Delft University of Technology, Bachelor Seminar of Computer Science and Engineering

Benchmarking the hyper-parameter sensitivity of VAE models for cancer treatment

Armin Korkic

25-06-2021

Responsible Professor: Marcel Reinders

Supervisors: Stavros Makrodimitris, Tamim Abdelaal, Mohammed Charrout, Mostafa elTager

Abstract

Cancer has been known as a deadly and complex disease to tackle. By applying machine learning algorithms we hope to improve personalized treatment for cancer patients. These machine learning algorithms are trying to learn a (latent) representation of the input. The problem is that this representation is hard to interpret and to observe the actual connections between the input and the output. That is why these algorithms are considered to behave like a black-box. In this research, a benchmark is conducted to measure how sensitive these algorithms are to changes in their hyper-parameters. The focus of this experiment are different types of variational auto encoders. We will measure how sensitive they are to changes in their: latent space dimension, learning rate and type of optimizer. The models will be trained on a dataset that contains the RNA gene-expression of different types of cancer tissues. To conclude that the optimizer may play the most important role performance wise for VAE models. Using the optimizer Adam and RMSprop results overall in lower reconstruction loss and overall in a more consistent performance.

1 Introduction

Cancer is a deadly disease that costs the lives of thousands of people yearly in the Netherlands alone [1]. While treatment for this disease continues to be very difficult, there are many factors at play when analyzing cancer growth and development. These factors can differ heavily between patients, which makes it difficult to converge treatment into a single methodology. That is why a wide set of possible treatments exists. The patient has many options of different treatments, but it may not be clear for the medical experts which one is the best. Personalized treatment through the use of machine learning could enable us to increase the success of cancer treatment, such as in the case of oral cancer [2]. By entering patient data into the algorithm and a certain type of treatment, it could be possible to predict the outcome of that treatment. One machine learning model that could make these predictions possible are VAEs (Variational Auto-Encoders) [3]. Studies have shown

the promising potential of using VAEs in the field of medicine. They have been used to estimate drug responses of different patients [4].

VAEs are a type of artificial neural network which is used to learn a latent space from the input data. The VAE could use this lower dimensional representation of the data to recreate the original data. VAEs are based on Auto-Encoders. Auto-Encoders can only encode and decode a single data point, but VAEs go beyond a single data point. They learn a latent distribution from the data, which allows them not only to encode and decode existing data, but also they can produce new data. This is one advantage in using VAEs, they are a generative model which gives them much more capabilities than regular Auto-Encoders. It are these generative features that can be used within the field of medicine to make personalized predictions for individual patients. However these algorithms are not easy to grasp or to use due to their black-box nature.

In this research different VAE models will be benchmarked with data retrieved from cancer cells to find how these different VAE models react to hyper-parameter changes. Answering this question gives more clarity to these black-box algorithms and would make it easier for data-scientists to tweak the hyper-parameters of their models to more optimum settings. Research related to this upcoming research is for example a study in the university of Pennsylvania [5]. They have attempted to extract biological relevant latent space from a cancer transcriptomes data set using a VAE. However this paper has only studied the use of a single type of VAE model on this data set and have not measured the performance by varying the hyper-parameters [6]. In this research the same data set will be used, but the performance of four different types of VAE models will be bench marked.

Section Two will contain an explanation of the Methodology used within this research. In Section Three all the Auto-Encoder models are briefly discussed and how they differ form each other. Following in Section Four the experiment setup is highlighted and all the detail needed for the reader to reproduce the experiment. Then in Section Five the results are presented within dot-plots. Section Six will discuss the ethical facets of doing research within the field of machine learning. Section 7 will contain a discussion about the results found in Section 5 and a conclusion is made. Finally in Section 8 a reflection of the previous research is discussed and how future research may proceed.

2 Methodology

2.1 Problem Analysis

The research question is; "How sensitive are different VAE models to the choice of hyper parameters". Due to the non-deterministic nature of machine learning algorithms, a purely analytical approach will be insufficient to better understand these algorithms and therefore an experimental approach will be needed. Machine learning algorithms got their name due to their capabilities of changing and or learning their internal logic and reasoning of certain problems. Their logic changes based on the problem they are solving, on how the model is set-up and the data from which they are learning from. Because of his highly variable internal structure, it is difficult to make a purely a-priori analysis of these models.

