Literature review: *Deep learning architectures applied to single cell RNA-seq data*

By Martin Banchero

Course code: XM_0007

Supervisors: Dr. Perry Moerland
Examiner: Dr. Huub Hoefsloot
April, 2021

Abstract

In less than a decade, single-cell RNA sequencing experiments gained broad popularity. In each of these experiments RNA mole cules expressed in thousands of cells can be sequenced generating large volumes of data, and with these, new challenges to develop methods to analyze these data[1]. Lately, several deep learning methods were implemented in this field to tackle different issues associated with this type of data[2]. The main goal of this review is to describe the most popular deep learning architectures that have been applied recently to analyze scRNA-seq data and discuss some of their applications.

1 Introduction

Nowadays single-cell sequencing technology gained broad popularity. This technology provides a high-resolution picture of biological processes allowing the identification of cell heterogeneity from tissue samples[3]. Within this field, different sub-fields arose like single-cell genomics, single-cell methylation sequencing, and single-cell transcriptomics also referred to as single-cell RNA sequencing (scRNA-seq)[4]. This type of experiments can generate large volumes of data, generating at the same time new challenges for data analysis. In general, data analysis pipelines for scRNA-seq data can be classified in two steps, pre-processing stage and downstream analysis[5]. The pre-processing stage consists of processing raw data, this can include normalization, correction of data that may include technical artifacts such as batch effects and high frequency of zeros present in the data (dropouts). Downstream analysis of scRNA-seq can be divided into cell-level and gene-level analysis[5]. The cell-level approach includes clustering[6], analysis of cluster composition[7] and annotation[6,7]. Also, cell-level analysis can be used to obtained information of dynamics processes of cell differentiation by using for instance methods to infer cell trajectories[8]. Gene-level analysis , depending on the problem to solve and considering large number of expressed genes per cell it is possible to perform feature selection and dimensionality reduction, study differential expression[9], gene set analysis[10], and gene regulatory networks[11].

As mentioned before scRNA-seq experiments generate large amounts of data making challenging the task of data analysis. Deep learning, a machine learning subdiscipline can incorporate high-level representation of the data and incorporate it to the model, i.e cell morphology, distance between cells are not captured by conventional machine learning algorithms[12]. Therefore, deep learning appears to be a tool capable to provide the field of single cell with robust methods to analyze high-dimensional scRNA-seq data[2]. Deep learning methods have been applied in different stages of the scRNA-seq analysis pipeline. Several of these methods have been use in data preprocessing stages, e.g. batch correction[13], correction for dropout events[14] but also in downstream analysis, for example, dimensionality reduction[15], to impute the levels of gene expression and clustering[16].

Many deep learning models used in scRNA-seq data analysis are based on the autoencoder architectures[17]. Within this type of architecture, there are different variations, depending on the task to tackle. For example, denoising autoencoders (DAE) for dimensionality reduction[18], and overcomplete autoencoders for sparse matrix for data imputation[14], among others.

Another deep learning approach used to analyze scRNA-seq data consists of generative models (GMs). The main idea of GMs is to learn a true distribution or model a distribution to be as close as possible to the true distribution of the training data, this allows to generates new data points that resemble the original data. Within GM, the most used architectures are variational autoencoders (VAEs)[19] and generative adversarial networks (GANs)[20]. These types of architectures started to gain popularity in the scRNA-seq field and used as a new approach to tackle imputation of gene expression levels[16], dimensionality reduction and visualization[9], or prediction of response dose-infection[21].

In this review, the goal is to describe different types of deep learning models use in scRNA-seq. The review is divided based on the type of deep learning architectures, while focusing mainly on the most predominant architectures, that is, autoencoders and generative models (VAEs and GANs). In addition to this, I add a brief description of applications of the different architectures in the context of scRNA-seq and results from benchmark studies comparing the methods described in this review with traditional machine learning methods.

2 Methods

2.1 Autoencoders

Autoencoders[22] are neural networks consisting of an encoder and a decoder. The encoder maps the input data x to a latent representation h, the decoder maps h to an output reconstruction r where the output target is a copy of its input (Figure 1). Each layer of the latent representation h is defined as h = f (Wx + b), where the function f() is the activation function and can be linear or non-linear, with the weight matrix (W) and bias (b) as parameters and x as input data. The output reconstruction r is defined as r = g (Wh + c), where g is the activation function that takes as input the latent representation h and parameters W representing the weight matrix and the bias c. The encoder and decoder can contain multiple layers that are connected by a bottleneck layer or latent representation space h . The number of nodes in this bottleneck layer is in most cases smaller than the size of the input and output layers. This type of autoencoder is known as a undercomplete autoencoder. In this type of autoencoders, the bottleneck layer enables encoding the input vector by compressing it into a lower dimension. To perform the compression, the encoder selects the most important features of the input data. On the other hand, the decoder decompresses the output of the encoder and returns an approximation of the original domain representation.

