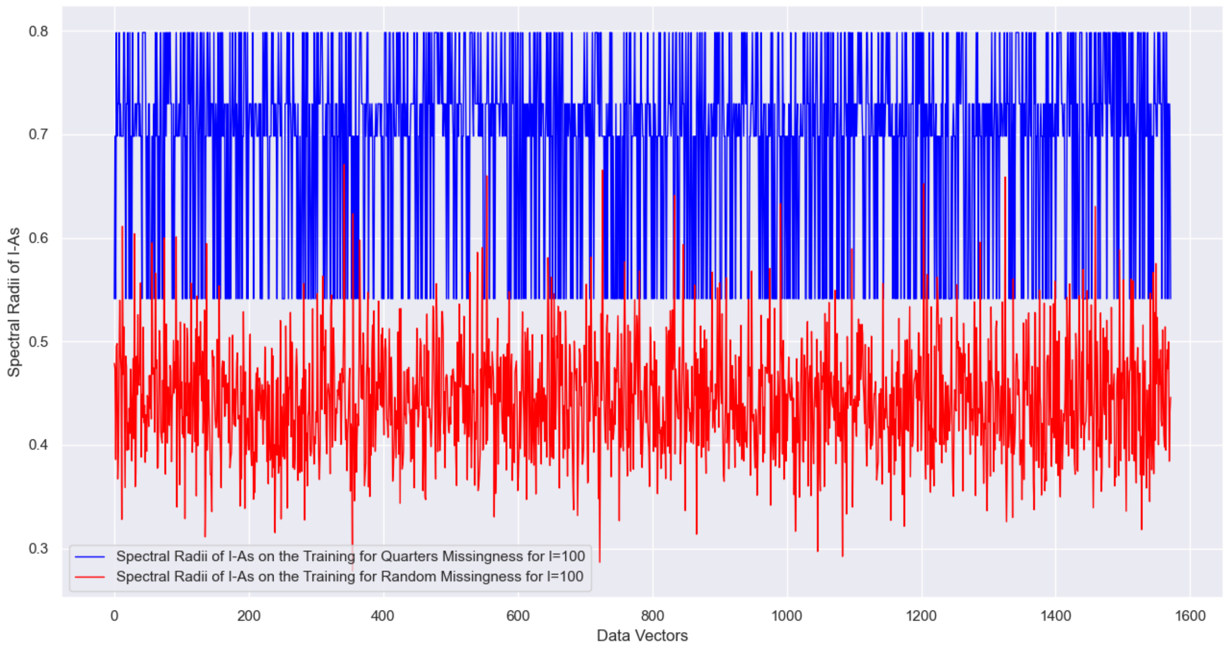


Figure 12: Behaviour of $[\rho(I_K - A_s)]^l$ for $l = 100$ for every image in the training set.

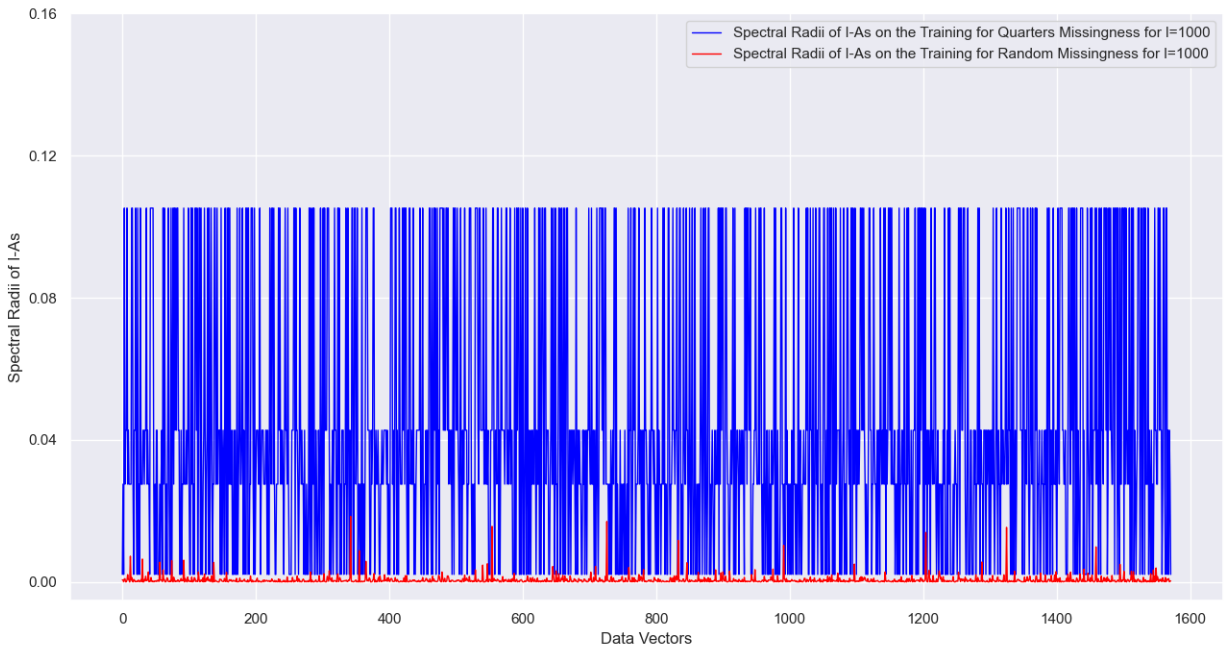


Figure 13: Behaviour of $[\rho(I_K - A_s)]^l$ for $l = 1000$ for every image in the training set.

Observe that for the quarters missingness, the spectral radii show a systematic pattern, which is to be expected because we have four missingness patterns that are repeating. Additionally, they are on average larger in magnitude. In addition, notice that as l increases the spectral radii shrink in magnitude and that the decrease happens much faster for random missingness. This becomes even more clear, if we average over the training set and study the average $[\rho(I - A_s)]$ raised to the l^{th} power, for a range of unrolling lengths. We demonstrate this in Figure (14), which shows an exponential downward trend and confirms the faster convergence rate for random missingness.

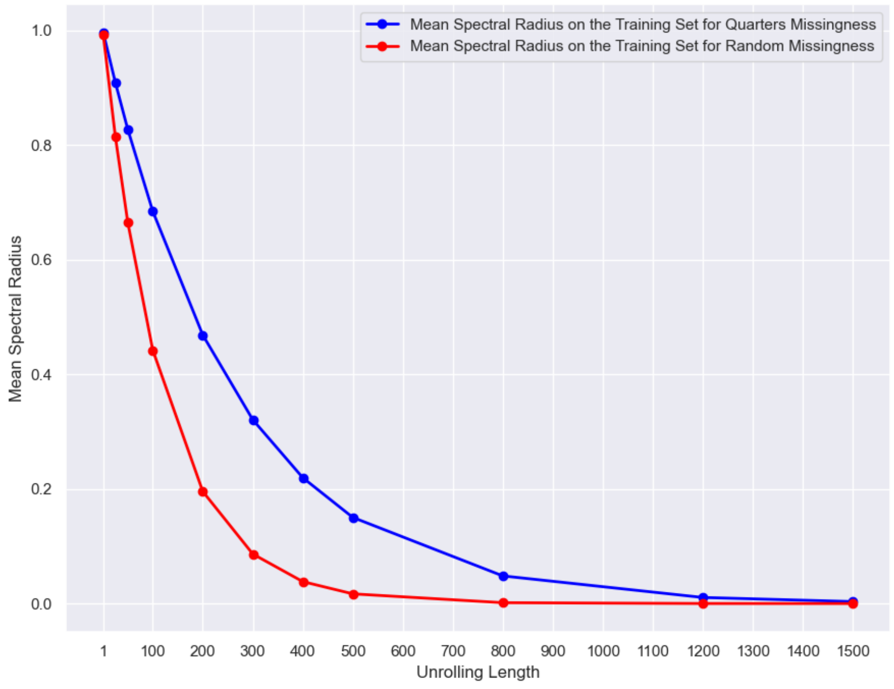


Figure 14: Rate of convergence on the training set.

An additional observation to be made in Figures (7),(8),(9),(10) is that for both patterns of missingness the qualitative behaviour is the same for the training and the test set, with both methods performing worse in the test set, as expected. Furthermore, it appears that dealing with the quarters missingness is a harder problem, as the reconstruction error is higher. A possible explanation is that this happens because we are imputing a very sparse area of the image at question, and therefore no information can be inferred from neighbouring pixels. In contrast, for random missingness, it is unlikely that a missing pixel will have all of its neighbours missing. Therefore, when we condition on the observed part of the image to compute the encoding, we obtain information through the neighbouring pixels. This comparison between random and quarters missingness, in terms of reconstruction error, is indicated in Figure (15).

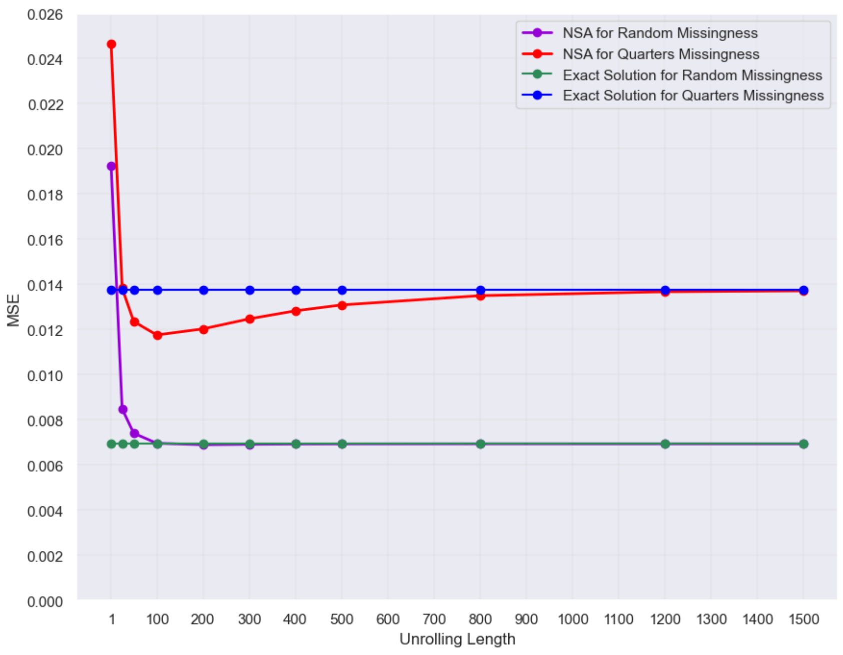


Figure 15: Random vs quarters missingness on the test set.

The fact that NSA has the capacity to produce better reconstructions than the exact solution is in fact surprising. We would like to gain some additional insight into how this occurs. For the quarters missingness, there are only four distinct missingness patterns. A good starting point is to look at the behaviour of the model for each of the four patterns, and the reconstruction error on the training and on the test set is averaged over them. This indicated in Figures (16),(17),(19),(18), where we can observe that the same qualitative behaviour does indeed persist for each quarter. There is a range of unrolling lengths (which might be different for each pattern), for which the Neumann series algorithm outperforms the exact solution. A minor difference is that the model appears to have some more difficulty imputing the right part of the image, with the reconstruction error being higher for the bottom and top right quarters.

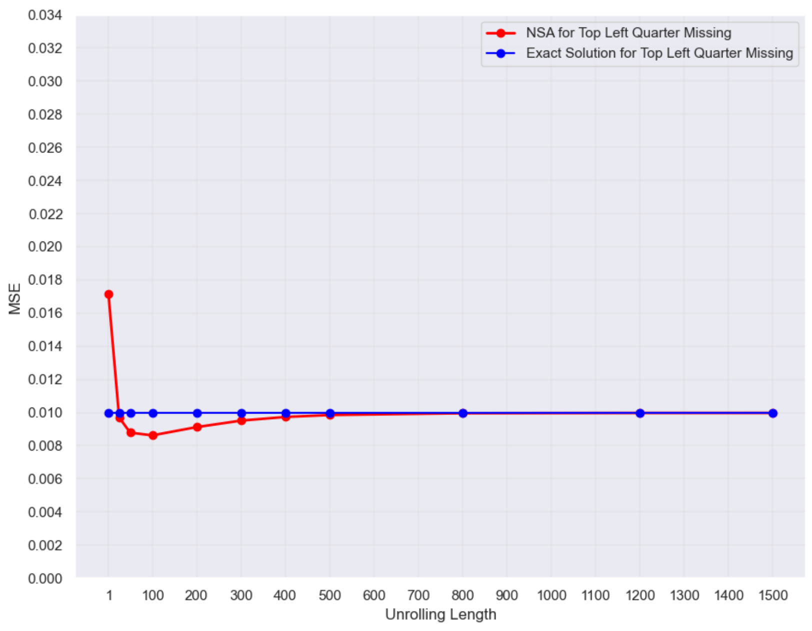


Figure 16: Reconstruction error as a function of the unrolling length of NSA vs the reconstruction error of the exact solution for the top left quarter missing on the test set.

