UNIVERSITY OF NOVI SAD

MASTER THESIS

Application of Autoencoders on Single-cell Data

Author:
Aleksandar ARMACKI

Supervisor:
Dr. Sanja BRDAR

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

at the

Faculty of Sciences
Department of Mathematics and Informatics

September 24, 2018

Declaration of Authorship

I, Aleksandar ARMACKI, declare that this thesis titled, "Application of Autoencoders on Single-cell Data" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Live as if you were to die tomorrow. Learn as if you were to live forever."

Mahatma Ghandi

UNIVERSITY OF NOVI SAD

Abstract

Faculty of Sciences
Department of Mathematics and Informatics

Master of Science

Application of Autoencoders on Single-cell Data

by Aleksandar ARMACKI

Single cell data allows for analysis of gene expression at cell level. Such data is of huge importance for establishing new cell types, finding causes of various diseases or differentiating between sick and healthy cells, to name a few.

One of the main challenges in the analysis of such data is big dimensionality, where each gene is a feature, therefore depending on the species observed, the data can have well above 30.000 features. Dimensionality reduction for such data is a necessary preprocessing step.

In our work, we evaluated dimension reduction capabilities of two neural networks based models – autoencoders and variational autoencoders and benchmarked them against a well known dimensionality reduction method – principal component analysis. The performances of all three dimension reduction models were tested on real life single-cell datasets with respect to different aspects – the quality of classification, clusterization and the reconstruction quality.

Obtained results indicate that the best method for dimensionality reduction for single cell data is the autoencoder, whereas the more powerful variational autoencoder in some aspects performed worse than the linear transformation based principal component analysis.

Acknowledgements

I would like to express my gratitude to Dr. Blaž Zupan and his team at the Faculty of Computer and Information Science, University of Ljubljana. The hospitality and kindliness they have shown throughout my time in Ljubljana went beyond the work in their lab and this master thesis wouldn't have been possible without their help and usefull suggestions.

Also, I would like to thank my mentor, Dr. Sanja Brdar for her guidence and patience. Her passion and keen insight motivated me and her ideas helped shape this thesis.

Finally, I owe a great debt of gratitude to my friends and family, for always being there for me, supporting me and providing useful suggestions whenever I needed them.

Contents

List of Figures

List of Tables

List of Abbreviations

AE	**A**uto**E**ncoder
AI	**A**rtificial **I**ntelligence
ARI	**A**djusted **R**and **I**ndex
BCE	**B**inary **C**ross-**E**ntropy
CPU	**C**entral **P**rocessing **U**nit
ELBO	**E**vidence **L**ower **BO**und
GPU	**G**raphical **P**rocessing **U**nit
IID	**I**ndependent **I**dentically **D**istributed
MSE	**M**ean-**S**quared **E**rror
PCA	**P**rincipal **C**omponent **A**nalysis
t-SNE	**t** **S**tochastic **N**eighbour **E**mbedding
VAE	**V**ariational **A**uto**E**ncoder
ReLU	**Re**ctified **L**inear **U**nit
RI	**R**and **I**ndex
RNA	**R**ibo**N**ucleic **A**cid
RNA seq	**RNA SEQ**uencing

To my mother, for her unwavering support thoroughout my studies.

Chapter 1

Introduction

In the modern era, terms like 'big data', 'machine learning', 'neural networks' or 'deep learning' are becoming ubiquitous. Large amounts of data supplemented by technological advances push machine learning algorithms onto new heights. In turns, these new and advanced algorithms crave for more data, thus creating a seemingly endless loop.

In order to build a profound knowledge and keep pace with a rapidly advancing area, we embarked on a challenge of applying deep learning models on datasets coming from an area of keen interest to a lot of machine learning practitioners – bioinformatics. Deep learning is gaining a significant foothold in this area, evident by the recent surge in papers concerning the application of deep learning models in bioinformatics ([26], [10], [14], [38], [31]).

In our work, we focused on a recently established type of data – single-cell data. Instead of inputing large amounts of cells and getting the average read counts of gene expressions, one can now obtain those results at cell level. One of the main challenges in the analysis of such data is big dimensionality, often measured in tens of thousands of dimensions. Dimensionality reduction for such data is a necessary preprocessing step.

We evaluated dimension reduction capabilities of two neural networks based models typically associated with deep learning – autoencoders and variational autoencoders. Their performances were benchmarked against a well known dimensionality reduction method – principal component analysis. Tests were performed on real life single-cell datasets with respect to different aspects – the quality of classification, clusterization and reconstruction quality.

All of the models used in our work are based on strong mathematical background. Neural networks themselves can be considered a composition of functions of the input data and their training is based on solving different[1] optimization problems iteratively. The variational autoencoder model offers deep insight into the probabilistic background and covers some underlying principles and ideas of probabilistic graphical models. Finally, a detailed approach to concepts such as PCA and entropy serves to rejuvenate the reader's knowledge about some fundamental mathematical concepts in machine learning.

The rest of the thesis is organized as follows: Chapter 2 explains the concept of single-cell data and provides an overview of the data used in our work. Chapter 3 concerns neural networks, giving a short historical introduction and the basic mathematical tools necessary for understanding neural networks. The rest of the chapter concerns neural network models used in our work – autoencoders and variational autoencoders. Chapter 4 gives thorough explanations of the metrics used and their suitability for the task at hand. Chapter 5 describes our experiments and presents results obtained and Chapter 6 concludes the thesis, providing a summary of the

[1] Depending on the nature of the problem at hand.

work done and some ideas for future work. Appendix A discusses components of neural networks in more details, Appendix B explains some concepts from information theory used in the field of neural networks and machine learning in general and Appendix C introduces the mathematical model of PCA.

A lot more can be written about the topics ahead, but without further hesitation, we encourage the reader to step deeper into this varied and interesting area, abounding with possibilities and promises. Take the first step by reading our work and then, hopefully, expand on it. Let us get started.

Chapter 2

Single-cell Data

This chapter introduces the concept of single-cell data. It is organized in two sections, the first one defining the notion of single-cell data and its main features, while the second section elaborates on the datasets used in our work.

2.1 What is single-cell data?

Technological advances have allowed for single cell RNA sequencing. Instead of inputing large amounts of cells and getting the average read counts of gene expressions, one can now obtain those results for a single cell. The potential is vast – such an approach can show variance in gene expressions in cells, potentially discovering novel cell types or helping establish causes of disease when measuring the difference in gene expressions of sick and healthy cells ([8], [19], [12], [15], [11]).

While such an approach comes with a lot of possibilities, the are some drawbacks of single-cell RNA seq. They come in two main forms:

- **High dimensionality** is a trait of single-cell data shared with bulk cell data. Depending on the species observed the number of genes can be tens of thousands. Combined with the prospect of a larger amount of data available nowadays in the era of big data, this trait makes the notion of dimension reduction a necessity when analysing single-cell (as well as bulk) data.

- **Dropout** is a trait typical for single-cell data. The main reason for using bulk RNA seq was the lack of RNA material contained in a single cell necessary to obtaine gene expressions. Although technology has advanced, it is still not sophisticated enough to flawlessly process the small amount of RNA material contained in a single cell. Dropouts are formally defined as false zeros, i.e. genes that have been expressed in a cell, but, due to the lack of RNA material or a technical error in the process used to obtain the data, they have been expressed as zeros. This aspect of single-cell data is quite a problematic one as it affects the quality of any analysis of the data and is a widely studied problem ([22], [20], [23], [30]).

Another peculiarity of single cell data is high sparsity. Sparsity stems from two sources – one is the true zero expressions, i.e. the genes not expressed in the cell, while the other is dropouts. While dropouts causing sparsity are an unwanted feature, in general, sparsity is not considered a bad property of single-cell data. If anything, it helps reduce computation costs and to some extent alleviate the cost caused by high dimensionality.

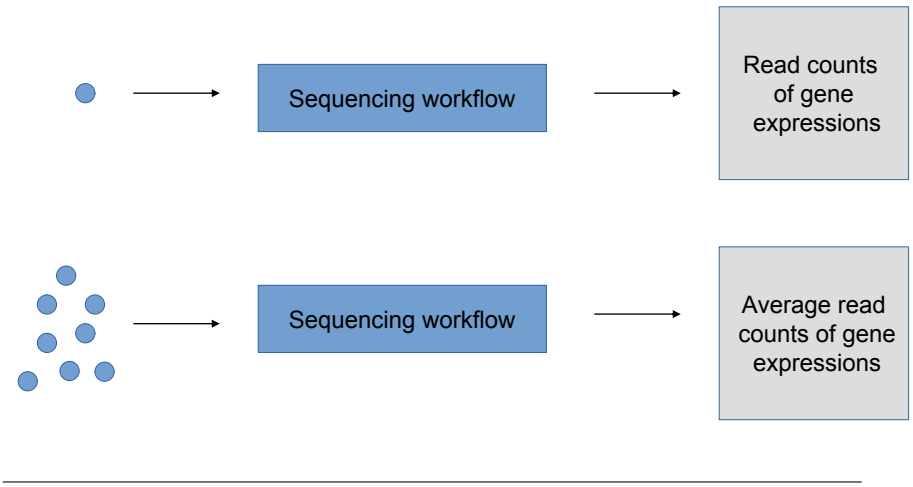

FIGURE 2.1: Single-cell and bulk RNA sequencing

2.2 Datasets used

2.2.1 Description

In our work we used six different datasets: Baron [3], Muraro [27], Xin [39], Deng [9], Goolam [13], Biase [4]. The former three datasets are human pancreas cells, while the later three are mouse embryo development cells. The idea behind this particular choice of datasets was simple – instead of just utilizing the standard cross-validation approach to training and testing, we also used one of the datasets from the same species and organ types for training and another one for testing. The results confirmed our assumptions that datasets originating from the same species and organs can be used intermittently as training and testing sets. The quality of results was comparable to results obtained via cross-validation, in some cases even eclipsing them. This was the case in spite of the fact that differences in protocols and techniques used to obtain different datasets are a source of technical noise. Further information regarding the datasets used is provided in Table 2.1.

The approach described in the passage above can be considered a form of transfer learning on single-cell data. A similar concept, as well as a more elaborate treatment of this topic, can be found in [18].

name	species	organ	cells	genes	sparseness	classes
Baron	human	pancreas	8569	20125	90.62%	14
Xin	human	pancreas	1492	38172	87.82%	4
Muraro[1]	human	pancreas	2292	7117	0	10
Deng	mouse	embryo devel	317	22958	60.72%	16
Goolam	mouse	embryo devel	124	40405	68.01%	5
Biase	mouse	embryo devel	56	25114	50.73%	4

TABLE 2.1: Overview of the datasets used

[1] A filtered dataset from the paper was used in our experiments, with the filtering process destroying its sparsity pattern. The original dataset had 3072 cells, 19059 genes and sparseness rate of 78.84%.

2.2.2 Preprocessing

Prior to doing any experiments, we preprocessed the data in the following way: when training on one dataset and testing on another, first the genes contained in both datasets were identified and only those genes were used as features in training. Then, the logarithmic transformation was applied. It was done as follows: if $X \in \mathbb{R}^{N \times d}$ is our data matrix, where each row is a sample and each column is a feature, then

$$Y = \log(X + J) \tag{2.1}$$

is the log transformed matrix, where

$$J = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & & \vdots \\ 1 & 1 & \dots & 1 \end{bmatrix}_{N \times d} \tag{2.2}$$

is the ones matrix of the same dimension as X. Then, for every sample vector $y^{(i)}$ from the matrix Y we divide the sample vector by its maximal element, i.e.

$$z^{(i)} = \frac{y^{(i)}}{\max_{j \in \{1,\dots,d\}} \{y_j^{(i)}\}} \tag{2.3}$$

where $y_j^{(i)}$ denotes the j^{th} component of the i^{th} sample vector. Finally, we get to the form of data matrix used as input for our models

$$Z = \begin{bmatrix} {z^{(1)}}^T \\ \vdots \\ {z^{(i)}}^T \\ \vdots \\ {z^{(N)}}^T \end{bmatrix}_{N \times d} . \tag{2.4}$$

There are two reasons for these transformations. First, raw read counts can sometimes be very large, so the log transformation makes computation more tractable. Secondly, since the activation function at the output of neural networks used is sigmoid, it is a necessity for the data matrix to have the following property

$$z_{ij} \in [0, 1], \forall (i, j) \in \{1, \dots, N\} \times \{1, \dots, d\} \tag{2.5}$$

where $\times$ denotes the Cartesian product of two sets. A more elaborate treatment of this necessity as well as activation functions in general is offered in Appendix A.

Chapter 3

Neural networks

This chapter introduces the concept of neural networks from a theoretical perspective. It is organized in three sections, the first providing a general overview of neural networks, with the other two elaborating on the particular models used in our work.

3.1 Overview

3.1.1 A brief history

Although their rise to prominence was abrupt and steep, neural networks have a long history. One of the first predecessors of this approach was Frank Rosenblatt's Perceptron [32], established all the way back in 1958. As the initial effervescence faded, the perceptron was exposed as a severely limited model. Combined with some other failures of highly publicized AI models of that time, the public interest waned and a lot of researchers fell into obscurity. A period known as AI winter[1] ensued and lasted for nearly 50 years.

Although that period was long and largely barren, it did produce a solid theoretical framework for what was to come. For example, Rumelhart, Hinton and Williams introduced error propagation for multilayer networks in 1986. [35], while Yann Lecun formulated the framework of backpropagation for neural networks in 1988. [21]. The latter is considered to be one of the most important contributions in the theory of neural networks, as it allowed for the gradient descent to be successfully applied in multilayer models. Finally, various advances, stemming mainly from the gaming industry[2], engendered deep neural networks and fields such as deep learning, moving the whole area towards the state we know today.

3.1.2 The mathematical model

To start, we will introduce some basic notation, while the remaining details will be explained subsequently. Quantity x denotes a vector, whereas x_i denotes the i-th scalar component of vector x. Symbol $\|\cdot\|$ denotes the vector (matrix) norm and referes to Euclidean (spectral) norm (respectively). Activation functions are scalar functions of one variable, i.e. if σ is an activation function, then

$$\sigma : \mathbb{R} \mapsto \mathbb{R}\,. \tag{3.1}$$

One can find different definitions of depth of a neural network. In our work, we define the depth of a network as the number of hidden layers in the network.

[1] The period is referenced as AI winter because of a lack of significant investment in this area.

[2] The term here is not to be confused with gambling industry, but is refered to computer games industry.

The framework presented here concerns neural networks with one (hidden) layer. It can easily be generalized for networks with an arbitrary number of layers, but for the sake of simplicity, we will stick with this model of network. A graphical representation of a network with one hidden layer is presented in Figure 3.1.

Assume that we are given a set of data pairs $\{(x_n, t_n)\}_{n=1}^N$, where $x_n \in \mathbb{R}^d$ represents a data point and $t_n \in \mathbb{R}^m$ is a target value. Define a_i as

$$a_i = \sum_{j=1}^{d} w_{ij}^{(1)} x_{ij} + w_{i0}^{(1)}, \; i = 1, \ldots, D \tag{3.2}$$

where D is the number of neurons in the first hidden layer, w_{ij} is the weight parameter associated with the i^{th} neuron and j^{th} scalar of vector x_i and the superscripts point to the fact that it is associated with the first hidden layer of the network. w_{i0} is often called the bias parameter. Let σ_1 and σ_2 be two activation functions. Then

$$z_i = \sigma_1(a_i) \tag{3.3}$$

represents the output value of the i^{th} neuron in the first hidden layer. Analogously, we define

$$b_i = \sum_{j=1}^{D} w_{ij}^{(2)} z_j + w_{i0}^{(2)} \tag{3.4}$$

and combining (3.2), (3.3) and (3.4) we get

$$y_i = \sigma_2(b_i) = \sigma_2\left(\sum_{j=1}^{D} w_{ij}^{(2)} \sigma_1\left(\sum_{k=1}^{d} w_{jk}^{(1)} x_{jk} + w_{j0}^{(1)} \right) + w_{i0}^{(2)} \right) \tag{3.5}$$

for $i = 1, \ldots, m$. We can sum the result up in a vector form

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \tag{3.6}$$

where y represents the output vector of the network. This is the basic model of a feed-forward network. The goal of training process is to make the values on the output of the neural network as close as possible to target values. For example, if one is dealing with a regression problem, a function of the form

$$O(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \| y_n(x_n, \mathbf{w}) - t_n \|^2 \tag{3.7}$$

could be used. Here $\mathbf{w}$ denotes all the weights of the network. In this context, function (3.7) is called a loss function. The goal of training is to find a set of weights $\mathbf{w}^*$ such that

$$\min_{\mathbf{w}} O(\mathbf{w}) = O(\mathbf{w}^*) \, . \tag{3.8}$$

There are various approaches to minimizing a loss function, with one of the most common being gradient descent [34].

Going back to $\mathbf{w}$, in the case of our network, it corresponds to $d \times D \times D \times m$ different weights. At this point, it should be clear that the number of weights tends to explode with the size of the network and the dimensionality of data.

Another property of the set of weights is that different sets of weights can result in the same mapping from the input to the output of the network. Assume that the activation function of the k^{th} hidden layer is the identity function

$$f(x) = x, \forall x \in \mathbb{R} \tag{3.9}$$

and for simplicity, let the bias parameters of the k^{th} and $k+1^{st}$ layers, $w_{i0}^{(k)}$ and $w_{i0}^{(k+1)}$, be equal to zero for all i. Then, by multiplying all the weights $w^{(k)}$ and $w^{(k+1)}$ with -1 without changing any other weights in the system, what we get is

$$a_i = \sum_{j=1}^{d}(-w_{ij}^{(k)})x_{ij} = -\sum_{j=1}^{d} w_{ij}^{(k)} x_{ij} \tag{3.10}$$

and combining (3.9) and (3.10)

$$\begin{aligned} b_i &= -\sum_{j=1}^{d} w_{ij}^{(k+1)} z_j = -\sum_{j=1}^{d} w_{ij}^{(k+1)} a_j \\ &= -\sum_{j=1}^{d} w_{ij}^{(k+1)} \left(-\sum_{m=1}^{d} w_{jm}^{(k)} x_{jm} \right) \\ &= \sum_{j=1}^{d} w_{ij}^{(k+1)} \left(\sum_{m=1}^{d} w_{jm}^{(k)} x_{jm} \right). \end{aligned} \tag{3.11}$$

So b_i, the input to $k+1^{st}$ activation function is the same as it would have been had we left both the weight vectors $w^{(k)}$ and $w^{(k+1)}$ unchanged. This goes on to show the combinatorial explosion of the number of different solutions to the optimization problem in (3.7).

Although there are various aspects that deserve attention, this short overview encompasses the basic and most important elements of neural networks. A more detailed treatment of some additional elements is presented in Appendix A. The reader is also refered to [5] for a comprehensive treatment of topics discussed here, as well as machine learning in general.

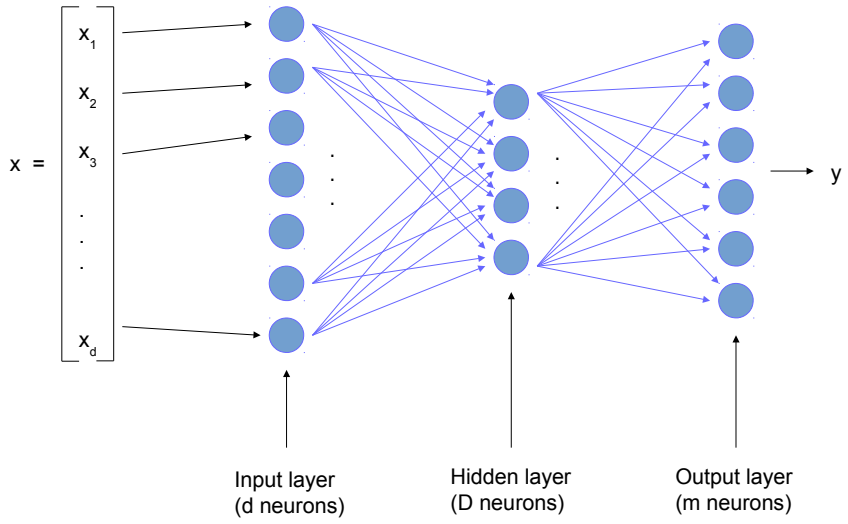

FIGURE 3.1: A neural network with one hidden layer

3.2 Autoencoder

AE [2] is a type of feed-forward neural network whose goal is to learn an efficient representation (encoding) of data. It is achieved by first mapping the original data to a (usually) lower dimensional space and then by mapping the lower dimensional representation back to the original space, with the goal of making the data obtained that way match the original data as close as possible[3]. Going back to our loss function from (3.7), in the case of AE, it would take the following form

$$O(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} \|y_n(x_n, \mathbf{w}) - x_n\|^2 . \tag{3.12}$$

In other words, the input vectors simultaneously play the role of input data as well as target vectors. Making the transformed data as similar as possible to the original data ensures that the lower dimensional representation learned by an autoencoder contains as much relevant information as possible. This idea is key to the main purpose of AE - dimensionality reduction.

Also, by creating a compact representation of the data, AE removes noise inherent to data, so the output of AE is a denoised version of the original data. Denoising is another common use of AE.

As can be seen in Figure 3.2, AE consists of two symmetric networks - an encoder and a decoder. The encoder is used to project the original data to a lower dimensional space, while the decoder does the opposite - projects the lower dimensional data back to the original space.

AEs saw their most prominent use in the area of image compression and denoising. In recent times, various extensions to the basic model have been proposed ([36], [40]), giving rise to a new model - the variational autoencoder.

[3]To clarify the ambiguity introduced here, the output needs to match the input with respect to a predefined criterion, depending on the application. For example, we could aim to minimize the mean-squared error or to maximize the cosine similarity between the input and the output data.

