1 2 9 0

UNIVERSIDADE Ð COIMBRA

Adriana Monteiro e Cunha

NEURAL NETWORKS FOR 2D REPRESENTATIONS OF CELL EXPRESSION

Dissertação no âmbito do Mestrado Integrado em Engenharia Biomédica, no ramo de Bioinformática, orientada pelo Professor Doutor Joel P. Arrais.

Outubro de 2020

UNIVERSIDADE D COIMBRA FACULDADE DE CIÊNCIAS E TECNOLOGIA

Adriana Monteiro e Cunha

Neural Networks for 2D Representations of Cell Expression

Thesis submitted to the University of Coimbra
for the degree of Master in Biomedical Engineering

Supervisor: Prof. Dr. Joel P. Arrais (CISUC, University of Coimbra)

Coimbra, 2020

This work was developed in collaboration with:

Center for Informatics and Systems of the University of Coimbra

Acknowledgements

Primeiramente agradeço ao Professor Joel P. Arrais pela orientação e apoio ao longo do último ano. Pela oportunidade que me foi concedida, pela liberdade e responsabilidade que me foram confiadas e pela preocupação para com o meu trabalho, bem-estar e enriquecimento pessoal. Desde as tardes de trabalho convividas com bolo, aos obstáculos impostos pelo trabalho de investigação, não poderia ter sido melhor recebida no laboratório que me acolheu durante este período.

Agradeço, seguidamente, à família que me carregou até aqui e que suportou o meu percurso académico. Pela incondicional presença, incontornáveis reprimendas e imensas celebrações. Pela paciência de responder a todos os porquês, de ler todos os livros a que a rotina de deitar obrigou e de pintar o mundo como sendo desconhecido e à espera de ser estudado.

Aos velhos amigos que me viram embarcar e comigo navegaram esta passagem. Pela constante compreensão e disponibilidade, por mesmo estando longe me terem acompanhado de tão perto. Um agradecimento pelas ilustrações, cartas e poemas, conselhos maternais, graças cantadas, espontâneos encontros e conversas estendidas. Particularmente, ao João Duarte pela lembrança e ajuda ao longo deste projeto e ao João Rocha e à Mafalda Fernandes pela constante preocupação e amparo.

Aos novos amigos que a viagem me trouxe. À Laura, à Mariana e à Natália pelos partilhados agoiros à porta dos exames e pelos lanches comemorativos que os seguiram. Por me terem trazido a saudade de Coimbra.

Financing

This research has been funded by the Portuguese Research Agency FCT, through D4 - Deep Drug Discovery and Deployment (CENTRO-01-0145-FEDER-029266).

Resumo

Os recentes avanços nas tecnologias de sequenciação do transcriptoma humano levaram ao aumento de estudos baseados em dados de expressão genética, com notável impacto nas áreas da biologia e medicina. Tipicamente, o trabalho desenvolvido com base neste tipo de informação recorre a técnicas de redução de *features* para combater os problemas que advêm da *curse of dimensionality* e associados à extração de dados de expressão (como eventos de *dropout*, ruído, *etc.*), sobretudo em projetos com tarefas de classificação.

Nesta dissertação apresenta-se um modelo de redução de dimensionalidade inspirado em redes neuronais, o *Autoencoder* Supervisionado, que acopla a arquitetura tradicional de *autoencoders* com uma camada de classificação *SoftMax*, para que as representações no espaço latente maximizem a separabilidade entre diferentes classes. De forma a considerar os recorrentes eventos *dropout* neste tipo de dados, foi usada uma camada *Dropout* na fase de treino, conferindo maior robustez ao modelo.

O estudo em causa foca-se em particular em reduções para duas dimensões, de forma a facilitar a visualização gráfica de informação. Além da análise do efeito da contabilização de classes no processo de redução de *features* (*a priori* de potenciais tarefas de classificação), explorou-se a possibilidade de o espaço latente obtido permitir aferir novos padrões de semelhança entre amostras.

O modelo foi validado usando três conjuntos de dados, comparando os seus resultados com os obtidos através de *Principal Component Analysis* e do *autoencoder* simples equivalente, bem como através da análise do mapa de calor dos dados completos de expressão genética agrupados através do *clustering* hierárquico das *features* reduzidas.

Os resultados mostram que o modelo é capaz de gerar representações adequadas dos dados originais, que permitem facilitar a tarefa de classificação quando comparadas com as resultantes das técnicas estado-da-arte. No entanto, não foi possível utilizá-las para estabelecer novos paralelos entre amostras.

Palavras-Chave: Redução De *Features*; Expressão Genética; *Autoencoder*; Aprendizagem Supervisionada; Visualização De Dados

Abstract

The recent advances in transcriptome sequencing technologies lead to the increase of gene expression studies, with significant impact in the fields of cellular biology and medicine. Typically, the work developed based on this type of data resorts to feature reduction techniques to combat the problems risen by the *curse of dimensionality* and from data extraction (such as dropout events, noise, *etc.*), especially in projects involving classification tasks.

This dissertation presents a novel dimensionality reduction model inspired by deep neural networks, the *Supervised Autoencoder*, which combines the architecture of traditional autoencoders with a SoftMax classification layer, so the latent space maximizes different classes' separability. To account for the recurring dropout events in this type of datasets, a Dropout layer was implemented during training, improving the model's robustness.

The present study focuses particularly on two-dimensional reductions to ease the information's visualisation. In addition to an analysis of the effect of label usage in the feature reduction process (prior to potential classification tasks), the possibility of inferring new similarity patterns between samples through the latent space was explored.

The model was validated with three datasets, comparing its results with those of *Principal Component Analysis* and the equivalent simple autoencoder, as well as by analysing the heatmap of the complete gene expression clustered based on the engineered features.

The results show the model is capable of meaningful representations of the original data that ease the classification task compared to the ones resultant of state-of-the-art techniques. However, it is not possible to draw new parallels between samples based on those features.

Keywords: Feature Reduction; Gene Expression Profiling; Autoencoder; Supervised Learning; Data Visualisation

Contents

List of Tables

List of Figures

Abbreviations

2D 2 dimensions

A Adenine

AE (Undercomplete) Autoencoder

AN Artificial Neuron

C Cytosine

DNA Deoxyribonucleic Acid

FR Feature Reduction

G Guanine

HPO Hyperparameter Optimisation

KL Kullback-Leibler

mRNA Messenger Ribonucleic Acid

MSE Mean Square Error

NN Neural Network

PC1 First Principal Component

PCA Principal Component Analysis

PP Pancreatic Polypeptide

RNA Ribonucleic Acid

PR Pattern Recognition

RS Random Search

SAE Supervised Autoencoder

SSD Sum of the Squared Distances

scRNA-seq Single-cell RNA sequencing

T Thymine

t-SNE t-Distributed Stochastic Neighbour Embedding

U Uracil

VAE Variational Autoencoder

1. Introduction

1.1 Context

The central dogma of molecular biology details the transfer process of sequential information in cells. Specifically, how genes that encode said information are transcribed into RNA to produce proteins, also referred to as gene expression [1].

Gene profiling techniques aim to measure gene expression levels at a given time, thus allowing researchers to obtain fundamental knowledge about cells' characteristics and regulation [2]. This understanding of cellular behaviour has shown a particularly significant role in medicine as it provides targeted molecular procedures [3].

Recently, Tang et al. (2009) [4] developed an innovative profiling technology, Single-cell RNA sequencing (scRNA-seq), capable of measuring the gene expression of individual cells. Thus, in the past decade, there was a growth in genomic studies that lead to a variety of discoveries about complex biological systems and health conditions [5] [6].

A significant portion of the mentioned progress lies in finding expression patterns within populations of cells, which was not possible using the outdated techniques [7].

The study of data similarities and differences, such as that of genetic research, usually falls under the scientific discipline of Pattern Recognition (PR). PR is the field dedicated to object description and classification. Figure 1.1.1 shows the workflow of PR systems, which is independent of the approach followed to design them [8].

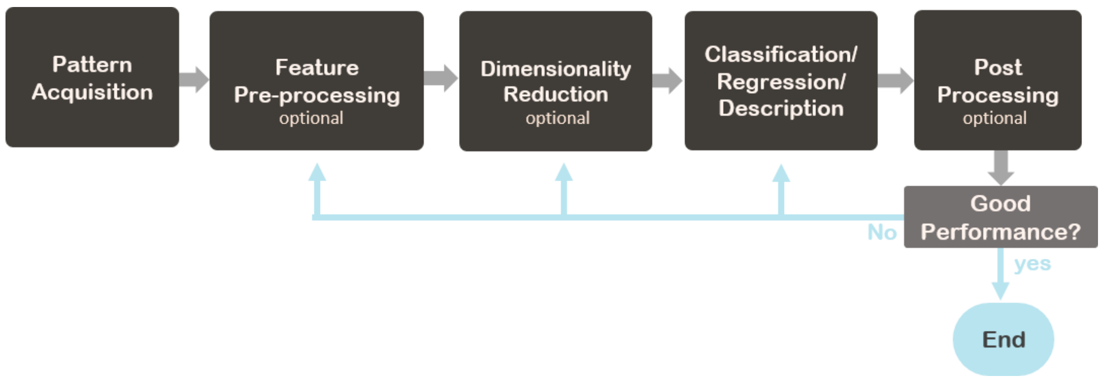
Pattern Acquisition
Feature Pre-processing optional
Dimensionality Reduction optional
Classification/ Regression/ Description
Post Processing optional
Good Performance?
No
yes
End

Figure 1.1.1 The workflow of a PR system (independent of the model design).

Note the optional steps, though not always necessary, play a crucial part in improving the task's performance. For example, dimensionality reduction, which could be accomplished through feature selection or feature reduction (which combines or transforms the features to originate new ones), helps to provide results with less information, avoiding redundancy and allowing a better understanding of the problem [9].

1.2 Motivation and Goals

Sample classification based on gene expression profiling is a widely used task with several biomedical purposes [10]–[12]. However, there are several challenges associated with this type of data when using PR systems. The *curse of dimensionality* dictates that the number of samples required to determine an accurate classification function grows exponentially with the number of features [13].

Since there are dozens of thousands of genes in each cell, dimensionality reduction becomes an essential step in the experimental setup [14]–[17]. Considering that commonly the responsible genes for the phenomenon of study are unknown, feature reduction (FR) is preferable.

As previously mentioned, the main purpose of FR is to improve the classifier's performance, as well as its learning efficiency. Noticeably, it also reduces the computational complexity, as more features translate into more classification parameters (like the weights in neural networks) [18]. Furthermore, feature reduction helps avoiding overfitting (it often happens, when the number of instances is too small, that the model learns the specific data characteristics and loses its generalisation ability); and dealing with noise (any property of the pattern due to extraction methods) [19].

Thus, the presented dissertation focuses on the development of a novel feature reduction model, the Supervised Autoencoder, designed to aid the pre-processing of gene expression data. This research should achieve the following goals:

- Enable the visualisation of gene expression data in 2 dimensions, easing its representation;

- Facilitate a posteriori classification;
- Study the effect of weighting the data's true label in the feature engineering process in addition to classification;
- Maximise separability between different classes in the two-dimensional space to optimise classification;
- Find new similarities between samples based on the engineered features and infer meaningful biological conclusions.

1.3 Document Structure

The remainder of this document is organised in 6 chapters. Chapter 2 – Background Knowledge provides an overview of the base concepts required to understand the project. Firstly, the notions of gene expression and its characteristics are better detailed. Then, a brief explanation of machine learning and neural networks, the algorithms used in this experiment and a practical analysis of Heatmaps and hierarchical clustering (used for the model validation) are provided.

Chapter 3 – State of the Art presents a summary of the main literature regarding feature reduction, autoencoders and the specific models developed for handling FR of gene expression data.

Chapter 4 – Data Description and Pre-processing credits the datasets used for this study, provides a short description of the data labels and introduces the first step of pre-processing, data normalisation.

Chapter 5 – Model and Experimental Setup schematises the proposed architecture and details the experimental setup, while providing a short reasoning behind its design.

Chapter 6 – Results and Discussion displays the results and their assessment.

In Chapter 7 – Conclusion the final takeaway from this project is summarised and future steps towards optimising the model are discussed.

2. Background Knowledge

2.1 Gene Expression

2.1.1 Basic Concepts

Nucleic Acids, named as such for being first discovered in the nucleus of cells, are biological polymers responsible for the storage and encoding of information essential to life. They are formed by a chain of units (monomers) called nucleotides.

There are two types of Nucleic Acids, Deoxyribonucleic acid (DNA) and ribonucleic acid (RNA). Both have similarly structured nucleotides, composed of a pentose, a nitrogenous base, and a phosphate group (illustrated in Figure 2.1.1.1 B). The differences between the two are the pentoses - a Deoxyribose in DNA and a Ribose in RNA - and the fact that DNA is structured in a double helix, contrasting with RNA's simple chain.