To tackle the research question an experiment is there to retrieve empirical data to then perform an analysis a-posteriori of these models. The experiment will include a benchmark different models, namely; VAE, IWAE, Info VAE and LogCosh VAE, from which their results will be documented. The four different models fall under the category of "Auto-Encoders", but they are all similar enough that the same benchmark experiment can be performed on

them all. Afterwards the results for every model will be reviewed and comparisons will be made between the different models. Based on the results conclusions are made on the sensitivity of hyper-parameter changes of the models.

2.2 Method Description

A purely analytical approach of answering the research question is very difficult due to the nature of these algorithms and therefore a more empirical approach is conducted. A benchmark is performed through a gird-search on every model. A selection of hyper parameters and their values is made and used for the gird search. The models are trained with a certain configuration and at the same time their performance is measured through a validation set. In this experiment performance is defined as the reconstruction loss and the amount of epochs until convergence. Training continues until the performance results of the validation set have converged. All the converged results are gathered and plotted on a dot plot to visualize the sensitivity of the results.

3 VAEs

This upcoming section will briefly explain the different VAE models on which research will be conducted on and how these models differ from each other. But first a brief explanation will be given about auto encoders. The list below will contain all the VAE models on which research will be conducted.

- Standard VAE [3]
- IWAE [7]
- InfoVAE [8]
- LogCosh VAE [9]

3.1 Autoencoder

An auto encoder is a type of neural network which exists out of two parts, the encoder and the decoder. The encoder exists out of neurons in which each layer is consequently smaller then the previous layer. The output of the encoder is known as the latent space. The decoder is often very similar to the encoder except that the layer sizes are reversed, so first the smallest size and then the larger ones. However there are models in which the encoder and the decoder differs very much.

The auto encoder functions as follows; the model takes as input a vector, which may represent an image, medical data or other type of data. The vector is fed through the encoder and then through the decoder. The model compares the input and the output with each other and propagates through the model to adjust the weights and biases. The model tries to recreate the original input by processing it into the latent space. And afterwards tries to recreate the original image from this laten space.

The loss of the auto encoder is calculated by taking the mean-squared error of the input and the output vector.

$$loss = MSE(input, output)$$

3.2 VAE

A VAE is a machine learning model which is derived from the machine learning model "Auto Encoders". A VAE performs seemingly similar, however the VAE learns a distribution of the data instead a single latent space, which means that the VAE can generate new unseen samples of the data. An auto encoder can learn to encoder and to decode different types of data, but a VAE could also generate new samples of this data. The VAE uses the reparameterization trick to learn this distribution [3].

The loss of this model is also calculated differently than the loss of the auto-encoder model. The loss is the sum of the mean-squared error of the input and output and the KL divergence multiplied by the KL-weight.

$$loss = MSE(input, output) + KLDweight * KL(N(\mu, \sigma), N(0, 1))$$

3.3 IWAE

The IWAE (Importance Weighted Auto Encoder) is variant of the regular VAE, but tries to improve this model by introducing importance sampling. The model takes into account multiple samples to approximate posterior. Through this the model can learn a more complex latent space. The loss function of this model is given below.

$$loss = \frac{exponent(xi)}{\sum exponent(xj)} * (MSE(input, output) + KLDweight * KL(N(\mu, \sigma), N(0, 1)))$$

3.4 Info VAE

The Info VAE is very similar the regular VAE, but it replaces ELBO (Evidence Lower Bound) criteron for MMD (Maximum Mean Discrepancy). By making this change to the VAE, the Info VAE tries to solve the problem of "Uninformative Latent Code" and "Variance Over-estimation in Feature Space" [10]. The loss function for this model is as follows.

$$recon\_loss = MSE(input, output) + KLDweight * KL(N(\mu, \sigma), N(0, 1)))$$

$$KLD\_loss = KL(N(\mu, \sigma), N(0, 1)))$$

$$bias\_correction = BatchSize * (BatchSize - 1))$$

$$loss = \beta * recon_loss + (1 - \alpha) * KLDweight * KLD\_loss + (\alpha + reg\_weight - 1)/bias\_correction * MMD\_loss$$