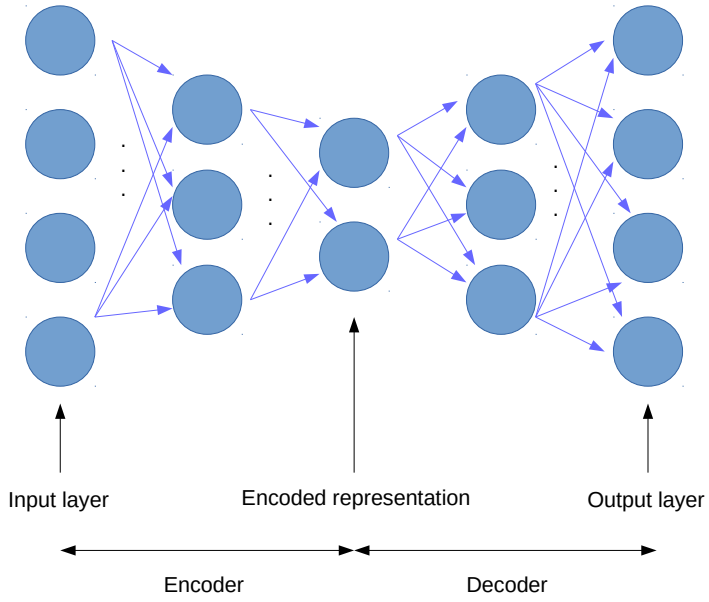

FIGURE 3.2: An autoencoder network

3.3 Variational Autoencoder

Introduced by Kingma and Welling [17] VAE is, at a first glance, a model fairly similar to a standard AE. The model consists of an encoder and a decoder and aims to produce an output similar to the input. However, there are some fundamental differences.

For starters, VAE is essentialy a probabilistic model. Assume we are given a set of iid data points $\{x_n\}_{n=1}^N$ originating from a random variable x. Let z be a latent variable whose distribution, as well as the samples generated, are unknown to us. The process works as follows: first, a value z_i is generated from a distribution $p(z_{\theta^*})$ and then a value x_i is generated from a distribution $p(x|z_i)$. As was already said, the values of θ^* and z_i are unknown, while the prior $p_{\theta^*}(z)$ and the likelihood function $p_{\theta^*}(x|z)$ come from parametric families of distributions $p_\theta(z)$ and $p_\theta(x|z)$ respectively. Assuming that the marginal $p_\theta(x)$ and the posterior $p_\theta(z|x)$ are intractable, which is often the case (e.g. a neural network with a nonlinear hidden layer), we can introduce $q_\phi(z|x)$ to approximate the intractable posterior. The approximation model $q_\phi(z|x)$ can be interpreted as a probabilistic encoder, since given the value of x, it encodes a latent representation z, while the model $p_\theta(x|z)$ can be interpreted as a probabilistic decoder, since given the value of the latent variable z it decodes z to a variable x.

In order to obtain parameters θ and ϕ, using the fact that $\{x_n\}_{n=1}^N$ are iid, first the marginal log likelihood function is computed as

$$\log p_\theta(x_1, \ldots, x_N) = \log \left(\prod_{i=1}^N p_\theta(x_i) \right) = \sum_{i=1}^N \log p_\theta(x_i) \tag{3.13}$$

where

$$\log p_\theta(x_i) = D_{KL}\left(q_\phi(z|x_i) \| p_\theta(z|x_i)\right) + L(\theta, \phi; x_i)\,. \tag{3.14}$$

The log likelihood function is used as an estimator of parameters. Maximizing the

log likelihood with respect to θ is a standard way to obtain reliable parameters of a distribution. In (3.13) the left hand side represents the log likelihood of the marginal distribution with respect to variables $x_1, \ldots, x_N$ and the term after the second equality sign is the sum of individual log likelihoods for each variable.

The right hand side of equation (3.14) consists of two terms: the Kullback-Leibler divergence of the approximated posterior from the true posterior and function L, given by

$$L(\theta, \phi; x_i) = \mathbf{E}_{q_\phi(z|x_i)}\big[-\log q_\phi(z|x_i) + \log p_\theta(x_i, z)\big] \tag{3.15}$$

which can be rewritten as

$$L(\theta, \phi; x_i) = -D_{KL}\big(q_\phi(z|x_i) \| p_\theta(z)\big) + \mathbf{E}_{q_\phi(z|x_i)}\big[\log p_\theta(x_i|z)\big]\,. \tag{3.16}$$

The first term on the right hand side of (3.16) represents the negative Kullback-Leibler divergence of the approximate posterior $q_\phi(z|x_i)$ from the prior $p_\theta(z)$, while the second term represents the expectation with respect to the approximate posterior $q_\phi(z|x_i)$ of the log likelihood of $p_\theta(x_i|z)$.

Since the true posterior $p_\theta(z|x_i)$ is untractable, we combine the fact that Kullback-Leibler divergence is always non-negative with (3.14) and obtain

$$\log p_\theta(x_i) \geq L(\theta, \phi; x_i) \tag{3.17}$$

which explains the name of function L, ELBO. The goal of the network is to optimize L with respect to both ϕ and θ.

Equation (3.16) provides further explanation for the VAE model. The right hand side of this equation consists of two terms that again can be interpreted with respect to our model: first one being Kullback-Leibler divergence of the encoder from the marginal $p_\theta(z)$ and second one being expectation of the decoder with respect to the encoder. While the second term concernes the quality of the output of our network, i.e. plays the role of the objective function of a standard AE, the first term is specific for VAE: assuming any distribution $p_\theta(z)$ we can control the encoder model by forcing the latent space created by it to closely resemble the given distribution p.

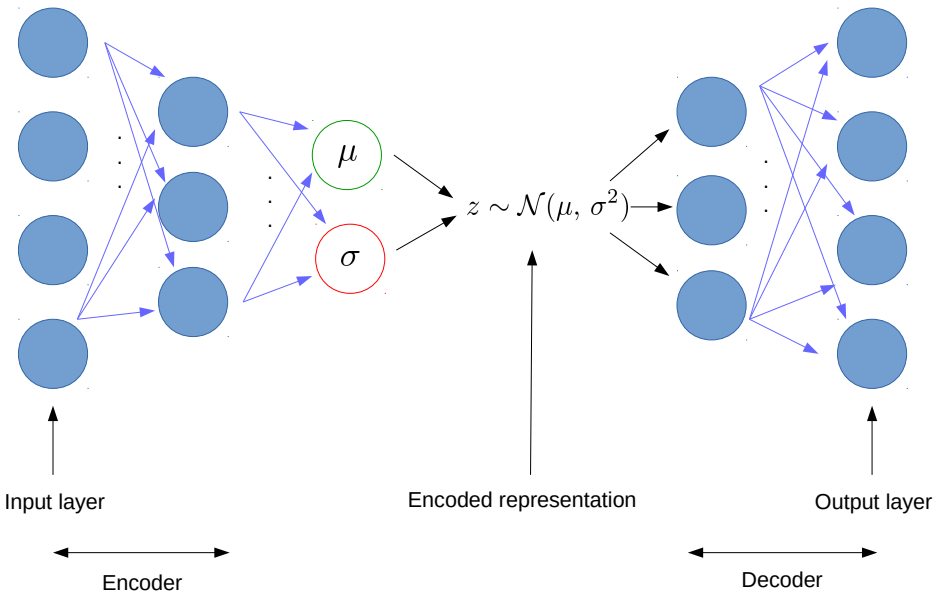

FIGURE 3.3: A variational autoencoder network

Instead of just mapping a point to the latent space, in the case of VAE, the network actually parametrizes a distribution from which samples are then drawn. For example, assuming that $p_\theta(z)$ is a standard normal distribution $\big($i.e. $\mathcal{N}(0,1)\big)$, the network will find parameters μ_i and σ_i for each input vector x_i and draw a sample from a Gaussian parametrized by these values, effectively making $q_\phi(z|x_i) \sim \mathcal{N}(\mu_i, {\sigma_i}^2)$. A VAE network is presented in Figure 3.3.

The advantages of such a model are multiple. For example, we have more control with respect to the structure of the latent space that is created, giving us application related adaptability.

The fact that the latent space is made to be more compact can be explained in the following way: we don't want our network to simply learn to map all the data points to points far from each other in the latent space, but we want it to be able to recognize similar points, learning similar parametrization for them and thus mapping them close to each other.

The most powerful concept of VAE, however, is the fact that in parametrizing a distribution for the latent space it can effectively generate new samples that closely resemble the learned data, thus making it a generative model as well.

Chapter 4

Evaluation Methods

Performances of models used were measured based on three different aspects: clusterization, classification and reconstruction quality. For each of the aspects, different evaluation metrics were used. This chapter is organized in three sections, each elaborating on the metrics used to evaluate the quality of the models, with respect to a particular aspect.

4.1 Classification

To measure the quality of classification on reduced data, we used F1 score.

4.1.1 F1 score

F1 score is a measure of classification quality based on precision and recall. For a binary classification problem all possible outcomes are described in Table 4.1 below.

True/Pred	Positive	Negative
Positive	a	b
Negative	c	d

TABLE 4.1: The possible outcomes of binary classification

Then, precision is defined as

$$P = \frac{a}{a+c} \tag{4.1}$$

while recall is defined as

$$R = \frac{a}{a+b}\,. \tag{4.2}$$

Using (4.1) and (4.2), F1 score is calculated as

$$F1 = \frac{2PR}{P+R}\,. \tag{4.3}$$

Another way to express F1 score is using values from Table 4.1 directly as

$$F1 = \frac{2a}{2a+c+b}\,. \tag{4.4}$$

In case of multiclass classification there are multiple methodologies used.

Micro methodology uses Table 4.1 for each class and then calculates the global F1 score as

$$F1_{micro} = \frac{2\sum_{i=1}^{N} a_i}{\sum_{i=1}^{N}(2a_i + c_i + b_i)} \tag{4.5}$$

where a_i, b_i and c_i are the true positives, false negatives and false positives for class i, $i = 1, \ldots, N$.

Macro methodology calculates F1 score for each class and then computes the overall F1 score as per class average, i.e.

$$F1_{macro} = \frac{1}{N} \sum_{i=1}^{N} F1_i \tag{4.6}$$

where $F1_i$ is the F1 score for class i.

Weighted methodology calculates the average F1 score per class weighted by the support of each class (the number of true instances), i.e.

$$F1_{weighted} = \frac{\sum_{i=1}^{N}(a_i + d_i)F1_i}{\sum_{j=1}^{N}(a_j + d_j)} \tag{4.7}$$

where a_i and d_i are the number of true positives and true negatives for class i, respectively.

Micro score has problems with small classes - since it considers only true positive rates small classes contribute little to overall score. Macro score weights all classes the same, but doesn't take into account label imbalance.

In our work, classes showed significant size difference and F1 score calculated using weighted and micro methodologies was almost identical. It suggested that more weight is given to larger classes, so we chose to use macro methodology for calculating F1 score.

4.2 Clusterization

To measure the quality of clusterization on the reduced data, we used adjusted rand index and silhouette score.

4.2.1 Adjusted Rand Index

ARI is a measure of clusterization quality based on similarity of two data clusterings. It is a form of *Rand index* adjusted for the chance grouping of elements. Given a set S of size n and two partitions of S, X and Y (where X and Y are not necessarily of the same size), let a be the number of pairs of elements in S that are in the same partitions in both X and Y and let b be the number of pairs of elements in S that are in different partitions in both X and Y. Then RI is calculated as

$$RI = \frac{a + b}{\binom{n}{2}}. \tag{4.8}$$

ARI uses the assumption that X and Y are of constant size. It is calculated as

$$ARI = \frac{index - expected\ index}{max\ index - expected\ index} \tag{4.9}$$

or, using Table 4.2, as

$$ARI = \frac{\sum_{i,j}\binom{n_{ij}}{2} - (\sum_i \binom{n_{i.}}{2} \sum_j \binom{n_{.j}}{2})/\binom{n}{2}}{\frac{1}{2}(\sum_i \binom{n_{i.}}{2} + \sum_j \binom{n_{.j}}{2}) - (\sum_i \binom{n_{i.}}{2} \sum_j \binom{n_{.j}}{2})/\binom{n}{2}}. \tag{4.10}$$

While RI $\in [0,1]$, ARI can take values from the interval $[-1,1]$.

$X \backslash Y$	v_1	v_2	...	v_s	sum
u_1	n_{11}	n_{12}	...	n_{1s}	$n_{1.}$
u_2	n_{21}	n_{22}	...	n_{2s}	$n_{2.}$
$\vdots$					
u_m	n_{m1}	n_{m2}	...	n_{ms}	$n_{m.}$
sum	$n_{.1}$	$n_{.2}$	...	$n_{.s}$	n

TABLE 4.2: Contingency table for comparing 2 partitions. n_{ij} is the number of objects that are both in u_i and v_j; $n_{i.}$ and $n_{.j}$ are the number of objects in u_i and v_j respectively

4.2.2 Silhouette score

Silhouette score is a metric for validation of consistency within clusters of data. It measures how similar an object is to its own cluster compared to other clusters. Assume we cluster N points into k clusters. Then, for each data point i, we define $a(i)$ as the average distance between i and all the other points in the same cluster

$$a(i) = \frac{1}{|c_i| - 1} \sum_{j \in c_i} d(i,j) \tag{4.11}$$

where c_i is the cluster point i belongs to, $|\ |$ denotes the number of elements of a set and d is the Euclidean distance between two points. Define the average dissimilarity of point i to cluster c (where $i \notin c$) as

$$dissim(i,c) = \frac{1}{|c|} \sum_{j \in c} d(i,j)\,. \tag{4.12}$$

Next, we define $b(i)$ as

$$b(i) = \min_{\{c | i \notin c\}} dissim(i,c)\,. \tag{4.13}$$

Silhouette of point i is defined as

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}\,. \tag{4.14}$$

Finally, silhouette score is then simply calculated as the average silhouette

$$SS = \frac{1}{N} \sum_{i=1}^{N} s(i)\,. \tag{4.15}$$

From (4.14) and (4.15) it is clear that silhouette score takes values from the interval $[-1,1]$.

4.3 Reconstruction

To measure the quality of reconstruction of data, we used cosine similarity, mean-squared error, Pearson correlation coefficient and binary cross-entropy.

4.3.1 Cosine similarity

Cosine similarity is a measure of similarity between two nonzero vectors in an inner product space. It measures the cosine of angle between the two vectors. Formally, it is defined as

$$\cos(x, y) = \frac{x^T y}{\|x\| \|y\|} . \tag{4.16}$$

Depending on the angle between the vectors, $\cos(x, y) \in [-1, 1]$. The main drawback of this similarity measure is that it does not consider the magnitude of the vectors, only the angle between them. So for two vectors having the same direction and slope, but different magnitude, cosine similarity will still return a score of 1, or a perfect match.

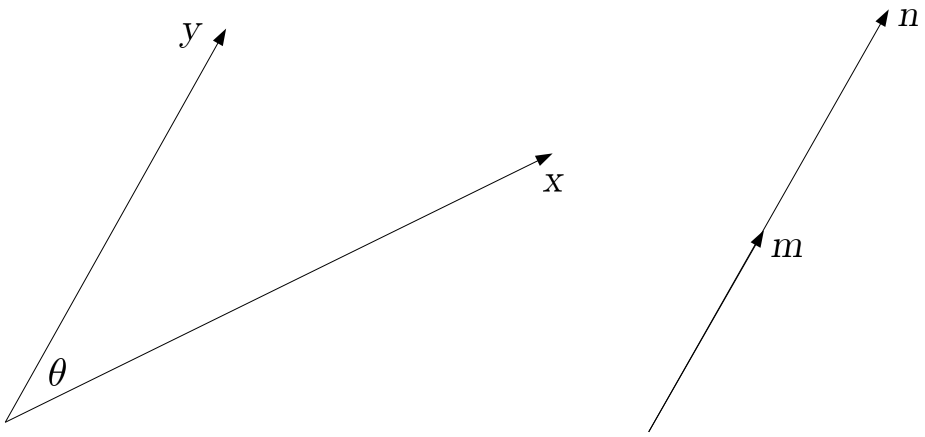

FIGURE 4.1: Two examples of cosine similarity. Cosine similarity of vectors x and y is calculated as $\cos(\theta)$. Cosine similarity of vectors m and n is 1, although there is an obvious difference in their respective magnitudes

4.3.2 Mean-squared error

Mean-squared error measures the average distance of estimated values from true/target values. If we are given a set $\{(x_i, t_i)\}_{i=1}^N$, $(x_i, t_i) \in \mathbb{R}^d \times \mathbb{R}^d$, where x_i are the estimated values and t_i are the true values, MSE is calculated as

$$MSE = \frac{1}{N} \sum_{i=1}^{N} \|x_i - t_i\|^2 . \tag{4.17}$$

The main downside of this metric is its susceptibility to outliers - a few estimates deviating significantly from their true/target values can skew the whole value, thus making it less reliable.

4.3.3 Binary cross-entropy

Cross-entropy is a measure used when dealing with probabilistic models. It was introduced in information theory, where it is interpreted as the number of bits needed to identify an occuring event, if a coding scheme is used that is optimized for a proximal probability distribution q, rather than the true distribution p. Formally, it is

defined as

$$CE(p,q) = \mathbf{E}_p[-\log q] = H(p) + D_{KL}(p\|q) = -\sum_x p(x)\log\big(q(x)\big) \tag{4.18}$$

where $H(p)$ is the entropy of p. Both the entropy and Kullback-Leibler divergence are covered in Appendix B.

In the case of binary classification tasks, where our values can be of the form $y \in \{0,1\}$, we define two distributions: p and q. p is the true distribution, which can simply be defined as

$$\begin{aligned} p_{\{y=1\}}(y) &= y \\ p_{\{y=0\}}(y) &= 1-y \end{aligned} \tag{4.19}$$

q is a proximal distribution, for example a neural network estimated distribution. It takes the values $q \in \{\hat{y}, 1-\hat{y}\}$ and can be defined as

$$\begin{aligned} q_{\{y=1\}}(y) &= \hat{y} \\ q_{\{y=0\}}(y) &= 1-\hat{y}\,. \end{aligned} \tag{4.20}$$

We obtain the binary cross-entropy formula using (4.18) as

$$BCE(p,q) = -p_0\log(q_0) - p_1\log(q_1) = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})\,. \tag{4.21}$$

Going back to equation (3.16) from Chapter 3, the second term on the right hand side of the equation is exactly the cross-entropy between input and output data as defined in (4.18).

From (4.21) it is clear that binary cross-entropy penalizes disagreement between true and predicted values. It is exactly that property that makes it a common objective function used in neural networks and an ideal fit for the networks used in our work.

Chapter 5

Experimental Results

This chapter desribes the models used, experiments performed and results. It is organized in two sections, the first describing computational resources used, parameters of the models and how each experiment was performed. The second section presents the results obtained.

5.1 Framework

5.1.1 Computational Resources

All experiments were executed on the server of Faculty of Computer and Information Science at the University of Ljubljana. The server has two CPUs, each one being a six core Intel(R) Xeon(R) CPU E5-1650 v3 @ 3.50GHz. It also has four GeForce GTX TITAN X GPUs each having 12GB memory. We used the GPUs for our experiments, since they provide a significant speed-up compared to the standard CPU.

All experiments were executed in Python [33], relying on libraries such as Keras [6], NumPy [28], scikit-learn [37], TensorFlow [1], SciPy [16] and Pandas [25]. For a more hands-on approach to deep learning and neural networks using Keras, the reader is refered to [7].

5.1.2 The Models

Benchmark

As a benchmark model, we used PCA, due to its long history and widespread use. It is a well known and used tool in biology, so making it a baseline for the quality of dimension reduction made a lot of sense from this perspective as well.

It was introduced by Karl Pearson [29]. Depending on the area of application, it has different names, but in the field of mathematics, it is known as Singular value decomposition. A more formal treatment of PCA can be found in Appendix C.

Concerning the model of PCA used in our work, in every experiment we would fit the PCA model on the training data and then implement it on the test data. Since PCA has no free parameters except L, the number of principal components to use, this model was very straightforward to use.

Autoencoder

As is the case with every neural network, there are many parameters that can be tuned. Here we will simply present each of the relevant parameters and our settings for them, without going into theory and mathematics behind them. A more elaborate discussion about the parameters of a neural network can be found in Appendix A.

Going back to the definition of number of layers given in Chapter 3, our AE network consisted of three layers. The encoder part of the network contained one layer, the decoder part contained one as well. The output layer of the encoder is the input layer of the decoder, so from the network wide perspective, that is the third hidden layer. All the layers used were dense or fully connected layers.

The initial, input layer, always contained m neurons, where m is the dimensionality of input data. The intermediate layers, both in the encoder and the decoder, contained 1024 neurons. The number of neurons in the latent space layer was not a fixed parameter, but was changed from experiment to experiment, in order to be able to follow the 'quality'[1] of latent space and reconstruction of the data with respect to its size. It was chosen from the set $\{5, 10, 15, \ldots, 95, 100\}$.

Activation funcitons used were ReLU and sigmoid. ReLU activation function was used in all layers except the input (for which we didn't use any activation functions) and the output layer. The output layer used a sigmoid activation function.

In order to have a fair comparison between AE and VAE, based on observations in Chapter 4, we decided to use binary cross-entorpy as the loss function for AE. We used Adam optimizer for the minimization step, as it was the only optimizer able to perform the optimization in all experiments.

In training phase, the number of epochs for AE was set to 100. An early break criterion was used, and it stated that training should be ended before it reaches 100 epochs if the loss at the end of epoch didn't decrease by at least 10^{-3} in three successive epochs. Such a criterion is a commonly used mean to prevent overfitting. Batch size was a function of the number of samples used in training. It was defined as

$$batch = round\left(\frac{samples}{30}\right) \tag{5.1}$$

in order to ensure that we have at least thirty passes through data in each epoch.

