Variational Autoencoder for HIV Drug Resistance Database

Muskaan Ali, Ashleigh Chen, Wenling Xu

STAT 469/563: Machine Learning

Dr. Xuekui Zhang

February 24, 2025

Introduction to the Data

HIV, the human immunodeficiency virus, attacks the body's immune system, which makes it harder for an individual to fight off illnesses. HIV mutates at a high rate, and these mutated viruses easily develop resistance to current drugs. To combat this, we predict resistance to multiple HIV drugs and select the most suitable one (Rhee et al., 2003). The HIV dataset in R contains resistance data for five NRTI drugs (3TC, ABC, AZT, D4T, and DDI), which serve as response variables in the matrix YY. Additionally, 228 mutation variables are predictors, representing genetic variation in the virus that may contribute to drug resistance, given as a matrix XX. In this analysis, we develop a Variational Autoencoder (VAE) model to capture the latent structure of the data, visualize drug resistance patterns, and analyze the reconstruction loss during training.

Tuning Parameters

To evaluate the impact of different tuning parameters on the performance of the VAE, we experimented with various configurations of the dimension of the latent space (start from 10), the batch size, number of epochs, and the number of neurons in the encoder and decoder layers. Initially, we set the neruons to 128 as it is suitable for medium-scale models with 10 dimensions in latent space (DigitalSreeni, 2020). Using this configuration, we trained 5 models with different combinations of epochs (50, 100, 150) and batch sizes (16, 32, 128) to observe their effects on convergence and reconstruction loss. Additionally, to assess the influence on the number of layers (neurons), we trained models with 64 and 256 units, using epoch=100 and batch_size = 32, as these are the intermediate values. These variations allowed us to

explore how increasing and decreasing the model parameters impacts overfitting, generalization, and training stability.

Analysis, Impact, and Interpretation

Encoder Summary:

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 228)]             0
 dense_1 (Dense)             (None, 128)               29312
 dense (Dense)               (None, 20)                2580
 lambda (Lambda)             (None, 10)                0
=================================================================
Total params: 31,892
Trainable params: 31,892
Non-trainable params: 0
```

The input is the XX dataset, which has a dimension of 228. The first hidden layer (dense_1) is a fully connected layer with 128 neurons using the ReLU activation function, containing 29312 parameters. The next layer (dense) represents the latent space, which has 20 dimensions and 2580 parameters. Finally, the lambda layer applies the reparameterization trick, reducing the latent space to 10 dimensions.

Decoder Summary:

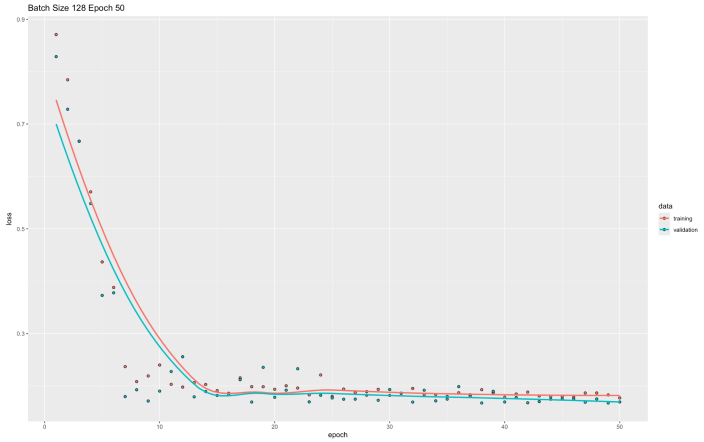
```
_________________________________________________________________
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 10)]              0
 dense_3 (Dense)             (None, 128)               1408
 dense_2 (Dense)             (None, 228)               29412
=================================================================
Total params: 30,820
Trainable params: 30,820
Non-trainable params: 0
```

The input comes from the latent space, referred to as lambda in the encoder summary. The first hidden layer (dense_3) has 128 neurons with the ReLU activation function. The final output layer (dense_2) reconstructs the input with the same 228 dimensions as XX.

Training and Validation Loss Curves:

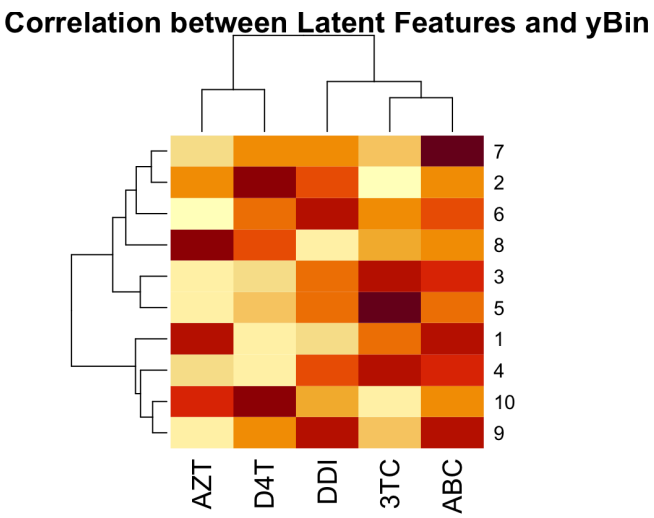
We plotted graphs showing how the loss function evolves during training. The red line represents the training loss while the blue line represents the validation loss. The points indicate the loss values

computed at each epoch. For effective training, both the training and validation loss should start from a high value and decrease over time, and the curves should be smooth without any oscillations (Underfitted, 2022). In our experiments, the loss converged and started from a low value for each configuration of the parameters. However, we determined that when the number of neurons is 128, batch size is 128, and epoch is 50, the model performs better. In this case, the loss started from a higher value and decreased rapidly. However, since the highest loss value remained below 0.9, none of the models were particularly efficient.



Heatmap:

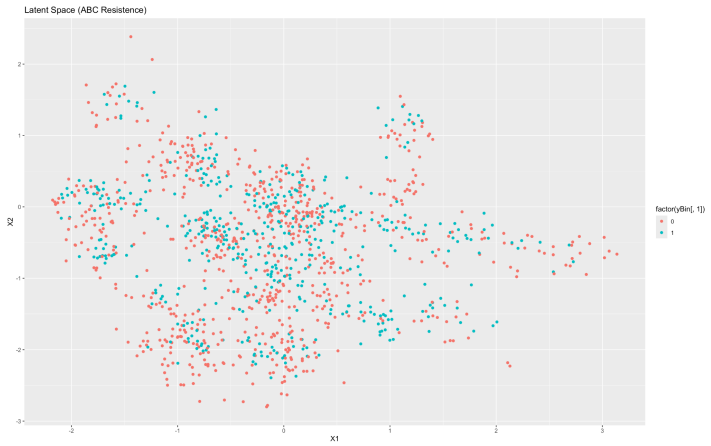
The heatmap demonstrates the correlation between the latent space dimensions and drug resistance, which helps us assess how well the VAE captures meaningful patterns. Each cell's shading represents the correlation strength between a latent dimension and drug resistance. The darker shades indicate a stronger correlation, so the drug ABC shows strong correlations across all dimensions, suggesting that the model successfully learned its resistance structure. However, AZT lacks strong associations, indicating weaker model performance for this drug.



Latent Distribution of the Drugs:

The latent space provides meaningful representations of the drugs so we further analyzed the model's classification effectiveness for each drug. The red points represent non-resistant, while the blue points represent resistance. Effective classification would result in a clear separation of these colours, indicating that the latent space has effectively captured the resistance patterns. The plots for each of the drugs provide the same distribution and same patterns for the spread of the points, so we only provide the ABC drug for this report. As a result of this plot, we can see that the resistant and non-resistant points remain intermixed, lacking distinct clustering, which matches our insights from the loss function graphs.