It is also relevant to note that some nitrogenous bases are exclusive to each polymer. There are five different nitrogenous bases – Adenine (A); Guanine (G); Cytosine (C); Thymine (T), only present in DNA; and Uracil (U), exclusive to RNA (Figure 2.1.1 A) [20].

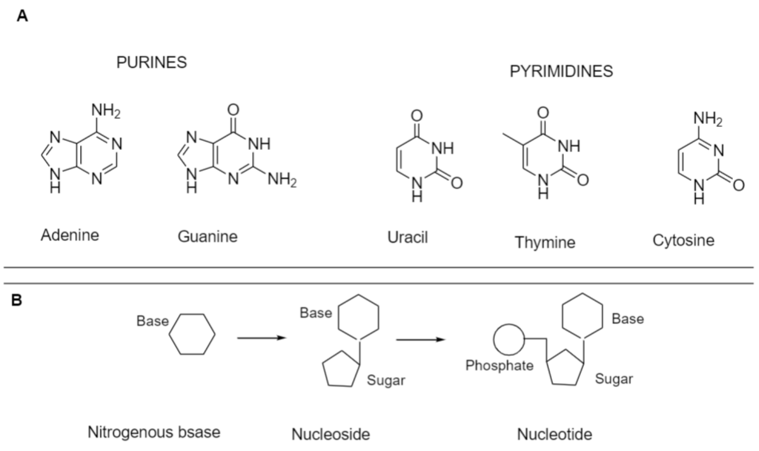

Figure 2.1.1.1 - A) Nitrogenous bases, B) Basic structure of a nucleotide [21].

Due to their chemical composition, hydrogenic bonds are formed between the nitrogenous bases of the two DNA strands that comprise the helix (or temporarily in RNA strands). These bonds only occur for G – C pairs and A – T or A – U pairs depending on whether the chain is DNA or RNA, respectively. Thus, the information in the two DNA strands of the double helix is redundant and can be fully encoded by a simple RNA strand with the pair bases [22].

2.1.2 Proteins Production

The information encoded in a DNA strand is tied to protein production. Proteins have a vital role in a cell's life. They monitor and execute all its functions, both general (e.g. intracellular digestion) and specific to the cell's type (e.g. Production of mucus in caliciform cells).

Figure 2.1.2.1 shows a simplified model of protein synthesis in eukaryotic cells. A portion of DNA is encoded into a messenger RNA (mRNA) strand with pairing nitrogenous bases (Transcription). The strand then undergoes a process called Translation, resulting in a protein. To the portion of DNA containing information to synthesise a protein, we call a gene [23]. Genes are said to be expressed when transcribed to RNA molecules.

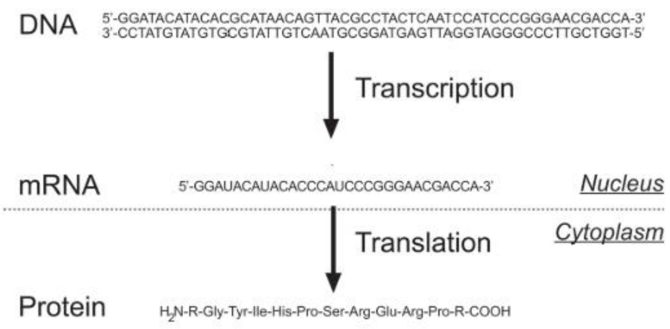
DNA
5'-GGATACATACACGCATAACAGTTACGCCTACTCAATCCATCCCGGGAACGACCA-3'
3'-CCTATGTATGTGCGTATTGTCAATGCGGATGAGTTAGGTAGGGCCCTTGCTGGT-5'
Transcription
mRNA
5'-GGAUACAUACACCCAUCCCGGGAACGACCA-3'
Nucleus
Translation
Cytoplasm
Protein
H2N-R-Gly-Tyr-Ile-His-Pro-Ser-Arg-Glu-Arg-Pro-R-COOH

Figure 2.1.2. 1 - Protein synthesis in eucaryotic calls. Adapted from [23].

2.1.3 Gene Expression Profiling

The gene expression profile, also designated transcriptome, corresponds to the set of mRNA molecules present in a cell [23]. Since it is quite dynamic and sensitive to environmental perturbations or cellular events, the transcriptome provides significant information about morphological and phenotypic characteristics of the cell [24]. Comparisons between gene expression profiles of different cells or of the same cell in distinct stages may also present knowledge about the function and role of certain genes

[25], the causes and development of diseases [26], how drugs affect said disease and cell, and what genes are appropriate targets for the drugs [27].

Current technology allows profiling the genome, epigenome and transcriptome of individual cells. This is called single-cell sequencing and, though it may lead to a deeper understanding of individual cells' responses to pathologies and phenomena [28], is still a rather flawed process. Gawad et al. (2016) [29] noted the physical isolation of individual cells and the amplification of the genetic material to sufficient proportions for sequencing create a challenge in analysing the data due to the biases and errors that are introduced during these procedures.

In fact, profiling methods are often prone to noise, dropout events or missing values for several reasons (e.g. contamination of samples or poor resolution). The quantity of mRNA extracted, reverse transcription biases and ambient conditions also cause systemic errors in the expression intensity measurements [20]. Thus, when experimenting on transcriptomes, it is essential to account for these problems, through data normalisation and the model design.

2.2 Neural Networks

2.2.1 Neurons

Artificial neural networks (NNs) are computational models inspired by the structure of the brain. Their processing units, the artificial neurons (ANs) (Figure 2.2.1.1), may be connected and transmit signals between themselves. However, similarly to their biological equivalents, not all signals stimulate the same neurons. The implementation of activation functions simulates this all-or-nothing effect. If activated, the neuron processes the signal, otherwise it returns zero [30]. The activation function is applied to the sum of the weighted input signals (with an added bias), resulting in an output that may serve as input to other neurons.

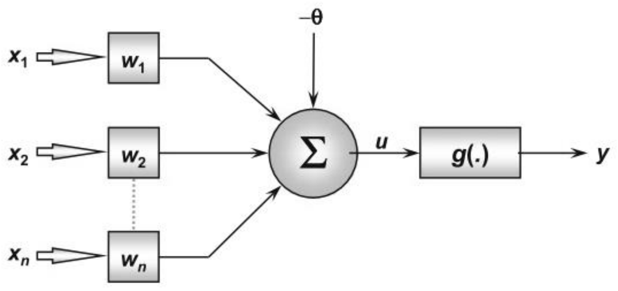

Figure 2.2.1. 1 - Neuron Structure [30].

The activation functions used in this project are described below:

- Sigmoid Function: The sigmoid function, presented in Figure 2.2.1.2, returns a value between 0 and 1.

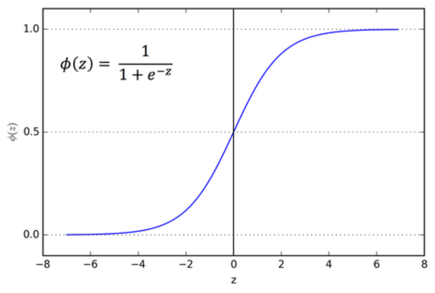

Figure 2.2.1. 2 - Sigmoid Function [31].

- Linear Function: The linear function, shown in Figure 2.2.1.3, returns its input value.

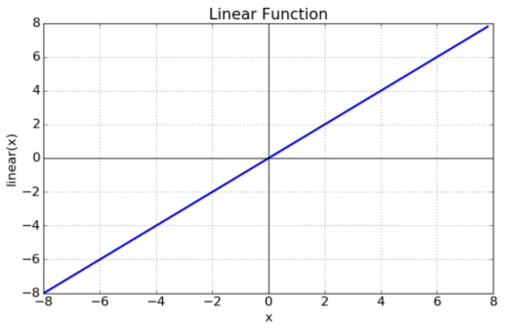

Figure 2.2.1. 3 – Linear Function [31].

- SoftMax Function: SoftMax activation functions, explained in Figure 2.2.1.4, are typically used for classification tasks. They return a categorical vector of values between 0 and 1 that sum to 1. Thus, their result could be interpreted as a probability distribution [32].

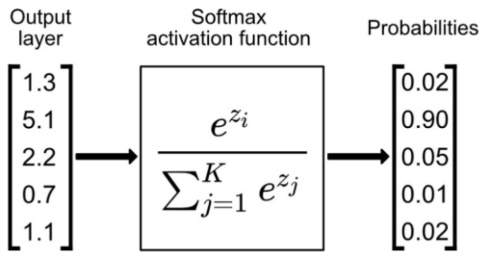
Output layer
Softmax activation function
Probabilities
1.3
5.1
2.2
0.7
1.1
$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
0.02
0.90
0.05
0.01
0.02

Figure 2.2.1. 4 - SoftMax function [33].

2.2.2 Layers

Figure 2.2.2.1 displays a simple NN.

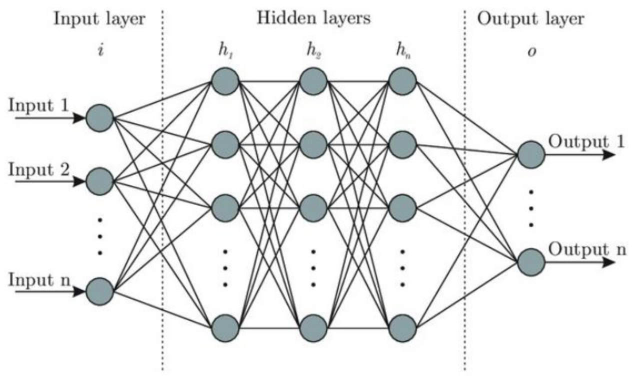

Figure 2.2.2. 1– Neural Network Architecture [34].

The neurons are organised in layers that can be categorised into three groups [35]:

- Input Layer - responsible for receiving the data. It does not transform the instances. Each node (i.e. neuron) corresponds to a feature, allowing the rest of the model to infer the data's shape;

- Hidden Layer(s) - any layer between the Input and Output layers. They perform most of the internal processing;
- Output Layer - returns the final network output (for example, a categorical vector with classifications resulting from a SoftMax function).

There are several types of layers. The ones represented in Figure 2.2.2.1 and mostly implemented throughout this research are designated Fully Connected layers because all outputs from a hidden layer are connected to each neuron of the next layer. Other examples are Dropout layers, which will be furthered detailed in future sections. They randomly set input units to 0 [36].

2.2.3 Parameter optimisation and learning strategies

The purpose of neural networks is to transform input data into a significant output. To learn the relationship between inputs and the desired output, the parameters (weights and biases) are tuned during a process called training (or learning).

There are three learning strategies, supervised learning (all desired outputs are labelled), unsupervised learning (there is no knowledge about the outputs) and semi-supervised learning (part of the labels is known). The present section focuses on parameter optimisation using the supervised learning approach [37].

The training process is iterative. In each iteration (epoch), a loss function is computed. Loss functions compare the output of the model with the true known labels, resulting in a similarity score. Thus, the weights and biases optimisation may be perceived as a minimisation problem of the loss function [38]. The Categorical Cross-Entropy (CCE) loss, ideal for multi-class classification, is shown in equation 1. For regression problems, the Mean Squared Error (MSE) (equation 2) is often chosen.

$$CCE = - \sum_{c=1}^{M} y_{i,c} log(p_{i,c}) \tag{1}$$

Where M is the number of classes, $y_{i,c}$ the indicator (0 or 1) if c is the true class and $p_{i,c}$ the predicted probability of c.

$$MSE = \frac{1}{\mathrm{n}} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2 \tag{2}$$

Where n is the number of samples, y the predicted labels and $\tilde{y}$ the true class.

At the end of each epoch, the parameters are updated based on the iteration's loss through a gradient descent method. Figure 2.2.3 illustrates the training process. The data goes through the model (forward propagation), resulting in a loss, which is used to calculate the updated weights (backpropagation).

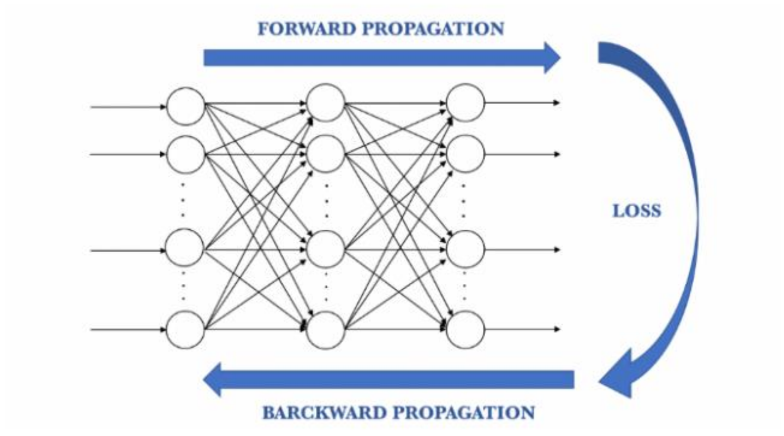

Figure 2.2.3. 1 - Training Process [39].