Variational Autoencoder

VAE network used in our experiments consisted of five layers. The intermediate layers used in the encoder and decoder parts of the autoencoder were used in the same form in the VAE as well. However, there were some differences in the middle part of the network. In particular, after the intermediate layer of encoder, two different latent layers were used to parametrize the mean and the standard deviation, as was explained in Chapter 3. Then, an additional sampling layer was used to obtain a sample from a normal distribution parametrized by the obtained mean and variance. Decoder was the same as in AE model. All the layers used were dense or fully connected layers, except the sampling layer.

The initial, input layer again contained m neurons, where m is the dimensionality of input data. The intermediate layers, both in encoder and decoder, contained 1024 neurons. The number of neurons in latent space layers was not a fixed parameter, but was changed from experiment to experiment, the same way as in AE model. It was again chosen from the set $\{5, 10, 15, \ldots, 95, 100\}$.

Activation funcitons used were ReLU, linear and sigmoid. ReLU activation function was used in the intermediate layers of encoder and decoder parts. Linear activation functions were used in latent layers that parametrized the mean and standard deviation of the normal distribution. Input and sampling layers didn't use any activation functions. Output layer again used a sigmoid activation function.

[1] In the sense of measures defined in Chapter 4.

As defined in (3.16), the loss function for VAE is a combination of binary cross-entorpy between input and output data and Kullback-Leibler divergence of the model's latent distribution from an assumed prior. In our experiments we used $\mathcal{N}(0,1)$ as the prior of choice. We used Adam optimizer for the minimization step, for the same resons as with AE.

In training phase, the number of epochs for VAE was set to 200. An early break criterion was used, defined the same way as for AE. The batch size used was, again, exactly the same as for AE.

5.1.3 Setting up the experiments

Classification

In order to test the quality of classification in the latent space, we had to use an algorithm for classification. Our algorithm of choice was Random Forest Classifier. The choice was based on the facts that Random Forest is an algorithm known to achieve good results and also a computationaly inexpensive one.

The classifier was implemented on encoded data and trained using three fold cross validation, each time training on two parts of the dataset and predicting labels on the third. For each prediction, F1 score was computed using scikit-learn's implementation, as defined in Chapter 4 and the final score was obtained as the average of three F1 scores calculated.

Clusterization

In order to test the quality of clusterization in latent space, we used two different approaches for two measures used.

When measuring the quality of clustering using ARI score, we used K-Means algorithm for clustering the data in latent space, using $K = true\ number\ of\ classes$. K-Means is a widely used clustering algorithm. Since we knew the number of classes for each dataset in advance, it made sense to use KMeans. After obtaining cluster labels assigned by KMeans, ARI score was computed from true and predicted labels, using scikit-learn's implementation of ARI score, as defined in Chapter 4.

When measuring the quality of clusterization using silhouette score, a different approach was used. *t-SNE* is an algorithm used for projecting higher dimensional data points to a two or three dimensional space, with the main goal - keeping neighboring relations between data points from higher dimensional space in the lower dimensional space. More on t-SNE can be found in [24]. After applying t-SNE on our encoded data and producing it to a 2D space, silhouette score was computed on 2D data, using scikit-learn's implementation of silhouette score, as defined in Chapter 4. Additionally, t-SNE was applied to output data of our models and silhouette score for this data was computed as well. This way, it can be interpreted as a signal for denoising capabilities of the models.

Reconstruction

Cosine similarity, MSE, Pearson correlation and BCE were all obtained in the same way. First, for each input sample the measure was applied with respect to its corresponding output and then the average over all samples were computed to give us the final values.

For cosine similarity and MSE scikit-learn's implementations were used, for Pearson correlation SciPy's implementation was used and for BCE Keras's implementation was used. All were based on metodologies defined in Chapter 4.

5.2 Results

5.2.1 Cross-validation

In this section we present results obtained through cross-validation. Four datasets were used: Xin, Muraro, Biase and Goolam. The same datasets were used as test datasets in the alternative, transfer learning approach. Before any experiments were run, we first reduced the number of features to match dataset standardization used in transfer learning approach, for a fair comparison. More on this approach will be said in the following subsection.

We used three fold cross-validation, which was a reasonable approach for Xin and Muraro datasets. Biase and Goolam are both smaller with respect to the number of samples, but their dimensionality is significantly higher, so due to computational restrictions, three fold cross-validation was performed on them as well. All results are presented in both tables and figures.

Muraro

The first dataset presented is Muraro. The dataset was reduced from its original size of 7117 features to 6836 features. All results are summarized in Figures 5.1, 5.2, 5.3 and 5.4 below. Additionaly, mean values with respect to latent dimension are presented in Tables 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7 and 5.8. More information about the dataset can be found in Table 2.1 and [27].

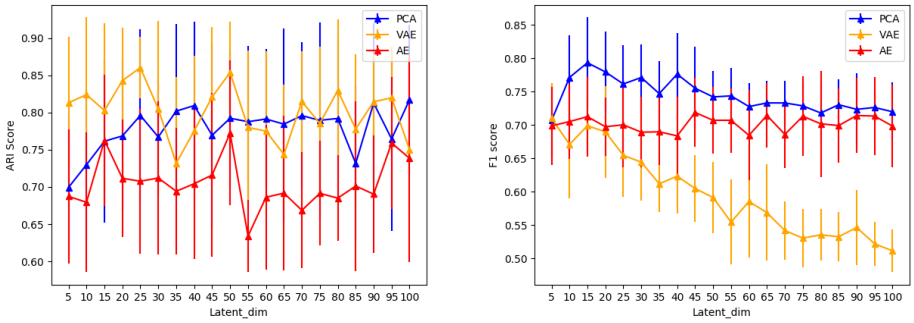

FIGURE 5.1: ARI and F1 scores - mean and standard deviation

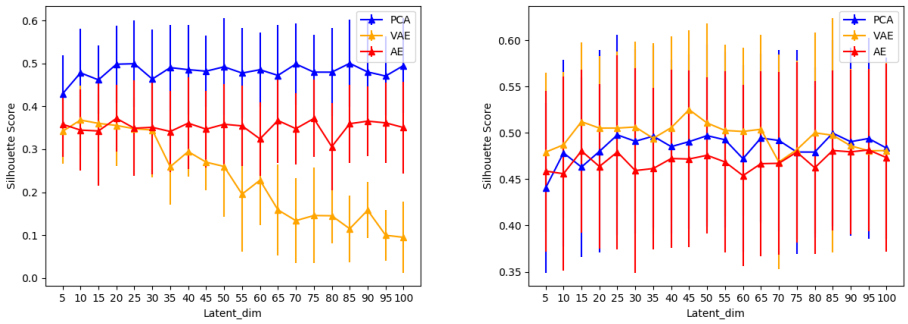

FIGURE 5.2: Silhouette scores on encoded and output data - mean and standard deviation

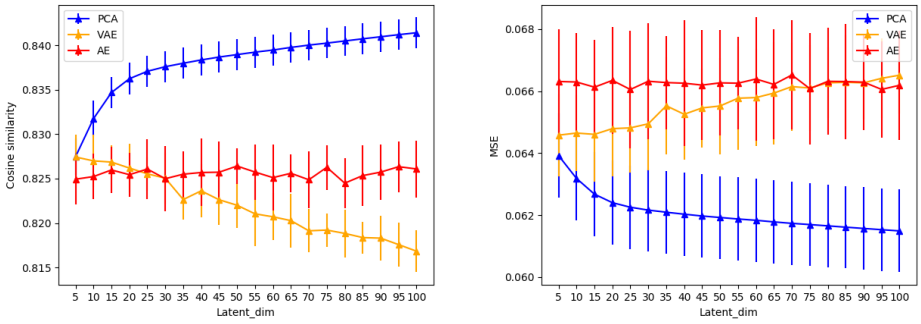

FIGURE 5.3: Cosine similarity and MSE - mean and standard deviation

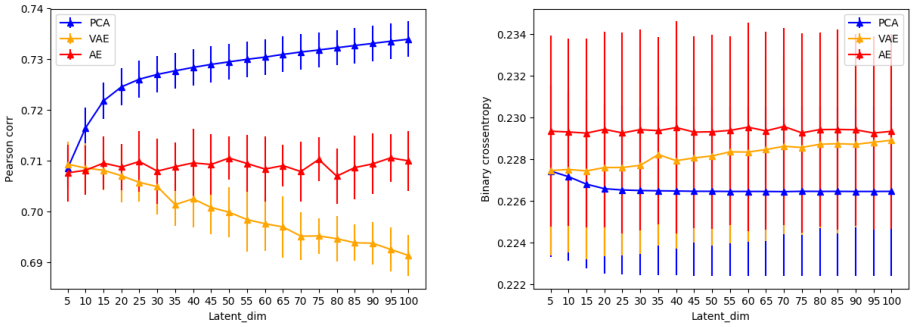

FIGURE 5.4: Pearson correlation and BCE - mean and standard deviation

From the figures above, one can conclude that, on average, PCA has the best results on this dataset. Since Muraro dataset used the fewest features in training, both neural network models achieved the early stoppage criterion in just a few epochs. This could offer an explanation for the fact that a much simpler model like PCA achieved better results than both neural network models - they slightly underfitted on this dataset. Unexpected results can be observed with respect to reconstruction quality of both AE and VAE - the increase in number of latent dimension does not yield an increase in reconstruction quality.

Xin

The second dataset presented is Xin. The dataset was reduced from its original size of 38172 features to 19415 features. All results are summarized in Figures 5.5, 5.6, 5.7 and 5.8 below. Additionaly, mean values with respect to latent dimension are presented in Tables 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7 and 5.8. More information about the dataset can be found in Table 2.1 and [39].

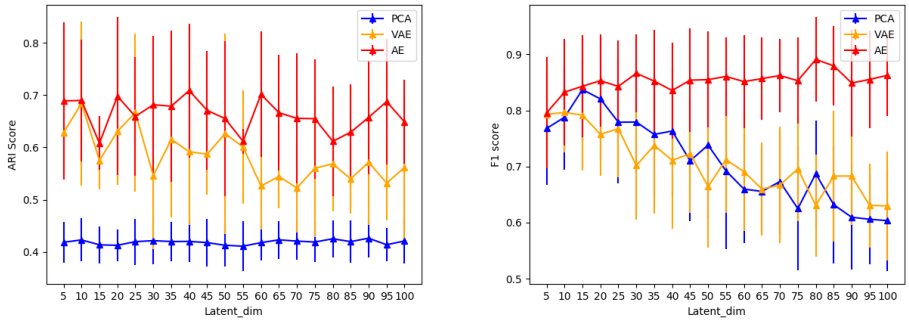

FIGURE 5.5: ARI and F1 scores - mean and standard deviation

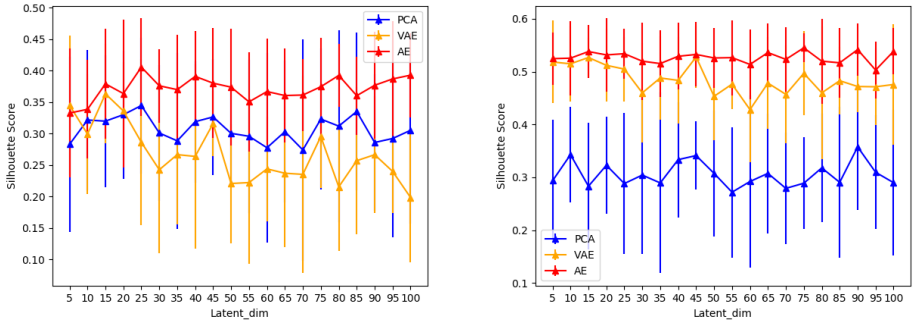

FIGURE 5.6: Silhouette scores on encoded and output data - mean and standard deviation

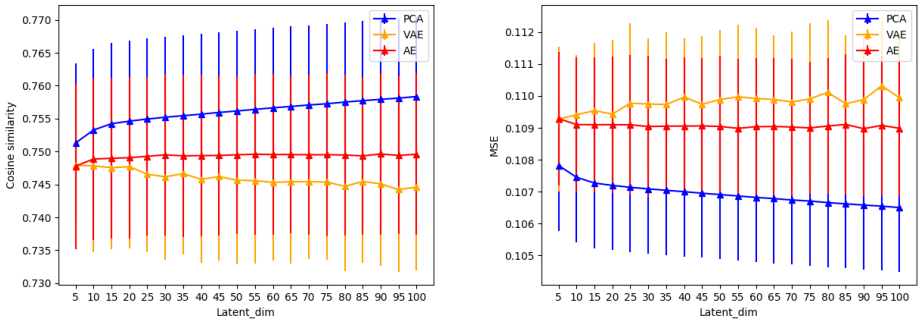

FIGURE 5.7: Cosine similarity and MSE - mean and standard deviation

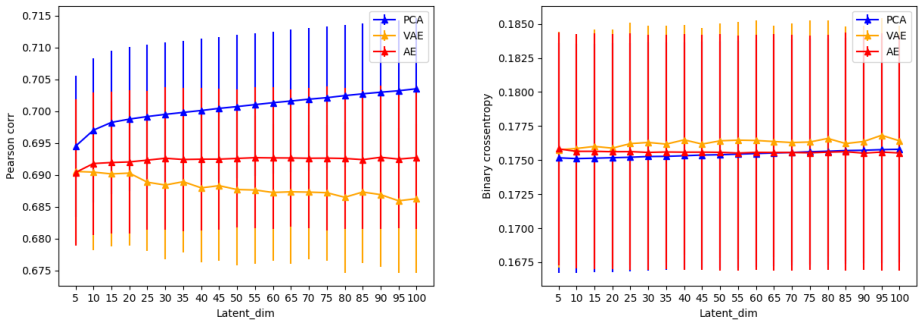

FIGURE 5.8: Pearson correlation and BCE - mean and standard deviation

From the figures above, one can conclude that, AE yields best results on this dataset. Again the increase in number of latent dimensions does not yield an increase in reconstruction quality.

Goolam

The third dataset presented is Goolam. The dataset was reduced from its original size of 40405 features to 19927 features. All results are summarized in Figures 5.9, 5.10, 5.11 and 5.12 below. Additionaly, mean values with respect to latent dimension are presented in Tables 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7 and 5.8. More information about the dataset can be found in Table 2.1 and [13].

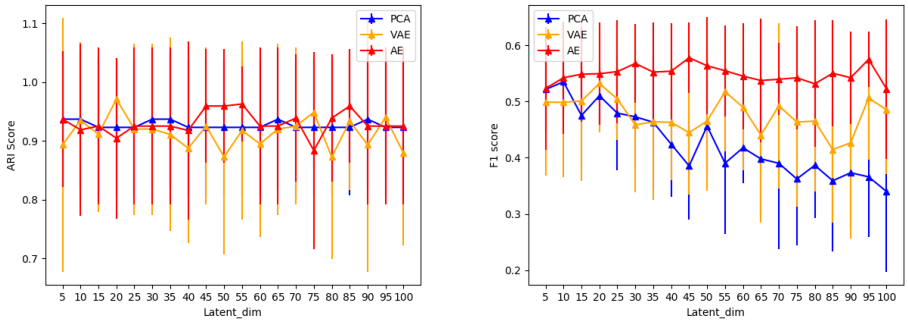

FIGURE 5.9: ARI and F1 scores - mean and standard deviation

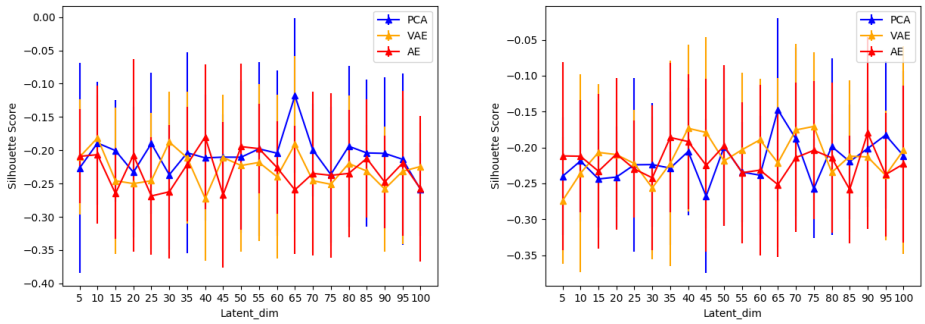

FIGURE 5.10: Silhouette scores on encoded and output data - mean and standard deviation

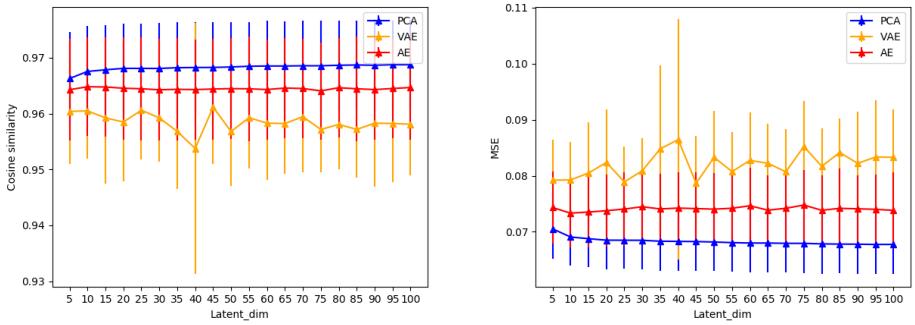

FIGURE 5.11: Cosine similarity and MSE - mean and standard deviation

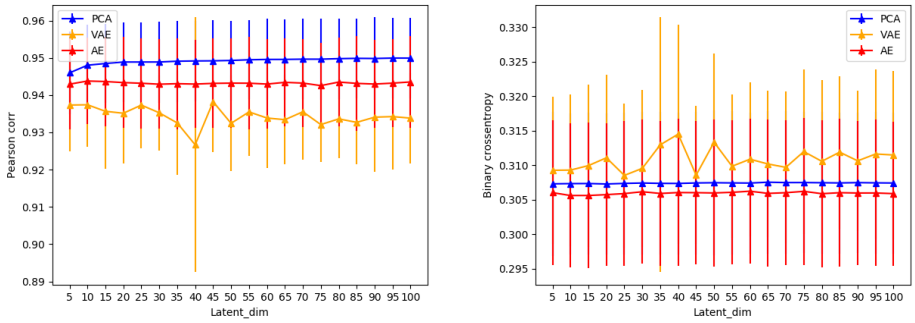

FIGURE 5.12: Pearson correlation and BCE - mean and standard deviation

Again, AE yields the best results based on figures above. On this dataset the increase in number of latent dimension does not yield an increase in reconstruction quality for both the neural network models as well as PCA.

Biase

The final dataset presented in the cross validation section is Biase. The dataset was reduced from its original size of 25114 features to 16229 features. All results are

summarized in Figures 5.13, 5.14, 5.15 and 5.16 below. Additionaly, mean values with respect to latent dimension are presented in Tables 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7 and 5.8. More information about the dataset can be found in Table 2.1 and [4].

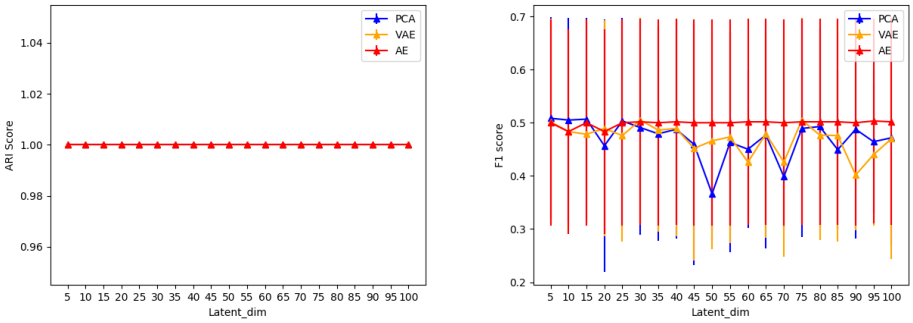

FIGURE 5.13: ARI and F1 scores - mean and standard deviation

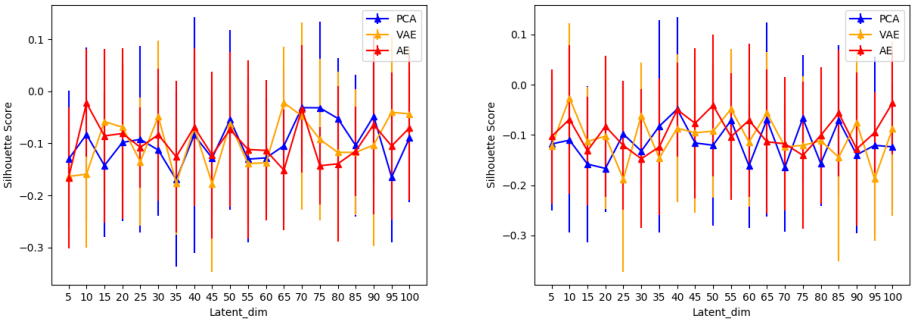

FIGURE 5.14: Silhouette scores on encoded and output data - mean and standard deviation

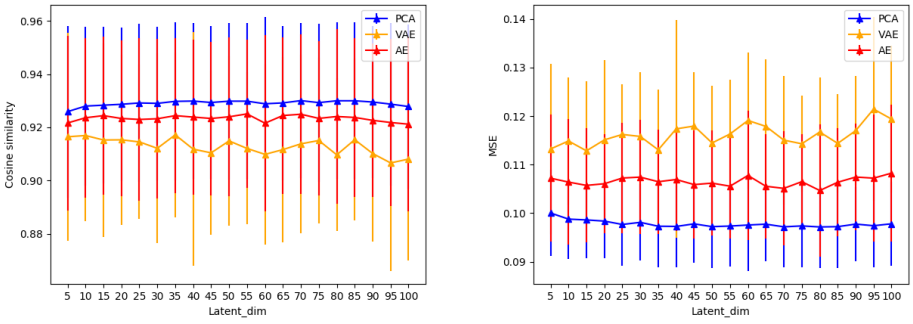

FIGURE 5.15: Cosine similarity and MSE - mean and standard deviation

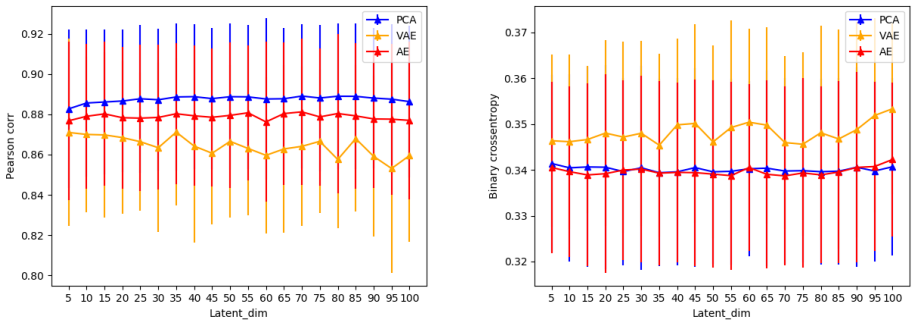

FIGURE 5.16: Pearson correlation and BCE - mean and standard deviation

Biase, the smallest data set, yields a perfect ARI score for all three models. Based on other metrics, one could give the nod to AE, but only by a slim margin. Again the increase in number of latent dimension does not yield an increase in reconstruction quality. Tables that summarize mean values for each metric and each dataset are presented below.

