AIML 425 Assignment 3

James Thompson
Victoria University of Wellington
300680096

I. Introduction

The report documents the implementation and results of Assignment 2 problem 2. I implement a Variational Autoencoder (VAE) and a basic Autoencoder (AE), compare their generative performance, and analyze the AE's latent space information.

II. Theory

Autoencoders and their variations are a type of neural networks architecture that are used for unsupervised learning of an efficient representation of the input data. This section will introduce auto encoders, variational auto encoders as well as some concepts for understanding the latent space and generative performance.

A. Autoencoders (AE)

An Autoencoder [1] has three important parts the encoder f_{enc}, the decoder f_{dec} and the latent layer z (typically with lower dimension than input). The encoder and decoder are neural networks with learnable parameters ϕ and θ.

$$z = f_{enc}(x;\phi) \quad \hat{x} = f_{dec}(z;\theta) \tag{1}$$

The weights are trained based on the reconstruction loss between the input and output data.

$$\mathcal{L}_{AE} = \|x - \hat{x}\|^2 + \beta MMD(z, \mathcal{N}(0, I)) \tag{2}$$

Training with the with $\beta = 0$ means that the model will learn a compressed representation of the data. However there is no reason that the distribution of the latent space will be smooth or continuous (i.e that all squares will be near each other in the latent space). This means that the generative capabilities of the model will be limited. Only certain part of the latent space will correspond to valid images. To fix this we can add a regularizer like Maximum Mean discrepancy [2] that will push the latent space to Gaussian distributions, this is what the second loss function is.

B. Variational Autoencoders (VAE)

In a VAE [3], the encoder does not produce a single point in the latent space but rather parameters for a Gaussian distribution. The goal is to learn the generative model $p_\theta(x)$, which we do by learning $p_\theta(x|z)$ where z is from a known distribution. To do this we also $q_\phi(z|x)$ that gives us the latent distribution given the input.

$$q_\phi(z|x) = \mathcal{N}(z; \mu(x), \text{diag}(\sigma^2(x))) \tag{3}$$

$$\text{where } \mu(x), \sigma(x) = f_{enc}(x;\phi) \text{ and } z \sim q_\phi(z|x)$$
$$\hat{x} = f_{dec}(z;\theta) \tag{4}$$

Like with traditional autoencoders, VAEs are trained to minimize a loss function that consists of two main components: the reconstruction loss and the regularization loss. The reconstruction error can be measured with Binary Cross Entropy and regularization with the Kullback-Leibler (KL) [4] divergence between the learned latent distribution and a prior distribution.

$$\mathcal{L}_{VAE} = \mathcal{L}_{reconstruction} + \beta\mathcal{L}_{KL} \tag{5}$$

Using BCE for reconstruction loss and Gaussian prior:

$$= -\sum_{i=1}^{d}\left[x_i \log(1+\exp(-\hat{x}_i)) + (1-x_i)\log(1+\exp(\hat{x}_i))\right]$$
$$+ \beta\frac{1}{2}\sum_{i=1}^{d}\left(\mu_i^2 + \sigma_i^2 - \log(\sigma_i^2) - 1\right) \tag{6}$$

Where d is the dimensionality of the input data and β is a hyperparameter. This loss function is derived using the Jensen's inequality to maximize the log likelihood of $p_\theta(x)$ [3]. To allow back propagation through the stochastic sampling process, VAEs employ the reparameterization trick.

C. Latent space information

1) Information rate: Information rate is the amount of information that can pass through the latent layer. Assuming that we add noise to the latent space, $z = y + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$, the information rate can be calculated as:

$$R = \sum_i \frac{1}{2}\log_2\left(1 + \frac{\sigma_{i,Z}^2}{\sigma_{i,\epsilon}^2}\right) \tag{7}$$

Where $\sigma_{i,Z}^2$ is the variance of the i-th dimension of the latent space and $\sigma_{i,\epsilon}^2$ is the variance of the noise added to that dimension.

2) Information meaning: The information that the latent space retains can be interpreted in multiple ways. One way is that each dimension of the latent space could represent a specific feature of the input data. We can test for this using correlation between known features and latent dimensions. Or we could test it by varying individual dimension and seeing how the reconstruction changes. In practice each dimension is unlikely to represent a single feature, but rather a combination of features.

