The Effect of Different Initialization Methods on VAEs for Modeling Cancer using RNA Genome Expressions

I.S. Kroskinski

June 26, 2021

Author:	I.S. Kroskinski
Responsible Professor:	M. Reinders
Other Supervisor:	{S .Makrodimitris, T.R.M. Abdelaal-1, M. Charrout, M.A.M.E. elTager}
Peer group members:	{R. d'Anjou, A. Korkic, B. van Groeningen, B. Pronk}

Abstract

Variational Auto-Encoders are a class of machine learning models that have been used in varying context, such as cancer research. Earlier research has shown that initialization plays a crucial part in training these models, since it can increase performance. Therefore, this paper studies the effect initialization methods on VAEs. This research shows that if using only one hidden layer, Uniform methods and Xavier methods perform best depending on the VAE model, where the standard VAE shows the most sensitivity to these methods. But, if using more hidden layers, the uniform method performs significantly worse than a method that uses the number of inputs of the layer such as the default implementation of PyTorch, Xavier Normal or Xavier Uniform. However, after enough epochs in all other models these initialization methods converge.

1 Introduction

Cancer is one of the deadliest diseases which causes tens of thousands of deaths in the Netherlands alone [1]. However, after years of research there is still no effective way of treating cancer. This is caused by the nature of the disease, which is constantly evolving and finding new ways to evade the immune system. Therefore, scientist have been looking at personalized medicines to treat patients suffering from cancer. However, to achieve this it is of utmost importance to predict drug response and expected survival rate from tumor data.

One of the methods for predicting drug response and expected survival rate is using machine and deep learning models such as Variational Auto-Encoders (VAE) [2]. VAEs try to perform dimensionalty reduction to create a latent space. Using this latent space, it reconstructs the original input. The difference between the original input and the reconstructed input is called the loss. Section 3.1 goes more in depth about the implementation

of different VAEs and how the loss function is calculated. Earlier research has shown that VAEs have had success in processing images [3] and text [4]. For this reason, research has been conducted by Way and Greene [5] on biological data using The Cancer Genome Atlas (TCGA) [6]. Using their VAE model [7] they were able to find disentangled representations and tissue specific patterns among other things. However, improved models are needed to accurately detect specific cancer types to enable personalized treatment plans. The earlier work by Sutskever [8] shows that for neural networks, initialization plays a crucial role in the performance and reproducibility of the model.

Therefore, this paper discusses the importance of initialization on VAEs on biological data. Firstly, we quantify the use of different initialization methods by assessing the impact of these methods on the training of the models. Additionally, different VAE models will be compared to conclude if certain models are more sensitive to initialization than other models.

This paper contributes to the discussion of how important initialization of machine learning models are, more specifically in the field of biological data. It quantifies this by analyzing the convergence rate and convergence point, which allows to conclude which methods of initialization perform best. In addition, it decides which VAEs models are more sensitive to initialization methods.

The paper starts with section 2 describing the methodology used, followed by section 3 which describes the contribution of this paper. Section 4 will discuss the experimental setup and results. Next is section 5, which addresses ethical aspects of this research and reproducibility of the methods used. Finally, a discussion and a conclusions is given in sections 6 and 7 respectively.

2 Methodology

2.1 Analysis

This research performs empirical analysis on different models of Variational Auto-Encoders. For each model, multiple initialization methods are used to benchmark the performance of each method with its specific model. Performance are assessed on multiple aspects of the validation loss of the different initialization methods and models. First, each lower bound of the validation loss are compared to conclude which of the methods and models performs best after a set amount of iterations, also known as epochs. Secondly, the convergence of rate of the models and initialization method to the validation loss lower-bound are also assessed. Finally, the variance of the validation loss are compared to conclude if some models are more sensitive to initialization methods.