	Muraro			Xin		
Latent dim	VAE	AE	PCA	VAE	AE	PCA
5	0.813	0.687	0.699	0.628	0.689	0.419
10	0.824	0.679	0.73	0.684	0.69	0.423
15	0.803	0.763	0.761	0.575	0.608	0.414
20	0.843	0.712	0.769	0.63	0.698	0.413
25	0.86	0.708	0.796	0.666	0.659	0.42
30	0.805	0.712	0.767	0.546	0.681	0.422
35	0.732	0.694	0.802	0.615	0.679	0.42
40	0.775	0.704	0.809	0.592	0.709	0.42
45	0.821	0.716	0.77	0.587	0.671	0.418
50	0.854	0.772	0.792	0.626	0.655	0.413
55	0.78	0.634	0.788	0.601	0.612	0.411
60	0.775	0.686	0.791	0.526	0.702	0.418
65	0.744	0.692	0.784	0.544	0.666	0.423
70	0.815	0.669	0.796	0.523	0.655	0.421
75	0.785	0.691	0.79	0.56	0.655	0.419
80	0.83	0.685	0.792	0.569	0.612	0.425
85	0.778	0.701	0.732	0.54	0.629	0.42
90	0.815	0.69	0.813	0.572	0.657	0.426
95	0.82	0.759	0.765	0.532	0.687	0.414
100	0.75	0.739	0.817	0.562	0.649	0.421
	Goolam			Biase		
Latent dim	VAE	AE	PCA	VAE	AE	PCA
5	0.894	0.937	0.937	1.0	1.0	1.0
10	0.935	0.919	0.937	1.0	1.0	1.0
15	0.911	0.925	0.923	1.0	1.0	1.0
20	0.971	0.905	0.923	1.0	1.0	1.0
25	0.92	0.925	0.923	1.0	1.0	1.0
30	0.92	0.925	0.937	1.0	1.0	1.0
35	0.911	0.925	0.937	1.0	1.0	1.0
40	0.888	0.918	0.923	1.0	1.0	1.0
45	0.925	0.96	0.923	1.0	1.0	1.0
50	0.873	0.96	0.923	1.0	1.0	1.0
55	0.918	0.963	0.923	1.0	1.0	1.0
60	0.895	0.925	0.923	1.0	1.0	1.0
65	0.92	0.925	0.937	1.0	1.0	1.0
70	0.925	0.939	0.923	1.0	1.0	1.0
75	0.949	0.884	0.923	1.0	1.0	1.0
80	0.873	0.939	0.923	1.0	1.0	1.0
85	0.934	0.96	0.923	1.0	1.0	1.0
90	0.893	0.925	0.937	1.0	1.0	1.0
95	0.94	0.925	0.923	1.0	1.0	1.0
100	0.879	0.925	0.923	1.0	1.0	1.0

TABLE 5.1: ARI scores - mean values with respect to the size of latent dimension

	Muraro			Xin		
Latent dim	VAE	AE	PCA	VAE	AE	PCA
5	0.71	0.699	0.707	0.793	0.796	0.768
10	0.671	0.705	0.771	0.796	0.832	0.788
15	0.699	0.712	0.793	0.792	0.843	0.837
20	0.69	0.697	0.78	0.757	0.853	0.82
25	0.654	0.7	0.761	0.767	0.843	0.779
30	0.644	0.689	0.771	0.702	0.866	0.779
35	0.612	0.69	0.747	0.738	0.852	0.757
40	0.623	0.684	0.776	0.711	0.836	0.763
45	0.605	0.719	0.755	0.723	0.854	0.71
50	0.591	0.707	0.742	0.664	0.855	0.738
55	0.554	0.707	0.744	0.712	0.86	0.692
60	0.585	0.684	0.728	0.691	0.851	0.66
65	0.568	0.714	0.733	0.66	0.857	0.656
70	0.542	0.686	0.733	0.667	0.862	0.673
75	0.53	0.713	0.729	0.696	0.853	0.625
80	0.535	0.702	0.718	0.631	0.891	0.687
85	0.532	0.699	0.73	0.683	0.879	0.632
90	0.546	0.714	0.723	0.683	0.849	0.61
95	0.522	0.713	0.726	0.631	0.855	0.606
100	0.511	0.698	0.72	0.63	0.863	0.603
	Goolam			Biase		
Latent dim	VAE	AE	PCA	VAE	AE	PCA
5	0.499	0.523	0.521	0.502	0.5	0.508
10	0.499	0.542	0.535	0.483	0.483	0.505
15	0.501	0.549	0.475	0.478	0.5	0.507
20	0.532	0.549	0.51	0.489	0.483	0.457
25	0.505	0.553	0.479	0.476	0.5	0.503
30	0.459	0.568	0.473	0.505	0.502	0.491
35	0.464	0.552	0.463	0.486	0.5	0.479
40	0.463	0.554	0.424	0.489	0.502	0.488
45	0.445	0.578	0.386	0.452	0.5	0.459
50	0.465	0.564	0.456	0.466	0.5	0.366
55	0.517	0.555	0.39	0.473	0.5	0.463
60	0.49	0.545	0.418	0.426	0.502	0.45
65	0.44	0.537	0.398	0.479	0.502	0.477
70	0.493	0.54	0.39	0.426	0.5	0.398
75	0.464	0.542	0.362	0.505	0.502	0.489
80	0.465	0.532	0.387	0.477	0.502	0.493
85	0.415	0.55	0.359	0.476	0.502	0.449
90	0.427	0.542	0.373	0.402	0.5	0.488
95	0.507	0.575	0.366	0.44	0.503	0.464
100	0.485	0.523	0.34	0.47	0.502	0.472

TABLE 5.2: F1 scores - mean values with respect to the size of latent dimension

	Muraro			Xin		
Latent dim	VAE	AE	PCA	VAE	AE	PCA
5	0.342	0.358	0.429	0.345	0.333	0.283
10	0.368	0.345	0.478	0.299	0.338	0.322
15	0.36	0.343	0.461	0.363	0.379	0.32
20	0.355	0.372	0.498	0.336	0.363	0.33
25	0.348	0.349	0.499	0.286	0.406	0.345
30	0.345	0.351	0.463	0.243	0.376	0.301
35	0.259	0.341	0.49	0.267	0.37	0.288
40	0.295	0.361	0.485	0.264	0.391	0.319
45	0.27	0.347	0.482	0.316	0.38	0.326
50	0.26	0.358	0.492	0.221	0.374	0.3
55	0.195	0.354	0.477	0.222	0.35	0.296
60	0.228	0.324	0.485	0.244	0.367	0.278
65	0.159	0.367	0.472	0.237	0.36	0.303
70	0.134	0.348	0.498	0.235	0.361	0.274
75	0.146	0.371	0.48	0.295	0.374	0.323
80	0.145	0.306	0.479	0.215	0.392	0.312
85	0.115	0.359	0.5	0.257	0.36	0.335
90	0.158	0.365	0.48	0.266	0.377	0.286
95	0.1	0.361	0.47	0.24	0.387	0.292
100	0.095	0.35	0.495	0.199	0.393	0.305
	Goolam			Biase		
Latent dim	VAE	AE	PCA	VAE	AE	PCA
5	-0.21	-0.209	-0.227	-0.164	-0.166	-0.13
10	-0.181	-0.207	-0.189	-0.159	-0.023	-0.083
15	-0.246	-0.265	-0.2	-0.058	-0.086	-0.143
20	-0.25	-0.207	-0.233	-0.069	-0.081	-0.098
25	-0.246	-0.269	-0.19	-0.135	-0.108	-0.092
30	-0.188	-0.262	-0.237	-0.049	-0.084	-0.113
35	-0.211	-0.221	-0.204	-0.176	-0.126	-0.17
40	-0.272	-0.18	-0.211	-0.07	-0.069	-0.084
45	-0.211	-0.267	-0.21	-0.177	-0.123	-0.128
50	-0.223	-0.194	-0.21	-0.062	-0.073	-0.055
55	-0.218	-0.198	-0.198	-0.139	-0.112	-0.13
60	-0.24	-0.226	-0.204	-0.137	-0.114	-0.128
65	-0.191	-0.259	-0.118	-0.022	-0.152	-0.105
70	-0.246	-0.235	-0.2	-0.047	-0.034	-0.032
75	-0.251	-0.238	-0.236	-0.093	-0.143	-0.032
80	-0.22	-0.235	-0.194	-0.117	-0.14	-0.052
85	-0.231	-0.212	-0.204	-0.117	-0.115	-0.104
90	-0.258	-0.248	-0.205	-0.103	-0.064	-0.048
95	-0.232	-0.219	-0.214	-0.04	-0.105	-0.165
100	-0.224	-0.258	-0.259	-0.044	-0.07	-0.089

TABLE 5.3: Silhouette scores on encoded data - mean values with respect to the size of latent dimension

	Muraro			Xin		
Latent dim	VAE	AE	PCA	VAE	AE	PCA
5	0.479	0.459	0.44	0.518	0.525	0.293
10	0.487	0.456	0.478	0.515	0.526	0.343
15	0.512	0.481	0.463	0.527	0.538	0.283
20	0.505	0.464	0.48	0.512	0.532	0.323
25	0.505	0.479	0.498	0.505	0.534	0.288
30	0.506	0.459	0.491	0.46	0.52	0.304
35	0.494	0.462	0.497	0.488	0.515	0.289
40	0.506	0.472	0.485	0.483	0.529	0.333
45	0.525	0.472	0.491	0.528	0.533	0.341
50	0.511	0.476	0.497	0.453	0.526	0.307
55	0.503	0.469	0.493	0.476	0.527	0.272
60	0.502	0.454	0.472	0.428	0.514	0.292
65	0.504	0.467	0.495	0.478	0.536	0.307
70	0.469	0.467	0.492	0.457	0.523	0.279
75	0.482	0.479	0.479	0.497	0.545	0.289
80	0.5	0.462	0.479	0.46	0.52	0.318
85	0.498	0.481	0.5	0.483	0.517	0.29
90	0.486	0.48	0.491	0.472	0.542	0.358
95	0.481	0.482	0.494	0.471	0.503	0.31
100	0.481	0.473	0.484	0.476	0.539	0.29
	Goolam			Biase		
Latent dim	VAE	AE	PCA	VAE	AE	PCA
5	-0.274	-0.212	-0.24	-0.122	-0.103	-0.118
10	-0.236	-0.212	-0.219	-0.026	-0.069	-0.111
15	-0.207	-0.233	-0.244	-0.113	-0.132	-0.158
20	-0.21	-0.209	-0.241	-0.103	-0.083	-0.167
25	-0.222	-0.23	-0.224	-0.188	-0.121	-0.098
30	-0.256	-0.242	-0.224	-0.061	-0.148	-0.132
35	-0.222	-0.186	-0.229	-0.146	-0.124	-0.083
40	-0.173	-0.192	-0.206	-0.087	-0.05	-0.049
45	-0.179	-0.225	-0.268	-0.096	-0.077	-0.116
50	-0.219	-0.198	-0.199	-0.093	-0.041	-0.121
55	-0.203	-0.235	-0.234	-0.049	-0.104	-0.071
60	-0.189	-0.232	-0.239	-0.114	-0.071	-0.162
65	-0.221	-0.252	-0.148	-0.056	-0.114	-0.069
70	-0.176	-0.214	-0.188	-0.124	-0.118	-0.165
75	-0.171	-0.204	-0.257	-0.121	-0.141	-0.067
80	-0.235	-0.214	-0.199	-0.112	-0.101	-0.157
85	-0.212	-0.259	-0.22	-0.145	-0.056	-0.071
90	-0.213	-0.18	-0.201	-0.075	-0.128	-0.14
95	-0.239	-0.238	-0.183	-0.188	-0.095	-0.121
100	-0.204	-0.223	-0.212	-0.088	-0.036	-0.124

TABLE 5.4: Silhouette scores on output data - mean values with respect to the size of latent dimension

	Muraro			Xin		
Latent dim	VAE	AE	PCA	VAE	AE	PCA
5	0.827	0.825	0.827	0.748	0.748	0.751
10	0.827	0.825	0.832	0.748	0.749	0.753
15	0.827	0.826	0.835	0.748	0.749	0.754
20	0.826	0.825	0.836	0.748	0.749	0.755
25	0.826	0.826	0.837	0.747	0.749	0.755
30	0.825	0.825	0.838	0.746	0.749	0.755
35	0.823	0.826	0.838	0.747	0.749	0.755
40	0.824	0.826	0.838	0.746	0.749	0.756
45	0.823	0.826	0.839	0.746	0.749	0.756
50	0.822	0.826	0.839	0.746	0.75	0.756
55	0.821	0.826	0.839	0.746	0.75	0.756
60	0.821	0.825	0.839	0.745	0.75	0.757
65	0.82	0.826	0.84	0.745	0.75	0.757
70	0.819	0.825	0.84	0.745	0.75	0.757
75	0.819	0.826	0.84	0.745	0.75	0.757
80	0.819	0.824	0.84	0.745	0.749	0.758
85	0.818	0.825	0.841	0.745	0.749	0.758
90	0.818	0.826	0.841	0.745	0.75	0.758
95	0.818	0.826	0.841	0.744	0.749	0.758
100	0.817	0.826	0.841	0.745	0.75	0.758
	Goolam			Biase		
Latent dim	VAE	AE	PCA	VAE	AE	PCA
5	0.96	0.964	0.966	0.916	0.922	0.926
10	0.96	0.965	0.968	0.917	0.924	0.928
15	0.959	0.965	0.968	0.915	0.924	0.928
20	0.958	0.965	0.968	0.915	0.923	0.929
25	0.961	0.964	0.968	0.915	0.923	0.929
30	0.959	0.964	0.968	0.912	0.923	0.929
35	0.957	0.964	0.968	0.917	0.924	0.93
40	0.954	0.964	0.968	0.912	0.924	0.93
45	0.961	0.964	0.968	0.91	0.923	0.929
50	0.957	0.964	0.968	0.915	0.924	0.93
55	0.959	0.964	0.968	0.912	0.925	0.93
60	0.958	0.964	0.969	0.91	0.922	0.929
65	0.958	0.965	0.969	0.912	0.924	0.929
70	0.959	0.964	0.969	0.914	0.925	0.93
75	0.957	0.964	0.969	0.915	0.923	0.929
80	0.958	0.965	0.969	0.91	0.924	0.93
85	0.957	0.964	0.969	0.915	0.924	0.93
90	0.958	0.964	0.969	0.91	0.923	0.93
95	0.958	0.964	0.969	0.907	0.922	0.929
100	0.958	0.965	0.969	0.908	0.921	0.928

TABLE 5.5: Cosine similarity - mean values with respect to the size of latent dimension

	Muraro			Xin		
Latent dim	VAE	AE	PCA	VAE	AE	PCA
5	0.065	0.066	0.064	0.109	0.109	0.108
10	0.065	0.066	0.063	0.109	0.109	0.107
15	0.065	0.066	0.063	0.11	0.109	0.107
20	0.065	0.066	0.062	0.109	0.109	0.107
25	0.065	0.066	0.062	0.11	0.109	0.107
30	0.065	0.066	0.062	0.11	0.109	0.107
35	0.066	0.066	0.062	0.11	0.109	0.107
40	0.065	0.066	0.062	0.11	0.109	0.107
45	0.065	0.066	0.062	0.11	0.109	0.107
50	0.066	0.066	0.062	0.11	0.109	0.107
55	0.066	0.066	0.062	0.11	0.109	0.107
60	0.066	0.066	0.062	0.11	0.109	0.107
65	0.066	0.066	0.062	0.11	0.109	0.107
70	0.066	0.067	0.062	0.11	0.109	0.107
75	0.066	0.066	0.062	0.11	0.109	0.107
80	0.066	0.066	0.062	0.11	0.109	0.107
85	0.066	0.066	0.062	0.11	0.109	0.107
90	0.066	0.066	0.062	0.11	0.109	0.107
95	0.066	0.066	0.062	0.11	0.109	0.107
100	0.067	0.066	0.061	0.11	0.109	0.106
	Goolam			Biase		
Latent dim	VAE	AE	PCA	VAE	AE	PCA
5	0.079	0.074	0.071	0.113	0.107	0.1
10	0.079	0.073	0.069	0.115	0.106	0.099
15	0.08	0.074	0.069	0.113	0.106	0.099
20	0.082	0.074	0.069	0.115	0.106	0.098
25	0.079	0.074	0.068	0.116	0.107	0.098
30	0.081	0.074	0.068	0.116	0.107	0.098
35	0.085	0.074	0.068	0.113	0.106	0.097
40	0.086	0.074	0.068	0.117	0.107	0.097
45	0.079	0.074	0.068	0.118	0.106	0.098
50	0.083	0.074	0.068	0.114	0.106	0.097
55	0.081	0.074	0.068	0.116	0.106	0.097
60	0.083	0.075	0.068	0.119	0.108	0.098
65	0.082	0.074	0.068	0.118	0.106	0.098
70	0.081	0.074	0.068	0.115	0.105	0.097
75	0.085	0.075	0.068	0.114	0.106	0.097
80	0.082	0.074	0.068	0.117	0.105	0.097
85	0.084	0.074	0.068	0.114	0.106	0.097
90	0.082	0.074	0.068	0.117	0.107	0.098
95	0.083	0.074	0.068	0.121	0.107	0.097
100	0.083	0.074	0.068	0.119	0.108	0.098

TABLE 5.6: MSE - mean values with respect to the size of latent dimension

	Muraro			Xin		
Latent dim	VAE	AE	PCA	VAE	AE	PCA
5	0.709	0.708	0.708	0.691	0.69	0.695
10	0.709	0.708	0.716	0.69	0.692	0.697
15	0.708	0.71	0.722	0.69	0.692	0.698
20	0.707	0.709	0.725	0.69	0.692	0.699
25	0.706	0.71	0.726	0.689	0.692	0.699
30	0.705	0.708	0.727	0.688	0.693	0.7
35	0.701	0.709	0.728	0.689	0.692	0.7
40	0.703	0.71	0.728	0.688	0.692	0.7
45	0.701	0.709	0.729	0.688	0.692	0.7
50	0.7	0.711	0.729	0.688	0.693	0.701
55	0.698	0.709	0.73	0.688	0.693	0.701
60	0.698	0.708	0.73	0.687	0.693	0.701
65	0.697	0.709	0.731	0.687	0.693	0.702
70	0.695	0.708	0.731	0.687	0.693	0.702
75	0.695	0.71	0.732	0.687	0.693	0.702
80	0.695	0.707	0.732	0.687	0.693	0.702
85	0.694	0.709	0.733	0.687	0.692	0.703
90	0.694	0.709	0.733	0.687	0.693	0.703
95	0.693	0.711	0.734	0.686	0.693	0.703
100	0.691	0.71	0.734	0.686	0.693	0.704
	Goolam			Biase		
Latent dim	VAE	AE	PCA	VAE	AE	PCA
5	0.937	0.943	0.946	0.871	0.877	0.883
10	0.937	0.944	0.948	0.87	0.879	0.886
15	0.936	0.944	0.948	0.87	0.88	0.886
20	0.935	0.943	0.949	0.868	0.878	0.887
25	0.937	0.943	0.949	0.866	0.878	0.888
30	0.935	0.943	0.949	0.863	0.878	0.887
35	0.933	0.943	0.949	0.871	0.88	0.889
40	0.927	0.943	0.949	0.864	0.879	0.889
45	0.938	0.943	0.949	0.861	0.879	0.888
50	0.932	0.943	0.949	0.867	0.879	0.889
55	0.935	0.943	0.949	0.863	0.881	0.889
60	0.934	0.943	0.95	0.86	0.876	0.888
65	0.933	0.943	0.95	0.863	0.88	0.888
70	0.936	0.943	0.95	0.864	0.881	0.889
75	0.932	0.943	0.95	0.867	0.879	0.888
80	0.934	0.943	0.95	0.858	0.88	0.889
85	0.933	0.943	0.95	0.868	0.879	0.889
90	0.934	0.943	0.95	0.859	0.878	0.888
95	0.934	0.943	0.95	0.853	0.878	0.888
100	0.934	0.944	0.95	0.86	0.877	0.886

TABLE 5.7: Pearson correlation - mean values with respect to the size of latent dimension

	Muraro			Xin		
Latent dim	VAE	AE	PCA	VAE	AE	PCA
5	0.227	0.229	0.227	0.176	0.176	0.175
10	0.228	0.229	0.227	0.176	0.176	0.175
15	0.227	0.229	0.227	0.176	0.176	0.175
20	0.228	0.229	0.227	0.176	0.176	0.175
25	0.228	0.229	0.227	0.176	0.176	0.175
30	0.228	0.229	0.227	0.176	0.176	0.175
35	0.228	0.229	0.226	0.176	0.176	0.175
40	0.228	0.23	0.226	0.176	0.176	0.175
45	0.228	0.229	0.226	0.176	0.176	0.175
50	0.228	0.229	0.226	0.176	0.176	0.175
55	0.228	0.229	0.226	0.176	0.176	0.175
60	0.228	0.23	0.226	0.176	0.176	0.175
65	0.228	0.229	0.226	0.176	0.176	0.176
70	0.229	0.23	0.226	0.176	0.176	0.176
75	0.229	0.229	0.226	0.176	0.176	0.176
80	0.229	0.229	0.226	0.177	0.176	0.176
85	0.229	0.229	0.226	0.176	0.176	0.176
90	0.229	0.229	0.226	0.176	0.176	0.176
95	0.229	0.229	0.226	0.177	0.176	0.176
100	0.229	0.229	0.226	0.176	0.176	0.176
	Goolam			Biase		
Latent dim	VAE	AE	PCA	VAE	AE	PCA
5	0.309	0.306	0.307	0.346	0.341	0.341
10	0.309	0.306	0.307	0.346	0.34	0.34
15	0.31	0.306	0.307	0.347	0.339	0.341
20	0.311	0.306	0.307	0.348	0.339	0.341
25	0.309	0.306	0.307	0.347	0.34	0.34
30	0.31	0.306	0.307	0.348	0.34	0.34
35	0.313	0.306	0.307	0.345	0.339	0.339
40	0.315	0.306	0.307	0.35	0.339	0.34
45	0.309	0.306	0.307	0.35	0.339	0.34
50	0.313	0.306	0.307	0.346	0.339	0.34
55	0.31	0.306	0.307	0.349	0.339	0.34
60	0.311	0.306	0.307	0.35	0.34	0.34
65	0.31	0.306	0.308	0.35	0.339	0.34
70	0.31	0.306	0.307	0.346	0.339	0.34
75	0.312	0.306	0.307	0.346	0.339	0.34
80	0.311	0.306	0.307	0.348	0.339	0.34
85	0.312	0.306	0.307	0.347	0.34	0.34
90	0.311	0.306	0.307	0.349	0.341	0.341
95	0.312	0.306	0.307	0.352	0.341	0.34
100	0.311	0.306	0.307	0.353	0.342	0.341

TABLE 5.8: BCE - mean values with respect to the size of latent dimension

5.2.2 Transfer learning

In this section we present results obtained through the second, transfer learning approach. Six combinations were used: Baron-Muraro, Baron-Xin, Xin-Muraro, Deng-Goolam, Deng-Biase and Goolam-Biase. The first dataset was always used for training, while the second one was always used as testing dataset.