3.5 LogCosh VAE

The LogCosh VAE is also very similar to the Standard VAE, however it differs mainly in its loss function. The standard VAE calculated the reconstruction loss by the Mean-Sqaured Error, but the LogCosh VAE replaces it with the log hyperbolic cosine loss. The formula for the loss function is given below.

$$loss = \frac{1}{\alpha} \sum log(cosh(\alpha(input - ouput))) + *KLDweight * KL(N(\mu, \sigma), N(0, 1))$$

4 Experimental Setup

4.1 Experimental Description

The models will be bench marked through the use of a grid search over the following hyper-parameter; the learning rate, latent space and optimizers.

By tweaking the learning rate it can be expected to see changes in how fast the models learns from the data, however when this parameter is set too high this model could "jump" over local optimum points and therefore not reach a good performance rate. But if this learning rate is too low, it might be that the model gets stuck within a local optimum point while unable to escape his current point and never find a global optimum.

The latent space size represents the size of the latent vector of the model. The bigger the size the more dimensions the model has to encode the data. However if this size is too big, the model takes longer to learn and might not even use the full dimensionality of the vector which renders a part of the latent space unused. However, if this number is too small then the model is forced to cram all the data into a small distribution, which could have an affect on the overall performance of the model, since some data needs a minimum of features to be represented sufficiently.

Finally the optimizer serves as a way for the model to adept certain training parameters during training to hope-fully better learn from the data. This adaptability of learning is achieved different by every optimizer. For this benchmark the grid search will be performed with 3 different optimizers; Adam, SGD and RMSprop.

The models will be trained on a RNA Gene-Expression dataset of different cancer tissues [6]. The model will have a single hidden layer of size 200. Then the grid search will be performed over the following set of hyper parameters.

- Learning rate = [0.00001, 0.0001 , 0.001 , 0.01, 0.1]
- Latent space = [10, 20, 30, 50, 100, 200]
- Optimizer = [Adam, SGD, RMSprop]

The data will be split into 2 different sets, a training set and a validation set. The models will be trained on the training set, while the validation set will be untouched by the models and will serve as a way to measure the performance of the models. Training will continue until the performance of the model has stopped increasing with a certain factor. Then the validation loss will be registered and the amount of epochs until convergences will be noted down.

4.2 Experimental Setup: The Data

The data used for training the models derives from the TCGA (The Cancer Genome Atlas) [6]. This data contains information about the RNA gene expression of different cancer tissues, in which the rows represent the cancer tissues and the columns are the RNA genes. There are roughly 11000 samples and every sample has roughly 17000 features. For this experiment the original data set has been processed to contain only the 5000 most variable genes. The reason for this is that some genes are not very interesting to investigate, since they are the same for all cancer or other bodily tissues, this way the algorithms can focus on more important genes while training. Secondly, this would also boost performance time wise since there is less data to process the algorithms can learn much quicker from the data.

This manner of processing the data before training is copied from another research that trains a VAE called 'tybalt' with the same dataset. [11]

After the data has been processed it is almost ready for training. Before training, the data is split between two sets, the training set and the validation set. The training has 90% of the original data while the validation set has 10% of the original data. The two sets have no samples that overlap, so they both contain unique samples.

4.3 Experimental Setup: The model

As previously mentioned this experiment does a grid search over 4 different VAE models over 3 different hyper parameters, however such models have more hyper parameters. Therefore a list will be given that presents the hyper-parameters that have stayed the same through out the grid search.

Hyperparameter	
Input layer size	5000
Amount of Hidden Layers	1
Hidden Layer size	256
Batch size	1000
Scheduler Gamma	0.95
Weight Decay	0.00
Number of Samples (IWAE)	5
MMD Weight (Info VAE)	110
KLD Weight (Info VAE)	-9.0
Reconstruction Weight (Info VAE)	10.5
Gradient Clip Val (Info VAE)	0.8
Alpha (LogCosh VAE)	10.0
Beta (LogCosh VAE)	1.0

Some details worthy of noting that can't be explained within the table given above. All models feed their calculation forward through linear neural networks, afterwards the calculations are normalized into values between '0' and '1'. After the results are normalized, a ReLu (Rectified Linear Unit) activation function is applied to the calculated results. However at the last layer of the decoder, the results are not normalized and instead of the ReLu, a sigmoid activation function is applied to the calculated results.