In autoencoders the learning process is given by minimizing the equation 1 in which the loss function quantifies the dissimilarity between each input instance x_i from a dataset D and the output instance r_i.

$$\text{loss} = \sum_i ||x_i - r_i||, D\{x_i\} \tag{1}$$

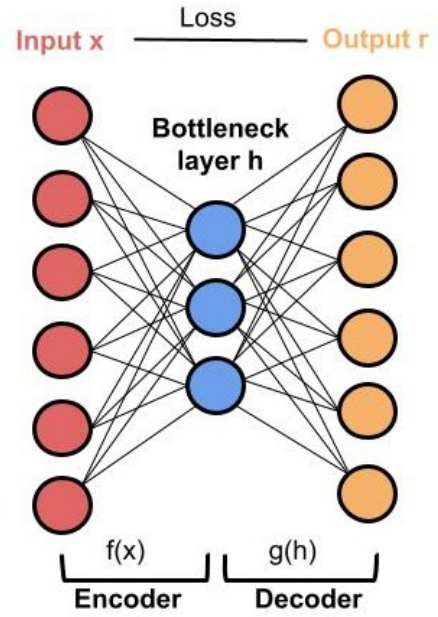

Figure 1: Autoencoder schematic representation. The input (x) is given to the encoder function f(x) to generate the compressed data h. This is the input for the decoder function g(h) that produces the reconstruction output r. Adapted from "Single-cell RNA-seq denoising using a deep count autoencoder" by G. Eraslan et al 2019, Nature Communications, 10, p.1-14.[12]

Autoencoders need to be constrained to avoid exactly copying the input into the output. In this way the autoencoder is forced to output an approximation of the input while learning useful characteristics of the data.
Due to the modular characteristics of autoencoders it is possible to generate different architectures that better suit different application domains. Next, these architectures are described with their application to scRNA-seq data analysis.

Applications of autoencoders to scRNA-seq data
A well-known issue related to scRNA-seq data is the presence of many zeroes in the gene expression values, which leads to a sparse gene expression matrix. This makes it difficult to distinguish zero gene expression values representing the lack of expression of a gene in a cell from zero values caused by low number of samples.

The autoencoder with "basic" architecture explained before can be used to impute sparse scRNA-seq data. One method able to impute highly sparse scRNA-seq data is LATE (Learning with AuToEncoder)[23]. This method trains an autoencoder which is initialized with random values for the parameters. An extension of LATE, called TRANSLATE (TRANSfer learning with LATE) s built on LATE. Here, the first step consists in train the autoencoder using a reference gene expression dataset.. These datasets can be for example RNA-seq data, or different available scRNA-seq datasets with similar cell types to the dataset to be impute. The second step trains the an autoencoder by using the transferred weights and biases obtained in the first step and using as input the scRNA-seq data to impute.

Technical variations in scRNA-seq experiments produce data containing noise due to sequence amplification and dropouts. Some approaches using autoencoders can tackle technical variations by denoising and imputing sparse count data in one step. One example is the deep count autoencoder (DCA) network[12]. This method takes into account that scRNA-seq data consists of over-dispersed counts with many missing values replacing the loss function (eq.1) by the likelihood of the distribution of the noise model, for example, a zero inflated negative binomial (ZINB) distribution. DCA enables removal of technical variation, which leads to an improvement of downstream analyses like clustering and differential expression analysis.

GOAE (Gene Ontology AutoEncoder) can incorporate prior biological information in order to gain understanding of the underlying biological mechanism. GOAE incorporates gene ontology (GO) terms into an autoencoder to improve dimensionality reduction followed by clustering of scRNA-seq data[24]. The input layer contains the genes in the scRNA-seq data set, while the first hidden layer of the autoencoder contains the nodes with GO information. In this first hidden layer, the GO term nodes are partially connected to its input, this mean that the genes in the input layer are connected with each GO term, either a biological process (BP) or molecular function (MF). Because of this, only those genes present in the data set with GO term annotation are passed to the GO neurons. The remaining hidden layers does not contain GO terms. The autoencoder then is used for dimensionality reduction and later for clustering. The model is trained using the mean squared error loss function.

The methods mentioned above are part of the pre-processing stage of scRNA-seq data analysis. Other methods using autoencoders have been developed to enhance biological interpretability of the data by deconvoluting the latent

space of the model[25]. This method evaluates the impact of features (genes) in the latent representation obtained by the encoder. To gain biological interpretation, different set of genes are evaluated by using saliency maps[26], a computer vision method that allows the visualization of the impact of the features (genes) in the latent representation over a heat map. The analysis of this heatmaps shows the differential expression of gene sets present in the latent representation of the autoencoder allowing biological interpretation. In addition to this, the model architecture contains a soft orthogonality constraint applied over the latent representation to represent in one hidden unit one dimension of the latent representation. Also, a Poisson loss function is used to verify the complexity capture by the autoencoder model.