2.3 Hierarchical Clustering and Heatmaps

Heatmaps are a powerful visualisation method for high-dimensional data [40]. Eisen et al. (1998) [41] proved them to be particularly well suited for gene expression data analysis when associated with hierarchical clustering. Figure 2.3.1 presents a simple heatmap, where each column represents a gene and each row a sample. The provided legend shows that light, bright colours mean higher expression levels and dark colours lower expression levels.

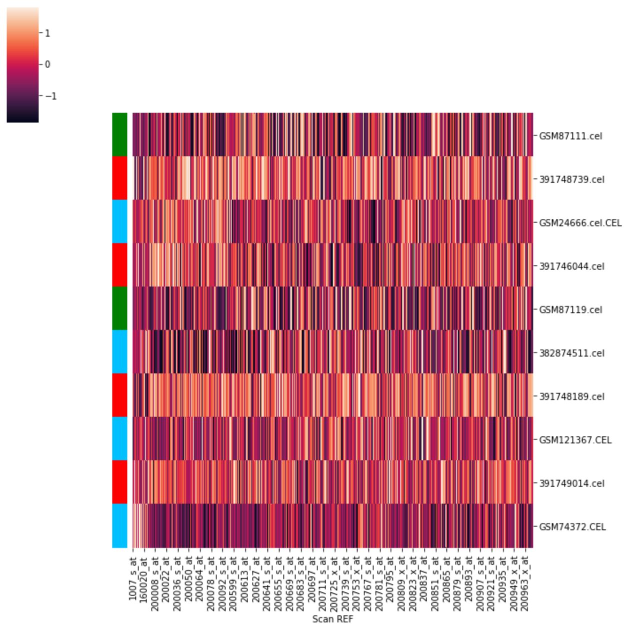

Figure 2.3.1 A simple heatmap of gene expression data. The red, blue and green strips identify the samples as three different types of cells.

It is worth emphasising the data was scaled. Scaling techniques will be further discussed in future sections. However, it is relevant to understand what they mean in terms of interpretability and their importance when visualising data [42]. Figure 2.3.2 illustrates the same data set without scaling. Notably, it is much more challenging to make any distinctions between samples. This is typically due to the presence of outliers - the first few columns of genes have significantly lower expression levels than the rest.

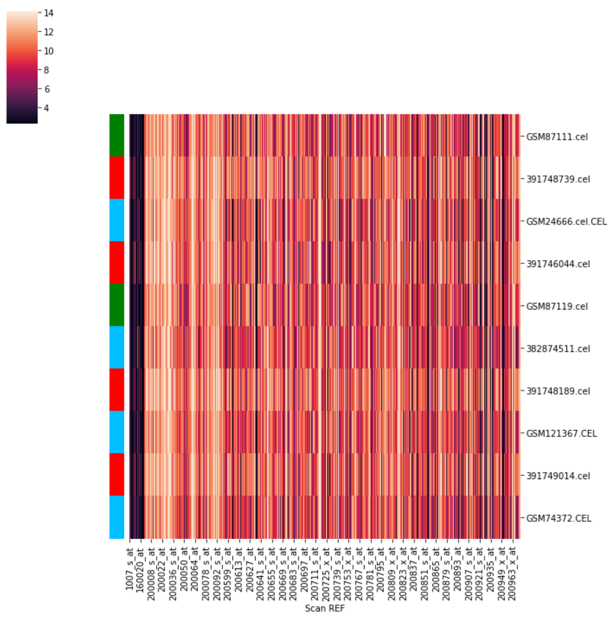

Figure 2.3. 2 - A simple heatmap without scaling.

The data displayed in Figure 2.3.1 was normalised per gene, i.e. each column was scaled independently. Consequently, the heatmap eases comparisons between samples but prevents drawing any conclusions within them (evaluate if a certain gene in the considered cell is more expressed than others). There are alternative scaling approaches [43], but they will not be detailed, since, in the context of this project, heatmaps were explored to compare instances, not features.

Figure 2.3.3 depicts the heatmap for the scaled dataset previously presented reassembled after hierarchical clustering. Each sample is a cluster of its own, which merges with the most similar one. This is repeated successively until all clusters are grouped in one. The result is represented in a dendrogram [42].

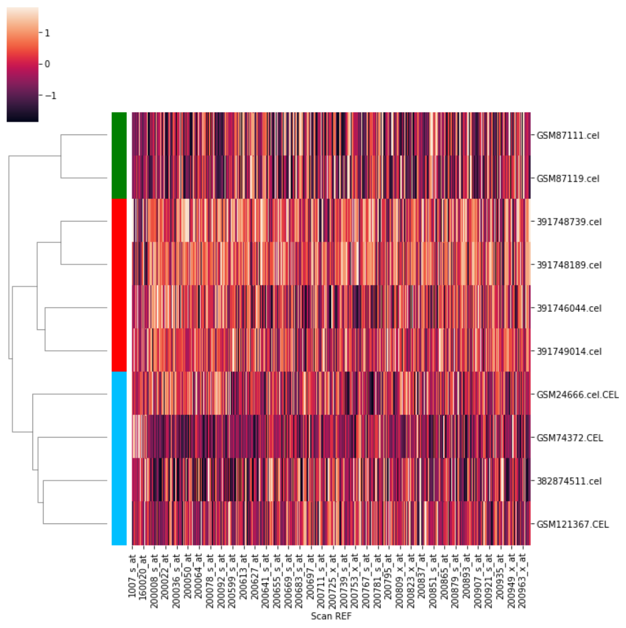

Figure 2.3. 3 – Heatmap with hierarchical clustering.

There are several algorithms and metrics when performing hierarchical clustering for assessing the similarity between clusters. The most common are described below:

- Single Linkage: the distance between two clusters is calculated using their two closest members [43].
- Complete Linkage the distance between two clusters is calculated with the two farthest-apart members [43].
- Group Average: the most used. The distance between two clusters is calculated using the average distance between each of their members [43].
- Euclidean Distance:

  It´s the most used metric [43]. Let d(p,q) be the Euclidean Distance between the two data points p and q with n features:

$$d(p,q) = \sqrt{\sum_{i=0}^{n}(p_i - q_i)^2} \qquad (3)$$

- Manhattan Distance:

Let d(p,q) be the Euclidean Distance between the two data points p and q with n features:

$$d(p,q) = \sum_{i=0}^{n} |p_i - q_i| \quad (4)$$

3. State of the Art

3.1 Principal Component Analysis

Harold Hotelling (1933) [44] introduced Principal Component Analysis (PCA) as the orthogonal projection of data onto a low dimensional linear space that maximises the data variance.

Figures 3.1.1 through 3.1.4 provide a simple explanation of this process for a scenario with five samples and two features, X1 and X2.

Figure 3.1.1 is a plot of the data and its first principal component (PC1), i.e., the linear projection that maximises the instances variance.

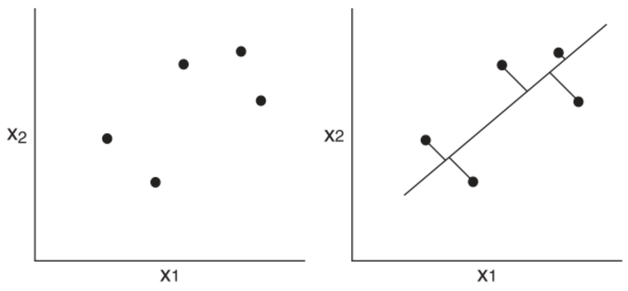

Figure 3.1. 1 - First Principal Component [45]

Since PCA uses variance as a criterion, it depends on units of measurement. Therefore, it is common practice standardise the variables [46]. Each data value x_{ij} is subtracted to the mean $\overline{x}_j$ and divided by the standard deviation of variable *j*, s_j, (equation 5).

$$z_{ij} = \frac{x_{ij} - \overline{x_j}}{\mathrm{sj}} \tag{5}$$

The result is illustrated in Figure 3.1.2.

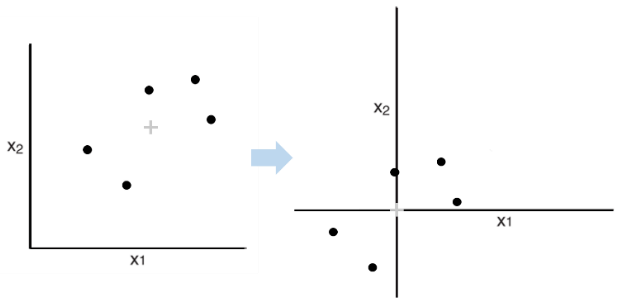

Figure 3.1. 2 – Data standardisation. Adapted from [45].

Note that finding the slope of the line that maximises the variances (i.e. of the principal component) corresponds to finding the slope of the line that maximises the sum of square distances (SSD) from the projected points to the origin (Figure 3.1.3). The singular vector that describes PC1 is then given by a linear combination of x1 and x2 (equation (6) and Figure 3.1.4) [47].

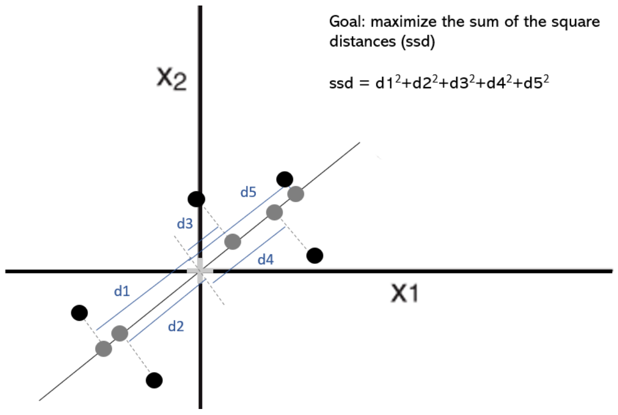
Goal: maximize the sum of the square distances (ssd)
ssd = d1^2+d2^2+d3^2+d4^2+d5^2
X2
X1
d1
d2
d3
d4
d5

Figure 3.1. 3 - Criteria for finding Principal Components. Adapted from [45].

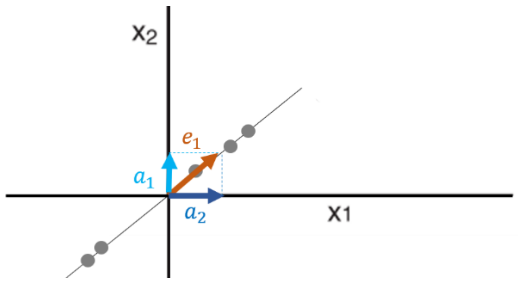

Figure 3.1. 4 – vector decomposition of PC1. Adapted from [45].

Let e_1 be the singular vector for PC1,

$$e_1 = a_1 x_1 + a_2 x_2 \tag{6}$$

Where a_1 and a_2 are loading scores. Notably, e_1 corresponds to the eigenvector and SSD to the eigenvalue of the correlation matrix between the data points and the features – an alternative calculus for PCA [48].

Lenz et al. (2016) [49] summarised in their work an evaluation of PCA and its suitability to gene expression data. They showed the intrinsic value of its representations to be higher than previously estimated. Moreover, they demonstrated the PCA's biological meaning to be proportional to the size of the relevant signal and the portion of samples containing it.

3.2 t-Distributed Stochastic Neighbour Embedding

Laurens van der Maaten and Geoffrey Hinton (2008) [50] developed a feature reduction technique, the t-Distributed Stochastic Neighbour Embedding (t-SNE), suited for the visualisation of high-dimensional data. The model was validated with five diversified datasets through a comparison of its results to those of seven state-of-the-art visualisation methods. Their model outperformed every other one for all datasets.

t-SNE uses random walks on neighbourhood graphs so that the underlying structure of all data influences the projections of its subsets. A detailed summary of the method along with the didactic example of a 2D to 1D reduction follows.

Consider the data represented in Figure 3.2.1, which is divided into 3 clusters.

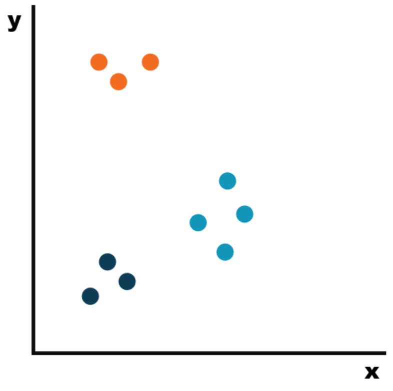

Figure 3.2. 1 – High dimensional space with three clusters.