2.2 VAEs and Initialization Methods

This research is conducted on four different VAEs and five different initialization methods. The four different VAEs are:

- Variation Auto-Encoder (VAE) [2]
- Importance Weighted Auto-Encoder (IWEA) [9]
- Information Maximizing Variational Auto-Encoders (InfoVAE) [10]
- Log Hyperbolic Cosine Variational Auto-Encoder (LogCoshVAE) [11]

Next, the following initialization methods are used to assess their performance and conclude if some VAEs are more sensitive to initialization methods than others:

- Default Implementation of PyTorch [12]
- Uniform
- Normal
- Xavier Normal [13]
- Xavier Uniform [13]

These initialization methods are chosen since they are already implemented in the PyTorch library which is used to train these methods.

2.3 Repositories

In order to implement these models and initialization methods, this paper uses the PyTorch-VAE repository [3]. This repository uses a framework for PyTorch [14] called PyTorch-Lightning [15] which makes implementing neural networks easier and faster. Originally this repository is used to process images, however this paper uses biological 1D data. This means that the code has to be adjusted accordingly by using linear layers instead of convolutional layers and changing settings in the configuration files of the models. More in depth implementation is discussed in section 4. The advantages of using this repository is that it has a basic framework and has the different VAE models implemented already. Besides that, the code base is available online for free and can be changed.

2.4 Data

The data used in this research is a data set of gene expression of RNA genes [16] from The Cancer Genome Atlas (TCGA) [6]. A gene expression is used in the body to produce molecules. This data set contains the gene expression values of 11060 samples from patients with different cancer types. In order to speed up the training of different VAE models and be more inline with previous research from the Tybalt paper [5], only the 5000 most variable genes will be used which are chosen using the Mean Absolute Deviation (MAD). Also the data will be normalized between the values zero and one in order for the activation functions of the model to perform better. Finally, the data set will be used in a 80% 20% split for training and validating respectively.

3 Benchmarking Different Initialization Methods

The following section will go further in depth of the implementation of these VAEs and initialization methods.

3.1 Models

3.1.1 VAE

The normal VAE is similar to a Auto Encoder [17] which is a form of neural network which tries to perform a dimensionality reduction on the number of features which describe the

data. However, it differs from a normal Auto Encoder by representing the data as a distribution instead of a single point. After the dimensionality reduction has been performed, it tries to reconstruct the original data point using a sample of the low-dimensional distribution. The difference between the original input and the reconstruction of the input is known as the loss. In this case the loss function is defined as follows:

$$\text{loss} = mse\_loss(\text{recons, input}) + \text{kld\_weight} \cdot KL(N(\mu, \sigma), N(0, 1))$$

Where mse_loss is defined as the Mean Squared error between the two inputs, kld_weight defined by the size of the batch size of the model. KL is the Kullback-Leibler loss [18] which compares the density function of the model with that of the standard normal distribution. The model tries to push the distribution to a standard normal distribution to reduce its complexity.

3.1.2 IWAE

IWEA is an Importance Weighted Auto Encoder which differs from the normal VAE by using importance weighting or else known by importance sampling. It uses multiple samples in order to perform variance reduction which can results in better estimations. The loss functions for this model is defined as follows:

$$\text{log\_weight} = mse\_loss(recons, input) + kld\_weight \cdot KL(N(\mu, \sigma), N(0, 1))$$

$$\text{loss} = mean(\sum(soft\_max(\text{log\_weight}) \cdot \text{log\_weight}))$$

Where softmax is defined as on the PyTorch documentation [19].

3.1.3 InfoVAE

InfoVAE takes a different approach in trying to resolve some issues that come with the normal VAE. Classic VAEs make use of Evidence Lower Bound (ELBO) which tends to overfit on the training data. InfoVAE tries to resolve this problem by using a different method called Maximum Mean Discrepancy (MMD) [20]. Therefore, the loss function is defined as follows:

$$\text{kld\_loss} = \text{kld\_weight} \cdot KL(N(\mu, \sigma), N(0, 1))$$

$$\text{bias\_corr} = \text{batch\_size} \cdot (\text{batch\_size} - 1)$$

$$\text{loss} = \beta \cdot mse\_loss(\text{recons, input}) + (1 - \alpha) \cdot \text{kld\_loss} + \frac{(\alpha + \text{reg\_weight} - 1)}{\text{bias\_corr} \cdot mmd\_loss(z)}$$

Where α, β and reg_weight are predefined in the configuration file and mmd_loss is the Maximum Mean Discrepancy on the distribution z defined by μ and σ taken as sample from the latent space.