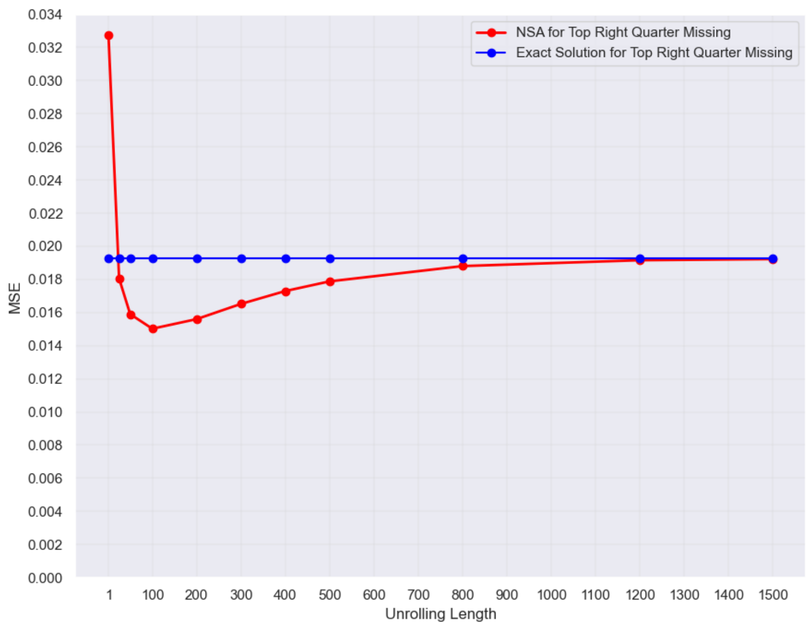


Figure 17: Reconstruction error as a function of the unrolling length of NSA vs the reconstruction error of the exact solution for the top right quarter missing on the test set.

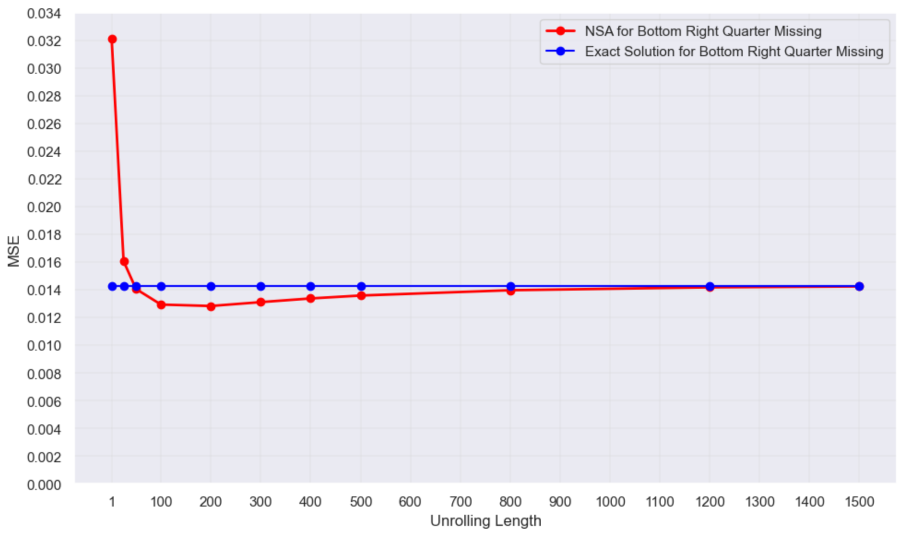


Figure 18: Reconstruction error as a function of the unrolling length of NSA vs the reconstruction error of the exact solution for the bottom right quarter missing on the test set.

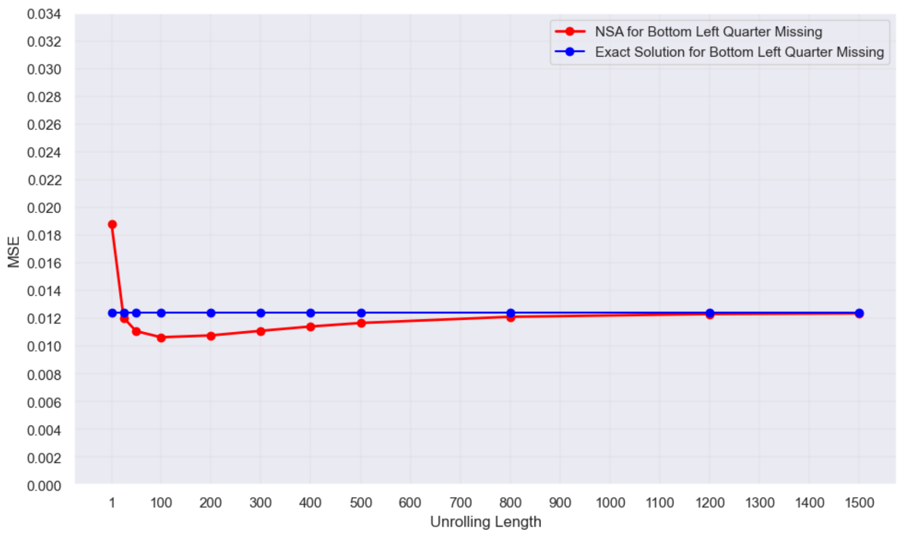


Figure 19: Reconstruction error as a function of the unrolling length of NSA vs the reconstruction error of the exact solution for the bottom left quarter missing on the test set.

Based on the performance on the test set we have chosen the Neumann series algorithm variant for $l = 100$ and compare it with the approximations of [WNN18]. The results we obtain are very similar to those of [WNN18] and are presented in Table (2). Note that if we treat l as a hyper-parameter, then a more principled approach would be to have a separate validation set, so that we do not fit the test set when we make choices related to the hyper-parameters, however as the Frey faces dataset is rather small, we are working directly with a test set.

		Imputation Methods				
		Mean Imputation	FCA	SCA	Exact Solution	NSA - 100 steps
R	Train	4.6215 ± 0.0013	1.9331 ± 0.0006	1.0648 ± 0.0003	0.6483 ± 0.003	0.6515 ± 0.003
	Test	4.6893 ± 0.0053	1.9932 ± 0.0026	1.1401 ± 0.0017	0.7146 ± 0.0016	0.7135 ± 0.0015
Q	Train	4.5877 ± 0.0018	2.7591 ± 0.0012	2.5140 ± 0.0011	1.2621 ± 0.0008	1.0619 ± 0.0007
	Test	4.7333 ± 0.0073	2.8862 ± 0.0043	2.6424 ± 0.0043	1.3741 ± 0.0040	1.1784 ± 0.0028

Table 1: Reconstruction error ($\times 10^{-2}$) with standard errors, where R and stands for random and Q stands for quarters missingness.

For random missingness, mean imputation is the worst performing method as expected, followed by FCA, SCA, the exact solution and NSA. The methods are in essence ranked by how much they respect each specific missingness pattern. Mean imputation and FCA essentially use a fixed encoder network and therefore do not respect the missingness. SCA is an intermediate solution, while NSA and the exact solution use a different encoder for each missingness pattern. Also, note that for random missingness NSA and the exact method present a very slight difference, and so NSA has essentially converged in 100 steps as is evident by Figure (7). For the quarters missingness, again mean imputation is the worst performing method, followed by FCA, SCA, the exact method and finally NSA. In this case, NSA noticeably outperforms the exact method. Additionally, in Table (2) we observe that the standard error for the exact solution is higher to that of NSA for the quarters missingness. Motivated by this, in Figures (20),(21),(22),(23) we look at the error distributions of both methods (the exact method and NSA for $l = 100$) to gain some insight on the reconstruction performance per image. All of the distributions are skewed which is to be expected, because the error cannot be lower than zero. The distributions are qualitatively similar, however it appears that for the exact method the distributions have longer tails. This is more evident on the test set, observe that the maximum error on the test set for the exact method is around 0.2 while for NSA is around 0.1. In Figures (24),(25) we look at the distribution of the difference of reconstruction errors (the error of the exact method minus the error of NSA). It is interesting to observe the difference distribution on the test set. For most images the difference falls close to zero. However the distribution has a long tail, which means that there a few *hard* examples in the test set for which NSA performs noticeably better and thus when we average to obtain the mean, the mean error for the exact method is noticeably higher. This means that NSA attains better generalisation, because it seems to handle harder data points better, where *hardness* refers to high error.

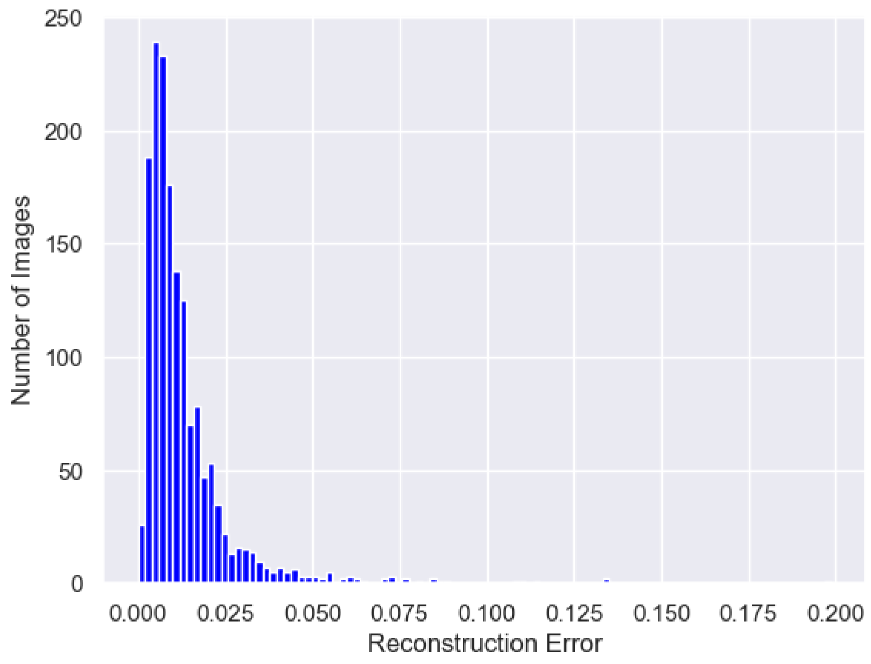


Figure 20: Reconstruction error distribution of the exact method on the training set.

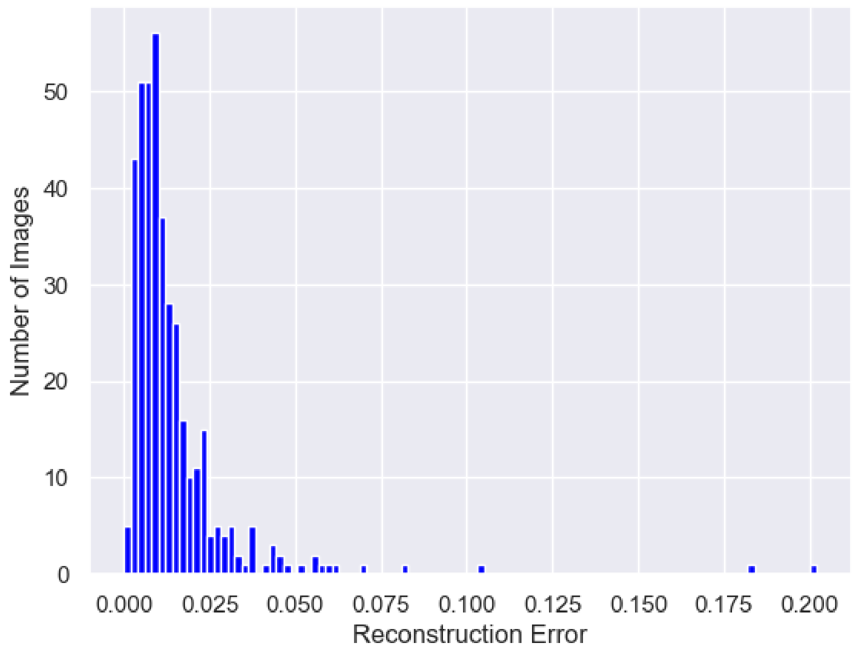


Figure 21: Reconstruction error distribution of the exact method on the test set.