Denoising autoencoders

The main idea of denoising autoencoders (DAEs) is to learn how to denoise input data corrupted by the addition of noise. This noise can be added by randomly setting to zero some input values (dropout)[27] or by addition of Gaussian noise[28]. The goal is to return a target output representing the original uncorrupted data (Figure 2) to control sequence. This is important to generate a robust method to find important features that allow the full reconstruction of denoised inputs.

Applications of DAEs to scRNA-seq data

A method call scSDAEs32 use an DAE architecture with several hidden layers. This architecture is call stacked denoising autoencoder[29] (SDAE) and is a more deep neural network compared with the autoencoders architectures mentioned before in this text, in which only a few hidden layers are used. In scSDAEs the SDAE architecture is used as a framework to impute single-cell data in combination with a loss function with two terms. One term is the mean squared error, and the second term is an L1 regularization term that penalizes the imputed non-zero values. This, in order to preserve true zeros that are caused by genes not transcribed in the cell and not by technical dropouts.

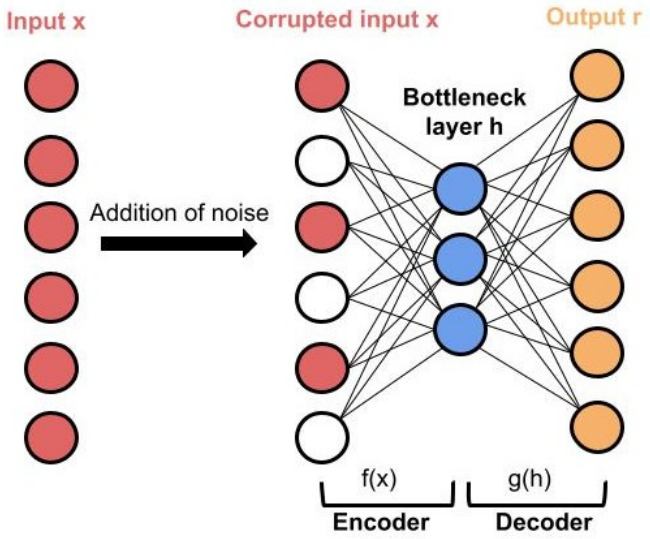

Figure 2: Denoising autoencoder. The data input x is corrupted by the addition of noise(Gaussian noise or dropouts). The corrupted input x in white shows the nodes that are replace by noise, nodes in red show original data values. The corrupted input x is the input layer for the encoder. The denoised output r is generated by the decoder and present the same dimension as the input data x.

DAEs can be use to initialize parameters that can be transferred to improve parameter initialization and generalization of a supervised model to perform dimensionality reduction[18]. In this DAE model, the noise is added from a standard normal distribution with zero mean and a standard deviation of 0.1. For all the layers of the DAE is used an activation function tanh and as loss function the mean squared error. It is important to note that the number of nodes in the DAE is the same as in the supervised neural network. This guarantees that the weights of the DAE can be used in the supervised model.

Overcomplete autoencoders and applications for scRNA-seq data
This type of autoencoder displays the same architecture as undercomplete autoencoders (Figure 1) presenting an encoder and a decoder but the bottleneck layer (h) in this case has a greater dimensionality than the input layer (x) and the output layer (r) (Figure 3).
A method using overcomplete autoencoders to handle dropouts is AutoImpute[14]. In this method the autoencoder receives a sparse gene expression matrix as input for the encoder. This autoencoder is used to learn the distribution of the sparse gene expression data by only considering the non-zero counts present in the sparse gene expression matrix.

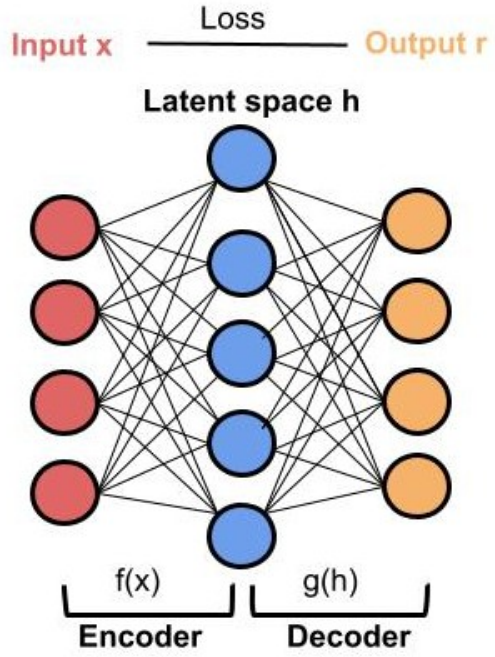

Figure 3: Schematic representation of Overcomplete autoencoder. Note that the bottleneck layer in the latent representation space h presents a higher dimensionality than the input layer x and output layer r.

2.2 Generative models approach

The principal idea of generative models (GM) is to model the probability distribution of the input data. Such a probabilistic model allows to generate new samples that resemble the original samples (training data) but are slightly different (Figure 4). In order to train a GM model is possible to add random noise to regularize the model and a loss function able to compare probability distributions of the original data and the generated by the model. The final generated probability distribution minimizes the loss and new samples can be drawn from it. There are two popular generative model approaches used in scRNA-seq data analysis, variational autoencoders[30] (VAEs), and generative adversarial networks (GANs)[20]. These models are discussed next.