Before any experiments were run, we first identified genes found in both datasets[2] and only those genes were used as features in training. This helped standardize the data as well as acting like an initial dimensionality reduction step. All results are presented in both tables and figures.

Baron-Muraro

The first pair presented is Baron-Muraro. The number of genes found in both datasets was 6836, so Baron was initially reduced from 20125 genes to 6836, while Muraro was reduced from 7117 genes to 6836. All results are summarized in Figures 5.17, 5.18, 5.19 and 5.20 below. Additionaly, mean values with respect to latent dimension are presented in Tables 5.9, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15 and 5.16. More information about the dataset can be found in Table 2.1 and [3], [27].

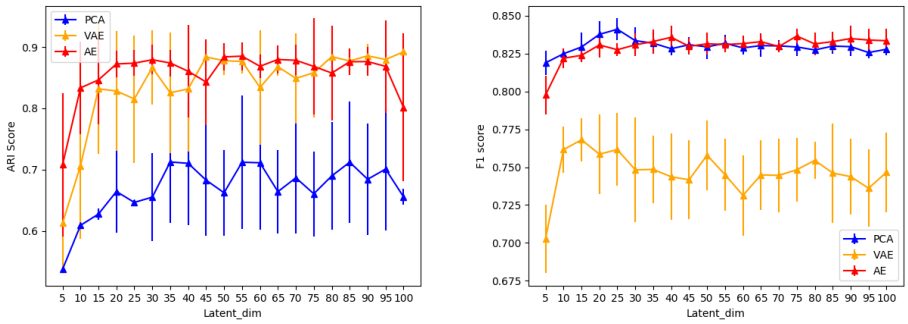

FIGURE 5.17: ARI and F1 scores - mean and standard deviation

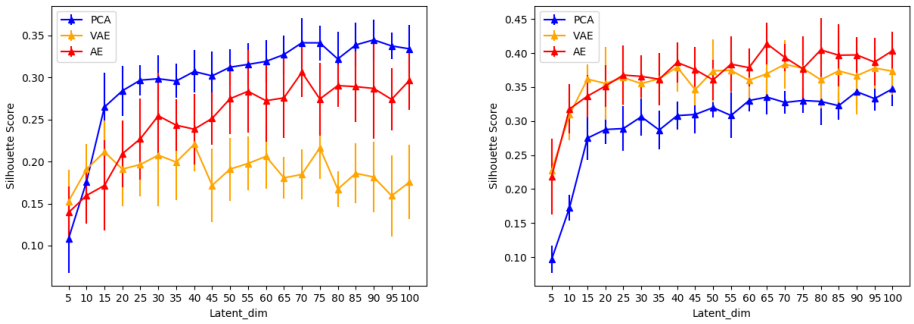

FIGURE 5.18: Silhouette scores on encoded and output data - mean and standard deviation

[2]Refering to each of the train-test pairs introduced.

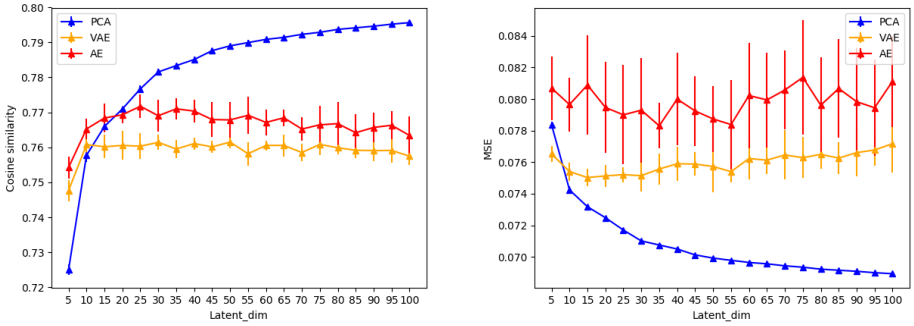

FIGURE 5.19: Cosine similarity and MSE - mean and standard deviation

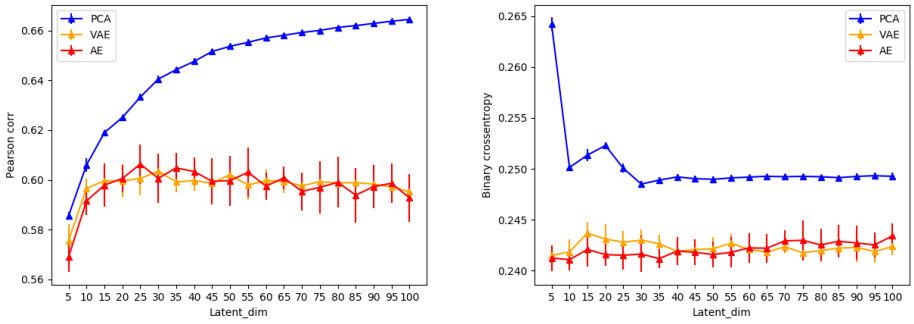

FIGURE 5.20: Pearson correlation and BCE - mean and standard deviation

From the figures above, one can conclude that, on average, AE has the best results on this pair. Again the increase in number of latent dimension does not yield an increase in reconstruction quality for both AE and VAE.

Baron-Xin

The second pair presented is Baron-Xin. The number of genes found in both datasets was 19415, so Baron was initially reduced from 20125 genes to 19415, while Xin was reduced from 38172 genes to 19415. All results are summarized in Figures 5.21, 5.22, 5.23 and 5.24 below. Additionaly, mean values with respect to latent dimension are presented in Tables 5.9, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15 and 5.16. More information about the dataset can be found in Table 2.1 and [3], [39].

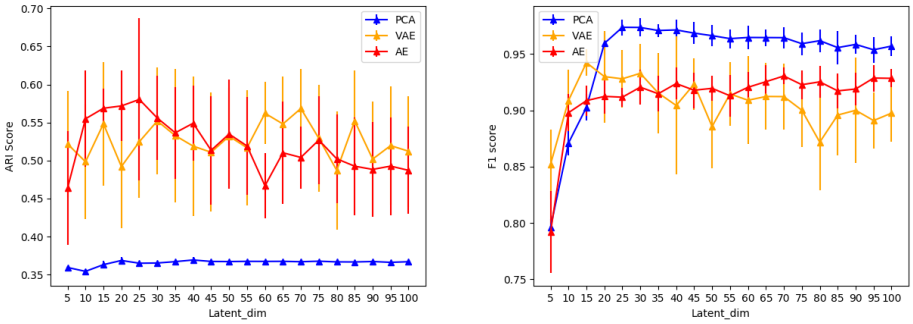

FIGURE 5.21: ARI and F1 scores - mean and standard deviation

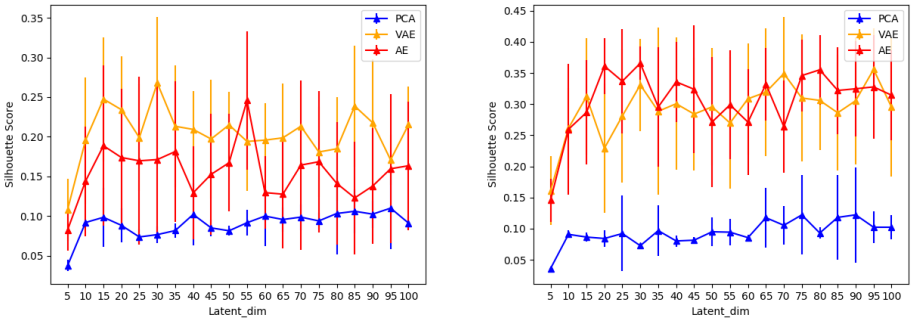

FIGURE 5.22: Silhouette scores on encoded and output data - mean and standard deviation

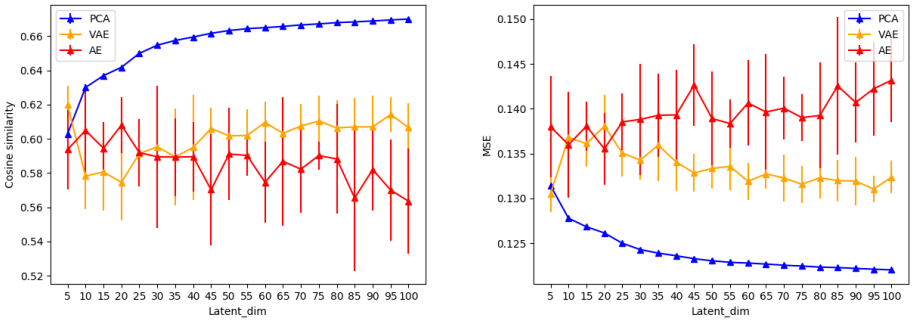

FIGURE 5.23: Cosine similarity and MSE - mean and standard deviation

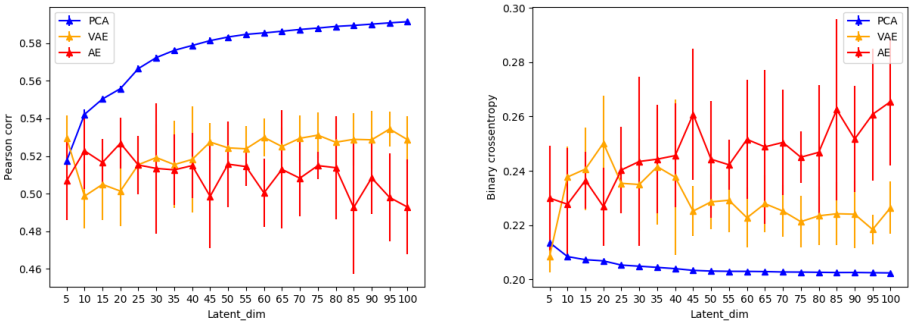

FIGURE 5.24: Pearson correlation and BCE - mean and standard deviation

For this pair, one can argue that both AE and VAE hold the advantage over PCA. Again the increase in number of latent dimension does not yield an increase in reconstruction quality for both neural network models, even the contrary can be observed.

Xin-Muraro

The third pair presented is Xin-Muraro. The number of genes found in both datasets was 6769, so Xin was initially reduced from 38172 genes to 6769, while Muraro was reduced from 7117 genes to 6769. All results are summarized in Figures 5.25, 5.26, 5.27 and 5.28 below. Additionaly, mean values with respect to the latent dimension are presented in Tables 5.9, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15 and 5.16. More information about the dataset can be found in Table 2.1 and [39], [27].

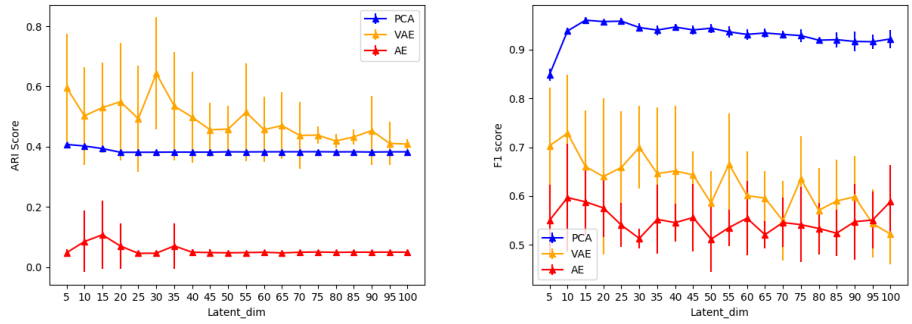

FIGURE 5.25: ARI and F1 scores - mean and standard deviation

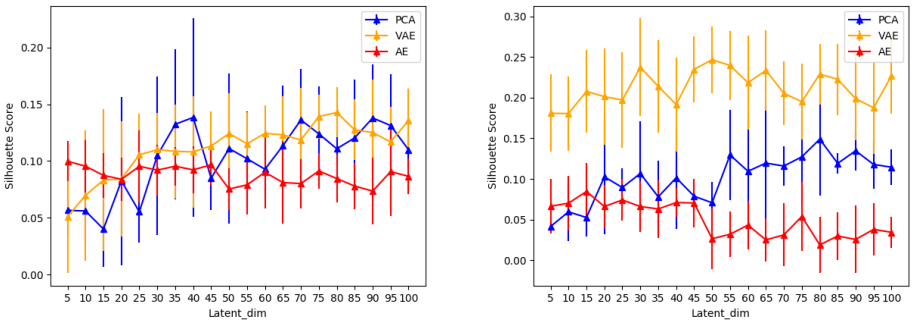

FIGURE 5.26: Silhouette scores on encoded and output data - mean and standard deviation

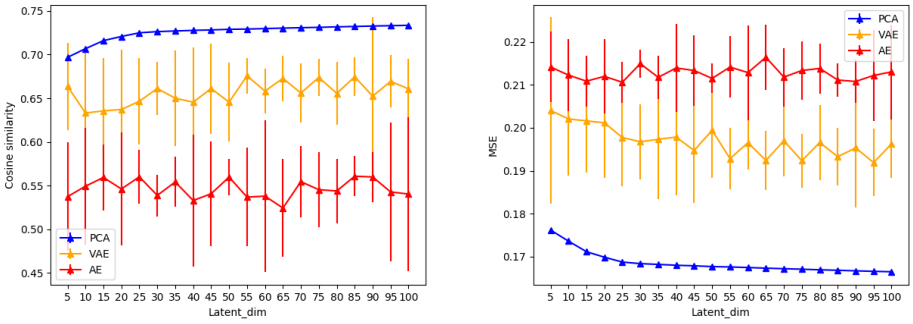

FIGURE 5.27: Cosine similarity and MSE - mean and standard deviation

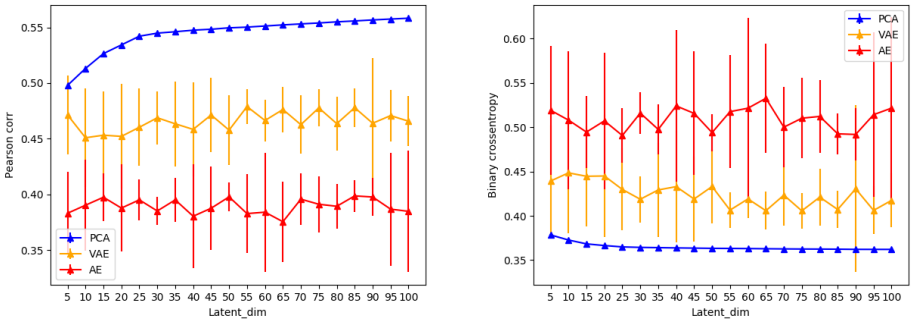

FIGURE 5.28: Pearson correlation and BCE - mean and standard deviation

From the figures above, one can conclude that, on average, PCA has the best results on this pair. On this dataset pair both neural network models achieved early stoppage criterion in just a few iterations, which could be a reason why PCA achieves a much better result here.

Deng-Goolam

The fourth pair presented is Deng-Goolam. The number of genes found in both datasets was 19927, so Deng was initially reduced from 22958 genes to 19927, while Goolam was reduced from 40405 genes to 19927. All results are summarized in Figures 5.29, 5.30, 5.31 and 5.32 below. Additionaly, mean values with respect to latent dimension are presented in Tables 5.9, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15 and 5.16. More information about the dataset can be found in Table 2.1 and [9], [13].

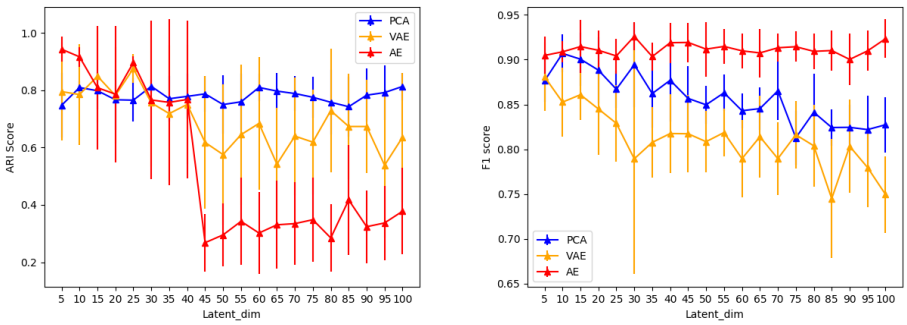

FIGURE 5.29: ARI and F1 scores - mean and standard deviation

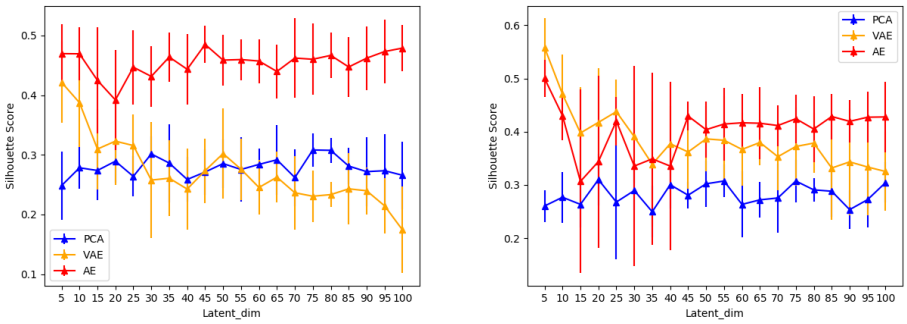

FIGURE 5.30: Silhouette scores on encoded and output data - mean and standard deviation

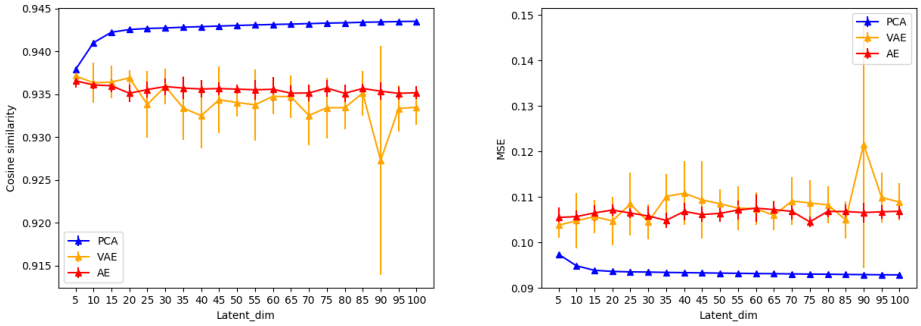

FIGURE 5.31: Cosine similarity and MSE - mean and standard deviation

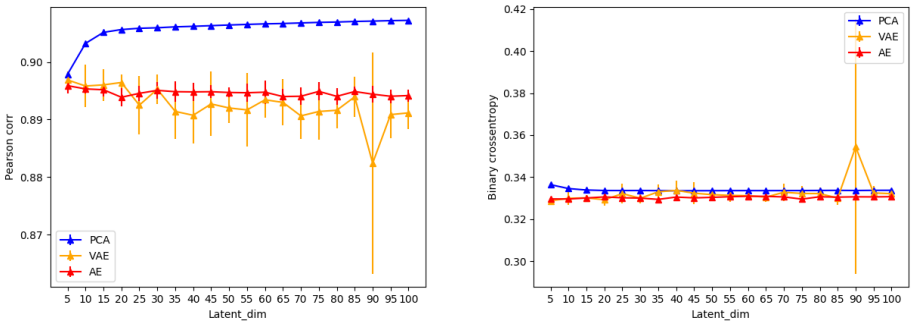

FIGURE 5.32: Pearson correlation and BCE - mean and standard deviation

From the figures above, one can conclude that, except for the strange nose dive of ARI score starting from 40 latent dimensions, AE has the best results on this pair.