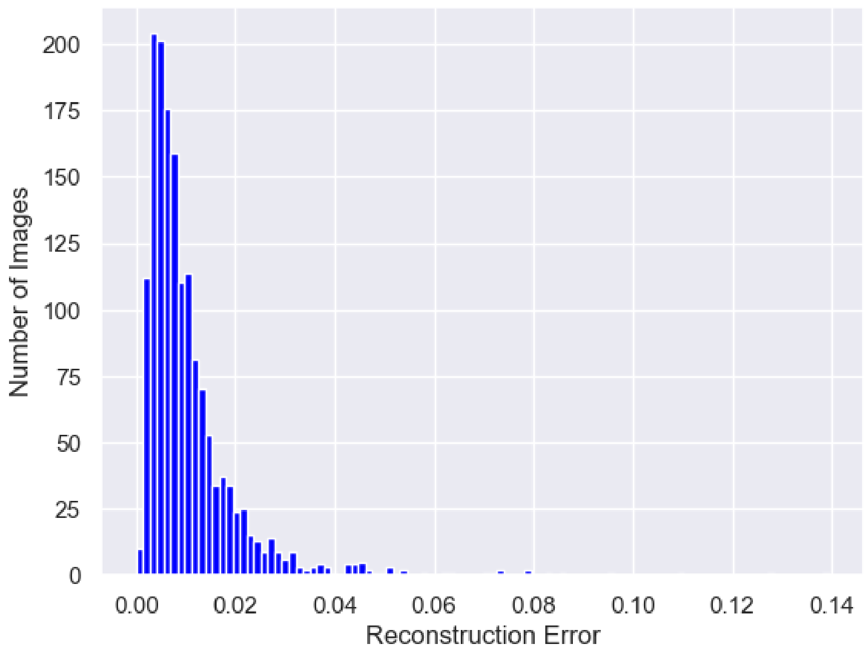


Figure 22: Reconstruction error distribution of NSA on the training set.

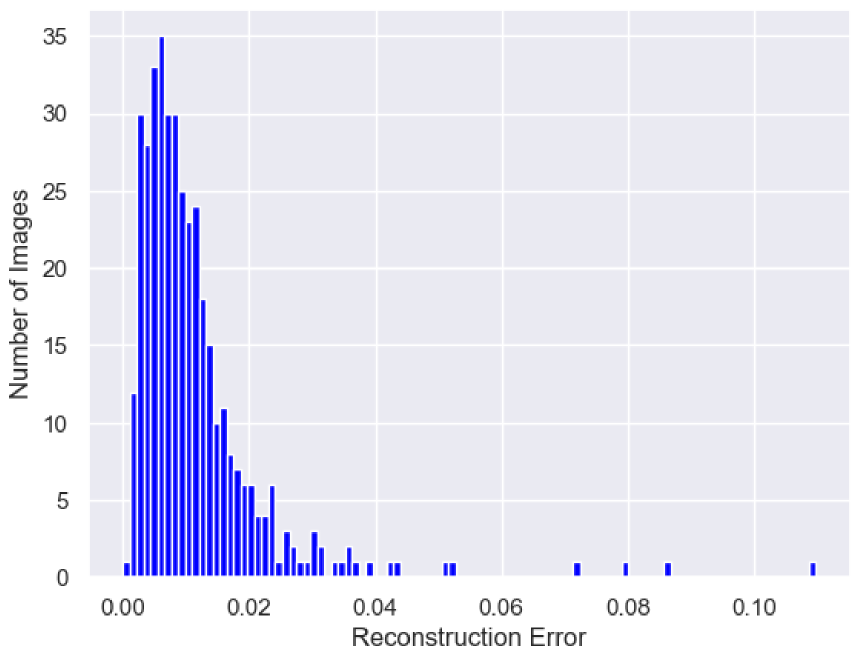


Figure 23: Reconstruction error distribution of NSA on the test set.

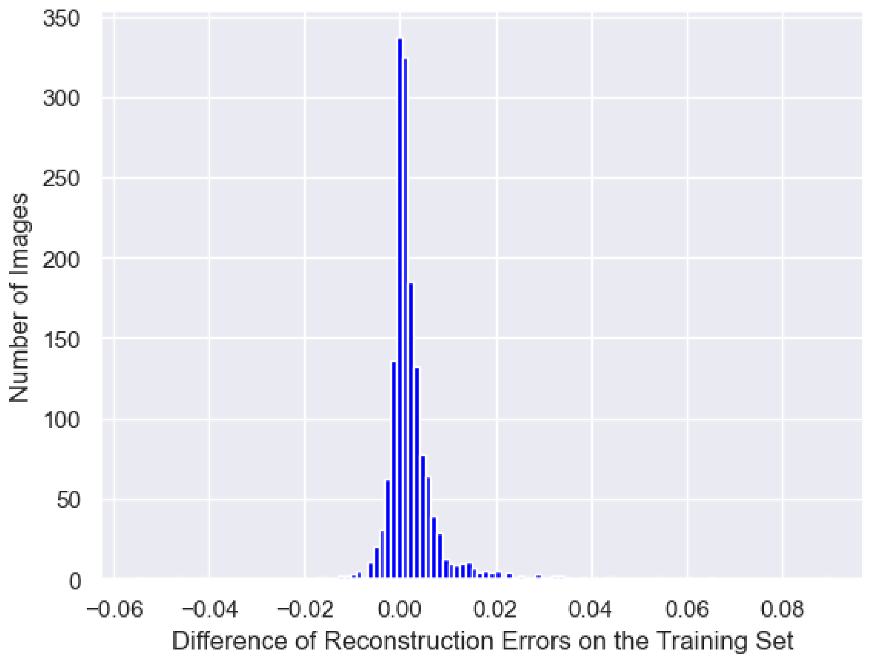


Figure 24: Distribution of the difference of the reconstruction errors on the training set.

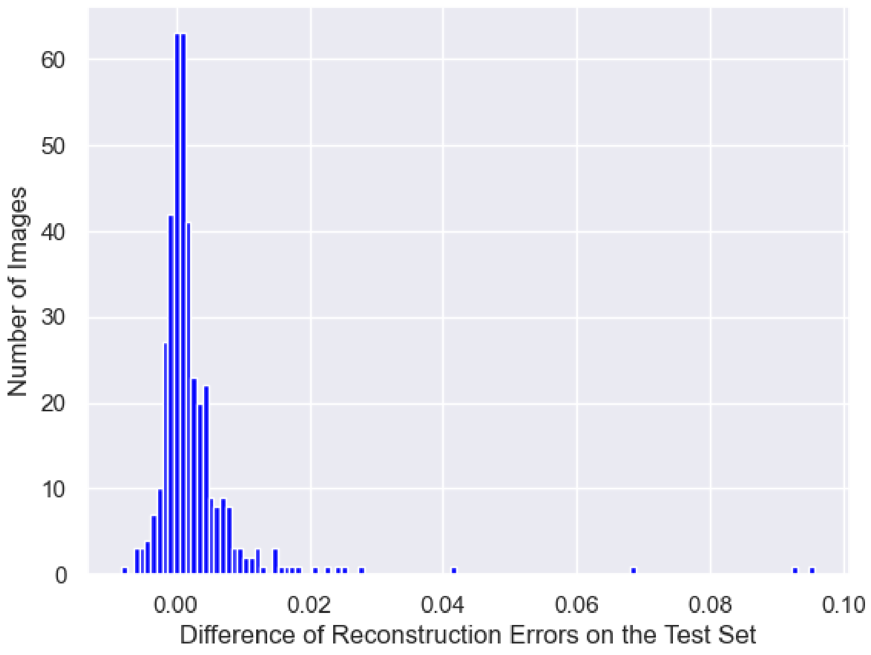


Figure 25: Distribution of the difference of the reconstruction errors on the test set.

Subsequently, in Figures (26) and (27) we present some examples of how the imputation produced by NSA looks visually, compared to the other methods. Additionally, in Figures (28),(29) we present how the imputation produced by the Neumann series looks as the step size increases for random missingness and the quarters missingness respectively. For both missingness patterns the imputation produced by NSA is very accurate as it is very close to the original image.

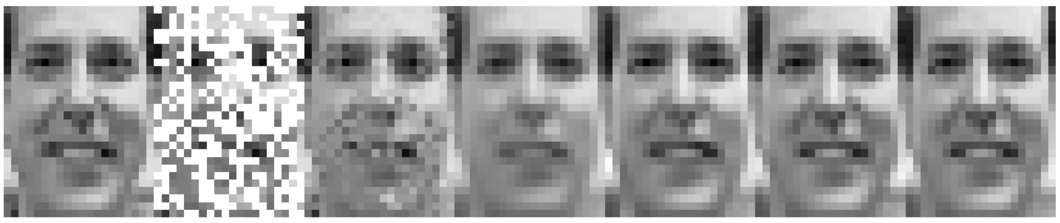


Figure 26: Imputation for random missingness: original image on the left followed by Mean Imputation, FCA, SCA, Exact Solution and NSA for $l = 100$.

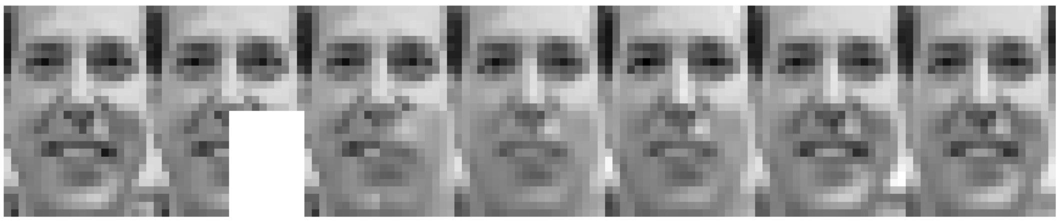


Figure 27: Imputation for the quarters missingness: original image on the left followed by Mean Imputation, FCA, SCA, Exact Solution and NSA for $l = 100$.

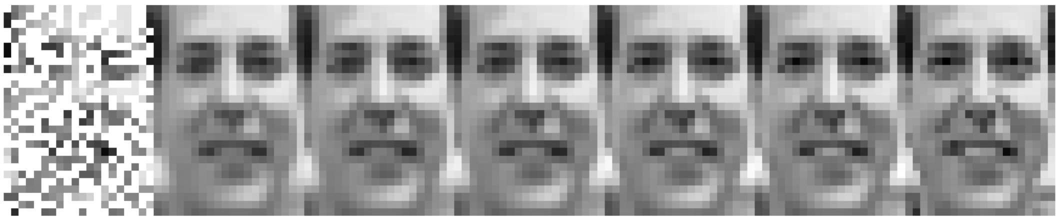


Figure 28: Improvement in NSA imputation quality as unrolling length increases for random missingness. Image with missingness on the left followed be NSA for $l = 1, 5, 10, 50, 100$ and the original image on the right.

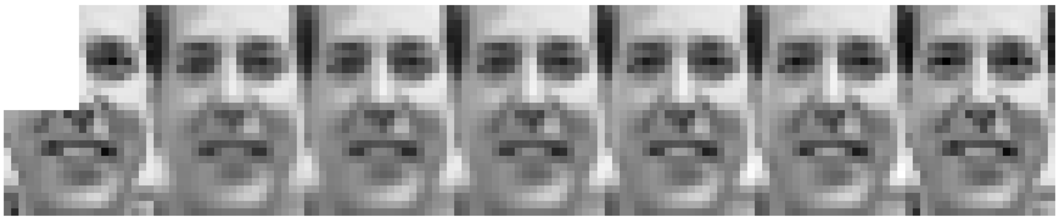


Figure 29: Improvement in NSA imputation quality as unrolling length increases for the quarters missingness. Image with missingness on the left followed be NSA for $l = 1, 5, 10, 50, 100$ and the original image on the right.