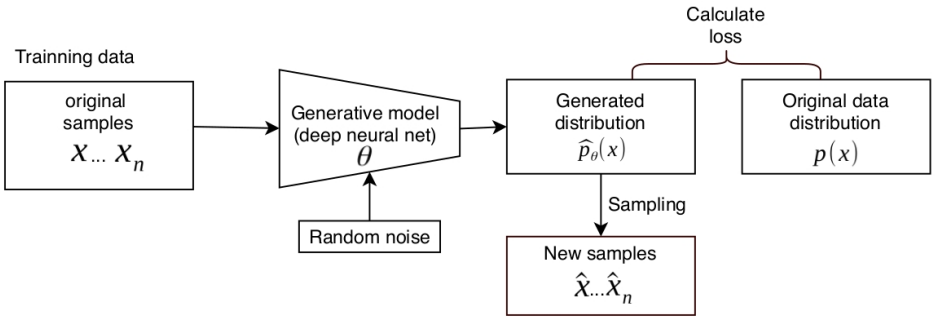

Figure 4: Basic scheme of generative models. The generative model is trained by using training data and by the addition of random noise to generate a probability distribution from which new samples that resemble the training data can be drawn. Adapted from Andrej Karpathy et al. "Generative Models", OpenAI, 16 Jun. 2016[31]

Variational autoencoders

Variational autoencoders present a similar architecture as the autoencoders described in Section 2.1. In VAEs the encoder instead of generating an encoding of the input data in the latent representation h generates encoded means (μ) and encoded standard deviations (σ). The encoded latent space h is a Gaussian distribution with means μ and standard deviations σ (Figure 2.2) from which the encoded data is randomly sampled and then decoded by the decoder generating as output new samples that resemble the instances of the training data.

Depending on the type of input data also other probability distributions than the Gaussian distribution can be used in VAEs. For example, a Bernoulli distribution can be used to model binary data[19].

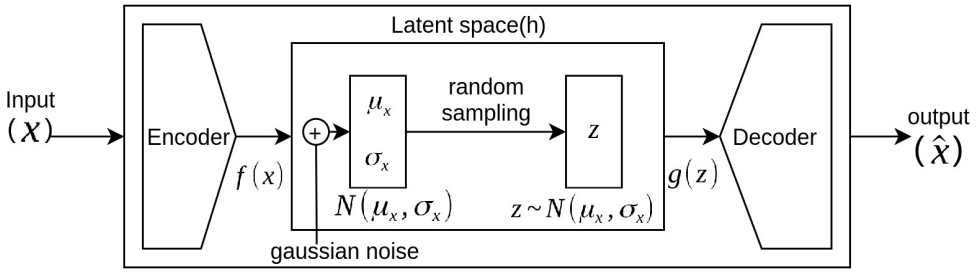

Figure 5: Variational autoencoder architecture. The encoder generates with the addition of noise an encoded mean µx and standard deviation σ_x, the parameters of a normal distribution. The encoded samples (z) are generated by random sampling from the normal distribution N (μ_x, σ_x) and then decoded into the output $\hat{x}$, that is a new sample that resembles the training data samples. Adapted from "Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow" by A. Géron, O'Reilly Media, p.587.[32]

In VAEs the loss function shown in equation 2 presents two terms, one is the general reconstruction term that takes into account the difference between the input and the output. The second part of the equation is the Kullback-Leibler divergence (D_{KL}) that compares how different the normal distribution with parameters μ_x and σ_x returned by the encoder is from a standard normal distribution.

$$\text{loss} = \sum_i ||x_i - \hat{x}_i||^2 + D_{KL}[N(\mu_x, \sigma_x) \text{ - } (\text{N}(0,1))], \text{D}\{\text{x}_i\} \quad (2)$$

Applications of VAEs to scRNA-seq data

Several methods have been developed using VAEs to tackle pre-processing steps in the scRNA-seq data analysis pipeline. The method scVAE can impute the gene expression matrix by estimating the gene expression levels[16]. In this case the VAE is extended using likelihood functions like Poisson or negative binomial to model discrete sparse count scRNA-seq data. These methods