Deng-Biase

The fifth pair presented is Deng-Biase. The number of genes found in both datasets was 16229, so Deng was initially reduced from 22958 genes to 16229, while Biase was reduced from 25114 genes to 19927. All results are summarized in Figures 5.33, 5.34, 5.35 and 5.36 below. Additionaly, mean values with respect to latent dimension are presented in Tables 5.9, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15 and 5.16. More information about the dataset can be found in Table 2.1 and [9], [4].

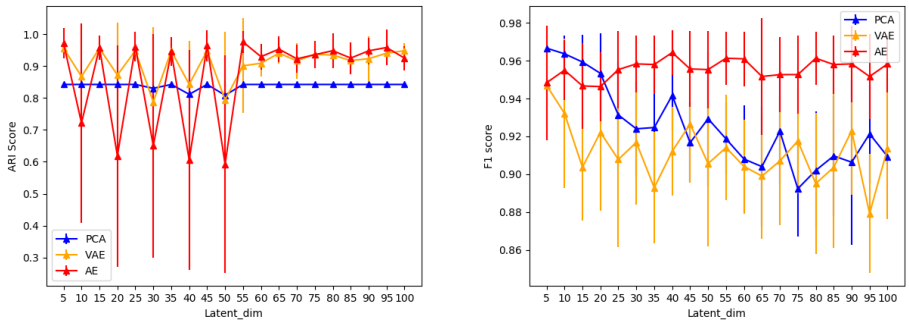

FIGURE 5.33: ARI and F1 scores - mean and standard deviation

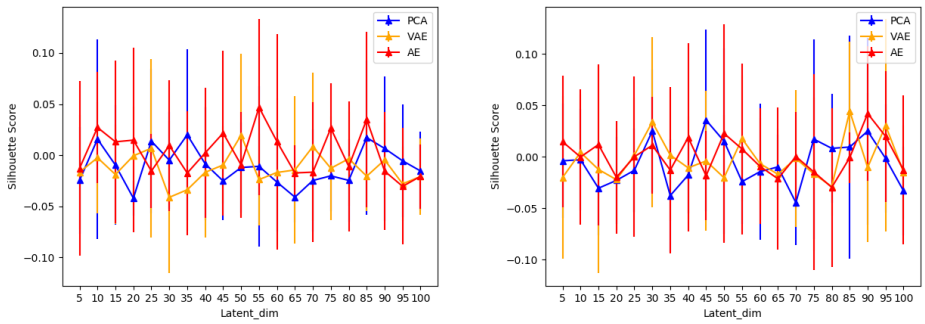

FIGURE 5.34: Silhouette scores on encoded and output data - mean and standard deviation

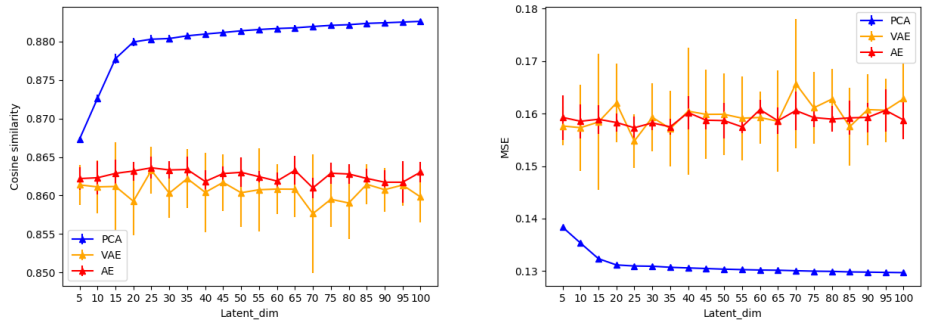

FIGURE 5.35: Cosine similarity and MSE - mean and standard deviation

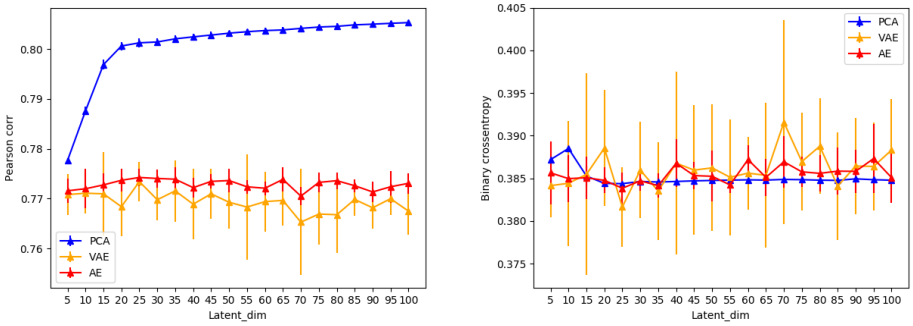

FIGURE 5.36: Pearson correlation and BCE - mean and standard deviation

From the figures above, one can conclude that AE has the best results on this pair. Again the increase in number of latent dimension does not yield an increase in reconstruction quality for both AE and VAE.

Goolam-Biase

The final pair presented is Goolam-Biase. In this case, all the genes from Biase dataset were found in Goolam set, so Goolam was initially reduced from 40405 genes to 25114, while no reduction was performed on Biase. All results are summarized in Figures 5.37, 5.38, 5.39 and 5.40 below. Additionaly, mean values with respect to latent dimension are presented in Tables 5.9, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15 and 5.16. More information about the dataset can be found in Table 2.1 and [13], [4].

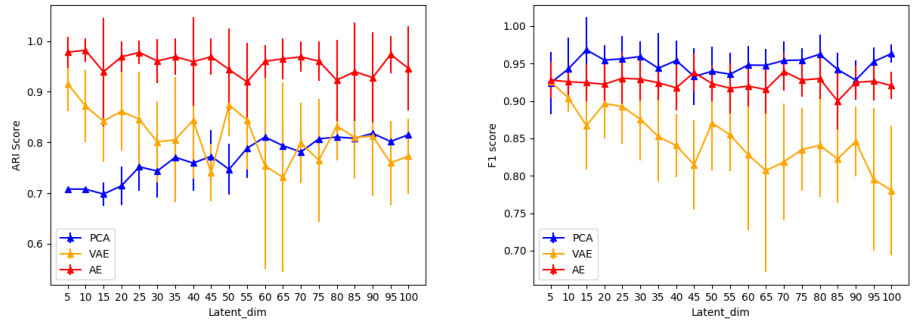

FIGURE 5.37: ARI and F1 scores - mean and standard deviation

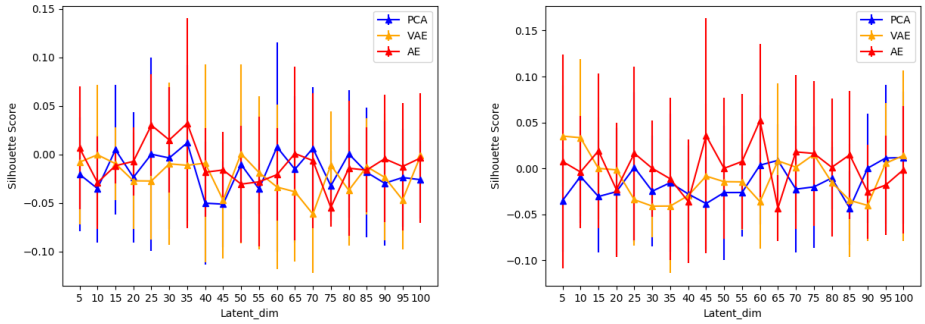

FIGURE 5.38: Silhouette scores on encoded and output data - mean and standard deviation

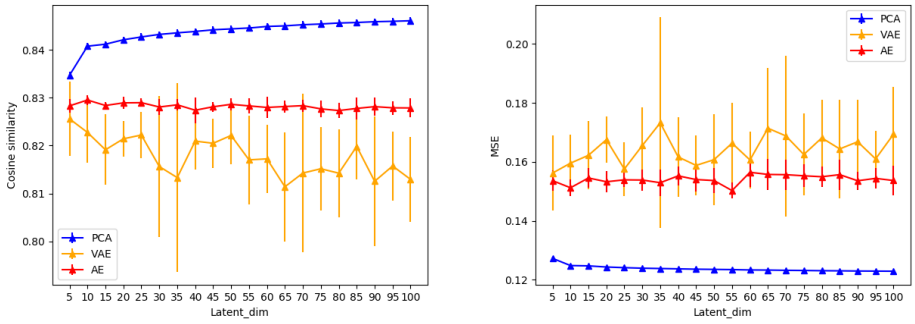

FIGURE 5.39: Cosine similarity and MSE - mean and standard deviation

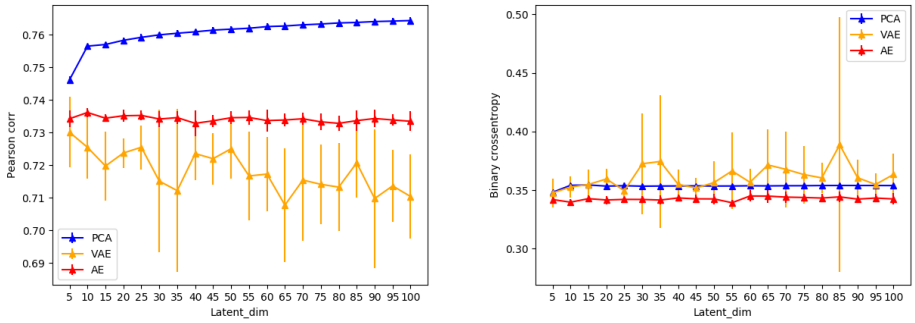

FIGURE 5.40: Pearson correlation and BCE - mean and standard deviation

For this pair one could give a slight advantage to AE, based on its far superior ARI score. Again the increase in number of latent dimension does not yield an increase in reconstruction quality for both AE and VAE. Tables that summarize mean values for each metric and each dataset pair are presented below.

	Baron-Muraro			Baron-Xin			Xin-Muraro		
Latent dim	VAE	AE	PCA	VAE	AE	PCA	VAE	AE	PCA
5	0.613	0.708	0.537	0.522	0.464	0.359	0.596	0.048	0.408
10	0.706	0.833	0.609	0.498	0.555	0.354	0.503	0.085	0.402
15	0.832	0.846	0.627	0.548	0.569	0.363	0.53	0.108	0.394
20	0.828	0.872	0.664	0.492	0.572	0.369	0.55	0.07	0.382
25	0.815	0.874	0.646	0.525	0.58	0.365	0.493	0.046	0.382
30	0.866	0.879	0.655	0.552	0.556	0.365	0.643	0.047	0.382
35	0.825	0.874	0.712	0.533	0.537	0.367	0.535	0.071	0.382
40	0.831	0.86	0.71	0.519	0.549	0.369	0.498	0.05	0.382
45	0.884	0.843	0.683	0.511	0.513	0.367	0.456	0.049	0.382
50	0.878	0.884	0.662	0.532	0.535	0.367	0.459	0.048	0.383
55	0.876	0.885	0.712	0.517	0.519	0.367	0.515	0.049	0.383
60	0.834	0.868	0.711	0.562	0.467	0.367	0.457	0.05	0.383
65	0.868	0.88	0.664	0.548	0.51	0.367	0.47	0.047	0.383
70	0.849	0.878	0.687	0.568	0.504	0.367	0.438	0.05	0.383
75	0.858	0.868	0.66	0.529	0.527	0.368	0.439	0.051	0.383
80	0.884	0.857	0.69	0.487	0.502	0.367	0.419	0.049	0.383
85	0.877	0.876	0.712	0.552	0.493	0.367	0.433	0.05	0.383
90	0.886	0.876	0.684	0.502	0.488	0.367	0.453	0.05	0.382
95	0.88	0.868	0.701	0.52	0.493	0.366	0.411	0.05	0.383
100	0.892	0.801	0.655	0.512	0.487	0.367	0.409	0.05	0.383
	Deng-Goolam			Deng-Biase			Goolam-Biase		
Latent dim	VAE	AE	PCA	VAE	AE	PCA	VAE	AE	PCA
5	0.795	0.943	0.747	0.956	0.972	0.842	0.916	0.978	0.708
10	0.785	0.917	0.81	0.867	0.721	0.842	0.872	0.982	0.708
15	0.849	0.809	0.798	0.956	0.958	0.842	0.842	0.939	0.698
20	0.782	0.786	0.767	0.872	0.619	0.842	0.861	0.969	0.714
25	0.876	0.897	0.766	0.948	0.961	0.842	0.845	0.977	0.752
30	0.756	0.766	0.814	0.788	0.65	0.83	0.801	0.961	0.743
35	0.719	0.758	0.771	0.948	0.945	0.842	0.805	0.969	0.771
40	0.75	0.768	0.779	0.842	0.606	0.811	0.844	0.959	0.759
45	0.618	0.268	0.787	0.948	0.964	0.842	0.74	0.969	0.773
50	0.576	0.295	0.75	0.794	0.592	0.808	0.874	0.944	0.747
55	0.645	0.343	0.76	0.901	0.975	0.842	0.844	0.919	0.788
60	0.684	0.302	0.81	0.909	0.929	0.842	0.754	0.96	0.811
65	0.542	0.33	0.797	0.94	0.953	0.842	0.731	0.965	0.793
70	0.64	0.335	0.789	0.916	0.923	0.842	0.799	0.969	0.781
75	0.619	0.348	0.776	0.937	0.936	0.842	0.765	0.96	0.807
80	0.729	0.285	0.758	0.934	0.948	0.842	0.832	0.923	0.811
85	0.674	0.417	0.743	0.917	0.925	0.842	0.81	0.94	0.808
90	0.674	0.324	0.783	0.923	0.948	0.842	0.813	0.928	0.818
95	0.538	0.337	0.792	0.94	0.958	0.842	0.759	0.974	0.802
100	0.636	0.378	0.813	0.948	0.925	0.842	0.773	0.946	0.815

TABLE 5.9: ARI scores - mean values with respect to the size of latent dimension

	Baron-Muraro			Baron-Xin			Xin-Muraro		
Latent dim	VAE	AE	PCA	VAE	AE	PCA	VAE	AE	PCA
5	0.703	0.798	0.819	0.852	0.792	0.796	0.703	0.55	0.849
10	0.762	0.822	0.825	0.909	0.898	0.871	0.729	0.597	0.938
15	0.768	0.824	0.829	0.942	0.909	0.903	0.66	0.588	0.96
20	0.759	0.831	0.838	0.93	0.912	0.959	0.64	0.576	0.957
25	0.762	0.828	0.841	0.928	0.912	0.974	0.659	0.541	0.958
30	0.748	0.831	0.834	0.933	0.921	0.974	0.7	0.513	0.945
35	0.749	0.833	0.832	0.915	0.915	0.971	0.646	0.552	0.94
40	0.744	0.836	0.828	0.904	0.924	0.971	0.651	0.546	0.946
45	0.742	0.83	0.831	0.923	0.918	0.969	0.643	0.556	0.94
50	0.758	0.832	0.83	0.886	0.92	0.966	0.586	0.512	0.944
55	0.745	0.831	0.832	0.915	0.913	0.964	0.665	0.535	0.936
60	0.731	0.832	0.829	0.909	0.92	0.965	0.601	0.555	0.931
65	0.745	0.833	0.83	0.912	0.925	0.965	0.596	0.521	0.934
70	0.745	0.83	0.83	0.912	0.93	0.965	0.549	0.546	0.931
75	0.748	0.837	0.829	0.9	0.923	0.959	0.635	0.542	0.929
80	0.754	0.831	0.827	0.872	0.925	0.962	0.571	0.534	0.919
85	0.746	0.833	0.83	0.896	0.917	0.956	0.59	0.524	0.92
90	0.744	0.835	0.83	0.9	0.919	0.959	0.598	0.547	0.917
95	0.736	0.834	0.826	0.891	0.929	0.954	0.544	0.551	0.916
100	0.747	0.834	0.828	0.898	0.928	0.957	0.522	0.589	0.922
	Deng-Goolam			Deng-Biase			Goolam-Biase		
Latent dim	VAE	AE	PCA	VAE	AE	PCA	VAE	AE	PCA
5	0.881	0.905	0.876	0.947	0.948	0.967	0.926	0.928	0.924
10	0.853	0.909	0.907	0.932	0.955	0.964	0.904	0.926	0.943
15	0.861	0.915	0.901	0.904	0.947	0.959	0.867	0.924	0.968
20	0.845	0.91	0.888	0.922	0.946	0.953	0.897	0.922	0.955
25	0.829	0.904	0.867	0.908	0.955	0.931	0.893	0.93	0.956
30	0.789	0.926	0.894	0.917	0.958	0.924	0.876	0.929	0.959
35	0.808	0.903	0.862	0.893	0.958	0.925	0.852	0.924	0.944
40	0.818	0.919	0.877	0.912	0.964	0.942	0.84	0.918	0.954
45	0.817	0.919	0.857	0.927	0.956	0.917	0.815	0.938	0.933
50	0.808	0.912	0.85	0.906	0.955	0.929	0.871	0.923	0.94
55	0.819	0.915	0.863	0.914	0.961	0.919	0.855	0.917	0.936
60	0.789	0.91	0.843	0.904	0.961	0.908	0.828	0.92	0.948
65	0.814	0.908	0.845	0.899	0.952	0.904	0.807	0.915	0.948
70	0.789	0.913	0.865	0.907	0.953	0.923	0.818	0.939	0.954
75	0.816	0.914	0.813	0.918	0.953	0.892	0.835	0.928	0.955
80	0.804	0.909	0.841	0.895	0.961	0.902	0.841	0.93	0.962
85	0.745	0.91	0.824	0.904	0.958	0.91	0.822	0.9	0.942
90	0.803	0.9	0.824	0.923	0.958	0.906	0.846	0.925	0.928
95	0.78	0.91	0.822	0.879	0.952	0.921	0.795	0.926	0.953
100	0.749	0.923	0.827	0.914	0.958	0.909	0.78	0.92	0.963

TABLE 5.10: F1 scores - mean values with respect to the size of latent dimension

	Baron-Muraro			Baron-Xin			Xin-Muraro		
Latent dim	VAE	AE	PCA	VAE	AE	PCA	VAE	AE	PCA
5	0.153	0.139	0.108	0.108	0.082	0.037	0.051	0.1	0.057
10	0.191	0.16	0.176	0.195	0.144	0.092	0.07	0.095	0.056
15	0.211	0.171	0.265	0.248	0.189	0.098	0.083	0.087	0.04
20	0.191	0.209	0.284	0.234	0.174	0.088	0.084	0.084	0.083
25	0.196	0.227	0.297	0.199	0.17	0.074	0.106	0.095	0.055
30	0.207	0.254	0.298	0.268	0.171	0.077	0.11	0.092	0.105
35	0.199	0.243	0.296	0.213	0.181	0.082	0.109	0.095	0.132
40	0.22	0.238	0.307	0.209	0.129	0.102	0.108	0.092	0.138
45	0.172	0.251	0.302	0.197	0.153	0.085	0.113	0.097	0.085
50	0.191	0.275	0.312	0.215	0.167	0.081	0.124	0.075	0.111
55	0.198	0.283	0.315	0.194	0.246	0.092	0.115	0.079	0.102
60	0.206	0.272	0.319	0.196	0.13	0.1	0.124	0.09	0.093
65	0.181	0.276	0.327	0.199	0.128	0.096	0.123	0.081	0.114
70	0.185	0.306	0.341	0.213	0.164	0.099	0.119	0.08	0.136
75	0.216	0.274	0.341	0.181	0.168	0.094	0.139	0.091	0.124
80	0.167	0.29	0.322	0.185	0.141	0.103	0.143	0.085	0.111
85	0.186	0.289	0.339	0.239	0.123	0.106	0.128	0.078	0.12
90	0.182	0.287	0.345	0.218	0.138	0.102	0.125	0.074	0.138
95	0.159	0.274	0.337	0.171	0.16	0.11	0.117	0.091	0.131
100	0.176	0.297	0.334	0.215	0.163	0.091	0.136	0.087	0.11
	Deng-Goolam			Deng-Biase			Goolam-Biase		
Latent dim	VAE	AE	PCA	VAE	AE	PCA	VAE	AE	PCA
5	0.422	0.47	0.248	-0.017	-0.013	-0.025	-0.008	0.007	-0.02
10	0.387	0.469	0.278	-0.002	0.027	0.016	-0.0	-0.029	-0.035
15	0.31	0.425	0.274	-0.019	0.013	-0.01	-0.009	-0.012	0.005
20	0.323	0.392	0.289	-0.001	0.015	-0.042	-0.027	-0.007	-0.024
25	0.316	0.447	0.264	0.007	-0.015	0.014	-0.027	0.03	0.0
30	0.258	0.431	0.301	-0.041	0.01	-0.005	-0.01	0.015	-0.004
35	0.261	0.464	0.286	-0.034	-0.017	0.02	-0.011	0.032	0.012
40	0.243	0.443	0.259	-0.017	0.002	-0.008	-0.009	-0.018	-0.05
45	0.273	0.485	0.271	-0.01	0.022	-0.025	-0.047	-0.016	-0.051
50	0.302	0.459	0.285	0.02	-0.009	-0.012	0.001	-0.031	-0.01
55	0.276	0.46	0.275	-0.024	0.047	-0.011	-0.019	-0.028	-0.036
60	0.245	0.457	0.284	-0.017	0.013	-0.026	-0.034	-0.021	0.008
65	0.263	0.44	0.292	-0.014	-0.017	-0.041	-0.038	0.001	-0.015
70	0.236	0.462	0.262	0.009	-0.017	-0.025	-0.061	-0.006	0.006
75	0.231	0.46	0.308	-0.013	0.026	-0.02	-0.011	-0.055	-0.032
80	0.233	0.467	0.308	-0.003	-0.011	-0.025	-0.037	-0.014	0.001
85	0.243	0.447	0.281	-0.021	0.035	0.017	-0.013	-0.016	-0.018
90	0.24	0.462	0.272	-0.005	-0.016	0.007	-0.023	-0.004	-0.03
95	0.215	0.473	0.273	-0.028	-0.03	-0.005	-0.047	-0.012	-0.024
100	0.174	0.479	0.266	-0.021	-0.021	-0.015	-0.001	-0.004	-0.026