On a final note, we discuss the experimental computational complexity of our approach. The experiments were conducted on a machine with an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz processor and 16GB RAM and while keeping every other implementation detail we test the NSA against Python's solver **np.linalg.solve()**. NSA (with optimised multiplication order) requires 60 ± 13 seconds to impute both the training and the test set for both patterns of missingness while the exact method with **np.linalg.solve()** requires 170 ± 15 seconds for the same task. We hypothesise that this discrepancy occurs because not many steps are required for convergence and therefore not many matrix-vector multiplications are required. Our method thus provides benefits both in terms of accuracy and computational efficiency.

3.3 Neumann Series for Mixtures of FAs with Missing Data

In this section we introduce the Neumann series algorithm for mixtures of FAs with missing data (NSA-M). As per the discussion in Section (2.4.1) the problem we have to solve is to compute the posterior responsibility of each mixture component for each data point. More specifically, from Equation (2.46), a core part of the problem is to compute the posterior mean for each component, with some additional matrix multiplications and a determinant computation. Although, we can use the Woodbury formula and the matrix determinant lemma to perform the bulk of the computation in the latent space, i.e. working with $K \times K$ matrices instead of $D \times D$ matrices. The computation of the posterior mean for each component suffers from the same problems we have discussed for the standard FA case and so we can apply the Neumann series algorithm to perform this computation. In essence we are proposing to approximate the visible log-posterior responsibility in the following manner

$$\log r_{v,ni}^{(l)} \propto \log p^{(l)}(x_v^{(n)}|i) + \log \pi_i, \tag{3.44}$$

where

$$\log p^{(l)}(x_v^{(n)}|i) = \frac{1}{2}(x^{mi,(n)} - \mu_i)^T \Psi_i^{-1}(x^{mi,(n)} - \mu_i) - \frac{1}{2}(x^{mi,(n)} - \mu_i)^T \Psi_i^{-1} W_i \mu_{z|x_v^{(n)},i}^{(l)} - \\ -\frac{1}{2}\log|\Sigma_{v,i}| - \frac{D}{2}\log(2\pi), \tag{3.45}$$

where $x^{mi,(n)}$ symbolises the fact that we have performed mean imputation on the n^{th} data vector $x^{(n)}$ as a pre-processing step, as we have discussed this does not affect the computation. Additionally, the posterior mean of each cluster is given by the Neumann series estimate

$$\mu^{(l)}_{z|x_v,i} = \frac{1}{s_i}\left[\sum_{j=0}^{l}(I_K - A_{s,i})^j u_i\right], \tag{3.46}$$

where, the scaling factor s_i has to be computed for each component, the notation essentially follows the standard FA case. In this way, we can compute the posterior mean for each component only once, store it, and then use it for the encoding. Note that we are essentially using NSA (see Algorithm (1)), for each cluster and so the convergence of the algorithm is guaranteed by the same principles presented in Section (3.1.1). Again the model parameters are pre-trained and so are passed in as inputs in tensor form, e.g. W is an $M \times D \times K$ tensor containing the weight matrices for each cluster, Ψ is $M \times D \times D$ and μ is $M \times D \times 1$ (which can trivially be thought of as a tensor), finally $\pi = [\pi_1, ..., \pi_M]$ is $M \times 1$. We can now present the following algorithm, which computes the posterior responsibility and the encoding for each cluster.

Algorithm 2 NSA-M

```
    Input: $\mu, W, \Psi, l, M, x^{(mi)}, \pi$
    Output: $\hat{x}, r$

compute $s = [s_1, ..., s_M]$                                        ▷ this is a vector
$D \leftarrow length(x^{mi})$
for i = 1,...,M do
    $u_i \leftarrow W^T\Psi^{-1}(x^{(mi)} - \mu_i)$
    $A_{s,i} \leftarrow \frac{1}{s[i]}(I_K + W_i^T M \Psi_i^{-1} M W_i)$                ▷ $M \times K \times K$
    $y_{0,i} \leftarrow u_i$
    $d_i \leftarrow det(I_K + W_i^T M \Psi_i^{-1} M W_i) det(\Psi_{i,v})$
    while count< $l$ do
        $y_i \leftarrow (I - A_{s,i})y_{0,i} + u_i$
        $y_{0,i} \leftarrow y_i$
        count ← count +1
    end while
    $\mu_{z|x_v,i} \leftarrow \frac{1}{s_i} y_i$
    $\hat{x_i} \leftarrow W_i\mu_{z|x_v,i} + \mu_i$
    $t1_i \leftarrow \frac{1}{2}(x^{mi} - \mu_i)^T\Psi_i^{-1}(x^{mi} - \mu_i)$              ▷ temporary variables
    $t2_i \leftarrow \frac{1}{2}(x^{mi} - \mu_i)^T\Psi_i^{-1}W_i\mu_{z|x_v,i}$
    $\log r_i \leftarrow t1_i - t2_i - \frac{1}{2}\log d_i - \frac{D}{2}\log(2\pi) + \log \pi_i$
end for
$r \leftarrow \exp(\log r)$                                           ▷ this is a vector
normalise $r$
return $\hat{x}, r$                          ▷ $\hat{x}$,r are vectors comprised of $\hat{x}_i, r_i$
```

3.4 Experiments for NSA-M

We are working with a subset of the MNIST dataset which consists of 28×28 images of handwritten digits. The full dataset has a training set of 60,000 examples, and a test set of 10,000 examples. We are working with a smaller subset of 4000 images of the digits one and seven, 80% is used as training set and the rest are kept as a test set. This decision is made to reduce computation time. The full dataset may be found at `(http://yann.lecun.com/exdb/mnist/)`.

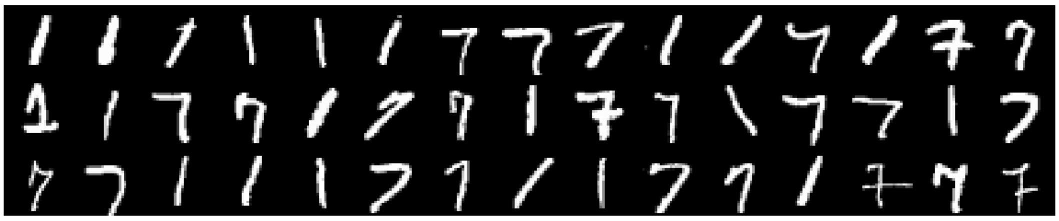


Figure 30: Examples of ones and sevens from the MNIST dataset.

We are training a mixture of PPCAs, see [TB99a] as an instance of a mixture of FAs. Again this decision is motivated by the fact that PPCA is more convenient to implement. We are using $M = 2$, i.e. we are fitting two mixture components. For the dimension of the latent space we are using $q = 100$ latent components for each cluster. To initialise the model we are using the k-means algorithm, see e.g. [Llo82]. The experimental task we are examining is image in-painting. We are again using the random and quarters missingness patterns, but this time we are adding an additional missingness pattern which we call the halfs missingness. Essentially, for the halfs missingness we remove the top half of the image. This pattern is interesting for this specific dataset because it allows us to study the uncertainty of the model, e.g. we may have an image for which the posterior responsibilities are close to being equal. For example, see Figure (33), for some images we cannot easily tell if we are dealing with a one or a seven just by looking at the bottom half. It is interesting to see how each cluster imputes such images. We test the performance of NSA-M against the exact solution for the mixture model (see Section (2.4.1)), NSA with $l = 300$ and the exact solution for standard FA, for both of these methods we are using $q = 100$ latent components. This comparison also allows us to check if fitting more than one clusters improves reconstruction performance. Again we are using the mean squared error as a measure for reconstruction quality.

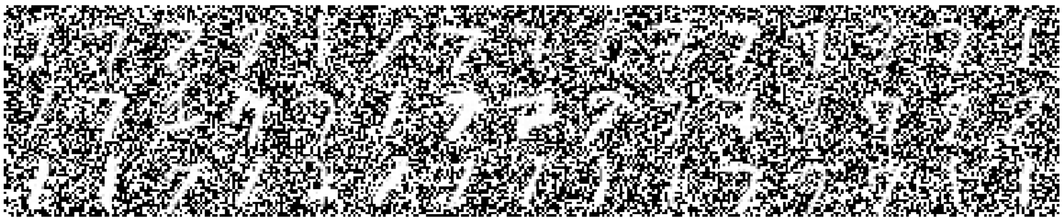


Figure 31: Examples of random missingness on the MNIST dataset.

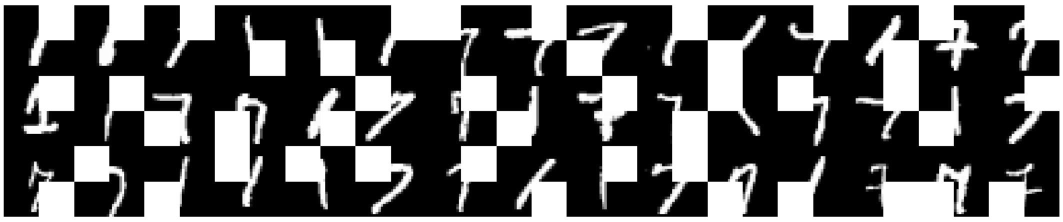


Figure 32: Examples of the quarters missingness on the MNIST dataset.

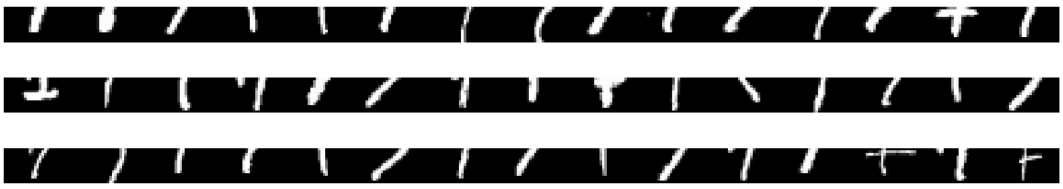


Figure 33: Examples of the halfs missingness on the MNIST dataset.

3.4.1 Results

In Figures (34),(35) we observe the fitted means of the two clusters. We can see that the first mixture component is modelling the distribution of the digit one, while the second component is modelling the distribution of the digit seven.

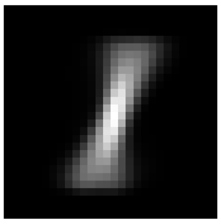


Figure 34: Fitted mean of component 1.

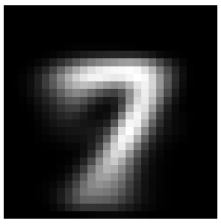


Figure 35: Fitted mean of component 2.

We are interested in seeing how the reconstruction error of NSA-M evolves as a function of the unrolling length. Note that the reconstruction error for a specific image is given as the minimum reconstruction error of the two clusters. The error as a function of unrolling length is showcased for all missingness patters in Figures (36),(37),(38),(39),(40),(41).

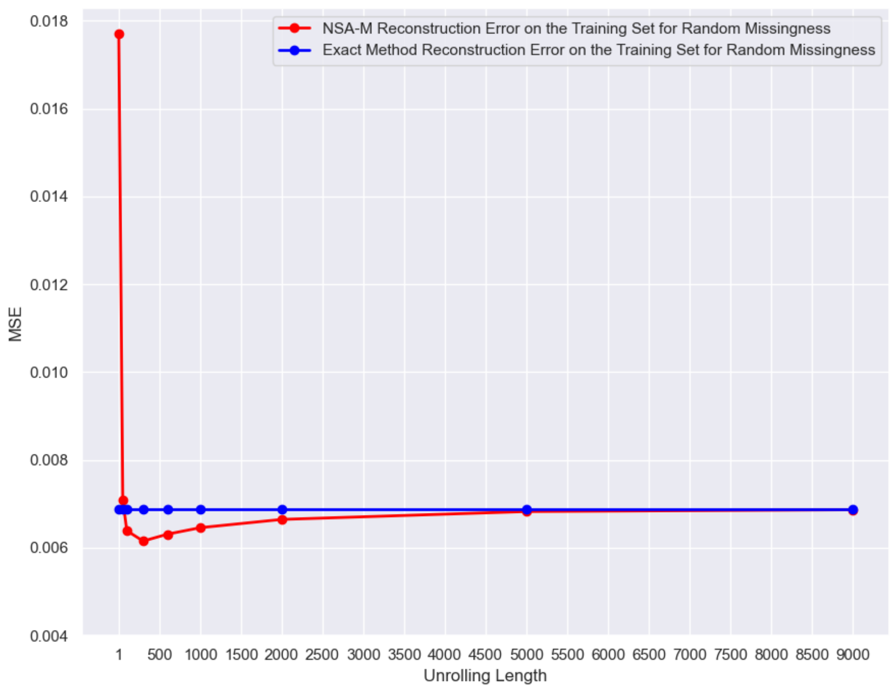


Figure 36: Reconstruction error as a function of the unrolling length of NSA-M vs the reconstruction error of the exact solution for random missingness on the training set.