receive the raw count data as input, thereby skipping pre-processing steps such as normalization or data transformation.
Following a similar approach, the method VASC can be used for dimensionality reduction and to improve visualization[9]. This method takes as input the scRNA-seq expression matrix, followed by adropout layer with some of the data points in the input expression matrix randomly set to zero. These layers are followed by the encoder layer that generates the latent Gaussian distribution space modeling the latent variable z. The decoder layer recovers the original expression matrix from z. Following the decoder layer one additional zero-inflated layer is added that allows to include zero values for many observations in count data. This is added to model dropout events.
The method single cell variational inference[33] (scVI) integrates into the framework different scRNA-seq data analysis tasks like batch effect removal, normalization, dimensionality reduction, clustering, and differential expression analysis. scVI performs dimensionality reduction using a VAE and uses this lower-dimensional representation to apply the different data analysis tasks mentioned before. scVI is a fully probabilistic approach based on hierarchical Bayesian models. The VAE architecture uses a Gaussian prior and Gaussian conditional probability distribution. Because the scVI deep learning architecture is composed of several modular blocks, it is possible to improve the model through the combination of different modules. For example, the Gaussian conditional distribution can be replaced by other distributions more suitable for sparse count data, like a negative-binomial, Poisson, or zero-inflated negative binomial distribution.
The method SOLO[34] has been developed to identify doublets (or multiplets). This are cells that have been captured and sequenced together, therefore, share the same barcode but can present different transcriptional states which can lead to incorrect inferences. SOLO uses a variational autoencoder from which the encoder appends to a feed forward neural network layer to form a supervised classifier used to separate doublets from observed data.
A recently developed method called scGen[21] combines the VAE architecture to study the dynamics of single cells in response to perturbations (i.e dose response, knockout of genes or treatment). The rationale behind scGen is to generate in silico data in order to recover biological processes that cannot be identified during the training process, that is to perform out-of-sample prediction.

Generative Adversarial Networks

The main idea behind this type of generative approach consists of training two neural networks. The first network is a generator that generates new samples similar to the original samples present in the training set. The second neural network is the discriminator that tries to detect whether a sample is produced by the generator or belongs to the training set[20].

The schematic representation in Figure 6 shows the architecture of a GAN. The generator receives a random input Z sampled from a standard normal distribution N(0,1) or uniform distribution U(-1,1) and generates a sample x' = G(z). On the other hand, the discriminator receives as input samples X that can be drawn from the training set (true samples) or from samples generated by the generator x' = G(z) (fake samples). The discriminator evaluates whether a sample is fake or real and outputs a probability value D(x). The probability D(x) represents the probability that the sample drawn from the model is real and not generated by the generator.

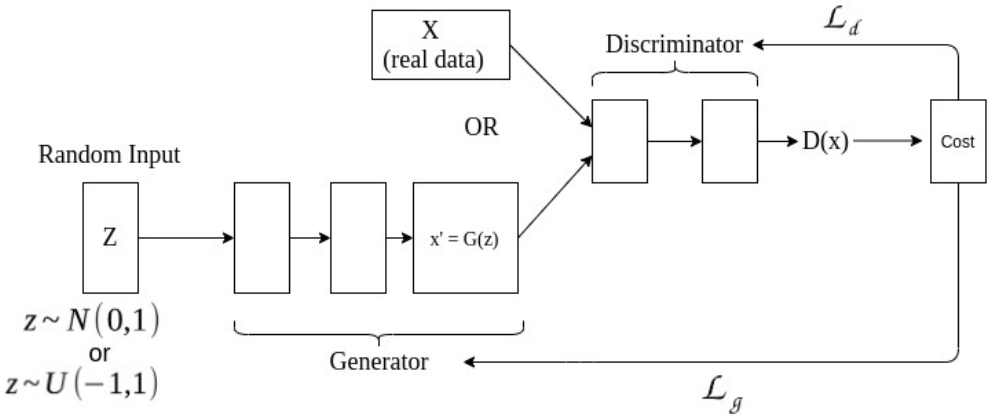

Figure 6: Generative adversarial network schematic representation. The random input z is sampled from a normal distribution N(0,1) or uniform U(-1,1) distribution and used as input of the generator G(z). The main objective of the generator is to produce samples that resemble the real data X. On the other hand, the objective of the discriminator is to discriminate real samples from in silico generated samples. The discriminator output D(x) is used to minimize the generator loss function (Lg) and maximize the discriminator loss function (Ld). Adapted from Hui, Jonathan."GAN — What is Generative Adversarial Networks GAN?" 2018, Medium, 19 Jun. 2018[35]

In GANs the generator and the discriminator are trained jointly. The discriminator D tries to maximize the loss while generator G tries to minimize it (see eq. 3). The loss function includes the probability D(x) of real instance data x_i belongs to the real data in data set d. With E_x as the expected value for all real data instances. The fake instances E_z are generated by the generator G(z_i). The discriminator estimate if a fake instance is real with a given probability D(G(z_i)).

$$\text{loss} = \sum_i E_x[log(D(x_i))] + E_z[log(1 - D(G(z_i)))], d\{x_i\} \tag{3}$$

Applications of GANs to scRNA-seq data

Recently, GANs were applied in the method scIGAN[36] for dropout imputation of scRNA-seq data. scIGAN converts the gene expression profiles of each individual cell into an image in which the pixels represent the normalized gene expression values. In this case the missing parts of the image correspond to dropout events. Using GANs allow the reconstruction of the image (impute dropouts) by avoiding information provided by observed cells. This approach in principle can reduce overfitting of cell types with large populations and be more robust to impute rare cells.

In the same line as scIGAN the method cscGAN[37] uses conditional generative adversarial neural networks with scRNA-seq data to augment on-demand specific sub-populations of cells.