Firstly, the algorithm calculates the instances similarity based on the Euclidean distances between them. To this intent, let us consider a data point (x_i) and centre a Gaussian Distribution around it (illustrated in Figure 3.2.2). By projecting the Euclidean Distance between x_i and another datapoint, x_j, onto the Gaussian's curve, we obtain the conditional probability $p_{j|i}$. Laurens van der Maaten and Geoffrey Hinton (2008) [50] described $p_{j|i}$ as the probability that 'x_i would pick x_j as its neighbour'.

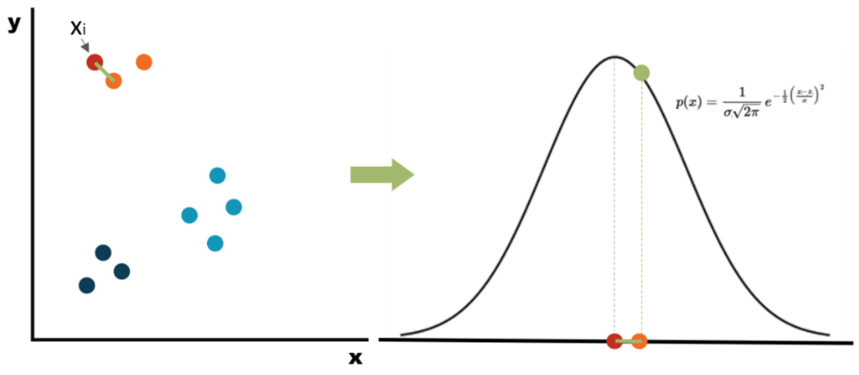
y
Xi
x
$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$

Figure 3.2. 2 – Converting Euclidean Distances to Similarities.

This process must be repeated for all potential neighbours of x_i (Figure 3.2.3). Note that due to the normal distribution, close points have higher probability values, i.e., higher similarity values.

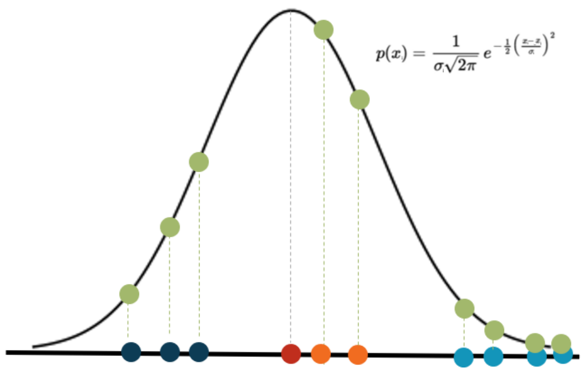
$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$

Figure 3.2. 3 – First step of the t-SNE method.

The process described must be repeated for all (k) data points and the conditional probabilities normalised so that their sum is 1. Thus, $p_{i|j}$ is given by equation 7.

$$p_{ij} = \frac{\exp\left(-\|x_i - x_j\|^2/2\sigma^2\right)}{\sum_{k\neq l}\exp\left(-\|x_k - x_l\|^2/2\sigma^2\right)}, \qquad (7)$$

Figure 3.2.4 illustrates the importance of normalising the probabilities. The algorithm calculates the width of each normal curve (determined by the variance σ) in dependency with points' density around the centre instance. Hence, sparser regions would mistakenly lead to lower similarities.

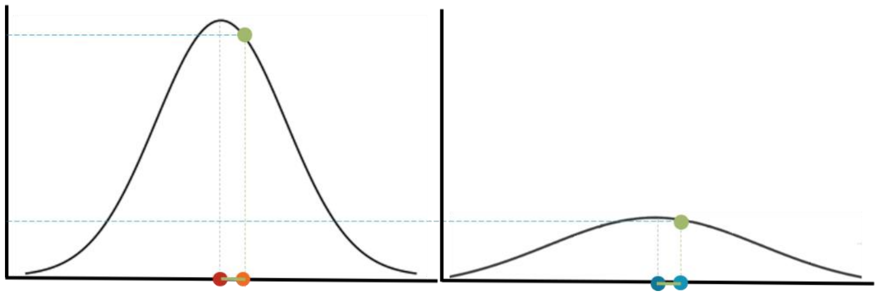

Figure 3.2. 4 – Importance of data normalisation. At the left, $p_{j|i}$ for a given data point. At the right, $p_{j|i}$ for a point at the same Euclidean Distance when the centre instance is in a sparse region.

Let P_{ij} be the set of probabilities previously obtained. The second step of this method is to randomly project the point onto the low dimensional space and repeat the process, replacing the Gaussian by a Student t-distribution (also known as a Cauchy distribution) with one degree of freedom (Figure 3.2.5). Let Q_{ij} be this new set of probabilities.

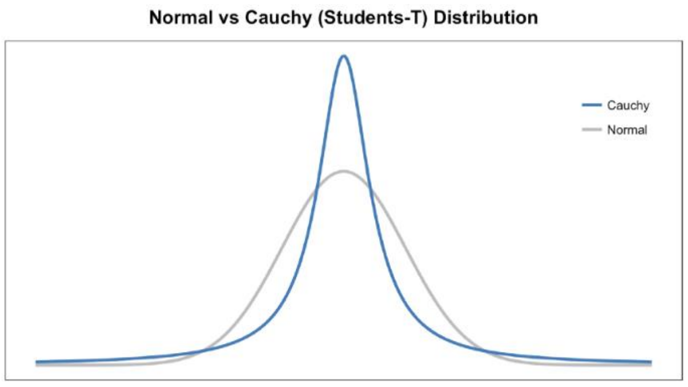

Figure 3.2. 5 - Normal vs Student t-distribution [51].

Finally, the Kullback-Leibler (KL) divergences between P_{ij} and Q_{ij} must be minimised, so that the instances on the lower space approximate the original distances represented by P_{ij}. t-SNE uses Gradient Descent to accomplish this.

Li et al. (2017) [52] studied this technique's suitability for human genetic data and compared it to PCA. They concluded t-SNE can accurately separate samples, is more robust in the presence of outliers.

However, it is worth mentioning that because the calculus of probabilities is computationally heavy, PCA is often preferred when dealing with high numbers of features.

3.3 Autoencoders

Autoencoders are unsupervised neural networks trained to learn efficient representations of the input data. Their architecture, depicted in Figure 3.3.1, can be perceived as two components: an encoder, formally described by function h=f(X) that converts X inputs to h codings; and a decoder, represented by X'=g(h), where X' is the inputs' reconstruction.

The goal of this process is not tied to X', but to the encoded layer h. When h has a smaller dimension than X, it captures the input's most dominant features, and the autoencoder is said to be 'undercomplete' [53], [54].

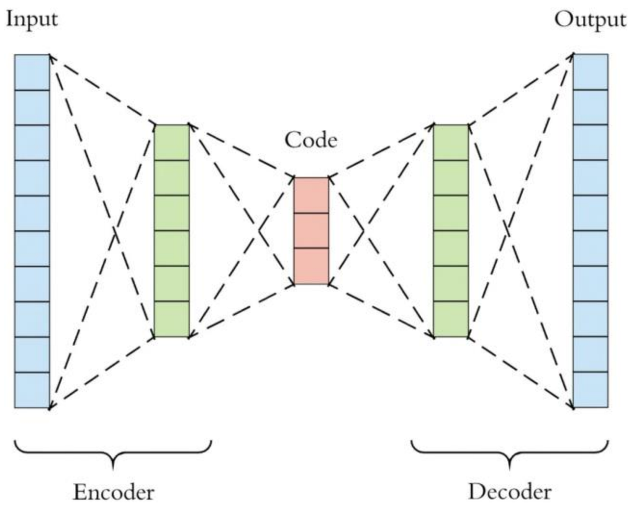

Figure 3.3. 1 - Autoencoder architecture [55]

The training process of an undercomplete autoencoder (AE) consists of minimising a loss function L(x, g(f(x))) that penalises X' (or g(f(x))) for being dissimilar from X.

Note that should the loss function be the mean squared error (MSE) and the activation functions linear, the autoencoder will generate the same subspace as PCA [56]. Thus, the use of non-linear activation functions tends to be more relevant when using AEs as a feature engineering technique.

Lei Le and Andrew White (2018) [57] proposed a model they named *Supervised Autoencoder* (SAE), that produces features that help improving generalisation performance. The examples displayed in Figure 3.2.2 illustrate the structure of two SAEs for linear (Figure 3.3.2 (a)) and deep (Figure 3.3.2 (b)) architectures. This model is suitable when data labels are known. The encoded layer, i.e. the codings, undergo a task (classification or regression). The learning loss is, then, given by the sum of a reconstruction loss (as is standard for all AEs) and a classification loss (for example, cross-entropy).

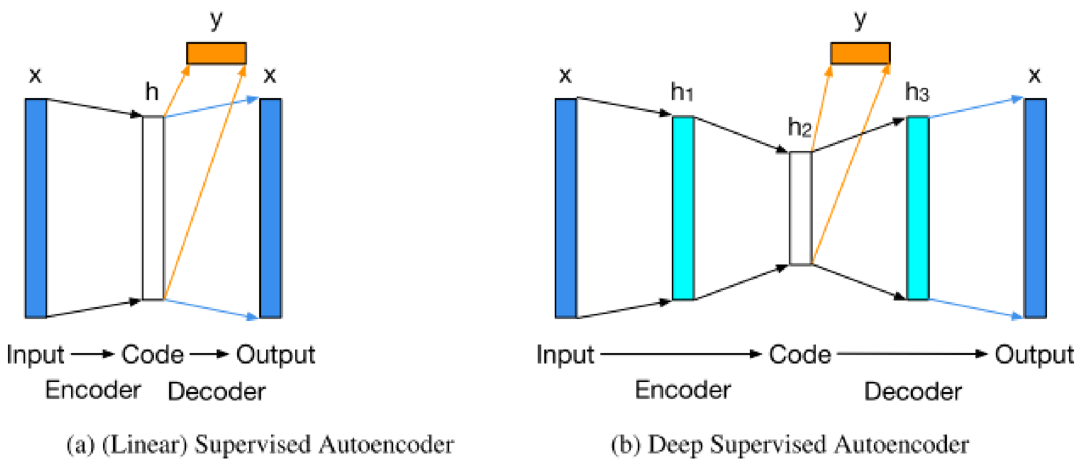

Figure 3.3. 2 - Two examples of Supervised Autoencoders [57].

Autoencoders are also often used for denoising purposes, to do so the input must go through a 'corruption' process. Vincent et al. (2008) [58] suggested setting random input values to 0 (manually, or adding a dropout layer) and training the model to reconstruct the original input. Thus, statistical dependencies between the altered and accurate signals are captured, and the learnt representations are robust to partial input corruption.

3.3.1 Autoencoders Applied to Gene Expression Data

Gupta et al. (2015) [59] showed denoising autoencoders to be suited options to learn compact representations of gene expression data. The model was evaluated by performing cluster on the encoded data.

Xiao et al. (2018) [60] developed a semi-supervised deep learning method for cancer prediction entitled 'stacked sparse auto- encoder'. Several autoencoders learn to reproduce one another's output. Their encoder and decoder portions are then reassembled as seen in Figure 3.3.1.1. Finally, the latent space of the overall model is connected to a classification neural network. The model was validated using three RNA-seq datasets of different types of cancer and comparing the classification performance to that of three standard methods.

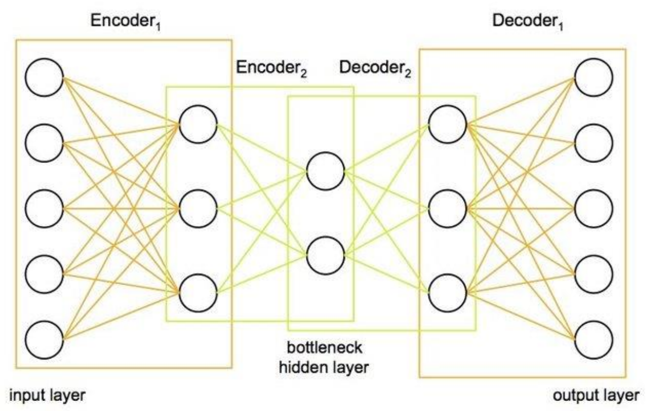

Figure 3.3.1. 1 - Stacked autoencoder architecture [61].

Aga Lewelt (2015) [62] encoded cancer transcriptomes using a variational autoencoder (VAE). The biological relevance of the engineered features was assessed comparing their 2D projections to those of other standard methods and by analysing the heatmap resulting from hierarchical clustering of the codings, showing these types of models are suited for gene expression data.