3.1.4 LogCoshVAE

LogCoshVAE is similar to the standard VAE except for a differentiation in the calculation of the reconstruction loss. Instead of using Mean Squared Error (MSE) it uses a hyperbolic cosine (log-cosh) function which behaves as MSE at small values but different at larger

values. According to the authors, this should lead to better performance on reconstruction loss without damaging the latent space. The loss function for this model is defined as follows:

$$\text{t} = \text{recons} - \text{input}$$

$$\text{recon\_loss} = \frac{1}{\alpha} \cdot mean(\alpha \cdot t + \log(1 + exp(-2 \cdot \alpha * t)) - \log(2))$$

$$\text{loss} = \text{recon\_loss} + \beta \cdot \text{kld\_weight} \cdot KL(N(\mu, \sigma), N(0, 1))$$

Where α is predefined in the configuration file.

3.2 Initialization Methods

3.2.1 Default PyTorch

The default initialization method that the PyTorch library uses is a form of uniform distribution [12]. It uses this distribution to set the values of the layer.

$$\mathcal{U}(-\sqrt{k}, \sqrt{k})$$

$$k = \frac{1}{\text{fan\_in}}$$

Where fan_in is defined as the number of inputs into the layer

3.2.2 Normal and Uniform

Two methods supported are Normal and Uniform initialization methods. When using a uniform initialization method, it sets all weights in the encoder and decoder of the model to values drawn from a $\mathcal{U}$(a,b) distribution. Where a is the lower bound and b is the upper bound of the uniform distribution. In this specific instance, a and b are set to zero and one respectively.

The Normal initialization method is similar to the Uniform method, where it sets the encoder and decoder weights. However, it sets the weights to values drawn from a $\mathcal{N}(\mu, \sigma^2)$ distribution. In this specific instance, μ and σ are set to zero and one respectively.

3.2.3 Xavier Normal and Xavier Uniform

The other two initialization methods are Glorot Normal and Glorot Uniform, or also known as Xavier Normal and Xavier Uniform [13]. The authors of these initialization methods claim when using sigmoid activation, random initialization performs poorly. Therefore, they claim there new method allow the model to converge substantially faster.

For Xavier Uniform initialization, the weights of the encoder and decoder are drawn from an $\mathcal{U}$(-a,a) distribution. Where a is defined as follows:

$$a = gain \cdot \sqrt{\frac{6}{fan\_in + fan\_out}}$$

In the implementation of the PyTorch library, gain is an optional parameter for a scaling factor depending on the activation function. Whereas fan_in and fan_out are the inputs and outputs of the layer.

Xavier Normal is again similar to the Xavier Uniform method where it draws the values of the weights for the encoder and decoder, however uses a $\mathcal{N}(0, \sigma^2)$ distribution. Where σ is defined as follows:

$$\sigma = gain \cdot \sqrt{\frac{2}{fan\_in + fan\_out}}$$

4 Experimental Setup and Results

As experimental setup, all the code will be run in a python script in an Anaconda environment using the High Performance Cluster (HPC) [21] from the Technical University of Delft or using a personal computer with a GTX1080. The code with the results, configuration files and seeds are available on the repository of TU Delft [22]. This repository is an edited version from the PyTorch VAE library on GitHub [3], such that it is compatible with the RNA data set obtained from the TCGA.

4.1 Hidden Layers

All four models are using the same neural network consisting of an encoder and decoder. It uses one hidden layer both for encoding and decoding. For the encoder network, it uses a hidden layer consisting of a linear network of input size with the number of genes and output size 256, a batch normalization of 256 and finally a ReLu activation layer. Then this layer is connected to the final layer using a linear network of input 256 and output the number of latent features.

The decoder uses a linear network with input being the number of latent features and output 256, followed by a batch normalization of size 256 and finally a ReLu activation layer. Then this layer is connected to the final layer using 256 as input and the number of genes as output with at the end a sigmoid activation function.