Then, we tried some different VAE models by changing the dimension of latent space and plotted figures. These figures still show an ineffective performance.



Conclusion

In summary, the VAE was able to extract some meaningful latent representations, and the loss function indicates some good results for certain parameter settings. But overall, our model has not effectively captured the characteristics of the data and struggles with classification. This could be due to our data set size as deep learning models typically require larger datasets while ours contains just 1246 samples. The underlying structure of our data may not be well suited for VAE, suggesting that an alternate model would be more efficient. The next steps to improve this could be to increase the dataset size, change the dataset, or try a different neural network.

References

Altosar, J. (2021, January 25). *Tutorial - what is a variational autoencoder?*. Jaan Lı 李. https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

Barnwal, K. (2024, January 4). Vae Image Classification. Medium. https://medium.com/@indubarnwal752/vae-image-classification-599675a13acf

Bergmann, D., & Stryker, C. (2023, November 19). *What is an autoencoder?*. IBM. https://www.ibm.com/think/topics/autoencoder

Bergmann, D., & Stryker, C. (2024, June). *What is a variational autoencoder?*. IBM. https://www.ibm.com/think/topics/variational-autoencoder

Birla, D. (2019, March 12). *Autoencoders*. Medium. https://medium.com/@birla.deepak26/autoencoders-76bb49ae6a8f

CodeEmporium. (2019, June 17). *Variational Autoencoders - EXPLAINED!*. YouTube. https://www.youtube.com/watch?v=fcvYpzHmhvA&ab_channel=CodeEmporium

DigitalSreeni. (2020, August 31). *154 - Understanding the training and validation loss curves*. [Video] YouTube. https://www.youtube.com/watch?v=p3CcfIjycBA&ab_channel=DigitalSreeni

Doersch, C. (2021). *Tutorial on variational autoencoders.* arXiv preprint arXiv:1606.05908v3. https://arxiv.org/abs/1606.05908

Hafner, D. (n.d.). Building Variational Auto-Encoders in TensorFlow. https://danijar.com/building-variational-auto-encoders-in-tensorflow/

Kingma, D. P., & Welling, M. (2014). *Auto-encoding variational Bayes*. arXiv preprint arXiv:1312.6114v10. https://arxiv.org/abs/1312.6114

Lianlio. (2022, December 11). *Variational Autoencoders, VAEs.* https://blog.csdn.net/weixin_42437114/article/details/125090943

Pykes, K. (2024, August 13). *Variational autoencoders: How they work and why they matter*. DataCamp. https://www.datacamp.com/tutorial/variational-autoencoders

R Core Team. (2019). *heatmap: Draw a heat map.* R Documentation. https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/heatmap

Soo-Yon Rhee, Matthew J. Gonzales, Rami Kantor, Bradley J. Betts, Jaideep Ravela, and Robert W. Shafer (2003) Human immunodeficiency virus reverse transcriptase and protease sequence database. Nucleic Acids Research, 31(1), 298-303. *https://hivdb.stanford.edu/*

Tom Lu, K.-H. (n.d.). | *VAE architecture and reconstruction examples. (A) vae architecture.... | download scientific diagram*. Research Gate. https://www.researchgate.net/figure/VAE-architecture-and-reconstruction-examples-A-VAE-architecture-The-encoder-and-the_fig2_330521293

Underfitted. (2022, September 17). *A Critical Skill People Learn Too LATE: Learning Curves In Machine Learning.* YouTube. https://www.youtube.com/watch?v=nt5DwCuYY5c&ab_channel=Underfitted

Vahdat, Arash, and Jan Kautz. NVAE: *A Deep Hierarchical Variational Autoencoder*. NVIDIA, 2020, Neural Information Processing Systems (NeurIPS) 2020, Vancouver, Canada. Available at: https://github.com/NVlabs/NVAE.

Variational Autoencoder (VAE). (n.d.). https://www.iterate.ai/ai-glossary/what-is-variational-autoencoder-vae#:~:text=VAEs%20are%20used%20in%20a,differ%20from%20a%20traditional%20autoencoder%3F