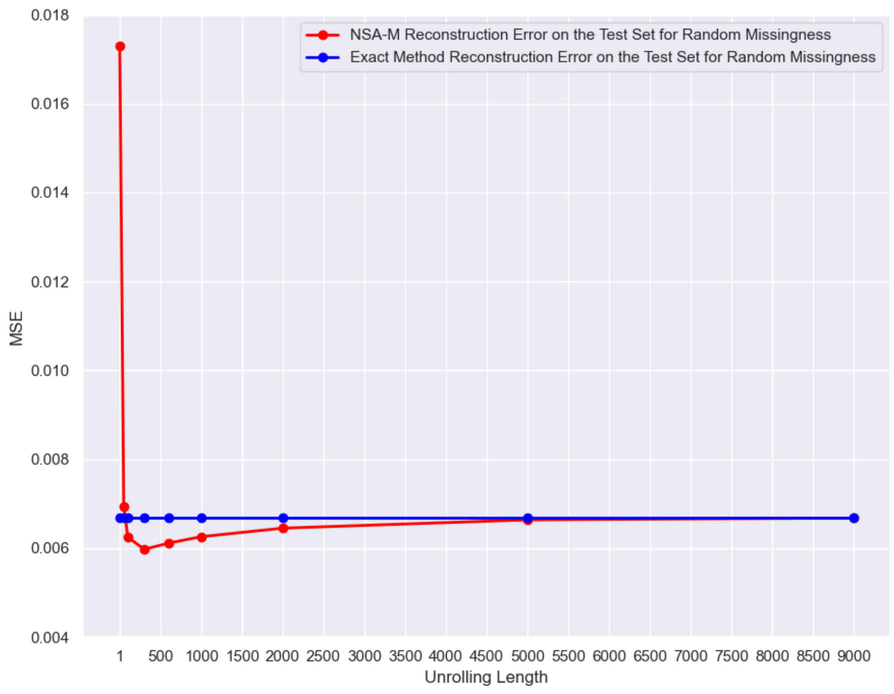


Figure 37: Reconstruction error as a function of the unrolling length of NSA-M vs the reconstruction error of the exact solution for random missingness on the test set.

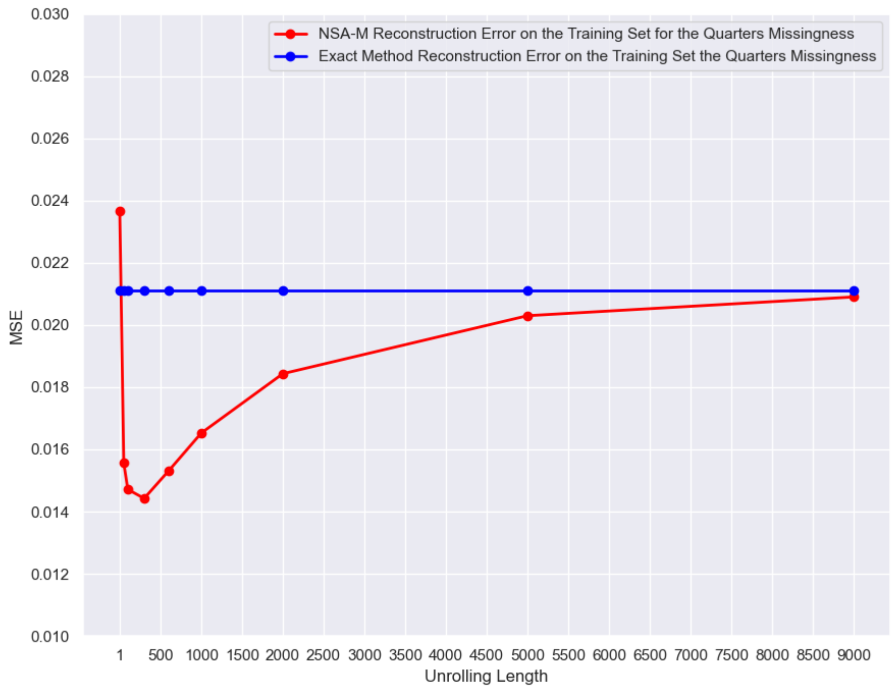


Figure 38: Reconstruction error as a function of the unrolling length of NSA-M vs the reconstruction error of the exact solution for the quarters missingness on the training set.

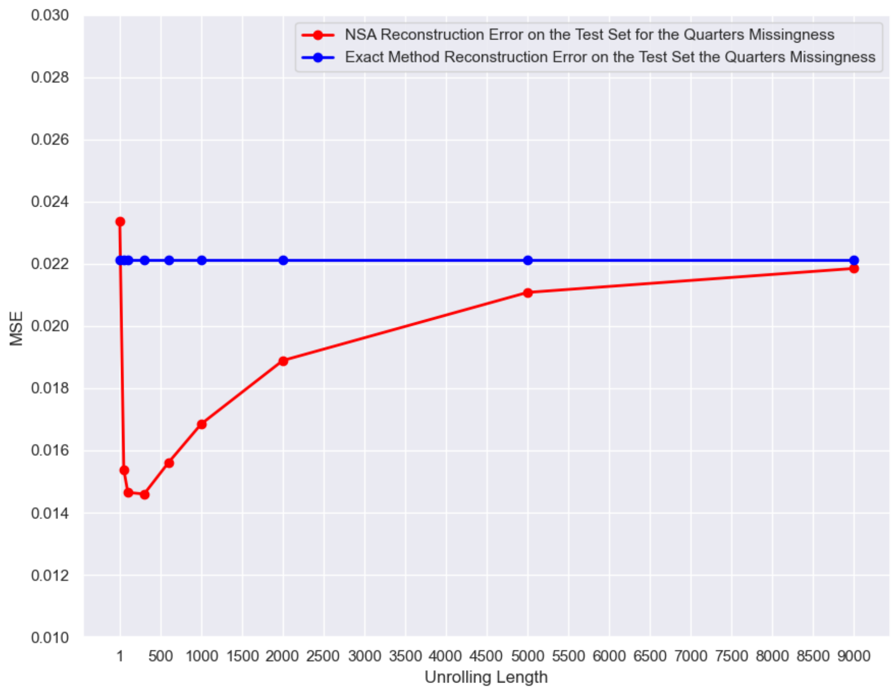


Figure 39: Reconstruction error as a function of the unrolling length of NSA-M vs the reconstruction error of the exact solution for the quarters missingness on the test set.

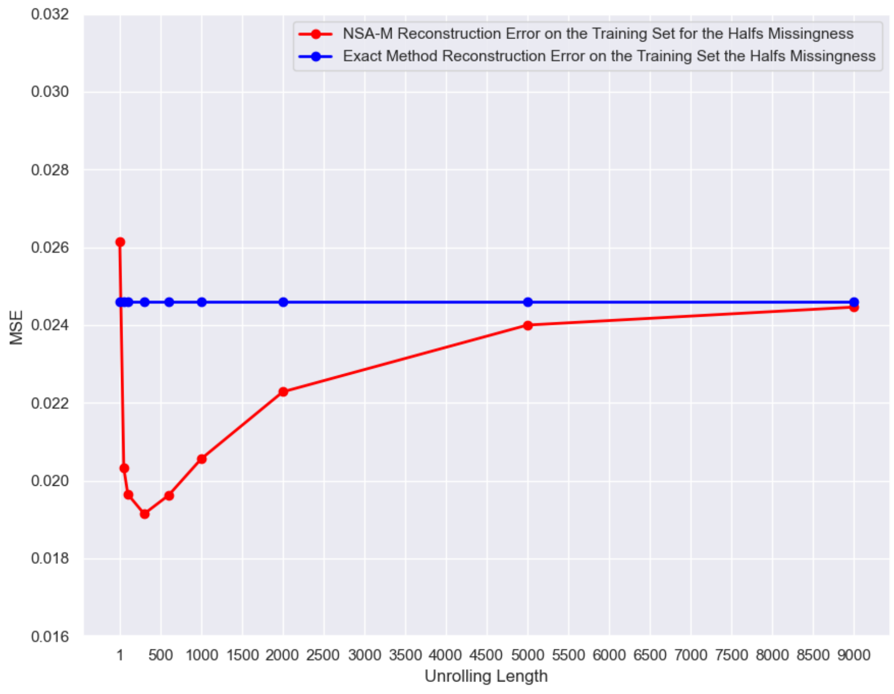


Figure 40: Reconstruction error as a function of the unrolling length of NSA-M vs the reconstruction error of the exact solution for the halfs missingness on the training set.

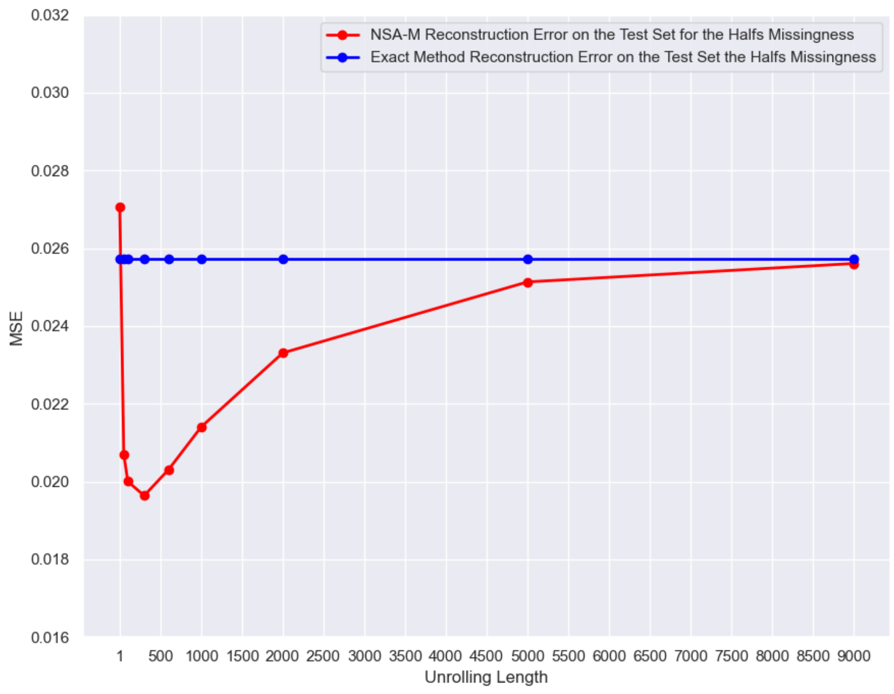


Figure 41: Reconstruction error as a function of the unrolling length of NSA-M vs the reconstruction error of the exact solution for the halfs missingness on the test set.