When these layers are initialized, the weights of these layers are set depending on the initialization method predefined in the configuration file of the VAE.

4.2 Parameters

Choosing the parameters, this paper follows mostly the parameters specified in the Tybalt paper [5]. The parameters used from the Tybalt in all VAE models are the latent dimensions used which are 100, the learning rate which is 0.005, weight decay is 0.0, scheduler gamma is 0.95 and max number of epochs is 50.

Other parameters used which are not from the Tybalt paper are: batch size, in channels and initialization method. Batch size is 1000 for the training set but the validation step uses the full length of the data set. In channels in the original code is used for images for colors such as RGB. However, since we are using 1D data from the TCGA, this has to be set to one. Finally, for this specific repository the initialization method needs to be specified and has to be one of the four described earlier in this paper. Otherwise, the default initialization method from the PyTorch library will be used.

The other parameters such as α, β and reg_weight, which are specific to some VAEs, are left as the default values from the PyTorch VAE library. The list of all values can be seen in the configuration files in the repository of this research [22].

4.3 Preprocessing

To better train the different VAE models and also train them faster, some preprocessing is performed on the TCGA data similar to that used in the Tybalt paper. Of the 20532 genes only the 5000 most variable genes will be used. The 5000 most variable genes are chosen based on Mean Absolute Deviation (MAD). Finally, the data was transposed to make sure the samples were indexed as rows and the genes as columns.

4.4 Runner

Since this research focuses on the effect of initialization methods and the sensitivity of different VAE models to these methods, each model with each initialization method is executed ten times. This is done in order to get the mean and the standard deviation of each model and initialization method to get to a conclusion. Besides that, it is done to account for variance across runs due to randomness. Also, for each configuration the train and validation set is randomly sampled from the TCGA data set. Finally, for each configuration a different seed is used to decide randomization, which is saved in the configuration file together with the results for reproducibility.

4.5 Results

The mean is taken of all ten runs for each method together with the variance between the configurations. The first data point is skipped since the model has not trained yet at that point, which causes some irregularities in the data.

4.5.1 VAE

Below in figure 1 is a graph of the different initialization methods ran ten times for 50 epochs and taken the mean of all methods. This figure concludes that the default method, Xavier Normal, Xavier Uniform perform extremely similar. With that, the Normal method performs the worst, although it does eventually converge together with both the Xavier Normal and Xavier Uniform method. It is clear as well that the Uniform method performs significantly better than the other three methods. Not only does it converge faster, it also converges lower.

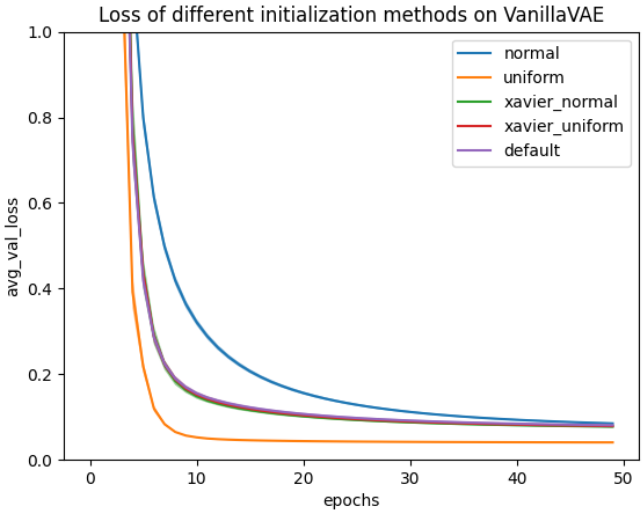

Figure 1: A graph of the mean of the validation loss of different initialization methods of VAE ran for 50 epochs. The y-axis is limited to 1.0 for a more clear image.

4.5.2 IWAE

In figure 2 is the graph of the IWAE model using different initialization methods. As in the standard VAE, the default method, Xavier Normal and Xavier Uniform perform extremely similar. Besides that, we can see that the Normal method starts off best, however it converges last of all methods at again around 50 epochs. This time the uniform method does not perform the best in the early stages of training, but does converge the lowest although with a minimal margin. All in all, all initialization methods converge around the same spot, at around 50 epochs.