Worthy of mentioning is also how in this experiment is decided when the models should stop training. The models have been set to train for 100 epochs however, when the validation loss is decreasing and the difference of the reconstruction loss between two epochs is 1% then the algorithm decides to stop learning.

4.4 Experimental Setup: Code and Experiment environment

Most of the code used for this experiment has been borrowed from a public python library called PyTorch-VAE [12]. This library contains different versions of VAE models which can be used for the dataset CelebA, which contains facial images of celebrities [13]. However, since the data doesn't contain images it was important that the implementation of these models within this library had to be changed. The models used convolutional neurons, so

the models were changed to use linear neurons to make calculations. Also the code was set-up to be used with the CelebA dataset, so it was changed that the models could be run on the TCPA cancer gene expression dataset. The library was originally copied and used on the TU Delft gitlab.

The training was done on a Dell laptop with 16gb of RAM, on a GPU with GTX-1050Ti on Windows 10. The code base used for this experiment is written in python 3.

5 Results

In the upcoming section the results of the grid search will be presented. The results are formatted within a dot-plot in which the x-axis represents the latent dimension, the y-axis represents the learning rate, the colour of the circles represents the reconstruction loss of the validation set and the size of the circle represents on which epoch the model is converged. There are 3 dot-plots per model. The 3 dot-plots are there because of the 3 optimizers that are grid-searched per model. The results are presented with short description of noticeable results.

5.1 VAE

Important to note for the Vanilla VAE model results is that when the model is run with RMSprop with a learning rate of 0.1, the model outputs NaN values. Within the dot-plots these values are not presented and a blank square is left within the dot-plots. It seems that the model becomes unstable under large learning rates with a low latent space. Further investigation needs to be done to understand why the model fails at these configurations.

This models performs overall the best with Adam and RMSprop as its optimizer. Also the best performance seems to come from a learning rate of 0.001 for models that use Adam and RMSprop as their optimizer.

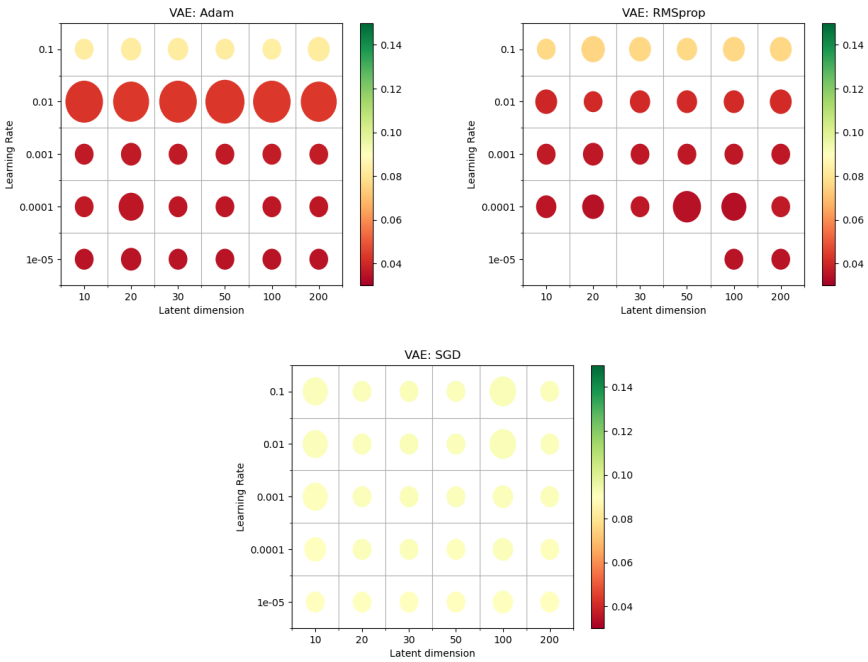

Figure 1: Results of the benchmark of the VAE with the optimizers: "Adam", "RMSprop" and "SGD". The x-axis represents the latent dimensions of the model and the y-axis represents the learning rate. The colour of the circles represent the reconstruction loss of the models and the size of the circles represents the amount of epochs until the model has converged. The largest circles denotes 28 epochs and the smallest circles represents 12 epochs.