The first thing to observe is that the qualitative behaviour is very similar to what we observe in the standard Factor Analysis case, where there is a range of unrolling lengths, before convergence is achieved, for which NSA-M outperforms the exact solution. However, it appears that the difference between the best performing variant of the NSA-M and the exact method is now more extreme than in the standard FA case. Also observe that in this case NSA-M outperforms the exact solution for the random missingness pattern as well. Additionally, it appears that the hardest problem in terms of reconstruction is imputing the halfs missingness, followed by the quarters missingness and finally random missingness. It seems that large sparse continuous missing areas make imputation harder. Notice for example that in an image with random missingness roughly half of the pixels will be missing, which means that we have about the same number of missing pixels as in the halfs missingness. However, imputing an image with the top half missing is a harder problem because excluding the boundary of the missing area, for every other missing pixel its neighbours are also missing and so no information can be inferred from them. The fact that the values of neighbouring pixels are not independent makes random missingness an easier problem. The comparison between missingness patterns on the training and the test set is indicated in Figures (42),(43). Another interesting observation, is that convergence takes many more steps than in the standard FA case to be achieved. In Figure (44) we look at the rate of convergence across missingness patterns. Convergence is slower for the halfs pattern, followed by the quarters and finally random missingness. This occurs because the halfs missingness produces matrices $I_K - A_s$ with larger spectral radii on average (we have discussed this argument in Section 3.2.1).

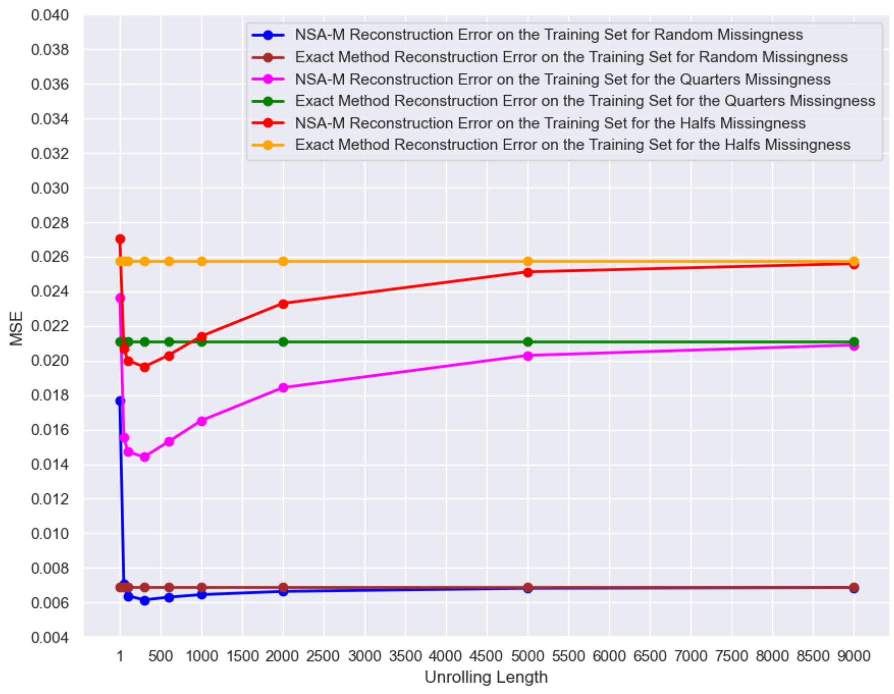


Figure 42: Reconstruction error comparison for the three missingness patterns on the training set.

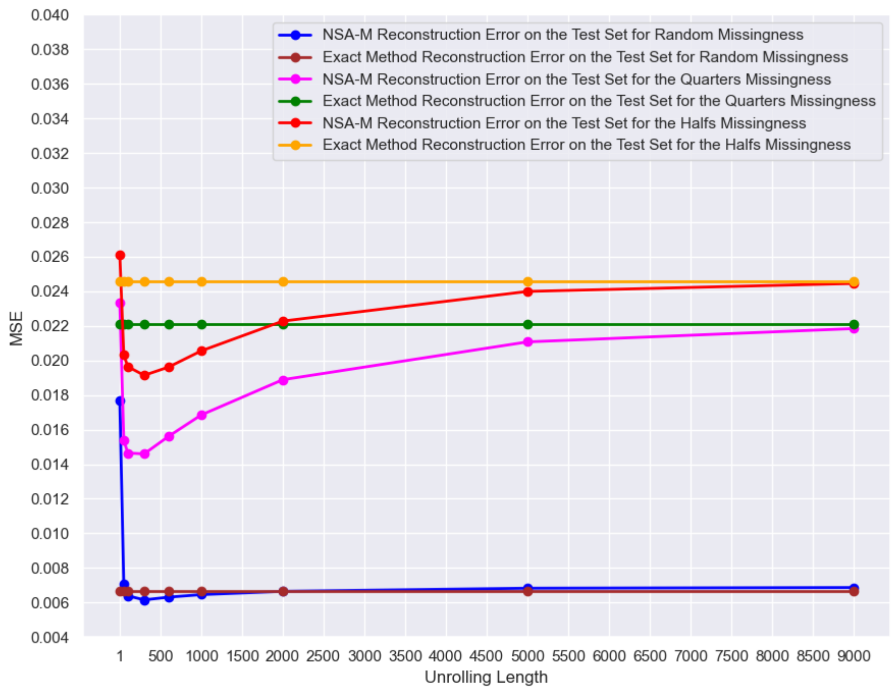


Figure 43: Reconstruction error comparison for the three missingness patterns on the test set.

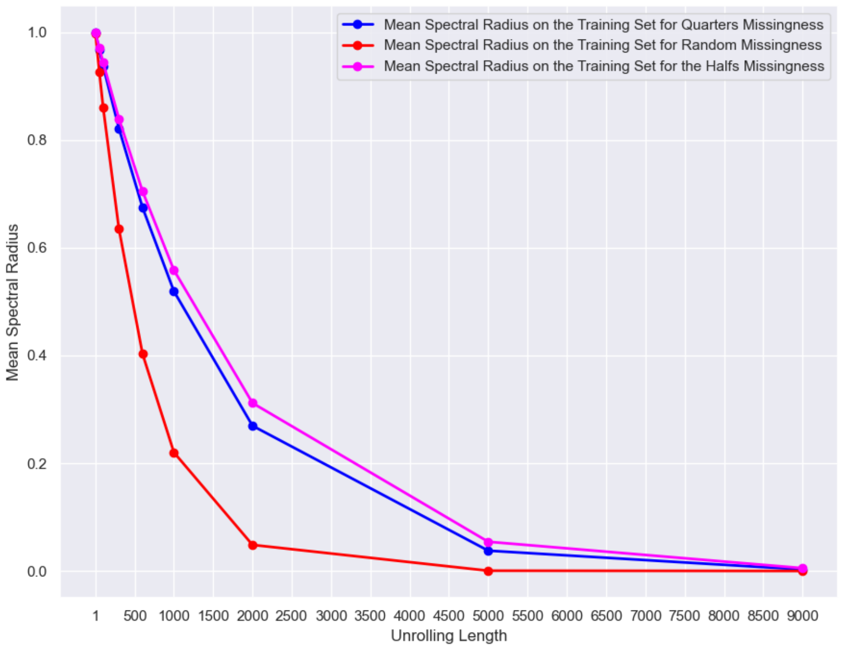


Figure 44: Rate of convergence per missingness pattern on the training set.

The best performing variant of NSA-M on the test occurs for $l = 300$. In Table 2 we compare it with the exact method for the mixture of FAs, NSA for $l = 300$ and the exact method for the standard FA, which we term the standard exact solution for convenience. With this comparison we aim to see if fitting more mixture components leads to a better reconstruction than just fitting a global linear model such as FA.

		Imputation Methods			
		Standard Exact Sol.	NSA 300 Steps	Exact Sol. - Mixture	NSA-M 300 Steps
R	Train	0.7688 ± 0.0001	0.7515 ± 0.0001	0.6685 ± 0.0001	0.5953 ± 0.0002
	Test	0.8256 ± 0.0005	0.8225 ± 0.0007	0.6865 ± 0.0001	0.6146 ± 0.0009
Q	Train	2.2619 ± 0.0006	2.2286 ± 0.0005	2.1240 ± 0.0006	1.4421 ± 0.0004
	Test	2.2944 ± 0.0029	2.2641 ± 0.0027	2.2214 ± 0.0029	1.5141 ± 0.0019
H	Train	2.5537 ± 0.0006	2.4284 ± 0.0005	2.4030 ± 0.0006	1.9621 ± 0.0006
	Test	2.8242 ± 0.0030	2.6842 ± 0.0040	2.5714 ± 0.0023	2.0741 ± 0.0025

Table 2: Reconstruction error ($\times 10^{-2}$) with standard errors, where R stands for random missingness, Q stands for quarters missingness, H stands for halfs missingness.

We can observe that NSA-M for $l = 300$ is the best performing method in terms of reconstruction error, followed by the exact solution for mixtures, NSA for $l = 300$ and finally the standard exact solution. Therefore, we can say that increasing the number of components leads to better reconstruction performance. Additionally, in Figures (45),(46),(47),(48),(49),(50) we look at the distribution of the difference of reconstruction errors for each missingness pattern on the training set and on the test set. Again we are interested in studying how reconstruction performance varies per image. Observe that the distributions for the test set have a longer tail, presenting similar behaviour to the standard FA case.

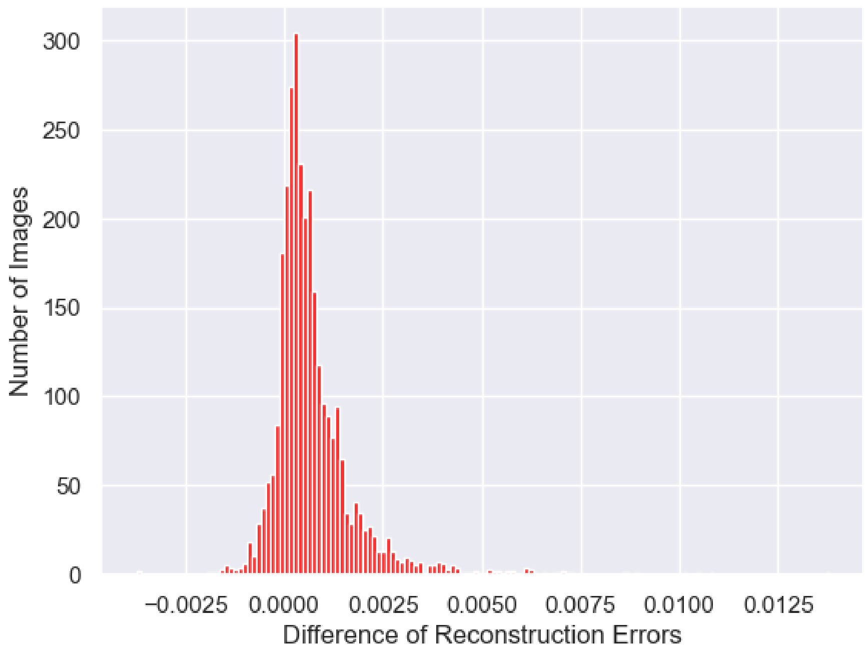


Figure 45: Difference of reconstruction errors on the training set for random missingness.

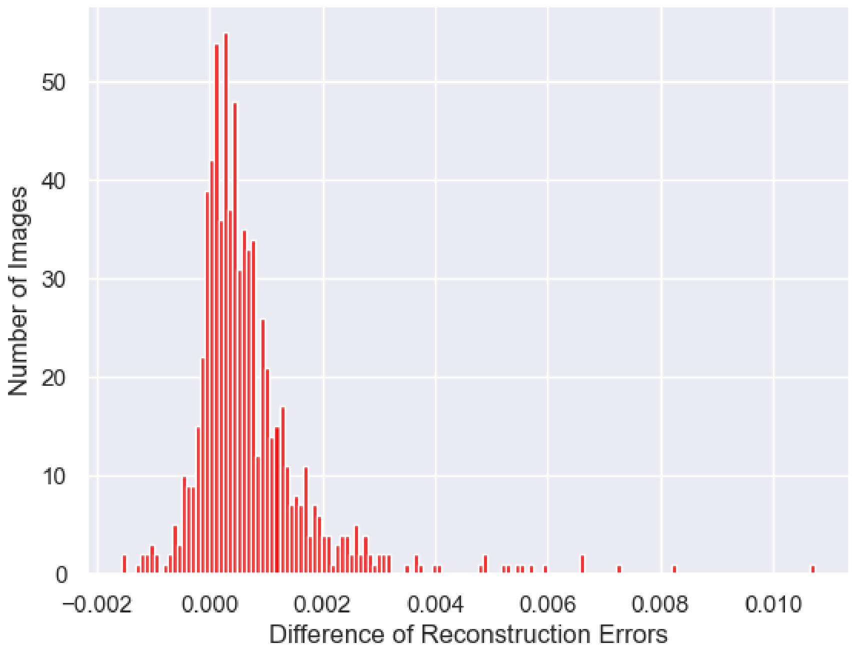


Figure 46: Difference of reconstruction errors on the test set for random missingness.