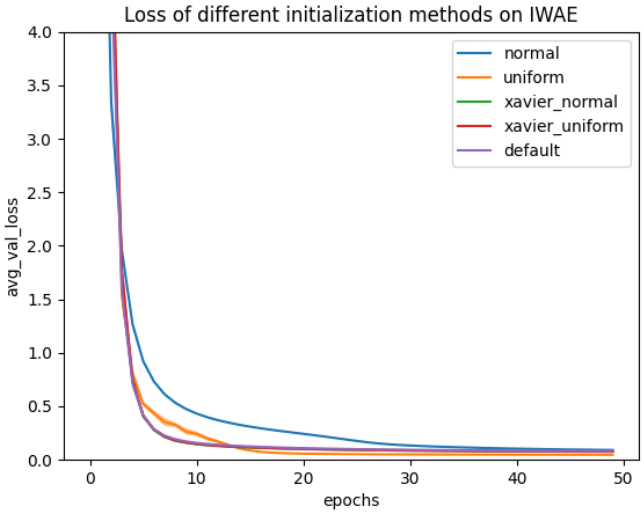

Figure 2: A graph of the mean of the validation loss of different initialization of IWAE methods ran for 50 epochs. The y-axis is limited to 4.0 for a more clear image.

4.5.3 InfoVAE

In figure 3 is the graph of the InfoVAE model. The one thing that is different from the other models is that this model has a lot of variance between eight and twenty epochs. This does not however influence the end results of the methods since they all converge around the 40 epochs. Again the Uniform method converges the fastest, but this time the normal method converges faster than both the default, Xavier Normal and Xavier Uniform which both perform similar again.

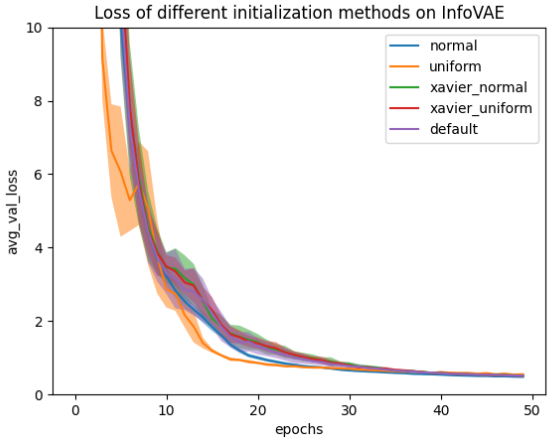

Figure 3: A graph of the mean of the validation loss of different initialization of InfoVAE methods ran for 50 epochs. The y-axis is limited to 10.0 for a more clear image.

4.5.4 LogCoshVAE

Finally the results of the LogCoshVAE are visible in figure 4. This model performs similar with the IWAE model. Although the default method, Xavier Normal and Xavier Uniform do perform a bit better in this model. Besides that, the Uniform method is a close second with Normal last.

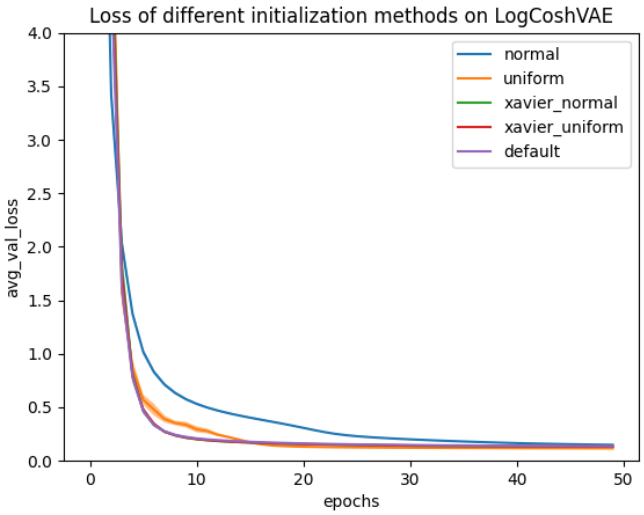

Figure 4: A graph of the mean of the validation loss of different initialization of LogCoshVAE methods ran for 50 epochs. The y-axis is limited to 4.0 for a more clear image.