5.2 IWAE

A similair phenomena arises for the IWAE model. For low learning rates, the model outputs NaN values for the optimizers Adam & RMSprop, which is represented as blank squares values within the dot-plot. It is assumed that the model performs instable for those values.

Similar observations could be made for this model compared to the previous model. It performs overall the best with Adam & RMSprop as its optimizer. Also the best performance seems to come from a learning rate of 0.001.

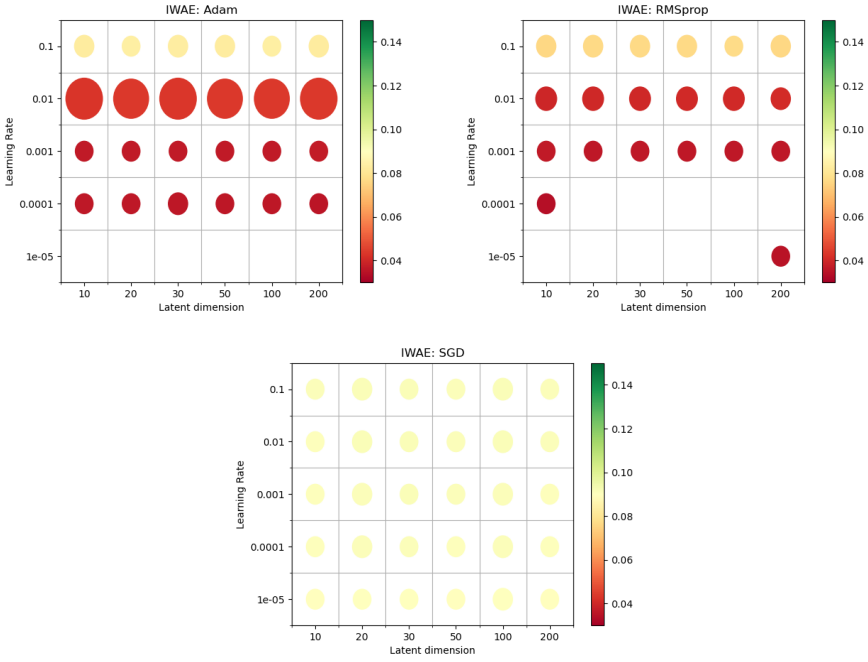

Figure 2: Results of the benchmark of the VAE with the optimizers: "Adam", "RMSprop" and "SGD". The x-axis represents the latent dimensions of the model and the y-axis represents the learning rate. The colour of the circles represent the reconstruction loss of the models and the size of the circles represents the amount of epochs until the model has converged. The largest circles denotes 24 epochs and the smallest circles represents 12 epochs.

5.3 Info VAE

Just as the Vanilla VAE and the IWAE, for some configurations of the Info VAE it returns NaN values. This happens for similar configurations just as with the Vanilla VAE and the IWAE. However there are also blank squares for the Info VAE configurations with a latent space of 200. These blank squares are there because of run time errors of the models. The model uses more RAM then the other models, so much RAM is needed that the GPU on which the experiment was run couldn't provide it. This only occurred on this model with a latent space of 200.

As with the previous model. It performs overall the best with Adam and RMSprop as its optimizer and a learning of 0.001.