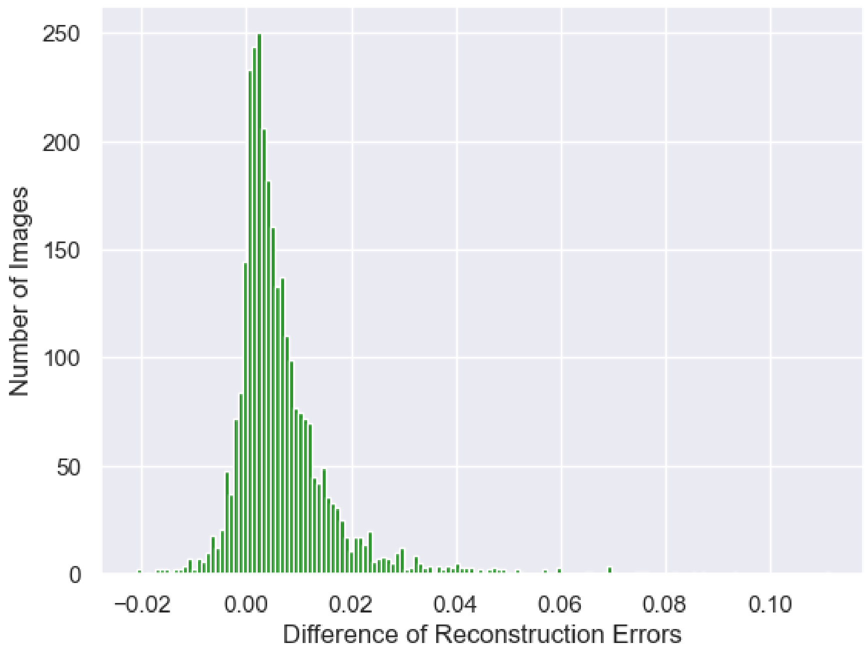


Figure 47: Difference of reconstruction errors on the training set for the quarters missingness.

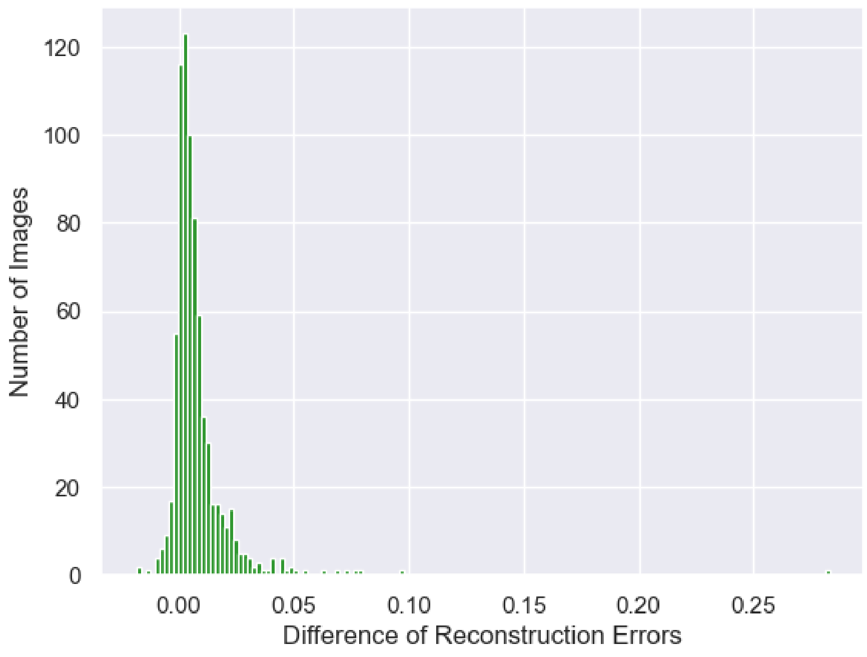


Figure 48: Difference of reconstruction errors on the test set for the quarters missingness.

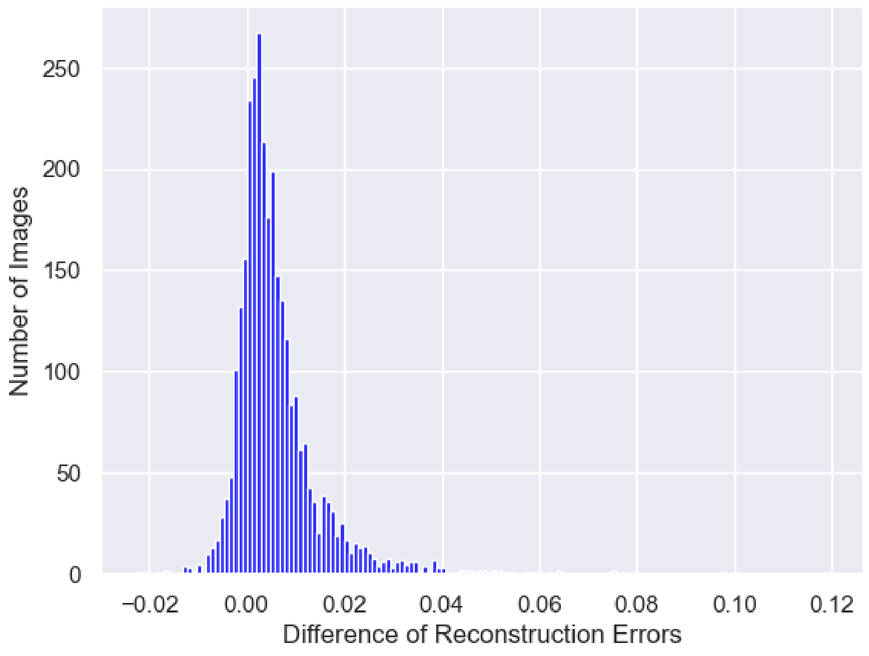


Figure 49: Difference of reconstruction errors on the training set for the halfs missingness.

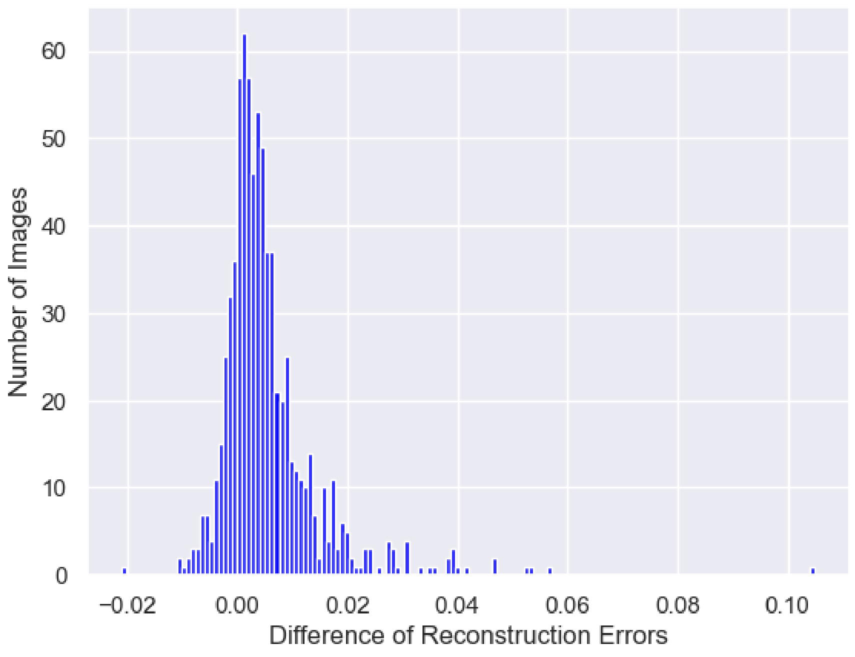
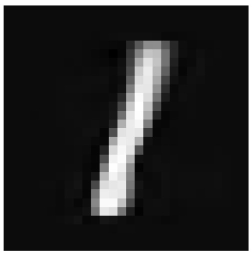


Figure 50: Difference of reconstruction errors on the test set for the halfs missingness.

Furthermore, we wish to study the behaviour of the posterior responsibilities computed by our model. Particularly, we look at images with the top half missing for which the posterior responsibilities for each cluster are close in magnitude. For example, see Figure (51). Observe how for the same image, both clusters can produce realistic imputations (the blurriness of the images can be attributed to the fact that our reconstructions are lossy because of the projection in the latent space). An interesting experiment is how frequently the NSA-M model guesses right about cluster membership, when there is halfs missingness. We check how frequently a true seven is assigned to cluster 2 via maximum posterior responsibility and respectively for true ones, where the characterisation true refers to the true label of each image. We find that the model is right 78% of the time on the training set and 74% on the test set. This means that, e.g. for the test set the model reconstructs the half images as the correct digit about 74% of the time.


(a) Imputation produced by cluster 1.

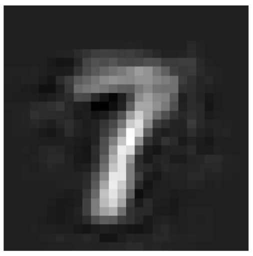

(b) Imputation produced by cluster 2.

Figure 51: A true one imputed by both clusters

In Figures (52),(53),(54) we present how the imputations look visually for each of the three missingness patterns. Additionally, in Figure (55) we show the improvement in imputation quality of NSA-M as the unrolling length increases.

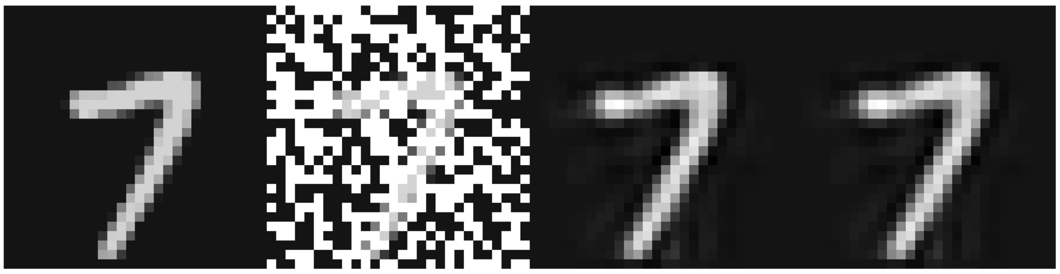


Figure 52: Imputation for random missingness: original image on the left followed by the image with missingness, exact method and NSA-M.

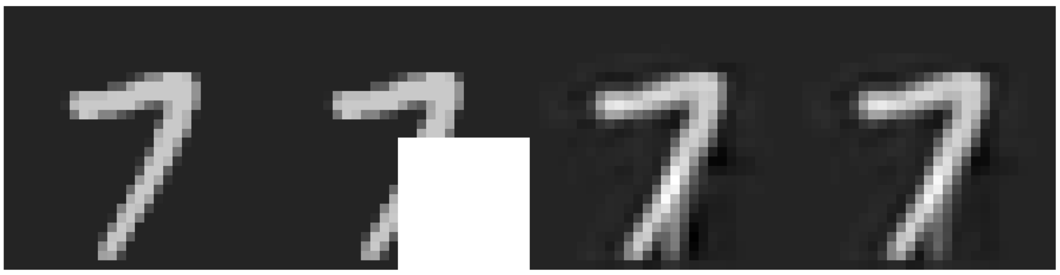


Figure 53: Imputation for the quarters missingness: original image on the left followed by the image with missingness, exact method and NSA-M

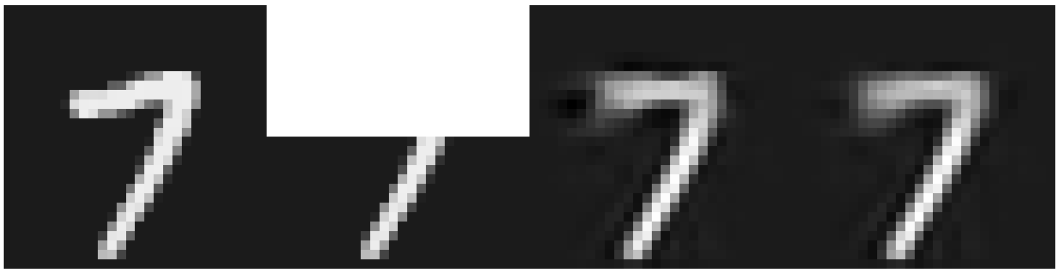


Figure 54: Imputation for the halfs missingness: original image on the left followed by the image with missingness, exact method and NSA-M

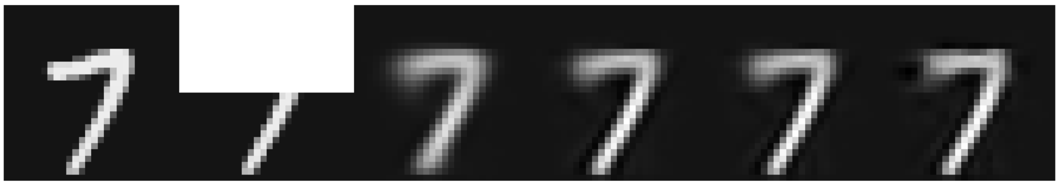


Figure 55: Improvement in the imputation quality for NSA-M: original image on the left followed by the image with missingness and NSA-M for $l = 1, 50, 100, 300$.