The method MAGAN[38] uses GANs to integrate scRNA-seq data with proteomics data by aligning the manifolds of these two different domains to find a pointwise relationship in order to integrate both domains.

3 Discussion and Conclusions

In this review, the focus is put on the most common deep learning architectures recently used to tackle different issues related to scRNA-seq data. I grouped these architectures within two general architectures, autoencoders and generative models. The main idea behind the autoencoders architecture is based on the advantage of this architectures to extract relevant features by compressing the data in the bottelneck layer to a lower dimension. This is a starting point to different applications for example to denoise scRNA-seq data that usually contains many zeros (dropouts), data imputation, or correction of batch effects. Furthermore, autoencoders can allow gaining better interpretability of the data by addition of prior biological information to improve the dimensionality reduction or by deconvolution of the hidden layers of the autoencoder to evaluate the impact of the selected features during dta compression.

Regarding generative models, the rationale behind this approach is based on the ability to performed data augmentation by learning the underlying probability of the training data. This allows the development of new approaches to tackle pre-processing tasks like data imputation, dimensionality reduction, and batch correction. Concerning performance VAEs have shown advantages over GANs in preliminary studies[21] where is mention that GANs are difficult to train for scRNA-seq data which is high-dimensional and structured data. Also, is mention that is not possible to direct map the input data into the latent space, making the process of augmentation of on-demand cells extremely difficult.

Most of the methods explained in this review have shown promising results not only to solve issues related to the scRNA-seq data analysis but also to integrate different sources of information. However, results of benchmark analysis for data imputation[39], batch correction[40], and doublet identification[41] do not show that deep learning methods clearly outperformance traditional ones. In this sense, more benchmark analysis is needed in order to help researchers to decide which method is more suitable for a given task. This is an important point because deep learning models contain many (hyper)parameters. Therefore, it is not a trivial task to train these models, which can be time consuming, and the resulting models can be difficult to interpret.

In summary, deep learning architectures in principle show a promising future to solve common issues related to scRNA-seq data, not only to pre-process

stage but also to gain biological interpretability. This review shows a comprehensive study showing that the high modularity of the deep learning architectures allows easy adaptation of these architectures to solve different problems related to scRNA-seq data. Furthermore, is an exciting time for both scRNA-seq and deep learning fields with many open questions to tackle in the upcoming future.

4 Acknowledgments

I would like to thank my supervisor Dr. Perry Moerland for his patience and support in writing this review.

References

[1] David Lähnemann et al. "Eleven grand challenges in single-cell data science". In: *Genome Biology* 21.1 (Feb. 2020), p. 53. ISSN: 1474760X. DOI: 10.1186/s13059-020-1926-6. URL: https://doi.org/10.1186/s13059-020-1926-6.

[2] Jie Zheng and Ke Wang. "Emerging deep learning methods for single-cell RNA-seq data analysis". In: *Quantitative Biology* 7.4 (Dec. 2019), pp. 247–254. ISSN: 20954697. DOI: 10.1007/s40484-019-0189-2. URL: https://doi.org/10.1007/s40484-019-0189-2.

[3] Tamim Abdelaal et al. "A comparison of automatic cell identification methods for single-cell RNA sequencing data". In: *Genome Biology* 20.1 (Sept. 2019), p. 194. ISSN: 1474760X. DOI: 10.1186/s13059-019-1795-z. URL: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1795-z.

[4] Jialong Liang, Wanshi Cai, and Zhongsheng Sun. "Single-Cell Sequencing Technologies: Current and Future". In: *Journal of Genetics and Genomics* 41.10 (2014), pp. 513–528. ISSN: 18735533. DOI: 10.1016/j.jgg.2014.09.005. URL: http://dx.doi.org/10.1016/j.jgg.2014.09.005.

[5] Malte D Luecken and Fabian J Theis. "Current best practices in single-cell RNA-seq analysis: a tutorial". In: *Molecular Systems Biology* 15.6 (June 2019), e8746. ISSN: 1744-4292. DOI: 10.15252/msb.20188746. URL: https://onlinelibrary.wiley.com/doi/10.15252/msb.20188746.

[6] Vladimir Yu Kiselev, Tallulah S. Andrews, and Martin Hemberg. "Challenges in unsupervised clustering of single-cell RNA-seq data". In: *Nature Reviews Genetics* 20.5 (May 2019), pp. 273–282. ISSN: 14710064. DOI: 10.1038/s41576-018-0088-9. URL: www.nature.com/nrg.

[7] Yue Cao et al. "ScDC: Single cell differential composition analysis". In: *BMC Bioinformatics* 20.S19 (Dec. 2019), p. 721. ISSN: 14712105. DOI: 10.1186/s12859-019-3211-9. URL: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-019-3211-9.

[8] Wouter Saelens et al. "A comparison of single-cell trajectory inference methods". In: *Nature Biotechnology* 37.5 (May 2019), pp. 547–554. ISSN: 15461696. DOI: 10.1038/s41587-019-0071-9. URL: https://doi.org/10.1038/s41587-019-0071-9.