4.5.5 Z-Score Normalization

In order to conclude if indeed the normalization method plays a role why the uniform outperforms the Xavier methods in some instances, a different normalization method will be used to compare results. Z-score normalization is used to compare the results using the VanillaVAE model and all different initialization methods.

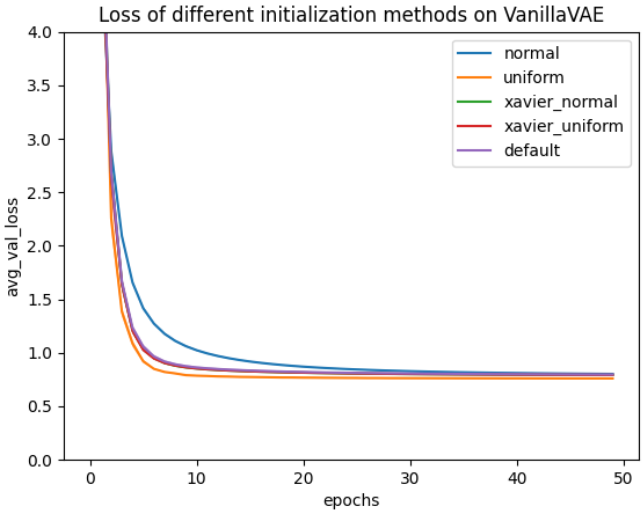

Figure 5: A graph of the mean of the validation loss of different initialization of VanillaVAE methods ran for 50 epochs using Z-scoring. The y-axis is limited to 4.0 for a more clear image.

If we compare the results from using normalizing between zero and one and using Z-scoring it is clear this does not influence the performs of the different initialization methods.

4.5.6 More Hidden Layers

Finally, this section discusses if using more hidden layers has an influence on the performance of each methods. In this instance, the model in question is again the VanillaVAE model using all different initialization methods. However, instead of using only one hidden layer, it makes use of three hidden layers [1024, 512, 256].

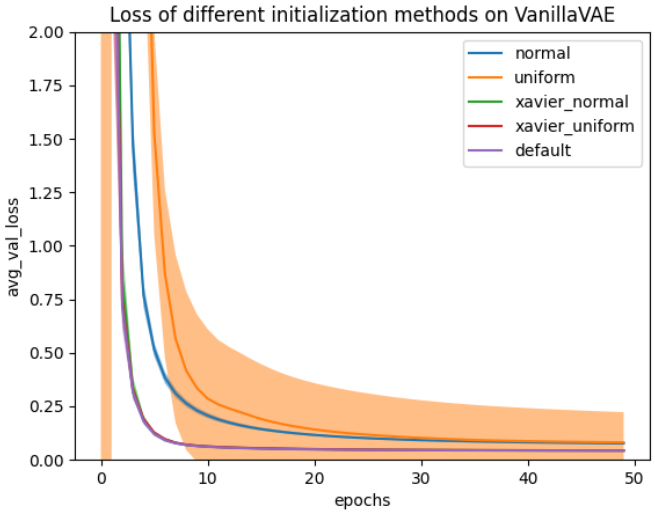

Figure 6: A graph of the mean of the validation loss of different initialization of VanillaVAE methods ran for 50 epochs with three hidden layers. The y-axis is limited to 2.0 for a more clear image.

It is clear in the images that the uniform method performs significantly worse and the default, Xavier Normal and Xavier Uniform converge the fastest. Not only does the uniform method converge slower, it also has a lot of variance as seen by the surface of the method.

5 Responsible Research

5.1 Ethics

The ethical consequences of this paper are concerned around patient data. First of all, when using real data from human patients it is of uttermost importance to value the patience privacy. The data must be anonymized in order to withhold that, although this is sometimes not enough. Studies have shown that it is possible to re-identify anonymized data to individual users [23], which leads to a breach in privacy protection. This is more often the case with rare diseases and the use of cross-correlation attacks.