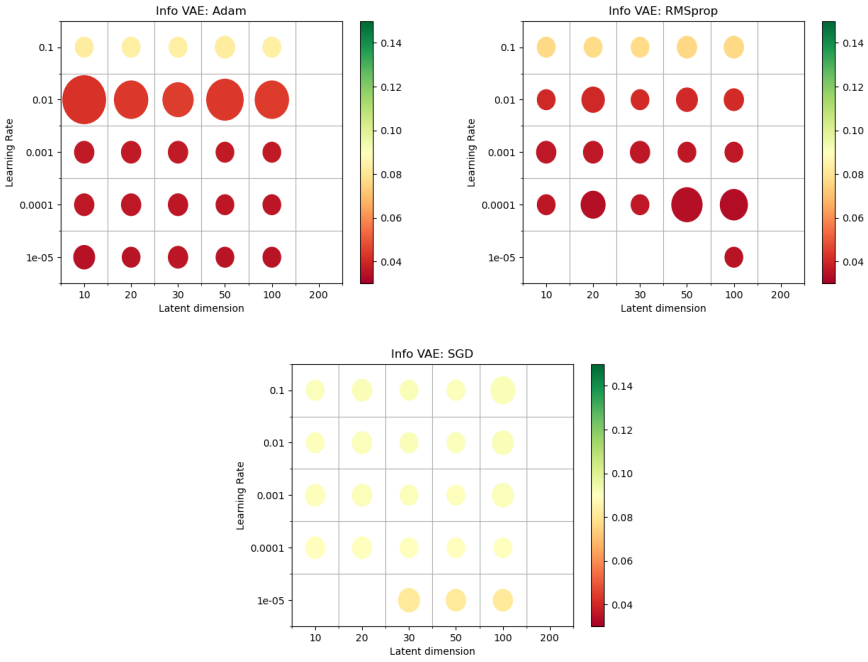

Figure 3: Results of the benchmark of the VAE with the optimizers: "Adam", "RMSprop" and "SGD". The x-axis represents the latent dimensions of the model and the y-axis represents the learning rate. The colour of the circles represent the reconstruction loss of the models and the size of the circles represents the amount of epochs until the model has converged. The largest circles denotes 28 epochs and the smallest circles represents 12 epochs.

5.4 LogCosh VAE

For the LogCosh VAE, similar things occurred as for the other models. For a low learning rate and for a low latent dimension the model returns NaN values with the optimizer RMSprop.

This model performs the best with RMSprop and Adam as optimizer. Also the best learning rate for these models is 0.001.

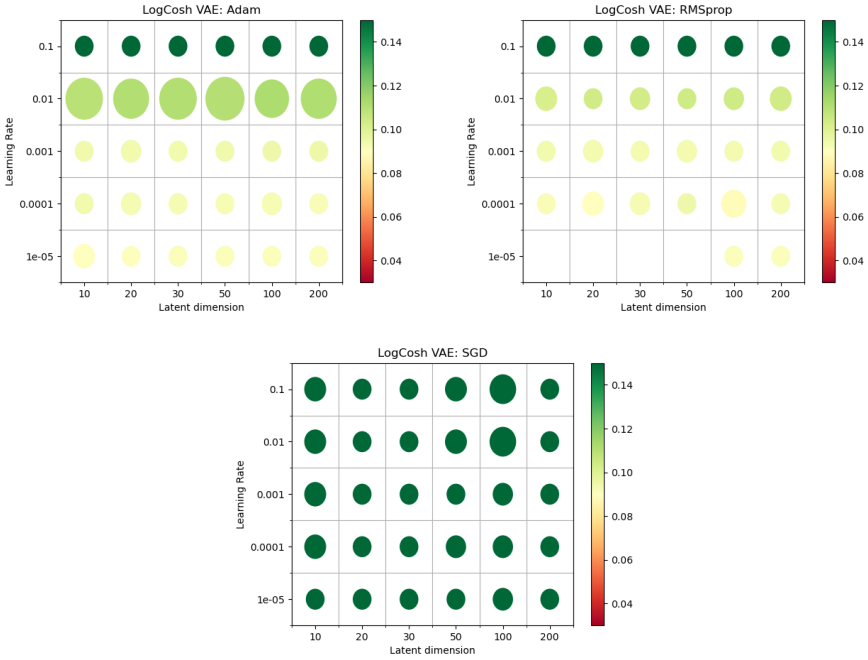

Figure 4: Results of the benchmark of the VAE with the optimizers: "Adam", "RMSprop" and "SGD". The x-axis represents the latent dimensions of the model and the y-axis represents the learning rate. The colour of the circles represent the reconstruction loss of the models and the size of the circles represents the amount of epochs until the model has converged. The largest circles denotes 25 epochs and the smallest circles represents 12 epochs.

6 Responsible Research

There are two important ethical aspects that need to be mentioned for this research. Firstly the medical data that has been used for the benchmark and secondly the reproducibility of the results and the experiments conducted within this research. The upcoming sections will firstly discuss the ethical aspects surrounding the data, such as the privacy of subjects from which the data has been retrieved and the availability of the data for the reproducibility of the experiments. Afterwards the reproducibility aspects of the experiments will be discussed and how the reproducibility of the experiments can be maintained.