[9] Dongfang Wang and Jin Gu. "VASC: Dimension Reduction and Visualization of Single-cell RNA-seq Data by Deep Variational Autoencoder". In: *Genomics, Proteomics and Bioinformatics* 16.5 (Oct. 2018), pp. 320–331. ISSN: 22103244. DOI: 10.1016/j.gpb.2018.08.003.

[10] Ying Ma et al. "Integrative differential expression and gene set enrichment analysis using summary statistics for scRNA-seq studies". In: *Nature Communications* 11.1 (Dec. 2020), pp. 1–13. ISSN: 20411723. DOI: 10.1038/s41467-020-15298-6. URL: https://doi.org/10.1038/s41467-020-15298-6.

[11] Giovanni Iacono, Ramon Massoni-Badosa, and Holger Heyn. "Single-cell transcriptomics unveils gene regulatory network plasticity". In: *Genome Biology* 20.1 (June 2019), p. 110. ISSN: 1474760X. DOI: 10.1186/s13059-019-1713-4. URL: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1713-4.

[12] Gökcen Eraslan et al. "Deep learning: new computational modelling techniques for genomics". In: *Nature Reviews Genetics* 20.7 (July 2019), pp. 389–403. ISSN: 14710064. DOI: 10.1038/s41576-019-0122-6. URL: www.nature.com/nrg.

[13] Uri Shaham et al. "Removal of batch effects using distribution-matching residual networks". In: *Bioinformatics* 33.16 (2017), pp. 2539–2546. ISSN: 14602059. DOI: 10.1093/bioinformatics/btx196. arXiv: 1610.04181. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5870543/%20https://www.you.

[14] Divyanshu Talwar et al. "AutoImpute: Autoencoder based imputation of single-cell RNA-seq data". In: *Scientific Reports* 8.1 (Dec. 2018), pp. 1–11. ISSN: 20452322. DOI: 10.1038/s41598-018-34688-x. URL: www.nature.com/scientificreports/.

[15] 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 (Dec. 2018), pp. 1–13. ISSN: 20411723. DOI: 10.1038/s41467-018-04368-5. URL: www.nature.com/naturecommunications.

[16] Christopher Heje Grønbech et al. "scVAE: variational auto-encoders for single-cell gene expression data". In: *Bioinformatics* 36.16 (Aug. 2020). Ed. by Bonnie Berger, pp. 4415–4422. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btaa293. URL: https://academic.oup.com/bioinformatics/article/36/16/4415/5838187.

[17] H Bourlard and Y Kamp. *Biological Cybernetics Auto-Association by Multilayer Perceptrons and Singular Value Decomposition*. Tech. rep. 1988, pp. 291–294.

[18] Chieh Lin et al. "Using neural networks for reducing the dimensions of single-cell RNA-Seq data". In: *Nucleic Acids Research* 45.17 (2017), p. 156. DOI: 10.1093/nar/gkx681.

[19] Diederik P. Kingma and Max Welling. "Auto-encoding variational bayes". In: *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, Dec. 2014. arXiv: 1312.6114. URL: https://arxiv.org/abs/1312.6114v10.

[20] Ian J Goodfellow et al. *Generative Adversarial Nets*. Tech. rep. 2014, pp. 2672–2680. URL: http://www.github.com/goodfeli/adversarial.

[21] Mohammad Lotfollahi et al. "scGen predicts single-cell perturbation responses". In: *Nature Methods* 16.8 (Aug. 2019), pp. 715–721. ISSN: 15487105. DOI: 10.1038/s41592-019-0494-8. URL: https://doi.org/10.1038/s41592-019-0494-8.

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[23] Md Bahadur Badsha et al. "Imputation of single-cell gene expression with an autoencoder neural network". In: *Quantitative Biology* 8.1 (Mar. 2020), pp. 78–94. ISSN: 20954697. DOI: 10.1007/s40484-019-0192-7. URL: https://doi.org/10.1007/s40484-019-0192-7.

[24] Jiajie Peng, Xiaoyu Wang, and Xuequn Shang. "Combining gene ontology with deep neural networks to enhance the clustering of single cell RNA-Seq data". In: *BMC Bioinformatics* 20.8 (June 2019), pp. 1–12. ISSN: 14712105. DOI: 10.1186/s12859-019-2769-6. URL: https://link-springer-com.vu-nl.idm.oclc.org/articles/10.1186/s12859-019-2769-6%20https://link-springer-com.vu-nl.idm.oclc.org/article/10.1186/s12859-019-2769-6.

[25] Savvas Kinalis et al. "Deconvolution of autoencoders to learn biological regulatory modules from single cell mRNA sequencing data". In: *BMC Bioinformatics* 20.1 (July 2019), pp. 1–9. ISSN: 14712105. DOI: 10.1186/s12859-019-2952-9. URL: https://link-springer-com.vu-nl.idm.oclc.org/articles/10.1186/s12859-019-2952-9%20https://link-springer-com.vu-nl.idm.oclc.org/article/10.1186/s12859-019-2952-9.