Secondly, there is also the issue of the data used in this specific research. Although it is data from real patients, it might not be representable since it is a small sample of a larger population. Therefore, the data might be a misrepresentation and can lead to inconclusive results. Thus, it is paramount that the sampled data set is an accurate representation of the actual population in order to produce significant results.

5.2 Reproducibility

There is also the discussion around the reproducibility of machine learning algorithms. Since VAEs act like a black box it is extremely difficulty to understand the inner workings of them. This causes that it is almost impossible to recreate the exact same model with the exact same data and parameters. Although it is not possible to create reproducibility, it is possible

to create replicability. However, there are some who say that replicability is not desired in the scientific community [24] and thus should not depended on heavily. Meanwhile there are some studies that confirm challenges in reproducibility in machine learning, but do produce desired results in the medical field [25] and thus should be considered carefully as tools.

Although reproducibility in this research is also an extremely difficult task, it is less important since the fact all models and initialization methods are run multiple times with different seed and different data splits. Then the mean of these are given and taken into consideration and thus only significant differences in performance are considered. Besides that, if another party would like to reproduce this study, all the seeds used in each configuration is saved in the configuration file.

6 Discussion

From the results it does seem that the uniform method performs best when using the InfoVAE or VanillaVAE. However, for the other two models (IWAE and LogCoshVAE) the default, Xavier Normal and Xavier Uniform perform best.

The results given in section 4.5 are somewhat different than the authors of the Xavier Normal and Xavier Uniform describe. They claimed that there method should perform significantly better than any other method especially in the top layers [13], although the Uniform method performed in some scenarios significantly better (InfoVAE and VAE).

There can be multiple explanations for this. First, uniform method only uses positive integers and thus performs better on this particular normalization method. Secondly, the models do not use enough hidden layers for the Xavier methods work as intended. Since they claim it should outperform significantly, especially in the top layers. Finally, this research uses a totally different data type than used in the research. The research uses images, whereas this research focuses on 1D data.

In order to conclude if the normalization method plays a role in the performance of the methods, Z-scoring was used. As seen in section 4.5.5 the results were very similar between both normalization methods. Therefore, it can be concluded that normalization does not play an important role between initialization methods.

Also, as seen in section 4.5.6 using more hidden layers has an enormous impact on the performance of initialization methods. Not only do the default method, Xavier Normal and Xavier Uniform outperform the uniform method now, the uniform method performs significantly worse with a lot of variance. So therefore, one might conclude that if using multiple hidden layers, using an initialization method that uses some form of input size of the layer, performs significantly better.

Finally, the lower bound of these models and initialization methods might not be optimal. This is since no extensive research has been done finding the optimal parameters for example the learning rate. Therefore, if someone may try to perform more research on this topic using different parameters they might conclude different results.

7 Conclusions and Future Work

7.1 Conclusion

If using only one hidden layer, the uniform method works best when using the standard VAE and InfoVAE, whereas the Xavier Normal and Xavier Uniform method work better

with IWAE and LogCoshVAE. Using a Normal distribution method is not recommended since it converges last in all models except InfoVAE. However, using the RNA genes data set from TCGA, running it for around 50 epochs results in all method converging around the same loss except for the standard VAE. That is where the uniform method performs significantly better than the other methods by the convergence rate and lower bound. All in all, the standard VAE is more sensitive to initialization methods than others.

However, if using multiple hidden layers, the uniform method performs extremely bad and it is recommended to use an initialization method that use some input of the hidden layer such as: default method of PyTorch, Xavier Normal or Xavier Uniform.

7.2 Future Work

As described in sections five and six, further research has to be performed using a larger data set, more hidden layers and different parameters for the Uniform and Normal methods. Also since this study mainly focused on the effect of initialization, no specific research has been done to find the optimal parameters for each of the models and initialization methods to keep it consistent. Thus, if one might find these parameters for the learning rate for example, it might lead to different results.