As mentioned in previous sections the data that has been used for the benchmark derives from the The Cancer Genome Atlas (TCGA) [6]. It is important to note that this data has been retrieved while considering the protection of the participants, this includes their consent of the use of data and the privacy of the participants. The TCGA has an ethics policy that considers the protection of their participants [14]. It can therefore be concluded that the data that has been used during this experiment is retrieved in an ethical manner. Also the data has been sufficiently anonymized and is publicly accessible. Therefore the dataset protects the identity of the individuals while also maintaining the highest degree of reproducibility.

Following in the heart of critical and public science it is important that researchers describe their experiments as sufficiently as possible, write down all methods that have been used and make a vivid description of the experimental set-up, so that the experiments are better reproducible. This is most definitely the case for the benchmarking of machine learning algorithms, since these algorithms are by nature non-deterministic and therefore results are much harder to reproduce. For every detail that is not mentioned within this report, makes reproducing the experiments only much harder. That is why in this report the reader is not only provided with a high-level overview of the experiments, but also descriptions of all configurations of the models, the language in which the models are programmed, the packages used for the models, on which machines the models were trained, the code base used for the models, etc. All of these details are found within section 4 Experimental Setup. By providing the reader as much information about the experiment, the reader could hopefully reproduce the experiments and come to the same or similar results.

7 Results and Discussion

All results have been presented in the previous section and in this section the results of the benchmark are discussed and a conclusions is drawn out of the results that considers the sensitivity of the models towards different hyper parameters. The results are discussed per hyper-parameter, so firstly the learning rate will be discussed, then the latent dimension and finally the optimizers. At the end a conclusion is drawn about the different hyper-parameters and how well each model has performed.

7.1 Learning Rate

For this experiment the models have been bench marked for a huge learning rate range, with the lowest learning rate being 1e-05 and the highest being 0.1. It can be observed from the dot-plots that the right learning rate plays an important role in the performance of the VAE models. Most models perform the best with a learning rate of 0.01.

7.2 Latent Dimension

Looking at the values of the dot-plots it seems that the Latent Dimension also has some impact on the performance of the models. The overall reconstruction loss tend to be very similar when the latent space is the only varying hyper-parameter. But it seems that sometimes a higher latent dimension results in more epochs until convergences. Which is to be expected from a higher latent dimension.

When readers want to extend this research it would be interesting to see how a lower latent dimension would affect performance. Currently the lowest latent dimension is 10, a benchmark could be extended that includes a latent dimension of 5 or 3.

7.3 Optimizer

It is clear from the dot-plots and their figures that some optimizers tend to perform better then other and require less specific configurations to perform decent. For example, the IWAE model when used with Adam and RMSprop performs decently well with most configurations, while the same model with SGD performs decently much worse. Similar observations can be made when looking at the dot-plots of the other models.

It can be concluded that when using Adam and RMSprop as optimizers, the models are less sensitive to hyper-parameter changes within the learning rate and latent dimension. For future research the grid search could be extended with other optimizers and see how sensitive they are to hyper-parameter changes.

7.4 Conclusion

A few important aspects can be noted down about these models and how they configure their hyper-parameter. Firstly, it was stated before that the learning rate needs to be not too high or not too low, however it seems that all models perform the best with a learning rate of 0.01. However for the latent space most models favour a lower latent space. This is probably due to the data not being very complex and therefore a higher latent space would result in more time needed to train the models.

But the most important hyper-parameter that effects the sensitivity of the models is the optimizer. If the correct optimizer is configured for the models, then the models tends to perform more consistent. The best performance and the least sensitivity comes from the optimizers Adam and RMSprop. In most configuration they tend to perform overall consistent and deliver decent results.

Hereby it can concluded while the correct learning rate and latent dimension can have much effect on the performance of these models, but the most important hyper-parameter sensitivity wise is the optimizer. Configuring the better optimizer for the model results more often in better results then the "correct" learning rate or latent dimension. In this case Adam and RMSprop, outperform and are more consistent then SGD, it can evenly be noted down that RMSprop tends to perform slightly better then Adam.