Variational Autoencoders, shown in Figure 3.3.1.2, are generative neural networks. They learn the distribution model of data points in a high-dimensional space, which allows them to create new instances with similar characteristic to those of the original input [63]. Figure 3.3.1.3 illustrates the difference between VAEs and AEs. VAEs train to encode a distribution over the latent space (an encoded vector of means and an encoded vector of variances), which is sampled and decoded [64]. The reconstruction error has a regularisation term between the returned distribution and a standard Gaussian, the Kulback-Leibler divergence. This avoids overfitting [63].

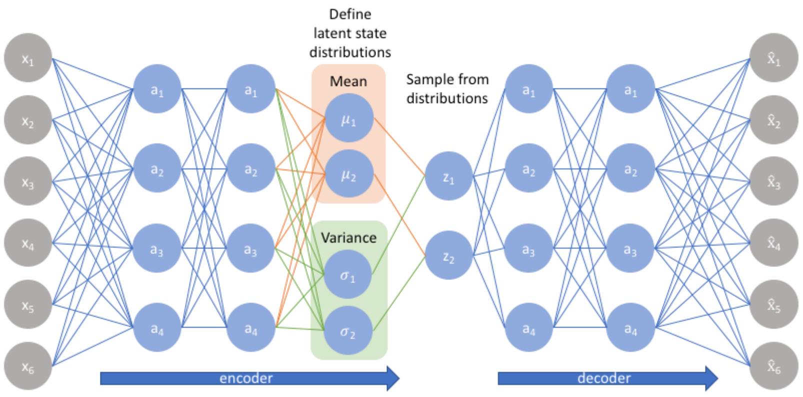
Define latent state distributions
Mean
Variance
Sample from distributions
x_1 x_2 x_3 x_4 x_5 x_6
a_1 a_2 a_3 a_4
μ_1 μ_2
σ_1 σ_2
z_1 z_2
$\hat{x}_1$ $\hat{x}_2$ $\hat{x}_3$ $\hat{x}_4$ $\hat{x}_5$ $\hat{x}_6$
encoder
decoder

Figure 3.3.1. 2 – Variational Autoencoder Architecture [65].

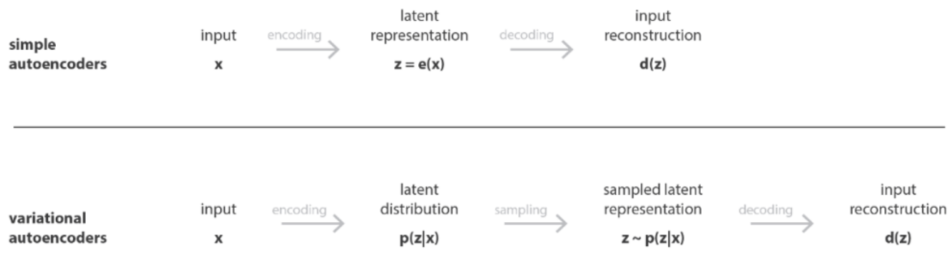

Figure 3.3.1. 3 5 - Difference between simple autoencoders and variational autoencoders [66].

Dongfang Wang and Jin Gu (2018) [67] applied a deep variational autoencoder they designated VASC to Single-cell RNA sequencing data. VASC (Figure 3.3.1.4) uses a Dropout layer and explicitly models its dropout events with a Zero-Inflated layer, designed based on a double-exponential distribution. The model was tested on 20 datasets and its performance measured by comparing the clustering results of the encoded features and four state-of-the-art dimension reduction methods. VASC achieved superior performances in most datasets.

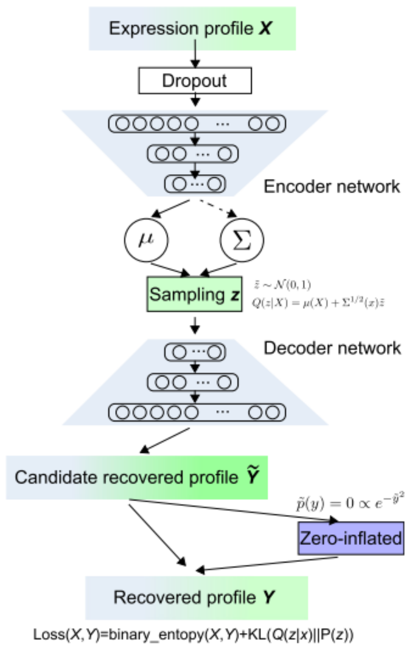
Expression profile **X**
Dropout
Encoder network
μ
Σ
Sampling **z**
$\tilde{z} \sim \mathcal{N}(0,1)$
$Q(z|X) = \mu(X) + \Sigma^{1/2}(x)\tilde{z}$
Decoder network
Candidate recovered profile $\tilde{Y}$
$\tilde{p}(y) = 0 \propto e^{-\tilde{y}^2}$
Zero-inflated
Recovered profile **Y**
Loss(X,Y)=binary_entopy(X,Y)+KL(Q(z|x)||P(z))

Figure 3.3.1. 4 - VASC workflow [67].

Geddes et al. (2020) [68] proposed an autoencoder-based cluster ensemble framework. A simple autoencoder reduces the data dimensions, the output is then classified by several clustering algorithms, and the final label being selected by a consensus equation. The model was validated using four different metrics. It showed better performances than using a single clustering method and than in the absence of the AE as a feature reduction technique.

4. Data Description and Pre-processing

To assess if the proposed architecture is suited for human gene expression data and meets the goals described in Section 1, three different datasets were used. Since the learning process relies on a classifying layer, they were filtered so that all the samples included proper labels and were from one of three classes. The following subsections describe each of the resulting subsets.

4.1 E-MTAB-62

The E-MTAB-62 dataset, provided by Lukk et al. (2010) [69], was retrieved from the ArrayExpress archive in February 2020. The selected samples were labelled as adipose tissue, bronchial epithelium, and bladder cells (displayed on Table 4.1.1).

Table 4.1. 1 – Number of samples per class in the E-MATB-62 dataset.

Class	Adipose Tissue	Bladder Cells	Bronchial Epithelia	Total
Number of Samples	50	48	33	158

To infer the biological significance of the 2D representations, other data annotations (referred to as descriptors) were collected, namely, the development stage; the location of the adipose tissue (from the thigh, abdomen or not specified); the type of bladder cell (from the mucosa or other); and the characteristics of the bronchial epithelium (from a current smoker, a former smoker or a non-smoker). These characteristics are described on Table 4.1.2 through 4.1.5.

Table 4.1. 2 – Number of descriptors in the E-MATB-62 dataset for the disease state.

Descriptor	Possible Values	Adipose Tissue	Bladder Cells	Bronchial Epithelia
Disease State	Normal	14	7	33
	Disease	33	41	-
	Not available	3	-	-

Table 4.1. 3 – Number of descriptors for the adipose tissue.

Descriptor	Possible Values	Number of Samples
Cell Type	Preadipocytes	24
	Not Specified	26
Development Stage	Adult	9
	Not Specified	41
Location	Abdomen	29
	Thigh	4
	Not Specified	17

Table 4.1. 4 – Number of descriptors for the bladder.

Descriptor	Possible Values	Number of Samples
Cell Type	Mucosa	4
	Not Specified	44

Table 4.1. 5 – Number of descriptors for the bronchial epithelium.

Descriptor	Possible Values	Number of Samples
Characteristics	Current Smoker	20
	Non-Smoker	7
	Former Smoker	6

4.2 E-GSE81608

The E-GSE81608 dataset, provided by Xin et al. (2016) [70], was retrieved from the GEO repository in April 2020. The selected samples were from alpha cells, beta cells, and pancreatic polypeptide (PP) cells. The descriptor considered was the presence or absence of diabetes type II (Table 4.2.1).

Table 4.2. 1 – Number of samples per class in the GSE81608 dataset.

Class	Alpha Cells	Beta Cells	PP cells	Total
Number of Samples	765	402	66	1233
Normal Cells	377	207	12	596
Diabetes type II	388	195	54	637

4.3 E-TABM-185

The E-TABM-185 dataset, provided by Lukk et al. (2007) [71], was retrieved from the ArrayExpress archive in May 2020. The selected samples were labelled as Skeletal Muscle cells, lung cells, and cerebellum cells (see Table 4.3.1). The descriptors considered were the development stage and disease state, described in Table 4.3.2, 4.3.3 and 4.3.4.

Table 4.3. 1– Number of samples per class in the E-TABM-185 dataset.

Class	Skeletal Muscle	Lung Cells	Cerebellum Cells	Total
Number of Samples	194	67	65	326

Table 4.3. 2 – Number of descriptors for the lung.

Descriptor	Possible Values	Number of Samples
Disease Type	Adenocarcinoma (four different stages)	49
	Normal	9
	Other	1
	Not Specified	8

Table 4.3. 3 – Number of descriptors for the cerebellum.

Descriptor	Possible Values	Number of Samples
Disease Type	Huntington's disease	38
	Not Specified	27

Table 4.3. 4 – Number of descriptors for the skeletal muscle.

Descriptor	Possible Values	Number of Samples
Disease Type	Dystrophy	63
	Myopathy	5
	Sclerosis	9
	Calpainopathy	10
	Dermatomyositis	26
	Methabolic Problem	39
	Other	54
	Not Specified	16

4.4 Data Scaling

Data scaling is an indispensable part of data pre-processing when training neural networks, as they use gradient descent as an optimisation technique [72]. Large input values and differences in ranges of features (common in gene expression) lead to large weight values, which often result in unstable models with poor performances [73].

The most popular scaling techniques are normalisation (also known as Min-Max scaling) and standardisation. Generally, standardisation is preferred when the data follows a normal distribution [72], [73]. Hence, the three datasets were normalised using the scikit-learn package [74].

Equation 8 shows how to obtain the normalised vector X', with X_{min} being the minimum value of the feature and X_{max} the maximum value. Note that normalisation forces the data to range between zero and one.

$$X' = \frac{X \quad - X_{min}}{X_{max} - X_{min}} \tag{8}$$

5. Model and Experimental Setup

5.1. Proposed Model

The work developed by Lei Le and Andrew White (2018) [57], described in section 3, inspired the development of a model capable of embedding gene expression measurements in a 2D space while taking into account the cells' characteristics.

The proposed architecture, illustrated in Figure 5.1.1, combines a SoftMax classification layer with a simple autoencoder so that the total model loss is given by the sum of the reconstruction and the classification losses (equation 9). The Dropout Layer, used for the learning process, avoids overtraining.

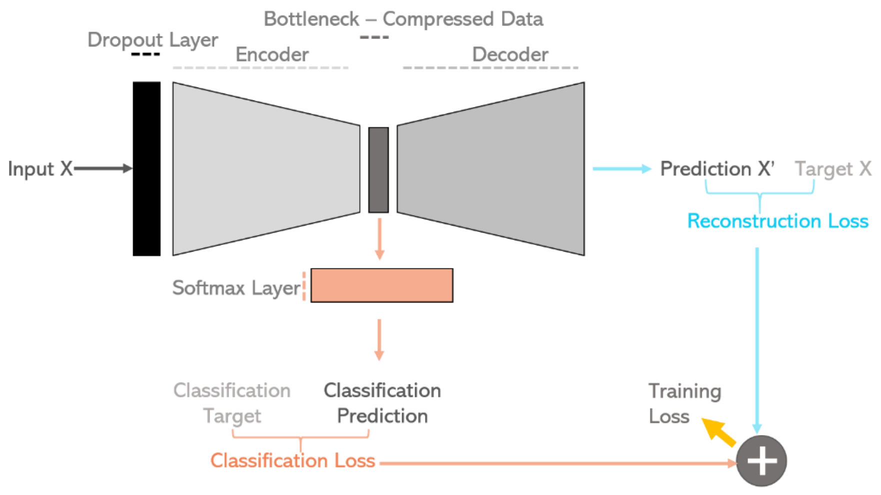

Figure 5.1. 1 – Scheme of the proposed architecture.

$$Training\ loss = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 - \sum_{c=1}^{M} y_{i,c} log(p_{i,c}) \qquad (9)$$

Where M is the number of classes, , n is the number of samples, $y_{i,c}$ the indicator (0 or 1) if c is the true class and $p_{i,c}$ the predicted probability of c, y the predicted labels and $\tilde{y}$ the true class

5.2 Random Search and Hyperparameter Optimisation

Random search (RS) is a simple hyperparameter optimisation (HPO) method often used as an alternative to grid search. It samples configurations at random within a given search budget, selecting the best one [75]. Bergstra et al. (2012) [76] demonstrated this approach is more efficient when training neural networks than grid search, for it achieves the same (or better) results within a small fraction of the computation time. Figure 5.2.1 shows a comparison between the two HPO methods when some hyperparameters are more relevant than others, which is the case of neural networks [75], [76].