References

[1] "Ranglijst doodsoorzaken op basis van sterfte | Volksgezondheidenzorg.info." [Online]. Available: https://www.volksgezondheidenzorg.info/ranglijst/ranglijst-doodsoorzaken-op-basis-van-sterfte

[2] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2014.

[3] A. Subramanian, "Pytorch-vae," https://github.com/AntixK/PyTorch-VAE, 2020.

[4] Z. Yang, Z. Hu, R. Salakhutdinov, and T. Berg-Kirkpatrick, "Improved variational autoencoders for text modeling using dilated convolutions," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 3881–3890. [Online]. Available: http://proceedings.mlr.press/v70/yang17d.html

[5] G. P. Way and C. S. Greene, "Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders," in *PACIFIC SYMPOSIUM ON BIOCOMPUTING 2018: Proceedings of the Pacific Symposium*. World Scientific, 2018, pp. 80–91.

[6] National Cancer Institute", "The Cancer Genome Atlas Program." [Online]. Available: https://www.cancer.gov/about-nci/organization/ccg/research/structural-genomics/tcga

[7] G. Way and C. Greene, "greenelab/tybalt," 01 2019. [Online]. Available: https://github.com/greenelab/tybalt

[8] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1139–1147. [Online]. Available: http://proceedings.mlr.press/v28/sutskever13.html

[9] Y. Burda, R. Grosse, and R. Salakhutdinov, "Importance weighted autoencoders," 2016.

[10] S. Zhao, J. Song, and S. Ermon, "Infovae: Information maximizing variational autoencoders," 2018.

[11] P. Chen, G. Chen, and S. Zhang, "Log hyperbolic cosine loss improves variational auto-encoder," 2019. [Online]. Available: https://openreview.net/forum?id=rkglvsC9Ym

[12] "PyTorch/Linear.py," 05 2021. [Online]. Available: https://github.com/pytorch/pytorch/blob/master/torch/nn/modules/linear.py#L53

[13] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[14] "PyTorch." [Online]. Available: https://pytorch.org/

[15] "PyTorch Lightning." [Online]. Available: https://www.pytorchlightning.ai/

[16] "UCSC Xena," 12 2016. [Online]. Available: https://xenabrowser.net/datapages/?dataset=EB%2B%2BAdjustPANCAN_IlluminaHiSeq_RNASeqV2.geneExp.xena&host=https%3A%2F%2Fpancanatlas.xenahubs.net&removeHub=https%3A%2F%2Fxena.treehouse.gi.ucsc.edu%3A443

[17] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, pp. 232–242, 2016, roLoD: Robust Local Descriptors for Computer Vision 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231215017671

[18] P. Hall, "On kullback-leibler loss and density estimation," *The Annals of Statistics*, vol. 15, no. 4, pp. 1491–1519, 1987. [Online]. Available: http://www.jstor.org/stable/2241687

[19] "Softmax PyTorch 1.9.0 documentation." [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.Softmax.html

[20] A. Gretton, K. Borgwardt, M. Rasch, B. Scholkopf, and A. Smola, "A kernel method for the two-sample-problem." 01 2006, pp. 513–520.

[21] "Delft High Performance Computing Cluster | Grootschalige wetenschappelijke infrastructuur." [Online]. Available: https://onderzoeksfaciliteiten.nl/node/3952

[22] I. Kroskinski, "rp-group-21-ikroskinski," 06 2021. [Online]. Available: https://gitlab.ewi.tudelft.nl/cse3000/2020-2021/rp-group-21/rp-group-21-ikroskinski

[23] C. C. Porter, "De-identified data and third party data mining: the risk of re-identification of personal information," *Shidler JL Com. & Tech.*, vol. 5, p. 1, 2008.

[24] D. C. Drummond, "Replicability is not reproducibility: Nor is it good science," June 2009. [Online]. Available: http://cogprints.org/7691/

[25] A. L. Beam, A. K. Manrai, and M. Ghassemi, "Challenges to the reproducibility of machine learning models in health care," *JAMA*, vol. 323, no. 4, pp. 305–306, 01 2020. [Online]. Available: https://doi.org/10.1001/jama.2019.20866