8 Reflection and Future Work

This section of this report is dedicated as a reflection of the experiment and how future researchers could extend and or build upon this research.

The most obvious manner to extend the current experiment is by extending the set of hyper-parameters for the grid search and or increasing the amount of elements per hyper parameter set. However this manner of extending the experiment would quickly increase the amount of configurations and therefore also increase the amount of time needed for training all the models. Therefore a careful selection should be made of important hyper parameters and set of elements per hyper parameter. An interesting hyper-parameter to add to this grid search is the amount of hidden layers within the network. Intuitively this would increase the amount of training needed for this model, but due to the extra layer the model could learn much more complex patterns within the data and could generate much more realistic and accurate samples of the original data.

Since these models are non-deterministic, it would also be important and interesting to rerun the algorithms and to also use different initialization function to measure their performance. It could be that some models are very much dependent on how their weights initialized while other models might perform much consistent no matter on how they are initialized.

Currently the experiment has been done on 4 different types of auto-encoders: the VAE, IWAE, Info VAE and LogCosh VAE. However there are more types of auto-encoders such as, Beta-VAE, MIWAE, SWAE etc. But experiments could also be extended to differents types of neural network models or even different types of machine learning models. Comparisons made between different types of auto-encoders would be easier to do, but it could be possible that some fruitfull results could be gained by benchmarking the different machine learning models.

Another way to extend the research is by including a different type of dataset. Currently the models are stuck on the TCGA cancer gene expression dataset. But it would be interesting to see how the models would perform on different types of datasets. These datasets could be similar to the current dataset in which the data represent some data about some biological phenomenon. By extending research to different data-sets, conclusions could be made of how different data-sets have what kind of impact on the models

References

[1] Volks Gezondheids Zorg (2019), Ranglijst doodsoorzaken op basis van sterfte, https://www.volksgezondheidenzorg.info/ranglijst/ranglijst-doodsoorzaken-op-basis-van-sterfte

[2] Chu, CS, Lee, NP, Adeoye, J, Thomson, P, Choi, S-W. Machine learning and treatment outcome prediction for oral cancer. J Oral Pathol Med. 2020; 49: 977â 985. https://doi.org/10.1111/jop.13089

[3] Kingma, Diederik & Welling, Max. (2014). Auto-Encoding Variational Bayes.

[4] Ladislav RampÃ¡Å¡ek, Daniel Hidru, Petr Smirnov, Benjamin Haibe-Kains, Anna Goldenberg, Dr.VAE: improving drug response prediction via modeling of drug perturbation effects, Bioinformatics, Volume 35, Issue 19, 1 October 2019, Pages 3743â3751, https://doi.org/10.1093/bioinformatics/btz158

[5] Way, G. P., & Greene, C. S. (2018). Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders. Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing, 23, 80-91.

[6] UCSC Xena. (2016, December 29). Gene Expression RNA seq. https://xenabrowser.net/datapages/?cohort=TCGA%20Pan-Cancer%20(PANCAN)&removeHub=https%3A%2F%2Fxena.treehouse.gi.ucsc.edu%3A443

[7] Burda, Yuri & Grosse, Roger & Salakhutdinov, Ruslan. (2015). Importance Weighted Autoencoders.

[8] Zhao, Shengjia Song, Jiaming Ermon, Stefano. (2017). InfoVAE: Information Maximizing Variational Autoencoders.

[9] Chen, Pengfei Chen, Guangyong Zhang, Shengyu. (2018). Log Hyperbolic Cosine Loss Improves Variational Auto-Encoder

[10] Zhao, Shengjia, A Tutorial on Information Maximizing Variational Autoencoders (InfoVAE). https://ermongroup.github.io/blog/a-tutorial-on-mmd-variational-autoencoders/

[11] A Variational Autoencoder trained on Pan-Cancer Gene Expression, https://github.com/greenelab/tybalt/blob/master/process$_data.ipynb$

[12] PyTorch VAE, https://github.com/AntixK/PyTorch-VAE

[13] CelebA, Large-scale CelebFaces Attributes (CelebA) Dataset, http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

[14] TCGA Ethics & Policies, https://www.cancer.gov/about-nci/organization/ccg/research/structural-genomics/tcga/history/policies