[26] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*. Tech. rep. Dec. 2013. arXiv: 1312.6034. URL: http://arxiv.org/abs/1312.6034.

[27] Pascal Vincent et al. "Extracting and composing robust features with denoising autoencoders". In: *Proceedings of the 25th International Conference on Machine Learning*. New York, New York, USA: Association for Computing Machinery (ACM), 2008, pp. 1096–1103. ISBN: 9781605582054. DOI: 10.1145/1390156.1390294. URL: http://portal.acm.org/citation.cfm?doid=1390156.1390294.

[28] Pascal Vincent. "A connection between scorematching and denoising autoencoders". In: *Neural Computation* 23.7 (July 2011), pp. 1661–1674. ISSN: 08997667. DOI: 10.1162/NECO_a_00142. URL: https://www-mitpressjournals-org.vu-nl.idm.oclc.org/doix/abs/10.1162/NECO%7B%5C_%7Da%7B%5C_%7D00142.

[29] Pascal Vincent@umontreal Ca et al. *Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion Pascal Vincent Hugo Larochelle Yoshua Bengio Pierre-Antoine Manzagol*. Tech. rep. 2010, pp. 3371–3408.

[30] Diederik P Kingma Google, Max Welling, and Boston -Delft. "An Introduction to Variational Autoencoders". In: *Foundations and Trends R in Machine Learning* xx, No. xx (2019), pp. 1–18. DOI: 10.1561/XXXXXXXXX. arXiv: 1906.02691v3.

[31] Wojciech Zaremba Andrej Karpathy Pieter Abbeel Greg Brockman Peter Chen Vicki Cheung Rocky Duan Ian Goodfellow Durk Kingma Jonathan Ho Rein Houthooft Tim Salimans John Schulman Ilya Sutskever. *Generative Models*. 2016. URL: https://openai.com/blog/generative-models/ (visited on 04/27/2021).

[32] Aurélien Géron. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, 2019, p. 587. ISBN: 9781492032649.

[33] Romain Lopez et al. "Deep generative modeling for single-cell transcriptomics". In: *Nature Methods* 15.12 (Dec. 2018), pp. 1053–1058. ISSN: 15487105. DOI: 10.1038/s41592-018-0229-2. URL: /pmc/articles/PMC6289068/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6289068/.

[34] Nicholas J. Bernstein et al. "Solo: Doublet Identification in Single-Cell RNA-Seq via Semi-Supervised Deep Learning". In: *Cell Systems* 11.1 (July 2020), 95–101.e5. ISSN: 24054720. DOI: 10.1016/j.cels.2020.05.010. URL: https://doi.org/10.1016/j.cels.2020.05.010.

[35] J. Hui. *GAN — What is Generative Adversarial Networks GAN? | by Jonathan Hui | Medium*. URL: https://jonathan-hui.medium.com/gan-whats-generative-adversarial-networks-and-its-application-f39ed278ef09 (visited on 04/27/2021).

[36] Yungang Xu et al. "scIGANs: single-cell RNA-seq imputation using generative adversarial networks". In: *Nucleic Acids Research* 48.15 (Sept. 2020), E85. ISSN: 13624962. DOI: 10.1093/nar/gkaa506. URL: https://academic.oup.com/nar/article/48/15/e85/5862684.

[37] Mohamed Marouf et al. "Realistic in silico generation and augmentation of single-cell RNA-seq data using generative adversarial networks". In: *Nature Communications* 11.1 (Dec. 2020), pp. 1–12. ISSN: 20411723. DOI: 10.1038/s41467-019-14018-z. URL: https://doi.org/10.1038/s41467-019-14018-z.

[38] Matthew Amodio and Smita Krishnaswamy. "MAGAN: Aligning Biological Manifolds". In: *35th International Conference on Machine Learning, ICML 2018* 1 (Feb. 2018), pp. 327–335. arXiv: 1803.00385. URL: http://arxiv.org/abs/1803.00385.

[39] Wenpin Hou et al. "A systematic evaluation of single-cell RNA-sequencing imputation methods". In: *Genome Biology* 21.1 (Aug. 2020), p. 218. ISSN: 1474760X. DOI: 10.1186/s13059-020-02132-x. URL: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-020-02132-x.

[40] Hoa Thi Nhu Tran et al. "A benchmark of batch-effect correction methods for single-cell RNA sequencing data". In: *Genome Biology* 21.1 (Jan. 2020), p. 12. ISSN: 1474760X. DOI: 10.1186/s13059-019-1850-9. URL: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1850-9.

[41] Authors Nan Miles Xi and Jingyi Jessica Li Correspondence. "Benchmarking Computational Doublet-Detection Methods for Single-Cell RNA Sequencing Data". In: *Cell Systems* 12 (2021), pp. 176–194. DOI: 10.1016/j.cels.2020.11.008. URL: https://doi.org/10.1016/j.cels.2020.11.008.