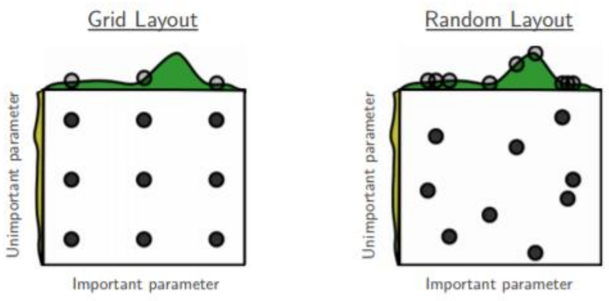

Figure 5.2. 1 - Grid and random search for optimising a function given by the sum of g(x), represented in green above the squares, and h(y) represented in yellow at the left [76].

The hyperparameters registered in Table 5.2.1., obtained through RS, describe the models used for each dataset. Note for each model the number of neurons in the input layer corresponds to the number of genes in the dataset.

Table 5.2. 1 Optimal hyperparameters resultant from grid search optimisation. *Not including the SoftMax Classification Layer or the Dropout Layer

Parameter/dataset	**E-MTAB-62**	**GSE81608**	**E-TABM-185**
Number of Layers*	7	7	7
Number of Neurons	[22283, 300, 20, 2, 20, 196, 22283]	[39851, 100, 20, 2, 20, 100, 39851]	[22283, 196, 20, 2 20, 196, 22283]
Activation Function	[sigmoid, sigmoid, sigmoid, linear, sigmoid, linear]	[linear, sigmoid, linear, sigmoid, linear, linear]	[sigmoid, sigmoid, linear, , linear]
Epochs	20	20	20
Batch Size	2	2	2
Optimiser	adadelta	adadelta	adadelta

5.3 K-Fold Cross-Validation

Cross-validation is a practice widely adopted to evaluate the generalisation ability of predictive models. The data is resampled to form a training and test set. In k-fold cross-validation, the training set is then randomly divided into k subsets of equal size, referred to as 'folds' [77].

As seen in Figure 5.3.1, the learning process is repeated k times, using each fold as a validation set. The final model is evaluated using the test set. This method is advantageous as it means all training data is used for validation and prevents overfitting [74].

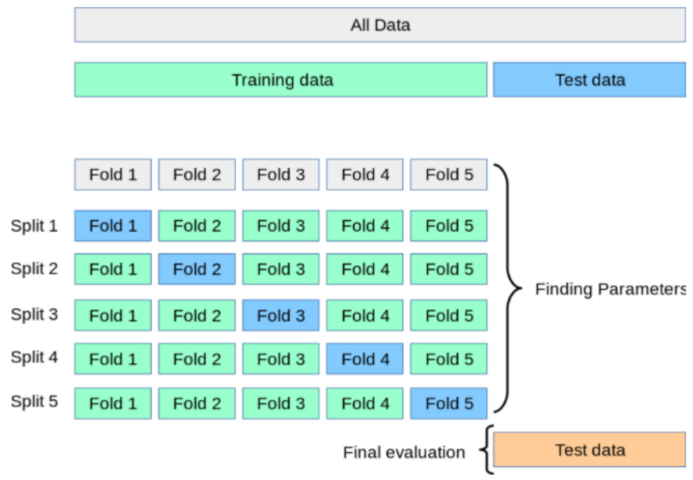

Figure 5.3. 1 - Data division using K-fold Cross Validation [74].

To determine the optimal parameters and hyperparameters, 20% of the data served as a test set. The rest was split into five folds. Tables 5.3.1 through 5.3.3 describe the divided datasets.

Table 5.3. 1 - Train and test sets for E-MATB-62.

	Number of Samples			
	Adipose Tissue	**Bladder Cells**	**Bronchial Epithelia**	**Total**
Train Set	36	43	25	131
Test Set	14	5	8	27
Total	50	48	33	158

Table 5.3. 2 - Train and test sets for GSE81608.

	Number of Samples			
	Alpha Cells	**PP Cells**	**Beta Cells**	**Total**
Train Set	765	66	402	1233
Test Set	181	27	101	309
Total	946	93	503	1542

Table 5.3. 3 - Train and test sets for E-TABM-185.

	Number of Samples			
	Skeletal Muscle	**Lung Cells**	**Cerebellum Cells**	**Total**
Train Set	157	51	52	260
Test Set	37	16	13	66
Total	194	67	65	326

5.4 Model Evaluation

After training, the model can be used to predict the test set's 2D representation. By plotting these results, it should be possible to observe the classes' separability and compare it to that obtained through standard-use methods such as PCA or AE. As mentioned in Section 3, a comparison between the classification accuracy using the encoded features and the standard methods projections is often used as a validation metric. However, because the proposed architecture already contains a supervised classification layer, this metric is not applicable.

Thus, to further investigate the biological meaning of this representation, and whether it allows inferring samples' properties, we examine the relationship between the samples' position in the encoded 2D space and their descriptors.

Finally, the two engineered features were used to perform hierarchical clustering. To easily visualise the outcome, a heatmap with complete genetic information was considered.

The clustering used the Euclidean Distance as metric, the group average as comparison criteria and the data was standardised per gene to avoid the effect of outliers.

6. Results and Discussion

6.1 The learning Process

Figure 6.1.1 shows the models' learning curves using the optimal hyperparameters described in Section 5.2.

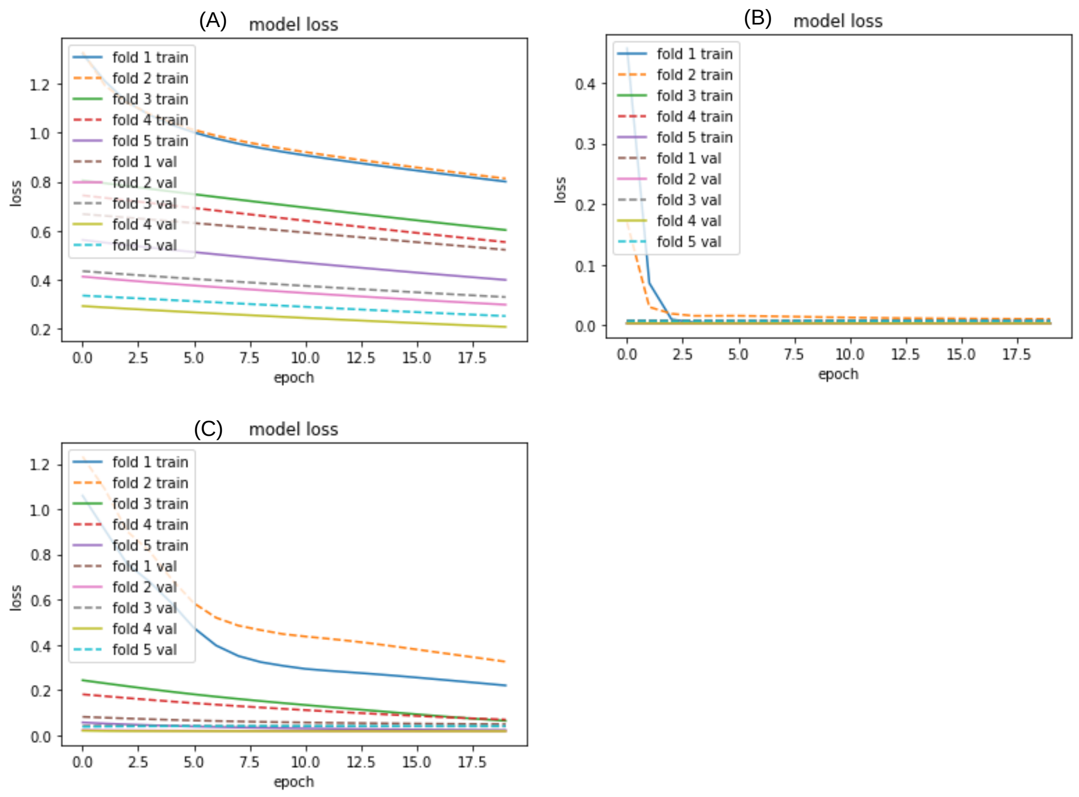

Figure 6.1. 1 - Models' learning curves. Each curve represents the loss for the training (train) or validation (val) set when a given fold is left out. (A) E-MTAB-62 (B) GSE81608, (C) E-TABM-185.

The loss values of the validation sets converge with those of the corresponding train sets, suggesting the models are not overfitted. Furthermore, the losses progressively decreased for both test and training data to values near zero, implying the latent spaces may be meaningful representations of the original inputs. A further analysis based on the test set follows.

6.2 Comparison with PCA and the AE

The 2D spaces generated for the test data using the SAE are presented in Figure 6.2, alongside the PCA and equivalent (i.e. with the same hyperparameters) unsupervised AE results for comparison purposes. Such analysis should determine if the model achieves its first goal – maximising class separability.

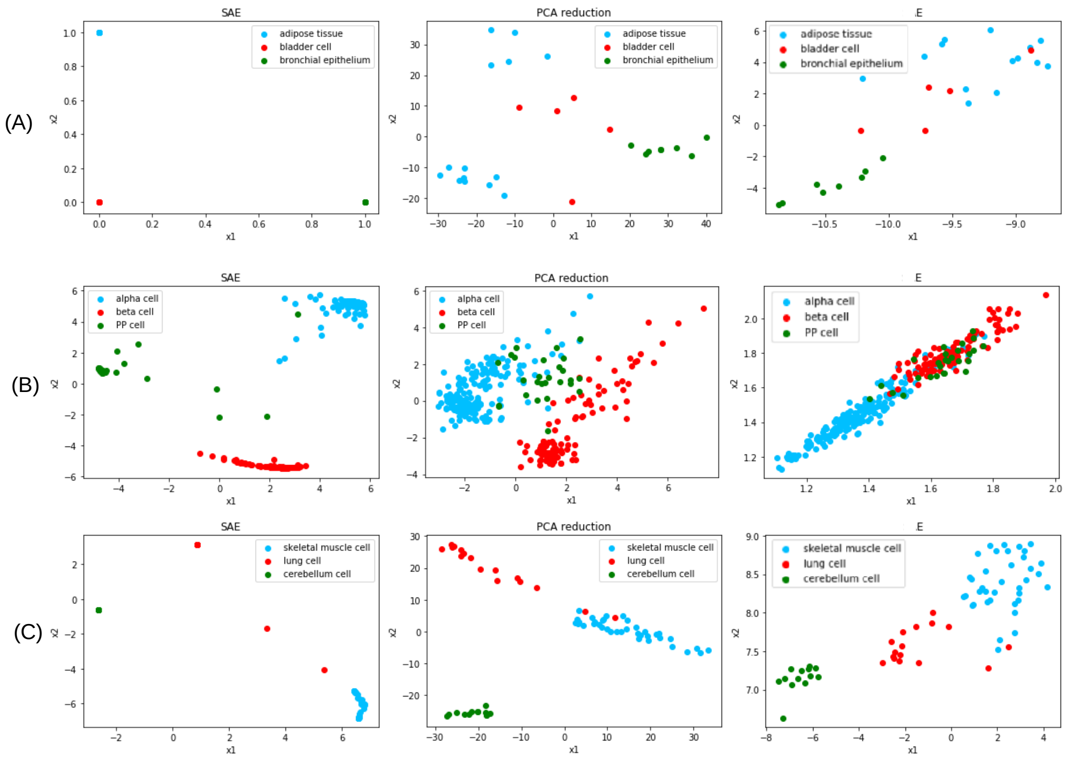

Figure 6.2. 1 – 2D representation of the gene expression data per class. At the right, the latent space generated from the Supervised Autoencoder (SAE) using the test data. At the centre, Principal Component Analysis (PCA) of the same data; At the right, the results

With the features engineered through PCA or the AE, classes are not always linearly separable or the distances between them not maximised. Figure 6.2.1 (A) shows the proposed model perfectly classifies every sample from the E-MATB-62 dataset, with the instances within each class overlapping, whereas both PCA and the AE would lead to several misclassifications.

The discrepancy between class separability across models is most noticeable in Figure 6.2.1 (B). Both state-of-the-art methods failed to distinguish delta cells representations from the remaining classes. Therefore, despite the four outliers produced by the SAE, it is fair to conclude it performs better.

The E-TABM-185 validation set contains two outliers across all reduced spaces, according to Figure 6.2.1 (C). Even so, the SAE is the only method that displays those instances as such (with PCA and the AE these samples are completely blended with skeletal muscle cells). Apart from said detail, all models' features result in reasonable class separability with our model maximising the distances between them.