TABLE 5.11: Silhouette scores on encoded data - mean values with respect to the size of latent dimension

	Baron-Muraro			Baron-Xin			Xin-Muraro		
Latent dim	VAE	AE	PCA	VAE	AE	PCA	VAE	AE	PCA
5	0.227	0.218	0.096	0.161	0.146	0.036	0.181	0.066	0.041
10	0.309	0.318	0.173	0.261	0.26	0.091	0.18	0.07	0.059
15	0.362	0.337	0.275	0.312	0.287	0.087	0.208	0.084	0.052
20	0.355	0.351	0.288	0.229	0.361	0.084	0.201	0.066	0.103
25	0.364	0.368	0.289	0.281	0.337	0.092	0.197	0.074	0.089
30	0.355	0.366	0.306	0.331	0.366	0.073	0.238	0.066	0.107
35	0.362	0.362	0.287	0.288	0.295	0.097	0.214	0.063	0.078
40	0.379	0.386	0.308	0.301	0.336	0.08	0.192	0.071	0.101
45	0.346	0.376	0.31	0.284	0.324	0.082	0.235	0.07	0.079
50	0.374	0.36	0.32	0.296	0.271	0.095	0.247	0.026	0.071
55	0.375	0.384	0.308	0.27	0.299	0.094	0.24	0.032	0.129
60	0.36	0.378	0.33	0.309	0.271	0.086	0.218	0.043	0.109
65	0.37	0.414	0.335	0.319	0.332	0.118	0.234	0.025	0.119
70	0.383	0.394	0.327	0.349	0.264	0.105	0.205	0.031	0.116
75	0.378	0.377	0.33	0.31	0.346	0.122	0.195	0.054	0.127
80	0.361	0.405	0.329	0.306	0.355	0.093	0.229	0.019	0.149
85	0.374	0.397	0.323	0.286	0.322	0.118	0.223	0.03	0.119
90	0.367	0.397	0.343	0.305	0.325	0.122	0.199	0.025	0.135
95	0.378	0.387	0.333	0.356	0.328	0.102	0.188	0.038	0.118
100	0.373	0.404	0.347	0.295	0.314	0.102	0.227	0.034	0.114
	Deng-Goolam			Deng-Biase			Goolam-Biase		
Latent dim	VAE	AE	PCA	VAE	AE	PCA	VAE	AE	PCA
5	0.558	0.5	0.26	-0.02	0.015	-0.004	0.035	0.008	-0.035
10	0.47	0.43	0.277	0.004	-0.0	-0.003	0.033	-0.004	-0.009
15	0.398	0.307	0.263	-0.012	0.012	-0.03	0.0	0.019	-0.031
20	0.417	0.344	0.31	-0.022	-0.02	-0.023	-0.002	-0.023	-0.025
25	0.438	0.42	0.268	0.002	0.0	-0.013	-0.034	0.016	0.001
30	0.39	0.335	0.289	0.034	0.011	0.025	-0.041	0.0	-0.025
35	0.339	0.349	0.25	0.002	-0.013	-0.038	-0.041	-0.011	-0.015
40	0.377	0.335	0.3	-0.011	0.019	-0.018	-0.03	-0.036	-0.028
45	0.362	0.429	0.281	-0.004	-0.018	0.036	-0.008	0.036	-0.038
50	0.386	0.404	0.302	-0.02	0.023	0.015	-0.014	0.0	-0.026
55	0.384	0.414	0.307	0.018	0.008	-0.024	-0.015	0.008	-0.026
60	0.367	0.417	0.263	-0.007	-0.009	-0.014	-0.036	0.052	0.004
65	0.38	0.416	0.272	-0.017	-0.021	-0.01	0.008	-0.043	0.009
70	0.353	0.411	0.275	-0.001	0.0	-0.044	0.001	0.018	-0.022
75	0.372	0.424	0.307	-0.017	-0.015	0.017	0.015	0.016	-0.02
80	0.379	0.405	0.291	-0.029	-0.03	0.008	-0.015	0.001	-0.01
85	0.332	0.428	0.288	0.045	-0.001	0.01	-0.035	0.015	-0.043
90	0.343	0.419	0.254	-0.01	0.042	0.025	-0.04	-0.025	0.0
95	0.334	0.427	0.272	0.031	0.02	-0.001	0.006	-0.018	0.011
100	0.325	0.428	0.304	-0.015	-0.013	-0.033	0.014	-0.001	0.012

TABLE 5.12: Silhouette scores on output data - mean values with respect to the size of latent dimension

	Baron-Muraro			Baron-Xin			Xin-Muraro		
Latent dim	VAE	AE	PCA	VAE	AE	PCA	VAE	AE	PCA
5	0.748	0.754	0.725	0.62	0.594	0.603	0.664	0.537	0.697
10	0.761	0.765	0.758	0.578	0.605	0.63	0.633	0.549	0.707
15	0.76	0.768	0.766	0.581	0.594	0.637	0.636	0.559	0.716
20	0.761	0.769	0.771	0.575	0.608	0.642	0.637	0.546	0.721
25	0.76	0.772	0.777	0.591	0.592	0.65	0.647	0.56	0.725
30	0.761	0.769	0.782	0.595	0.59	0.655	0.661	0.539	0.726
35	0.76	0.771	0.783	0.589	0.589	0.657	0.65	0.554	0.727
40	0.761	0.77	0.785	0.595	0.59	0.659	0.646	0.533	0.728
45	0.76	0.768	0.788	0.606	0.57	0.662	0.661	0.541	0.728
50	0.762	0.768	0.789	0.602	0.591	0.663	0.646	0.56	0.729
55	0.758	0.769	0.79	0.602	0.59	0.664	0.676	0.537	0.729
60	0.761	0.767	0.791	0.61	0.575	0.665	0.658	0.538	0.73
65	0.761	0.768	0.791	0.603	0.587	0.666	0.673	0.524	0.73
70	0.759	0.765	0.792	0.607	0.582	0.667	0.656	0.555	0.731
75	0.761	0.766	0.793	0.61	0.59	0.667	0.674	0.545	0.731
80	0.76	0.767	0.794	0.606	0.588	0.668	0.656	0.544	0.732
85	0.759	0.764	0.794	0.607	0.565	0.668	0.675	0.561	0.732
90	0.759	0.766	0.795	0.607	0.582	0.669	0.652	0.56	0.733
95	0.759	0.766	0.795	0.614	0.57	0.669	0.669	0.543	0.733
100	0.757	0.763	0.796	0.607	0.564	0.67	0.661	0.54	0.734
	Deng-Goolam			Deng-Biase			Goolam-Biase		
Latent dim	VAE	AE	PCA	VAE	AE	PCA	VAE	AE	PCA
5	0.937	0.937	0.938	0.861	0.862	0.867	0.826	0.828	0.835
10	0.936	0.936	0.941	0.861	0.862	0.873	0.823	0.83	0.841
15	0.936	0.936	0.942	0.861	0.863	0.878	0.819	0.828	0.841
20	0.937	0.935	0.943	0.859	0.863	0.88	0.821	0.829	0.842
25	0.934	0.936	0.943	0.863	0.864	0.88	0.822	0.829	0.843
30	0.936	0.936	0.943	0.86	0.863	0.88	0.816	0.828	0.843
35	0.933	0.936	0.943	0.862	0.863	0.881	0.813	0.829	0.844
40	0.933	0.936	0.943	0.86	0.862	0.881	0.821	0.827	0.844
45	0.934	0.936	0.943	0.862	0.863	0.881	0.82	0.828	0.844
50	0.934	0.936	0.943	0.86	0.863	0.881	0.822	0.829	0.844
55	0.934	0.936	0.943	0.861	0.862	0.882	0.817	0.828	0.845
60	0.935	0.936	0.943	0.861	0.862	0.882	0.817	0.828	0.845
65	0.935	0.935	0.943	0.861	0.863	0.882	0.811	0.828	0.845
70	0.933	0.935	0.943	0.858	0.861	0.882	0.814	0.828	0.845
75	0.933	0.936	0.943	0.86	0.863	0.882	0.815	0.828	0.845
80	0.933	0.935	0.943	0.859	0.863	0.882	0.814	0.827	0.846
85	0.935	0.936	0.943	0.861	0.862	0.882	0.82	0.828	0.846
90	0.927	0.935	0.943	0.861	0.862	0.882	0.813	0.828	0.846
95	0.933	0.935	0.943	0.861	0.862	0.883	0.816	0.828	0.846
100	0.933	0.935	0.944	0.86	0.863	0.883	0.813	0.828	0.846

TABLE 5.13: Cosine similarity - mean values with respect to the size of latent dimension

	Baron-Muraro			Baron-Xin			Xin-Muraro		
Latent dim	VAE	AE	PCA	VAE	AE	PCA	VAE	AE	PCA
5	0.077	0.081	0.078	0.13	0.138	0.131	0.204	0.214	0.176
10	0.075	0.08	0.074	0.137	0.136	0.128	0.202	0.212	0.174
15	0.075	0.081	0.073	0.136	0.138	0.127	0.202	0.211	0.171
20	0.075	0.079	0.072	0.138	0.136	0.126	0.201	0.212	0.17
25	0.075	0.079	0.072	0.135	0.139	0.125	0.198	0.211	0.169
30	0.075	0.079	0.071	0.134	0.139	0.124	0.197	0.215	0.168
35	0.076	0.078	0.071	0.136	0.139	0.124	0.197	0.212	0.168
40	0.076	0.08	0.07	0.134	0.139	0.124	0.198	0.214	0.168
45	0.076	0.079	0.07	0.133	0.143	0.123	0.195	0.213	0.168
50	0.076	0.079	0.07	0.133	0.139	0.123	0.199	0.211	0.168
55	0.075	0.078	0.07	0.134	0.138	0.123	0.193	0.214	0.168
60	0.076	0.08	0.07	0.132	0.141	0.123	0.197	0.213	0.167
65	0.076	0.08	0.07	0.133	0.14	0.123	0.192	0.216	0.167
70	0.076	0.081	0.069	0.132	0.14	0.123	0.197	0.212	0.167
75	0.076	0.081	0.069	0.132	0.139	0.122	0.192	0.213	0.167
80	0.076	0.08	0.069	0.132	0.139	0.122	0.197	0.214	0.167
85	0.076	0.081	0.069	0.132	0.143	0.122	0.193	0.211	0.167
90	0.077	0.08	0.069	0.132	0.141	0.122	0.195	0.211	0.167
95	0.077	0.079	0.069	0.131	0.142	0.122	0.192	0.212	0.167
100	0.077	0.081	0.069	0.132	0.143	0.122	0.196	0.213	0.166
	Deng-Goolam			Deng-Biase			Goolam-Biase		
Latent dim	VAE	AE	PCA	VAE	AE	PCA	VAE	AE	PCA
5	0.104	0.106	0.097	0.158	0.159	0.138	0.156	0.154	0.127
10	0.105	0.106	0.095	0.157	0.159	0.135	0.16	0.151	0.125
15	0.106	0.107	0.094	0.158	0.159	0.132	0.162	0.155	0.125
20	0.105	0.107	0.094	0.162	0.158	0.131	0.168	0.153	0.124
25	0.108	0.106	0.094	0.155	0.157	0.131	0.158	0.154	0.124
30	0.105	0.106	0.094	0.159	0.158	0.131	0.166	0.154	0.124
35	0.11	0.105	0.093	0.157	0.157	0.131	0.173	0.153	0.124
40	0.111	0.107	0.093	0.16	0.16	0.131	0.162	0.155	0.124
45	0.109	0.106	0.093	0.16	0.159	0.13	0.159	0.154	0.124
50	0.109	0.106	0.093	0.16	0.159	0.13	0.161	0.154	0.124
55	0.108	0.107	0.093	0.159	0.157	0.13	0.166	0.15	0.123
60	0.107	0.108	0.093	0.159	0.161	0.13	0.161	0.156	0.123
65	0.106	0.107	0.093	0.159	0.159	0.13	0.171	0.156	0.123
70	0.109	0.107	0.093	0.166	0.161	0.13	0.169	0.156	0.123
75	0.109	0.105	0.093	0.161	0.159	0.13	0.163	0.155	0.123
80	0.108	0.107	0.093	0.163	0.159	0.13	0.168	0.155	0.123
85	0.105	0.107	0.093	0.158	0.159	0.13	0.164	0.156	0.123
90	0.122	0.107	0.093	0.161	0.159	0.13	0.167	0.154	0.123
95	0.11	0.107	0.093	0.161	0.161	0.13	0.161	0.154	0.123
100	0.109	0.107	0.093	0.163	0.159	0.13	0.169	0.154	0.123

TABLE 5.14: MSE - mean values with respect to the size of latent dimension

	Baron-Muraro			Baron-Xin			Xin-Muraro		
Latent dim	VAE	AE	PCA	VAE	AE	PCA	VAE	AE	PCA
5	0.576	0.569	0.586	0.53	0.507	0.517	0.471	0.383	0.498
10	0.596	0.591	0.606	0.499	0.523	0.542	0.451	0.39	0.513
15	0.6	0.598	0.619	0.505	0.517	0.55	0.453	0.397	0.526
20	0.6	0.601	0.625	0.501	0.527	0.556	0.452	0.388	0.534
25	0.601	0.606	0.633	0.515	0.515	0.566	0.46	0.395	0.542
30	0.603	0.601	0.64	0.519	0.513	0.572	0.469	0.385	0.545
35	0.599	0.605	0.644	0.515	0.513	0.576	0.463	0.395	0.546
40	0.6	0.603	0.648	0.518	0.515	0.579	0.458	0.38	0.547
45	0.599	0.599	0.652	0.527	0.499	0.581	0.471	0.388	0.548
50	0.602	0.6	0.654	0.524	0.516	0.583	0.458	0.398	0.55
55	0.598	0.603	0.655	0.524	0.514	0.585	0.479	0.383	0.55
60	0.6	0.597	0.657	0.53	0.5	0.585	0.466	0.384	0.551
65	0.599	0.601	0.658	0.525	0.513	0.586	0.476	0.375	0.552
70	0.598	0.595	0.659	0.529	0.508	0.587	0.463	0.396	0.553
75	0.599	0.597	0.66	0.531	0.515	0.588	0.478	0.391	0.554
80	0.599	0.599	0.661	0.527	0.514	0.589	0.464	0.389	0.555
85	0.599	0.594	0.662	0.529	0.493	0.589	0.478	0.399	0.556
90	0.598	0.597	0.663	0.529	0.508	0.59	0.464	0.398	0.557
95	0.597	0.599	0.664	0.534	0.498	0.591	0.471	0.387	0.557
100	0.595	0.593	0.664	0.529	0.493	0.591	0.466	0.385	0.558
	Deng-Goolam			Deng-Biase			Goolam-Biase		
Latent dim	VAE	AE	PCA	VAE	AE	PCA	VAE	AE	PCA
5	0.897	0.896	0.898	0.771	0.772	0.778	0.73	0.734	0.746
10	0.896	0.895	0.903	0.771	0.772	0.788	0.725	0.736	0.756
15	0.896	0.895	0.905	0.771	0.773	0.797	0.72	0.734	0.757
20	0.896	0.894	0.906	0.768	0.774	0.801	0.724	0.735	0.758
25	0.893	0.895	0.906	0.773	0.774	0.801	0.726	0.735	0.759
30	0.895	0.895	0.906	0.77	0.774	0.801	0.715	0.734	0.76
35	0.891	0.895	0.906	0.772	0.774	0.802	0.712	0.735	0.76
40	0.891	0.895	0.906	0.769	0.772	0.802	0.724	0.733	0.761
45	0.893	0.895	0.906	0.771	0.773	0.803	0.722	0.734	0.761
50	0.892	0.895	0.906	0.769	0.774	0.803	0.725	0.735	0.762
55	0.892	0.895	0.907	0.768	0.772	0.803	0.717	0.735	0.762
60	0.893	0.895	0.907	0.769	0.772	0.804	0.717	0.734	0.762
65	0.893	0.894	0.907	0.77	0.774	0.804	0.708	0.734	0.763
70	0.891	0.894	0.907	0.765	0.771	0.804	0.715	0.734	0.763
75	0.891	0.895	0.907	0.767	0.773	0.804	0.714	0.733	0.763
80	0.892	0.894	0.907	0.767	0.774	0.805	0.713	0.733	0.764
85	0.894	0.895	0.907	0.77	0.773	0.805	0.721	0.734	0.764
90	0.882	0.894	0.907	0.768	0.771	0.805	0.71	0.734	0.764
95	0.891	0.894	0.907	0.77	0.772	0.805	0.714	0.734	0.764
100	0.891	0.894	0.907	0.767	0.773	0.805	0.71	0.733	0.764

TABLE 5.15: Pearson correlation - mean values with respect to the size of latent dimension

	Baron-Muraro			Baron-Xin			Xin-Muraro		
Latent dim	VAE	AE	PCA	VAE	AE	PCA	VAE	AE	PCA
5	0.242	0.241	0.264	0.208	0.23	0.214	0.439	0.519	0.378
10	0.242	0.241	0.25	0.238	0.228	0.208	0.448	0.508	0.373
15	0.244	0.242	0.251	0.241	0.237	0.207	0.445	0.495	0.368
20	0.243	0.242	0.252	0.25	0.227	0.207	0.445	0.507	0.366
25	0.243	0.242	0.25	0.235	0.24	0.205	0.43	0.491	0.365
30	0.243	0.242	0.249	0.235	0.243	0.205	0.419	0.516	0.364
35	0.243	0.241	0.249	0.242	0.244	0.204	0.429	0.498	0.364
40	0.242	0.242	0.249	0.238	0.246	0.204	0.433	0.524	0.364
45	0.242	0.242	0.249	0.225	0.261	0.203	0.419	0.516	0.364
50	0.242	0.242	0.249	0.229	0.244	0.203	0.433	0.494	0.363
55	0.243	0.242	0.249	0.229	0.242	0.203	0.406	0.518	0.363
60	0.242	0.242	0.249	0.223	0.252	0.203	0.419	0.521	0.363
65	0.242	0.242	0.249	0.228	0.249	0.203	0.406	0.533	0.363
70	0.242	0.243	0.249	0.225	0.25	0.203	0.423	0.5	0.363
75	0.242	0.243	0.249	0.221	0.245	0.203	0.406	0.51	0.363
80	0.242	0.243	0.249	0.223	0.247	0.203	0.421	0.512	0.362
85	0.242	0.243	0.249	0.224	0.263	0.203	0.407	0.493	0.362
90	0.242	0.243	0.249	0.224	0.252	0.203	0.43	0.492	0.362
95	0.242	0.243	0.249	0.218	0.261	0.203	0.406	0.514	0.362
100	0.242	0.243	0.249	0.226	0.265	0.202	0.417	0.522	0.362
	Deng-Goolam			Deng-Biase			Goolam-Biase		
Latent dim	VAE	AE	PCA	VAE	AE	PCA	VAE	AE	PCA
5	0.329	0.33	0.336	0.384	0.386	0.387	0.348	0.342	0.348
10	0.33	0.33	0.335	0.384	0.385	0.388	0.353	0.34	0.354
15	0.33	0.33	0.334	0.385	0.385	0.385	0.355	0.343	0.354
20	0.329	0.331	0.334	0.389	0.385	0.384	0.36	0.342	0.353
25	0.332	0.33	0.334	0.382	0.384	0.384	0.349	0.342	0.354
30	0.33	0.33	0.334	0.386	0.385	0.385	0.373	0.342	0.353
35	0.333	0.329	0.334	0.384	0.384	0.385	0.374	0.342	0.353
40	0.334	0.331	0.334	0.387	0.387	0.385	0.355	0.343	0.354
45	0.332	0.33	0.334	0.386	0.385	0.385	0.352	0.343	0.354
50	0.332	0.33	0.334	0.386	0.385	0.385	0.356	0.342	0.354
55	0.331	0.331	0.334	0.385	0.384	0.385	0.366	0.339	0.354
60	0.331	0.331	0.334	0.386	0.387	0.385	0.357	0.345	0.354
65	0.33	0.331	0.334	0.385	0.385	0.385	0.372	0.345	0.354
70	0.333	0.331	0.334	0.392	0.387	0.385	0.368	0.344	0.354
75	0.332	0.33	0.334	0.387	0.386	0.385	0.363	0.344	0.354
80	0.332	0.331	0.334	0.389	0.386	0.385	0.36	0.343	0.354
85	0.33	0.33	0.334	0.384	0.386	0.385	0.389	0.344	0.354
90	0.354	0.331	0.334	0.386	0.386	0.385	0.361	0.342	0.354
95	0.332	0.331	0.334	0.386	0.387	0.385	0.355	0.343	0.354
100	0.332	0.331	0.334	0.388	0.385	0.385	0.363	0.342	0.354

TABLE 5.16: BCE - mean values with respect to the size of latent dimension

Chapter 6

Conclusion

The goal of this thesis was to evaluate dimensionality reduction capabilities of two neural networks based models – autoencoders and variational autoencoders on single-cell data. An extensive analysis was performed and all models were tested on real life datasets, with respect to various aspects of quality of the reduced data. Results obtained suggest that the best model for dimensionality reduction on single-cell data is AE.

However, some irregularities have been observed, creating a lot of ideas for future work. Probably the strangest results came from reconstruction quality metrics, where the increase in size of latent space did not yield an increase in reconstruction quality for both AE and VAE. An interesting challenge would be to find the cause for this anomaly and establish whether single-cell data can truly be represented by only a few latent factors.