On a final note, the run-time was similar for both methods and no significant discrepancy was observed. This may be due to the fact that the method is taking many steps to converge. If the number of steps required is large then in practice the cost of our method will eventually be close to cubic, and so in regard to computational complexity there is not much gain.

4 Conclusion

Our primary objective with this project was to test if using Neumann series is a viable approach in the context of our problem. This objective has largely been fulfilled. We have given a theoretically justified algorithm that produces reconstructions of higher quality with better computational complexity, than the exact method of Williams et al. which is its direct counterpart. We have mainly examined the discrepancy in performance, between the two methods, in an experimental manner. However, a conclusive mathematical argument explaining why the discrepancy occurs has eluded us. This is the most important question that has been left open by our research. Consequently, it is a good starting point for future research. In addition, as a secondary objective, we wanted to explore if the Neumann series approach we developed can be extended in any way to accommodate more flexible models than the standard Factor Analysis model. This objective has been partly fulfilled with the development of NSA-M. An initial direction we took with this question was to look at nonlinear autoencoder architectures and try to fit the Neumann series approach in that framework. However, it seems that this idea is a dead-end, because the Neumann series is an approximator for linear operators and therefore an extension to the nonlinear is not theoretically justified. The NSA-M extension we have presented is a globally nonlinear model but each of its components is linear, and so we do not consider this an extension to the full nonlinear case. It could more accurately be described as a piece-wise linear model. A more promising approach for future research regarding the nonlinear case, would be to apply a decomposition tailored to nonlinear operators, e.g. the Adomian decomposition see [Gab94]. Another limitation of our model is that it only covers the MAR case, so a final avenue for future expansions could be to try an extension to the harder MNAR case, Le Morvan et al. [LMJM$^+$20] have worked with a similar problem, which suggest that an extension might indeed be possible. However, in that case, the exact solution presented in our Section (2) would have to be re-derived from the start. This is because, it is based on the premise that the missingness is MAR. A naive first step in that direction could be to apply the method anyway and examine how extreme the bias is in the results.

References

[Bar12] David Barber. *Bayesian Reasoning and Machine Learning.* Cambridge University Press, 2012.

[BFB15] Richard L. Burden, J. Douglas Faires, and Annette M. Burden. *Numerical Analysis.* Cengage Learning, 2015.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*, volume 4. Springer, 2006.

[BKM11] David J. Bartholomew, Martin Knott, and Irini Moustaki. *Latent Variable Models and Factor Analysis: A Unified Approach*, volume 904. John Wiley & Sons, 2011.

[CNW20] Mark Collier, Alfredo Nazabal, and Christopher K. I. Williams. VAEs in the Presence of Missing Data. *arXiv preprint arXiv:2006.05301*, 2020.

[DLR77] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society: Series B (methodological)*, 39(1):1–22, 1977.

[DS88] Nelson Dunford and Jacob T. Schwartz. *Linear Operators, Part 1: General Theory*, volume 10. John Wiley & Sons, 1988.

[Gab94] L. Gabet. The Theoretical Foundation of the Adomian Method. *Computers & Mathematics with Applications*, 27(12):41–52, 1994.

[GJ94] Zoubin Ghahramani and Michael I. Jordan. Learning from Incomplete Data. *Technical report CBCL-108, Massachusetts Institute of Technology*, 1994.

[GOW19] Davis Gilton, Greg Ongie, and Rebecca Willett. Learning to Regularize Using Neumann Networks. In *2019 IEEE Data Science Workshop (DSW)*, pages 201–207, 2019.

[GPAM$^+$20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *Communications of the ACM*, 63(11):139–144, 2020.

[GVL13] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* JHU press, 2013.

[HDR97] Geoffrey E. Hinton, Peter Dayan, and Michael Revow. Modeling the Manifolds of Images of Handwritten Digits. *IEEE Transactions on Neural Networks*, 8(1):65–74, 1997.

[HJ12] Roger A. Horn and Charles R. Johnson. *Matrix Analysis.* Cambridge University Press, 2012.

[HZ93] Geoffrey E. Hinton and Richard Zemel. Autoencoders, Minimum Description Length and Helmholtz Free Energy. *Advances in Neural Information Processing Systems*, 6, 1993.

[IR10] Alexander Ilin and Tapani Raiko. Practical Approaches to Principal Component Analysis in the Presence of Missing Values. *The Journal of Machine Learning Research*, 11:1957–2000, 2010.

[KW13] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[KW$^+$19] Diederik P. Kingma, Max Welling, et al. An Introduction to Variational Autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[Llo82] Stuart Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[LMJM$^+$20] Marine Le Morvan, Julie Josse, Thomas Moreau, Erwan Scornet, and Gaël Varoquaux. NeuMiss networks: Differentiable Programming for Supervised Learning with Missing Values. *Advances in Neural Information Processing Systems*, 33:5980–5990, 2020.

[LR19] Roderick J.A. Little and Donald B. Rubin. *Statistical Analysis with Missing Data*, volume 793. John Wiley & Sons, 2019.

[Mac03] David J.C. MacKay. *Information Theory, Inference and Learning Algorithms.* Cambridge University Press, 2003.

[MS23] Carl D. Meyer and Ian Stewart. *Matrix Analysis and Applied Linear Algebra.* SIAM, 2023.

[NOGV20] Alfredo Nazabal, Pablo M. Olmos, Zoubin Ghahramani, and Isabel Valera. Handling Incomplete Heterogeneous Data Using VAEs. *Pattern Recognition*, page 107501, 2020.

[PKD$^+$16] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context Encoders: Feature Learning by Inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

[RMW14] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *International Conference on Machine Learning*, pages 1278–1286. PMLR, 2014.

[Row97] Sam Roweis. EM Algorithms for PCA and SPCA. *Advances in Neural Information Processing Systems*, 10, 1997.

[Rub76] Donald B. Rubin. Inference and Missing Data. *Biometrika*, 63(3):581–592, 1976.

[Sch97] Joseph L. Schafer. *Analysis of Incomplete Multivariate Data.* CRC press, 1997.

[TB99a] Michael E. Tipping and Christopher M. Bishop. Mixtures of Probabilistic Principal Component Analyzers. *Neural computation*, 11(2):443–482, 1999.

[TB99b] Michael E. Tipping and Christopher M. Bishop. Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 61(3):611–622, 1999.

[VB18] Stef Van Buuren. *Flexible Imputation of Missing Data.* CRC press, 2018.

[vBGOR$^+$15] Stef van Buuren, Karin Groothuis-Oudshoorn, Alexander Robitzsch, Gerko Vink, Lisa Doove, Shahab Jolani, et al. Package 'mice'. *Computer software*, 2015.

[Wer06] Dirk Werner. *Funktionalanalysis.* Springer-Verlag, 2006.

[WNN18] Christopher K. I. Williams, Charlie Nash, and Alfredo Nazábal. Autoencoders and Probabilistic Inference with Missing Data: An Exact Solution for the Factor Analysis Case. *arXiv preprint arXiv:1801.03851*, 2018.

[Woo50] Max A. Woodbury. *Inverting Modified Matrices.* Department of Statistics, Princeton University, 1950.

[ZZ09] Yan Zi-Zong. Schur Complements and Determinant Inequalities. *Journal of Mathematical Inequalities*, 3(2):161–167, 2009.

Appendices

A Proofs

In this Appendix we present the proofs of two theorems appearing in the main text. The reason we present the proofs for those specific results, is because we were not able to locate them in the exact form we are using them in the bibliography (we were able to find equivalent variants of them). For the sake of completeness we prove them in the form we are using them

Proof of Theorem 2: In essence we have to prove that the bound for the approximation error is given by Equation (2.28). Indeed, consider that, from the triangle inequality we can write

$$\left|\left|\sum_{j=0}^{m} A^j\right|\right| \leq \sum_{j=0}^{m} ||A||^j \leq \sum_{j=0}^{+\infty} ||A||^j = \frac{1}{1-||A||}, \tag{A.1}$$

where we have used the fact that the norm is a real number and as a result we can use the geometric series for real numbers. The geometric series is convergent because of the conditions we have on A. Letting $m \to \infty$, in Equation (A.1), we obtain

$$\left|\left|\sum_{j=0}^{+\infty} A^j\right|\right| \leq \sum_{j=0}^{+\infty} ||A||^j = \frac{1}{1-||A||}, \tag{A.2}$$

which implies

$$||(I_n - A)^{-1}|| = \left|\left|\sum_{j=0}^{+\infty} A^j\right|\right| \leq \frac{1}{1-||A||}. \tag{A.3}$$

Finally, since

$$u - u_l = A^l g + A^{l+1} g + ... = A^l (I_n + A + ...) g = A^l (I_n - A)^{-1} g, \tag{A.4}$$

and using Equation (A.3), we can write

$$||u - u_l|| = \left|\left|A^l (I_n - A)^{-1} g\right|\right| \leq ||A||^l ||(I_n - A)^{-1}|| ||g|| \leq ||A||^l \frac{1}{1-||A||} ||g||, \tag{A.5}$$

where we have made use of the Cauchy-Schwarz inequality. This concludes the proof. □

Proof of Theorem 3: First we introduce some terms for convenience.

Definition A.1 Let f, g be real-rooted polynomials of degree n. Then, if $\alpha_n \leq ... \leq \alpha_1$ are the roots of f, and $\beta_n, ..., \beta_1$ are the roots of g, we say that g *interlaces* f, if and only if

$$\beta_n \leq \alpha_n \leq ... \leq \beta_1 \leq \alpha_1, \tag{A.6}$$

and we write $g \to f$ to signify that f has the largest root.

Definition A.2 Let A be a square matrix, then the *characteristic polynomial* of A is defined as

$$\chi(A)(x) = det(xI - A), \tag{A.7}$$

where the variable is x.

Now, let us prove the following statement.

Statement: If A is a symmetric matrix and v is a vector, then

$$\chi(A)(x) \to \chi(A + vv^T)(x). \tag{A.8}$$

Indeed, let us use a useful rank one update formula for the determinant, see e.g. [MS23].

$$\begin{aligned}\chi(A+vv^T)(x) = det(xI-A-vv^T) = det(xI-A)det(I-(xI-A)^{-1}vv^T) = \\ = \chi(A)(x)(1-v^T(xI-A)^{-1}v).\end{aligned} \tag{A.9}$$

Letting (λ_i, u_i) be the i^{th} eigenvalue-eigenvector pair of A, we have

$$\chi(A+vv^T)(x) = \chi(A)(x)\left(1-\sum_{j=1}^{n}\frac{\langle v-u_j\rangle}{x-\lambda_j}\right). \tag{A.10}$$

Therefore, an eigenvalue of $A+vv^T$ is either an eigenvalue of A, if v is orthogonal to the corresponding eigenvector, or such that $\sum_{j=1}^{n}\frac{\langle v-u_j\rangle}{x-\lambda_j}=1$. This establishes the statement. Notice that the roots of the characteristic polynomials are the eigenvalues, which implies that we have concluded the proof of Theorem (3). $\square$