Thus, in the presence of labels, the proposed model is preferable, as it facilitates the classification problem and eases the visualisation of each sample's properties. However, this analysis does not suffice as it is predictable that a supervised method generates such results when compared to unsupervised alternatives. Since there are no standard supervised methods when dealing with gene expression information, to assess if the SAE can capture relevant biological characteristics, the samples' descriptions were examined.

6.3 Descriptor Analysis and Heatmap Visualisation

Figure 6.3.1 exemplifies the observations made. The known descriptors of the cells do not correlate to the spatial distribution of their representations, *i.e.* they do not confirm the two dimensions are biologically interpretable. Though this is true for the four datasets, there are few descriptors (some are even missing for several samples), and they are too vague to reject the hypothesis that samples with similar representations have similar expression patterns or phenotypes. Therefore, Figures 6.3.2, 6.3.3 and 6.3.4 show the heatmap of the test samples' expression levels, clustered based on the encoded features.

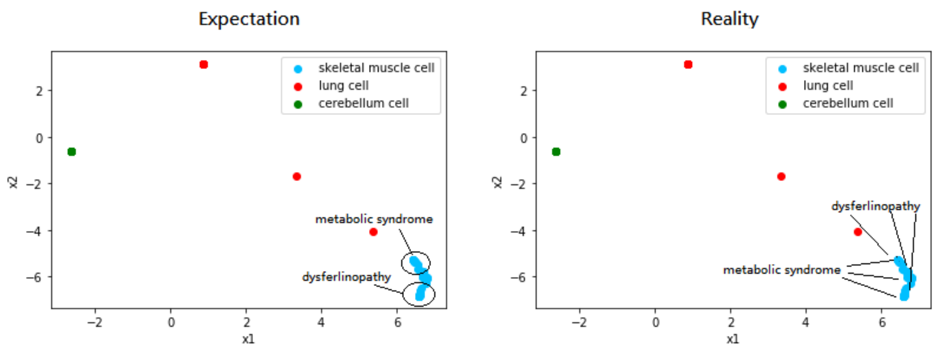

Figure 6.3. 1 Illustration of descriptors analysis. At the right, the hypothesis - the points scattering is related to the cells' phenotypes. At the left, the observations - the descriptors do not correlate to the dispersion of points.

Figure 6.3.2 further confirms the observations previously made. The encoded space allows to perfectly separate each class of cells. However, since all the data points in the encoded space overlap, the representation does not allow any further distinctions between instances of the same class. Unfortunately, this indicates the second goal – inferring new biological patterns through the latent space - may be undertaken by the effect of the *SoftMax* layer.

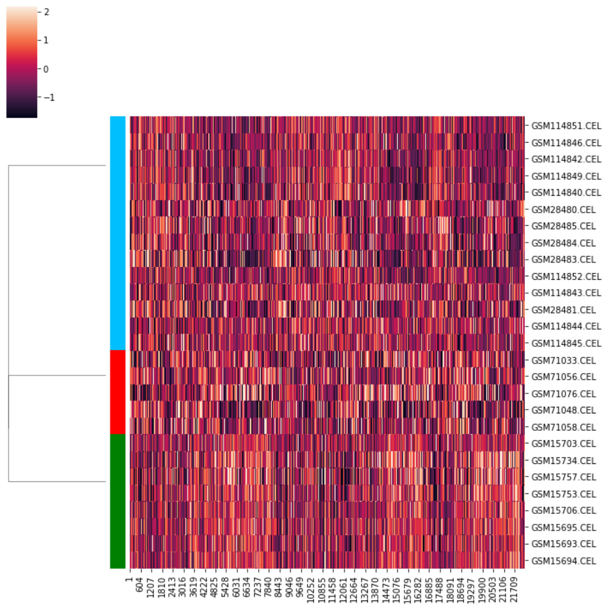

Figure 6.3. 2– Heatmap of the gene expression of the test set from E-MATB-62. The hierarchical clustering is based on the engineered features.

Figure 6.3.3 is significantly harder to analyse, exemplifying the importance of simpler visualisation techniques for large datasets of gene expression data. Due to the amount of information displayed, it is not possible to assess the biological meaning behind the data relative location.

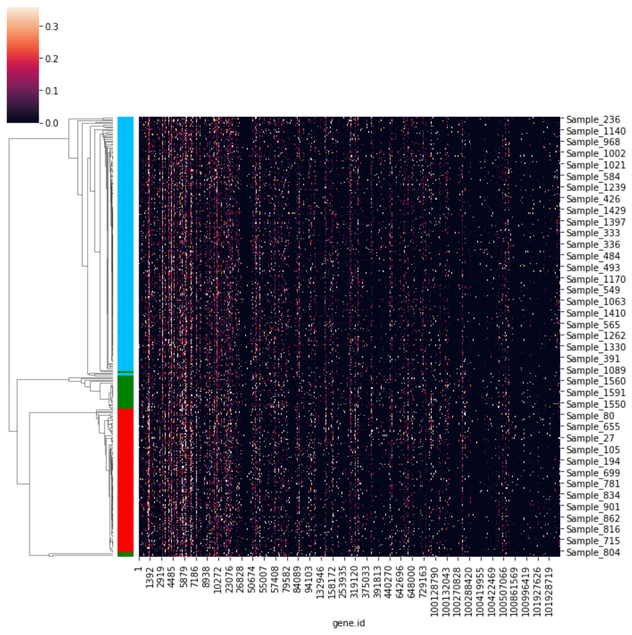

Figure 6.3. 3 – Heatmap of the gene expression of the test set from GSE81608. The hierarchical clustering is based on the engineered features. Note the white rows correspond to genes that share the same expression levels in all samples.

The clustering resulting from the encoded space using the E-TABM-185 dataset (Figure 6.3.4) is reasonably homogeneous within each class, although there are some outliers, and there is a clear separation between classes. Furthermore, the two misclassified samples have a notably different expression pattern than the other samples of their class, resembling the class they were misclassified has. Since the two samples have no further annotations, it is not possible to find a biological explanation for this. However, it is fair to conclude the model’s classification to be reasonable.

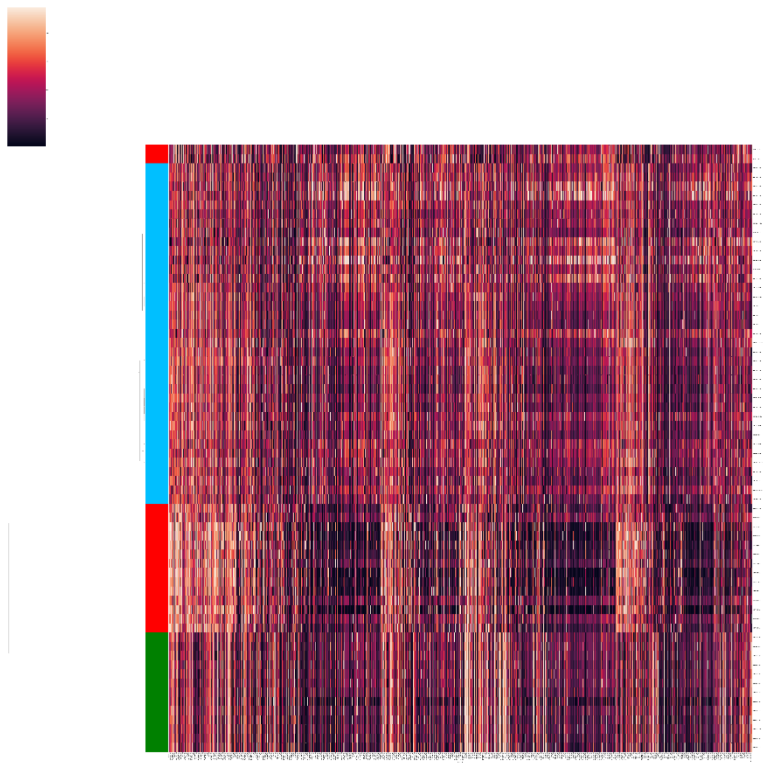

Figure 6.3. 4 – Heatmap of the gene expression of the test set from E-TABM-185. The hierarchical clustering is based on the engineered features.

In conclusion, the model does not provide a representation capable of capturing data patterns beyond the ones characterised by the labels. This is mostly because there seems to be a trade-off between maximising the separability between classes and the separability between diverse instances within classes. Hence the suitability of the proposed feature reduction method depends on the researcher's priority.

7. Conclusion

This dissertation focused on a supervised deep learning model, the SAE, designed to engineer gene expression features. Two goals were set for the model. The representation of data in the 2D latent space should maximise class separability, easing the visualisation of different labelled samples and optimising a posteriori classification. Furthermore, the relative position of instances in the latent space should show meaningful patterns, allowing to infer similarities between cells' genotypes.

The model's performance was validated on 3 datasets. The first objective was evaluated by visualising the encoded features and those of state-of-the-art methods such as PCA and simple AEs. The SAE performed better than the other techniques despite some misclassification errors and the classes' imbalance, proving that weighting the true labels during the pre-processing phase can enhance and facilitate the task of supervised learning algorithms.

To analyse how samples' relative distances related to their genotype and phenotypical dissimilarities, the instances annotations were studied as well as the expression heatmap based on the encodings hierarchical clustering.

The former study was inconclusive mainly due to scarcity of descriptors, incomplete information and vague labels, emphasising the need for better organised datasets in this field and for investment in more effective sequencing technologies.

The latter showed some of the outliers in the 2D representations better resembled the classes they were misclassified as. These results indicate the latent space does have some biological interpretability that may be used to infer patterns between data. However, it doesn't fully achieve this purpose, which can be explained by the plot of encodings. Notably, the classification loss often endorsed the overlap of same labelled samples. Thus, the two goals seem conflictive and a compromise between the two must be made when using this method.

7.1 Future Work

To better understand this architecture's robustness, compatibility with different datasets and protocols, and generalisation capability, further work exploring a semi-supervised learning approach should be considered. Moreover, as accessing datasets with detailed and complete annotations proved to be a challenging problem and learning algorithms' performance tends to improve with more instances, a semi-supervised approach would facilitate training and lead to broader practical application in gene expression research.

References

[1] F. Crick, “Central Dogma of Molucular Biology,” *Nature*, vol. 227, no. 5258, pp. 561–563, 1970, [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1078143908003268%5Cnhttp://www.hmg.oxfordjournals.org/cgi/doi/10.1093/hmg/ddl046%5Cnhttp://www.tandfonline.com/doi/abs/10.4161/rna.7.5.13216%5Cnhttp://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=304.

[2] B. Jordan, “Historical background and anticipated developments,” *Ann. N. Y. Acad. Sci.*, vol. 975, no. 0, pp. 24–32, 2002, doi: 10.1111/j.1749-6632.2002.tb05938.x.

[3] M. P. Richards, “Techniques for Gene Expression Profiling,” *Med. Biomethods Handb.*, pp. 507–518, 2005, doi: 10.1385/1-59259-870-6:507.

[4] F. Tang *et al.*, “mRNA-Seq whole-transcriptome analysis of a single cell,” *Nat. Methods*, vol. 6, no. 5, pp. 377–382, 2009, doi: 10.1038/nmeth.1315.

[5] S. Nomura, “Single-cell genomics to understand disease pathogenesis,” *J. Hum. Genet.*, 2020, doi: 10.1038/s10038-020-00844-3.

[6] B. Hwang, J. H. Lee, and D. Bang, “Single-cell RNA sequencing technologies and bioinformatics pipelines,” *Exp. Mol. Med.*, vol. 50, no. 8, 2018, doi: 10.1038/s12276-018-0071-8.

[7] A. Haque, J. Engel, S. A. Teichmann, and T. Lönnberg, “A practical guide to single-cell RNA-sequencing for biomedical research and clinical applications,” *Genome Med.*, vol. 9, no. 1, pp. 1–12, 2017, doi: 10.1186/s13073-017-0467-4.

[8] J. P. M. De Sa, *Pattern Recognition: Concepts, Methods and Applications*. .

[9] D. Hutchison and J. C. Mitchell, *Subspace, Latent Structure and Feature Selection - Statistical and Optimization Perspectives Workshop, SLSFS 2005, Revised Selected Papers*, vol. 3940 LNCS. 2006.

[10] A. Ben-Dor, L. Bruhn, N. Friedman, I. Nachman, M. Schummer, and Z. Yakhini, “Tissue Classification with Gene Expression Profiles,” in *Proceedings of the Fourth*

Annual International Conference on Computational Molecular Biology, 2000, pp. 54–64, doi: 10.1145/332306.332328.

[11] S. Tarek, R. Abd Elwahab, and M. Shoman, “Gene expression based cancer classification,” *Egypt. Informatics J.*, vol. 18, no. 3, pp. 151–159, 2017, doi: https://doi.org/10.1016/j.eij.2016.12.001.