Secondly, the most powerful of the models, VAE performed the worst on most of datasets. This result was also unexpected and it would be interesting to perform a more extensive study to determine whether these results represent an irregularity or is VAE simply not compatible with single-cell data. One possible hypothesis could be that the prior used for modeling the latent space, standard normal distribution, is not a good fit for single-cell data. A prior more befitting single-cell data could possibly lead to better performance of VAE.

Finally, there is always the argument of optimizing network parameters. As was already mentioned, neural networks have a huge number of parameters. In our models, we used a fixed parameter set, considered to be reasonable choices with respect to our data. Optimizing the network parameters would most certainly improve the quality of dimensionality reduction performance of both neural network models. Such a task was, due to time and computational resource constraints, impossible for the current project, but leaves an interesting idea for future work.

Appendix A

A Background on Neural Networks

Appendix A examines the basic elements that make up a neural network. Each element is described and its particular function in the network is explained. It consists of six sections, dedicated to layers, activation functions, loss functions, optimizers, epochs and batch size, respectively.

A.1 Layers

A layer represents the basic building blocks of neural networks. Layers consist of neurons that, given some input values, output a particular numerical value. The number of neurons in a given layer determines the dimensionality of the output of that layer.

Layers can have various structures. For example, the most common type of layers is a dense, or fully connected layer. As its name suggests, a dense layer is such that, if each neuron from the current and previous layer are viewed as nodes in a graph, then the graph between induced by those nodes would be a fully connected one. It is possible for a layer's output to be the input of the layer before it (recurrent networks) or of the layer a few places after it (for example the output of the third layer can be the input to the sixth layer directly).

Layers can also have different functions. For example, in Chapter 5 we mentioned a sampling layer, whose sole purpose is to sample a point from a defined distribution. Another specific type of layer is called the dropout layer and is used to prevent overfitting by randomly setting some inputs to the layer after it to zero in order force the network to learn instead of memorizing.

As was mentioned in Chapter 3, the number of layers in a network determines its depth. The more layers a network has, the larger its learning capacity is. At this point, one would be inclined to ask 'why don't we build a network with an arbitrarily large number of layers and solve all of our problems?'. There are two different reasons why this approach would not yield good results. One is that, with the size of layers, neural networks become computationaly intractable. The other one is that after a certain number of layers, the network starts to overfit the problem. Due to its large capacity, it becomes capable of completely learning the solution of a given problem, but in turns, becomes uncapable of generalizing and adapting to solve similar problems.

A.2 Activation functions

Activation functions are functions that are applied at the end of the computational process of each neuron.

Various functions can be used as activation functions, but there are some desirable properties we would like our activation functions to have. A key property of neural networks is that activation functions can be nonlinear. It is a well known fact that a composition of linear functions yields a linear function. Since training a neural network can be interpreted as learning the best mapping of input data to target data, and the network as a whole is a composition of functions in each layer, one could only solve a fairly limited class of problems using linear activation functions. On the other hand, using a composition of nonlinear functions can approximate almost any function.

Some of the most common activation functions are ReLU, sigmoid and tanh. ReLU is currently the most widely used activation function in machine learning. It is defined as

$$ReLU(x) = \max\{0, x\}\,. \tag{A.1}$$

ReLU allows only for nonnegative values and creates a nonlinear function from a simple identity function. Although simple to interpret, it is criticised for outputing only nonnegative values and flattening all the negative values to zero.

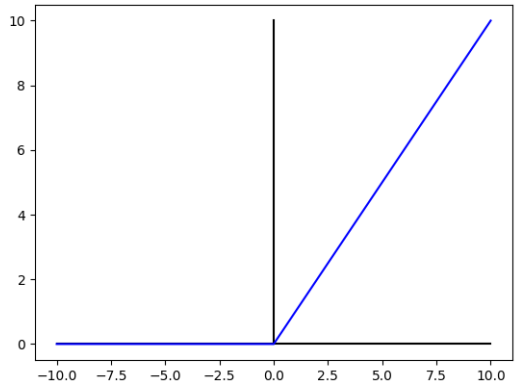

FIGURE A.1: ReLU

Sigmoid, unlike ReLU, outputs values in the interval $[0, 1]$. Its primary use is in classification and is defined as

$$sigmoid(x) = \frac{1}{1 + e^{-x}}\,. \tag{A.2}$$

Since it is differentiable everywhere, it is easy to work with using the backpropagation algorithm.

Sigmoid functions have a probabilistic interpretation - the closer to one the output of a sigmoid function is, the more certain we can be about an event/prediction. For example, say we are faced with a binary classification task, and sigmoid is the activation function used in the last layer. The output vector will be given by

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{A.3}$$

where $x_1, x_2 \in [0,1]$. If, for example $x_1 > x_2$, our network claims that the current input belongs to class 1. Moreover, the closer

$$x^T v \tag{A.4}$$

is to one, the more certain we can be in its classification. Here,

$$v = \begin{bmatrix} 1 \\ -1 \end{bmatrix} . \tag{A.5}$$

However, if (A.4) is close to zero, we have a signal that the degree of certainty of our classification is low. The same criticism of ReLU is aimed at sigmoid, as it only outputs nonnegative values.

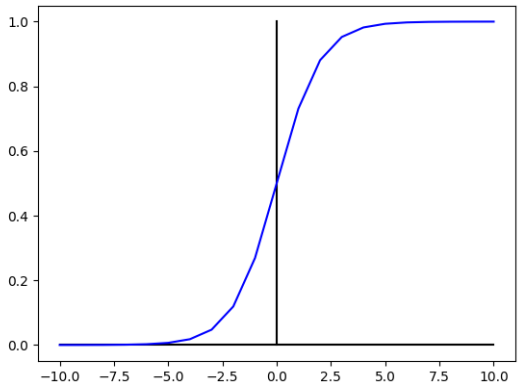

FIGURE A.2: Sigmoid

Finally, an activation function that deals with the criticism aimed at both ReLU and sigmoid is the tanh function. It is defined as

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1 . \tag{A.6}$$

From the definition above, it is clear that tanh takes values from the interval $[-1, 1]$.

Again, like the sigmoid, it is a differentiable function, mainly used in binary classification tasks.

This is the proper time to explain the condition (2.5) imposed on our datasets, mentioned in Chapter 2. Since sigmoid function outputs values in the interval $[0, 1]$ in order to train our models, we had to make sure that our input data is also in the interval $[0, 1]$. Otherwise, the networks would not be able to optimize for similarity between inputs that are in the interval $[0, \infty]$ and outputs that are in the interval $[0, 1]$.

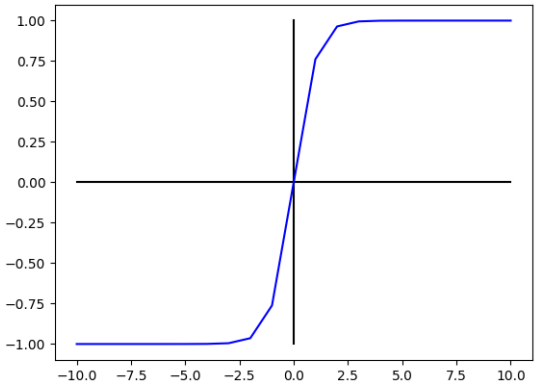

FIGURE A.3: Tanh

A.3 Loss functions

Loss function represents the objective function that the network optimizes in order to produce a good model. In Chapter 3 we introduced one such function, but depending on the application, loss functions can have various forms. For example, all the metrics used to measure reconstruction quality, introduced in Chapter 4, could be used as loss functions for a neural network model.

A.4 Optimizers

Optimizers are the algorithms used to minimize the loss function. Almost all are based on the stochastic gradient descent algorithm and present various application related extensions of it.

A.5 Epochs

An epoch is defined as one training instance where the network trains on the whole training set. Since neural networks are robust and iteratively optimize their models, unlike the older machine learning algorithms, they need multiple passes through the dataset.

A.6 Batch size

Batch size determines the number of points used for calculating the gradient during each epoch. It controls the number of times we update our solution during one epoch. The ideal (and the most computationaly expensive) case is to do just one gradient step using all the points in our dataset. Since nowadays the number of points tends to be huge, this approach is usually impossible to adopt. Instead, a number is chosen in a way that is perceived as a resonable choice with respect to our dataset[1]. For example, using batches containing only a single point will result in many noisy gradient steps taken within an epoch. Usually, a good choice is 32 or 64.

[1]Depending mainly on the number of points in our dataset, but also on the dimensionality.

Appendix B

A Background on Information-theoretic Metrics and Methods

Appendix B explaines some important concepts from information theory introduced in the thesis in more details. It consists of two sections, dedicated to the entropy and Kullback-Leibler divergence, respectively.

B.1 Entropy

The concept of entropy and information gain was introduced by Claude Shannon in 1948. It measures the average rate at which information is produced by a stochastic source of data.

If we have N observable elements, the entropy of the system can be calculated using the formula

$$H(p) = \mathbf{E}[-\log p] = -\sum_{i=1}^{N} p_i \log(p_i)\,. \tag{B.1}$$

The entropy can be interpreted as a measure of diversity of the system observed, in the sense of unpredictability of state, or equivalently, of the average information content of the system. For example, if there is only one element observable in the system, its probability would be 1 and using (B.1), we would get

$$H(p) = -1\log(1) = 0 \tag{B.2}$$

which would signal for a stable, information depraved system. On the other hand, having a uniformly distributed system with N elements observable, we would get an entropy of

$$H(p) = -\sum_{i=1}^{N} \frac{1}{N} \log\left(\frac{1}{N}\right) = -\log\left(\frac{1}{N}\right) \tag{B.3}$$

which implies that, as $N \to \infty$, the information content of the system explodes.

B.2 Kullback-Leibler Divergence

Kullback-Leibler divergence, another concept originating from information theory, was introduced by Solomon Kullback and Richard Leibler in 1951. It measures the similarity, or the 'distance', of one probability distribution from another.

Kullback-Leibler divergence from distribution q to distribution p can be calculated using the formula

$$D_{KL}(p\|q) = E_p\left[\log\left(\frac{q}{p}\right)\right]. \tag{B.4}$$

In the discrete case, (B.4) evaluates to

$$D_{KL}(p\|q) = \sum_i p_i \log\left(\frac{q_i}{p_i}\right). \tag{B.5}$$

In machine learning, Kullback-Leibler divergence is interpreted as *information gain* achieved when using q instead of p. In information theory, it is interpreted as the *relative entropy* of p with respect to q.

Kullback-Leibler divergence has two key properties, both of which are easily deduced. The first one is

$$D_{KL}(q\|p) \geq 0\,. \tag{B.6}$$

The second one is the fact that Kullback-Leibler divergence is not necessarily symmetrical, i.e.

$$D_{KL}(q\|p) \neq D_{KL}(p\|q) \tag{B.7}$$

in most cases.

Appendix C

Principal Component Analysis

Appendix C explains the mathematical model of PCA.

C.1 The Model

PCA is a model mainly used for dimensionality reduction and denoising. Based on orthogonal linear transformations, the aim is to convert a set of possibly correlated variables to a set of linearly uncorrelated variables called principal components.

Given a data matrix of size $n \times m$, the number of distinct principal components is $\min\{n-1, m\}$. They are ordered in such a way that the first principal component accounts for the most variance in the data set, while the next one accounts for most variance with respect to the constraint of being orthogonal to the previous principal component(s).

The basic outlay of the model goes as follows: assume we are given a data matrix $X \in \mathbb{R}^{n\times m}$ where for each column of X its empirical mean is equal to zero. Then, the transformation is given as

$$T = XW \tag{C.1}$$

where $W \in \mathbb{R}^{m\times m}$ is the matrix defining the transformation and $T \in \mathbb{R}^{n\times m}$ is the resulting matrix of principal components. The columns of W are calculated as

$$w_{(1)} = arg\max\{\frac{w^T X^T X w}{w^T w}\} \tag{C.2}$$

for the first column of W and

$$w_{(k)} = arg\max\{\frac{w^T \hat{X}_k^T \hat{X}_k w}{w^T w}\} \tag{C.3}$$

for the k^{th} column of W where

$$\hat{X}_k = X - \sum_{i=1}^{k-1} X w_{(i)} w_{(i)}^T . \tag{C.4}$$

The dimensionality reduction step is done simply by using a smaller number of principal components, that is

$$T_{n\times L} = X_{n\times m} W_{m\times L} \tag{C.5}$$

where $L < m$.

Bibliography

[1] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[2] Pierre Baldi. "Autoencoders, Unsupervised Learning and Deep Architectures". In: *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop - Volume 27*. UTLW'11. Washington, USA: JMLR.org, 2011, pp. 37–50. URL: http://dl.acm.org/citation.cfm?id=3045796.3045801.

[3] Maayan Baron et al. "A Single-Cell Transcriptomic Map of the Human and Mouse Pancreas Reveals Inter- and Intra-cell Population Structure". In: *Cell Syst* 3.4 (2016). 27667365[pmid], 346–360.e4. ISSN: 2405-4712. DOI: 10.1016/j.cels.2016.08.011. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC5228327/.

[4] Fernando H Biase, Xiaoyi Cao, and Sheng Zhong. "Cell fate inclination within 2-cell and 4-cell mouse embryos revealed by single-cell RNA sequencing." In: *Genome research* 24 11 (2014), pp. 1787–96.

[5] Christopher M. Bishop. *Pattern recognition and machine learning*. Textbook for graduates.;Includes bibliographical references (pages 711-728) and index. New York : Springer, [2006] ©2006, [2006]. URL: https://search.library.wisc.edu/catalog/9910032530902121.

[6] François Chollet et al. *Keras*. https://keras.io. 2015.

[7] Francois Chollet. *Deep Learning with Python*. 1st. Greenwich, CT, USA: Manning Publications Co., 2017. ISBN: 1617294438, 9781617294433.

[8] Valerio Costa et al. "RNA-Seq and human complex diseases: recent accomplishments and future perspectives". In: *Eur J Hum Genet* 21.2 (2013). 22739340[pmid], pp. 134–142. ISSN: 1018-4813. DOI: 10.1038/ejhg.2012.129. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3548270/.

[9] Qiaolin Deng et al. "Single-Cell RNA-Seq Reveals Dynamic, Random Monoallelic Gene Expression in Mammalian Cells". In: *Science* 343 (2014), pp. 193–196.

[10] Jiarui Ding, Anne Condon, and Sohrab P. Shah. "Interpretable dimensionality reduction of single cell transcriptome data with deep generative models". In: *Nature Communications* 9.1 (2018), p. 2002. ISSN: 2041-1723. DOI: 10.1038/s41467-018-04368-5. URL: https://doi.org/10.1038/s41467-018-04368-5.

[11] James Eberwine et al. "The promise of single-cell sequencing". In: *Nature Methods* 11 (2013), 25 EP –. URL: http://dx.doi.org/10.1038/nmeth.2769.

[12] Amir Giladi and Ido Amit. "Single-Cell Genomics: A Stepping Stone for Future Immunology Discoveries". In: *Cell* 172.1 (2018), pp. 14–21. ISSN: 0092-8674. DOI: 10.1016/j.cell.2017.11.011. URL: http://dx.doi.org/10.1016/j.cell.2017.11.011.

[13] Mubeen Goolam et al. "Heterogeneity in Oct4 and Sox2 Targets Biases Cell Fate in 4-Cell Mouse Embryos". In: *Cell* 165.1 (2016). S0092-8674(16)30061-7[PII], pp. 61–74. ISSN: 0092-8674. DOI: 10.1016/j.cell.2016.01.047. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4819611/.

[14] Christopher Heje Grønbech et al. *scVAE: Variational auto-encoders for single-cell gene expression data*. 2018. DOI: 10.1101/318295. URL: https://doi.org/10.1101/318295.

[15] Daniel Hebenstreit. "Methods, Challenges and Potentials of Single Cell RNA-seq". In: *Biology (Basel)* 1.3 (2012). biology-01-00658[PII], pp. 658–667. ISSN: 2079-7737. DOI: 10.3390/biology1030658. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4009822/.

[16] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. [Online]. 2001–. URL: http://www.scipy.org/.

[17] D. P Kingma and M. Welling. "Auto-Encoding Variational Bayes". In: *ArXiv e-prints* (Dec. 2013). arXiv: 1312.6114 [stat.ML].

[18] Vladimir Yu Kiselev, Andrew Yiu, and Martin Hemberg. "scmap: projection of single-cell RNA-seq data across data sets". In: *Nature Methods* 15 (2018), 359 EP –. URL: http://dx.doi.org/10.1038/nmeth.4644.

[19] Olsen Thale Kristin and Baryawno Ninib. "Introduction to Single-Cell RNA Sequencing". In: *Current Protocols in Molecular Biology* 122.1 (), e57. DOI: 10.1002/cpmb.57. eprint: https://currentprotocols.onlinelibrary.wiley.com/doi/pdf/10.1002/cpmb.57. URL: https://currentprotocols.onlinelibrary.wiley.com/doi/abs/10.1002/cpmb.57.

[20] Il-Youp Kwak et al. "DrImpute: Imputing dropout events in single cell RNA sequencing data". In: *bioRxiv* (2017). DOI: 10.1101/181479. eprint: https://www.biorxiv.org/content/early/2017/08/28/181479.full.pdf. URL: https://www.biorxiv.org/content/early/2017/08/28/181479.

[21] Y. Lecun. "A theoretical framework for Back-Propagation". In: (). URL: http://yann.lecun.com/exdb/publis/pdf/lecun-92b.pdf.

[22] Wei Vivian Li and Jingyi Jessica Li. "An accurate and robust imputation method scImpute for single-cell RNA-seq data". In: *Nature Communications* 9.1 (2018), p. 997. ISSN: 2041-1723. DOI: 10.1038/s41467-018-03405-7. URL: https://doi.org/10.1038/s41467-018-03405-7.

[23] Peijie Lin, Michael Troup, and Joshua W. K. Ho. "CIDR: Ultrafast and accurate clustering through imputation for single cell RNA-Seq data". In: *bioRxiv* (2016). DOI: 10.1101/068775. eprint: https://www.biorxiv.org/content/early/2016/08/10/068775.full.pdf. URL: https://www.biorxiv.org/content/early/2016/08/10/068775.

[24] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605. URL: http://www.jmlr.org/papers/v9/vandermaaten08a.html.

[25] Wes McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 51 –56.

[26] Seonwoo Min, Byunghan Lee, and Sungroh Yoon. "Deep learning in bioinformatics". In: *Briefings in Bioinformatics* 18.5 (2017), pp. 851–869. DOI: 10.1093/bib/bbw068. eprint: /oup/backfile/content_public/journal/bib/18/5/10.1093_bib_bbw068/1/bbw068.pdf. URL: http://dx.doi.org/10.1093/bib/bbw068.

[27] Mauro?J Muraro et al. "A Single-Cell Transcriptome Atlas of the Human Pancreas". In: *Cell Systems* 3.4 (2016), 385–394.e3. ISSN: 2405-4712. DOI: 10.1016/j.cels.2016.09.002. URL: http://dx.doi.org/10.1016/j.cels.2016.09.002.

[28] Travis E. Oliphant. *Guide to NumPy*. 2nd. USA: CreateSpace Independent Publishing Platform, 2015. ISBN: 151730007X, 9781517300074.

[29] K. Pearson. "On lines and planes of closest fit to points in space". In: *Philos. Mag* 2, (1901), pp. 559–572.

[30] Sandhya Prabhakaran et al. "Dirichlet Process Mixture Model for Correcting Technical Variation in Single-Cell Gene Expression Data". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 2016, pp. 1070–1079. URL: http://proceedings.mlr.press/v48/prabhakaran16.html.

[31] Sabrina Rashid et al. "Project Dhaka: Variational Autoencoder for Unmasking Tumor Heterogeneity from Single Cell Genomic Data". In: *bioRxiv* (2018). DOI: 10.1101/183863. eprint: https://www.biorxiv.org/content/early/2018/04/19/183863.full.pdf. URL: https://www.biorxiv.org/content/early/2018/04/19/183863.

[32] F. Rosenblatt. "The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain". In: *Psychological Review* (1958), pp. 65–386.

[33] Guido Rossum. *Python Reference Manual*. Tech. rep. Amsterdam, The Netherlands, The Netherlands, 1995.

[34] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *CoRR* abs/1609.04747 (2016). arXiv: 1609.04747. URL: http://arxiv.org/abs/1609.04747.

[35] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1". In: ed. by David E. Rumelhart, James L. McClelland, and CORPORATE PDP Research Group. Cambridge, MA, USA: MIT Press, 1986. Chap. Learning Internal Representations by Error Propagation, pp. 318–362. ISBN: 0-262-68053-X. URL: http://dl.acm.org/citation.cfm?id=104279.104293.

[36] L. Theis et al. "Lossy Image Compression with Compressive Autoencoders". In: *ArXiv e-prints* (Mar. 2017). arXiv: 1703.00395 [stat.ML].

[37] Bertrand Thirion et al. *scikitlearn*. http://mloss.org/software/view/240/. 2016.

[38] Dongfang Wang and Jin Gu. "VASC: dimension reduction and visualization of single cell RNA sequencing data by deep variational autoencoder". In: *bioRxiv* (2017). DOI: 10.1101/199315. eprint: https://www.biorxiv.org/content/early/2017/10/06/199315.full.pdf. URL: https://www.biorxiv.org/content/early/2017/10/06/199315.

[39] Yurong Xin et al. “RNA Sequencing of Single Human Islet Cells Reveals Type 2 Diabetes Genes”. In: *Cell Metabolism* 24.4 (2016), pp. 608–615. ISSN: 1550-4131. DOI: 10.1016/j.cmet.2016.08.018. URL: http://dx.doi.org/10.1016/j.cmet.2016.08.018.

[40] L. Zamparo and Z. Zhang. “Deep Autoencoders for Dimensionality Reduction of High-Content Screening Data”. In: *ArXiv e-prints* (Jan. 2015). arXiv: 1501.01348 [cs.LG].