[12] L. Zakaria, H. M. Ebeid, S. Dahshan, and M. F. Tolba, “Analysis of Classification Methods for Gene Expression Data BT - The International Conference on Advanced Machine Learning Technologies and Applications (AMLTA2019),” 2020, pp. 190–199.

[13] L. Chen, “Curse of Dimensionality BT - Encyclopedia of Database Systems,” L. LIU and M. T. ÖZSU, Eds. Boston, MA: Springer US, 2009, pp. 545–546.

[14] Z. M. Hira and D. F. Gillies, “A Review of Feature Selection and Feature Extraction Methods Applied on Microarray Data,” *Adv. Bioinformatics*, vol. 2015, p. 198363, 2015, doi: 10.1155/2015/198363.

[15] S. Liu *et al.*, “Feature selection of gene expression data for Cancer classification using double RBF-kernels,” *BMC Bioinformatics*, vol. 19, no. 1, p. 396, 2018, doi: 10.1186/s12859-018-2400-2.

[16] S. Vanjimalar, D. Ramyachitra, and P. Manikandan, “A Review on Feature Selection Techniques for Gene Expression Data,” in *2018 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, 2018, pp. 1–4, doi: 10.1109/ICCIC.2018.8782294.

[17] H. Omara, M. LAZAAR, and Y. Tabii, “Effect of feature selection on gene expression datasets classification accuracy,” *Int. J. Electr. Comput. Eng.*, vol. 8, pp. 3194–3203, Oct. 2018, doi: 10.11591/ijece.v8i5.pp.3194-3203.

[18] S. Theodoridis and K. Koutroumbas, “Chapter 5 - Feature Selection,” *Pattern Recognit. (Fourth Ed.*, pp. 261–322, 2009, doi: 10.1016/B978-1-59749-272-0.50007-4.

[19] “Richard O. Duda, Peter E. Hart, David G. Stork - Pattern classification (2001, Wiley)

- libgen.lc.pdf.” .

[20] *Advanced Analysis of Gene Expression Microarray Data Aidongzhang*. .

[21] E. Żymańczyk-Duda, M. Klimek-Ochab, M. Brzezińska-Rodak, and M. Duda, “Biochemistry - knowledge to practical applications - examples.,” 2016.

[22] D. W. Ball, J. W. Hill, and R. J. Scott, “The Basics of General, Organic, and Biological Chemistry,” 1st ed., Saylor Foundation, 2011.

[23] P. Cullen and M. Werner, *Gene Expression: Basic Concepts*. 2008.

[24] D. J. Lockhart and E. A. Winzeler, “Genomics, gene expression and DNA arrays [In Process Citation],” *Nature*, vol. 405, no. 0028-0836 SB-M SB-X, pp. 827–836, 2000.

[25] P. D. Maningat *et al.*, “Gene expression in the human mammary epithelium during lactation: the milk fat globule transcriptome.,” *Physiol. Genomics*, vol. 37, no. 1, pp. 12–22, Mar. 2009, doi: 10.1152/physiolgenomics.90341.2008.

[26] S. Jain, J. Arrais, N. J. Venkatachari, V. Ayyavoo, and Z. Bar-Joseph, “Reconstructing the temporal progression of HIV-1 immune response pathways,” *Bioinformatics*, vol. 32, no. 12, pp. i253–i261, Jun. 2016, doi: 10.1093/bioinformatics/btw254.

[27] C. Nóbrega and S. Alves, “Editorial: Gene silencing and editing strategies for neurodegenerative diseases,” *Front. Neurosci.*, vol. 12, no. JUN, pp. 1–2, 2018, doi: 10.3389/fnins.2018.00425.

[28] J. Eberwine, J. Y. Sul, T. Bartfai, and J. Kim, “The promise of single-cell sequencing,” *Nat. Methods*, vol. 11, no. 1, pp. 25–27, 2014, doi: 10.1038/nmeth.2769.

[29] C. Gawad, W. Koh, and S. R. Quake, “Single-cell genome sequencing: Current state of the science,” *Nat. Rev. Genet.*, vol. 17, no. 3, pp. 175–188, 2016, doi: 10.1038/nrg.2015.16.

[30] I. N. da Silva, D. H. Spatti, and R. A. F. L. H. B. L. S. F. dos R. Alves, *Artificial Neural Networks A Practical Course*. 2017.

[31] S. Sharma, “Activation Functions in Neural Networks,” 2019. https://towardsdatascience.com/activation-functions-neural-networks-

1cbd9f8d91d6.

[32] F. Chollet and others, “Keras.” 2015.

[33] D. Radečić, “Softmax Activation Function Explained,” 2020. https://towardsdatascience.com/softmax-activation-function-explained-a7e1bc3ad60 (accessed Sep. 20, 2020).

[34] F. Bre, J. Gimenez, and V. Fachinotti, “Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks,” *Energy Build.*, vol. 158, Nov. 2017, doi: 10.1016/j.enbuild.2017.11.045.

[35] D. Graupe, *Principles of Artificial Neural Networks*. World Scientific, 2007.

[36] L. De Marchi and L. Mitchell, *Hands-on neural networks : learn how to build and train your first neural network model using Python*. 2019.

[37] J. Brownlee, “Machine Learning Algorithms,” 2016, pp. 16–19.

[38] M. A. Razzaque and M. R. Karim, *Hands-On Deep Learning for IoT: Train neural network models to develop intelligent IoT applications*. Packt Publishing, 2019.

[39] J. Torres, “Learning process of a neural network,” 2018. https://towardsdatascience.com/how-do-artificial-neural-networks-learn-773e46399fc7 (accessed Sep. 25, 2020).

[40] A. Krause and M. OConnell, *A Picture is Worth a Thousand Tables: Graphics in Life Sciences*. Springer US, 2012.

[41] D. (1998). Eisen, M. B., Spellman, P. T., Brown, P. O., & Botstein, “Cluster analysis and display of genome-wide expression patterns.,” *Proc. Natl. Acad. Sci. United States Am. 95(25)*, pp. 14863-14868., doi: https://doi.org/10.1073/pnas.95.25.14863.

[42] A. Kassambara, “Practical guide to cluster analysis in R: unsupervised machine learning,” *J. Comput. Graph. Stat.*, p. 187, 2017.

[43] C. H. Map, “Clustered Heat Maps (Double Dendrograms),” pp. 1–16.

[44] H. Hotelling, “Analysis of a complex of statistical variables into principal components,” *J. Educ. Psychol.*, vol. 24, no. 6, pp. 417–441, 1933, doi: 10.1037/h0071325.

[45] C. Syms, “Principal components analysis,” *Encycl. Ecol.*, pp. 566–573, 2018, doi: 10.1016/B978-0-12-409548-9.11152-2.

[46] I. T. Jolliffe, J. Cadima, and J. Cadima, “Principal component analysis : a review and recent developments Subject Areas,” *Phil.Trans.R.Soc.A*, vol. 374, no. 2065, pp. 1–16, 2016.

[47] J. Shlens, “A Tutorial on Principal Component Analysis,” 2014, [Online]. Available: http://arxiv.org/abs/1404.1100.

[48] C. Bishop, “Pattern Recognition and Machine Learning,” in *Journal of Electronic Imaging*, vol. 16, 2006, pp. 559–595.

[49] M. Lenz, F. J. Muller, M. Zenke, and A. Schuppert, “Principal components analysis and the reported low intrinsic dimensionality of gene expression microarray data,” *Sci. Rep.*, vol. 6, no. November 2015, pp. 1–11, 2016, doi: 10.1038/srep25696.

[50] L. van der Maaten and G. Hinton, “Visualizing Data using t-SNE,” *J. Mach. Learn. Res.*, vol. 9, 2008, doi: 10.1007/s10479-011-0841-3.

[51] A. Violante, “An Introduction to t-SNE with Python Example,” 2018. https://towardsdatascience.com/an-introduction-to-t-sne-with-python-example-5a3a293108d1 (accessed Oct. 29, 2020).

[52] W. Li, J. E. Cerise, Y. Yang, and H. Han, “Application of t-SNE to human genetic data,” *J. Bioinform. Comput. Biol.*, vol. 15, no. 04, p. 1750017, Jun. 2017, doi: 10.1142/S0219720017500172.

[53] I. Goodfellow, Y. Bengio, and C. Aaron, *Deep Learning*. MIT Press, 2016.

[54] B. Boehmke and B. Greenwell, *Hands-On Machine Learning with R*. 2019.

[55] M. Alkhayrat, M. Aljnidi, and K. Aljoumaa, “A comparative dimensionality reduction study in telecom customer segmentation using deep learning and PCA,” *J. Big Data*, vol. 7, no. 1, p. 9, 2020, doi: 10.1186/s40537-020-0286-0.

[56] E. Plaut, “From Principal Subspaces to Principal Components with Linear Autoencoders,” pp. 1–6, 2018, [Online]. Available: http://arxiv.org/abs/1804.10253.

[57] L. Le, A. Patterson, and M. White, “Supervised autoencoders: Improving

generalization performance with unsupervised regularizers," *Adv. Neural Inf. Process. Syst.*, vol. 2018-Decem, no. NeurIPS, pp. 107–117, 2018.

[58] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, *Extracting and composing robust features with denoising autoencoders*. 2008.

[59] A. Gupta, H. Wang, and M. Ganapathiraju, "Learning structure in gene expression data using deep architectures, with an application to gene clustering," in *2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2015, pp. 1328–1335, doi: 10.1109/BIBM.2015.7359871.

[60] Y. Xiao, J. Wu, Z. Lin, and X. Zhao, "A semi-supervised deep learning method based on stacked sparse auto-encoder for cancer prediction using RNA-seq data," *Comput. Methods Programs Biomed.*, vol. 166, pp. 99–105, 2018, doi: 10.1016/j.cmpb.2018.10.004.

[61] "Deep Learning Techniques for Music Generation - A Survey - Scientific Figure on ResearchGate." .

[62] Aga Lewelt, "Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders," *Physiol. Behav.*, vol. 176, no. 3, pp. 139–148, 2015, doi: 10.1016/j.physbeh.2017.03.040.

[63] C. Mellon and U. C. Berkeley, "Tutorial on Variational Autoencoders," pp. 1–23, 2016.

[64] I. Goodfellow, Y. Bengio, and C. Aaron, "Deep Learning," MIT Press, 2016.

[65] J. Jordon, "Variational autoencoders," *Jeremy Jordan*, 2018. https://www.jeremyjordan.me/variational-autoencoders/ (accessed Oct. 13, 2020).

[66] J. Rocca, "Understanding Variational Autoencoders (VAEs)," *towards data science*, 2019. https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73 (accessed Oct. 13, 2020).

[67] D. Wang and J. Gu, "VASC: Dimension Reduction and Visualization of Single-cell RNA-seq Data by Deep Variational Autoencoder," *Genomics, Proteomics Bioinforma.*, vol. 16, no. 5, pp. 320–331, 2018, doi: 10.1016/j.gpb.2018.08.003.

[68] T. A. Geddes *et al.*, “Autoencoder-based cluster ensembles for single-cell RNA-seq data analysis,” *BMC Bioinformatics*, vol. 20, no. 19, p. 660, 2019, doi: 10.1186/s12859-019-3179-5.

[69] M. Lukk *et al.*, “A global map of human gene expression,” *Nat. Biotechnol.*, vol. 28, no. 4, pp. 322–324, 2010, [Online]. Available: https://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-62/?s_sortby=col_31&s_sortorder=ascending.

[70] Y. Xin *et al.*, “RNA Sequencing of Single Human Islet Cells Reveals Type 2 Diabetes Genes.,” *Cell Metab.*, vol. 24, no. 4, pp. 608–615, Oct. 2016, doi: 10.1016/j.cmet.2016.08.018.

[71] M. Lukk *et al.*, “A global map of human gene expression,” *Nat. Biotechnol.*, vol. 28, no. 4, p. 322—324, 2010, doi: 10.1038/nbt0410-322.

[72] A. Bhandari, “Feature Scaling for Machine Learning: Understanding the Difference Between Normalization vs. Standardization,” 2020. https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/ (accessed Sep. 12, 2020).

[73] J. Brownlee, *Data Preparation for Machine Learning: Data Cleaning, Feature Selection, and Data Transforms in Python*. Machine Learning Mastery, 2020.

[74] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in {P}ython,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[75] M. Feurer and F. Hutter, “Hyperparameter Optimization BT - Automated Machine Learning: Methods, Systems, Challenges,” F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Cham: Springer International Publishing, 2019, pp. 3–33.

[76] B. James and B. Yoshua, “Random Search for Hyper-Parameter Optimization,” *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 281–305, 2012.

[77] D. Berrar, “Cross-Validation,” 2018.