D. Generative performance

To understand the generative performance of the models we can use the set of valid images to compare against. This allows us to ask the question; how many of the valid images can the model generate (coverage), how close are the generated images to the nearest valid image (nearest neighbor performance) and how similar is the distribution of generated images to the distribution of valid images (KL divergence). These metrics are defined as follows:

$$\text{coverage} = \frac{|\{x \in \text{images} \mid \min_{\hat{x}\in\text{generated}} d(x,\hat{x}) < \text{thres}\}|}{|\text{images}|}$$

$$\text{nearest neighbor} = \frac{1}{|\text{images}|} \sum_{x_i \in \text{images}} \min_{\hat{x}_j \in \text{generated}} d(x_i, \hat{x}_j)$$

$$\begin{aligned}\text{KL divergence} &\approx \sum_{x_i \in \text{images}} p_{data}(x_i) \log \frac{p_{data}(x_i)}{p_{model}(x_i)} \\ &= \sum_{x_i \in \text{images}} \frac{1}{|images|} \log \frac{1}{q_{counts}(x_i)}\end{aligned}$$

Where $d(x_i, \hat{x}_j)$ is the distance between two images, and the thres is a hyperparameter, and $q_{counts}(x_i)$ is the number of times the model has generated an image close enough to x_i (within the threshold).

III. Experiments

A. Dataset

The dataset used for training both models is a generated dataset of simple shapes (circles, squares, triangles) on a 28x28 canvas. There are 8 sizes($[7, 14]$) which means that there are a total 8002 unique images. The images are white on black. I used 90% for training and 10% for validation.

B. Variational autoencoder

1) setup: The VAE model is two fully connected neural networks. The encoder outputs 2 values (mean and variance) while the decoder outputs a single scalar. The simplicity of the network was chosen due to the relatively simple images. The loss function used was the ELBO loss function with a β of 0.65. A complete list of hyperparameters is shown in table II. The fine tuning of parameters was done through trial and error using the loss curve and generative performance metrics as a guide.

2) generative performance: I use euclidean as the distance metric and the threshold of 3 in the generative metrics. I choose this by looking at sample images and aligning the cutoff and distance metric with my intuition. The results of the generative performance can be found in I along with example generations in figure 1. The model is found to be able to generate just over half of the input images and the nearest neighbor distance is about 2.3 pixels.

C. Autoencoder

1) setup: The autoencoder was built off the same architecture as the VAE, except that the encoder outputs a single value and the loss function is the modified MSE loss function. A complete list of hyperparameters is shown in table II. It took more fine-tuning to get any generative performance out of the model.

2) Information rate: The information rate of the latent space was calculated by getting the encoder outputs from all of the possible images. Then using equation 7 to calculate the information rate. The estimated information was 15.7 bits. This means it uses just less information than a single half float value. It allows us to compress the data from 784 bits (28x28 image) to 15.7 bits which is a ratio of 50:1. There is about 14.2 bits of information required to encode the 4 parameters of the shapes (x, y, size, shape) so the model is using about 1.9 bits of extra information to encode the data.

3) Information meaning: There are two methods that I will use to understand the latent space. The first is to check for correlation between known features (x position, y position, size, shape) and the latent dimensions. The second is to change the latent dimensions and see how the reconstruction changes, I do by both zeroing out and conducting traversals.

Figure 3 shows that there is some correlation between some latent dimensions. Particularly the size has some correlations around 0.6, other latent have some weak correlations 0.3 but nothing strong. Furthermore looking at the distribution of figure 2 we can see that there is no strong correlation between shape and any latent dimension.

Looking at figure 4 shows that some dimensions have a large effect. For example latent dimension 8 can be seen to reduce the size of the shape as it increase. We can see in most of the dimensions that if you move too far away from the mean (0) then the generations become meaningless, this shows how only a small part of the latent space is understood by the AE model.

D. Autoencoder vs Variational autoencoder

Using the metrics discussed above we can measure the generative performance of both models quantitatively.

TABLE I
Comparison of performance between the VAE and AE models.

Metric	VAE	AE
Avg. Nearest Neighbor Distance	2.2377	3.2141
Coverage (%)	57.1%	43.7%
D_kl (bits)	8.4022	8.2702
BCE Error (bits)	519.8196	607.9106

We can see in I that the VAE model outperforms the AE models on almost all the metrics. It has noticeably lower average nearest neighbor and higher coverage. Both models however still fail to capture the full distribution of the data missing about 40% of the valid images.

IV. Conclusion

I trained two different models a VAE and an AE on a simple dataset of shapes. Both models were able to effectively compress the data. The VAE model had better generative performance. An investigation of the latent space of the AE model showed some latent dimensions have correlated effects to known features. I would expect that more complex models and hyperparameter search could yield better results.

Statement

The code and report of the Assignment was solely completed by myself (Thompson James). The complete source code can be found here https://gitea.james-server.duckdns.org/james/AIML425_assignment_3, with a link to a colab notebook found here: https://colab.research.google.com/github/1jamesthompson1/AIML425_assignment_3/blob/main/main.ipynb. A complete run through of the notebook takes about 10 minutes on a GPU enabled machine.

I have kept the appendix as concise as possible, however the code is setup to produce many more figures and tables that can be used to understand the models.

I completed my work using the following tools:

- **Jupyter Notebook [5] and JupyterText [6]:** For interactive development and hosting.
- **LaTeX**: For writing the report.
- **VSCode [7]:** As IDE, with Copilot to help with plotting and debugging.
- **JAX [8] and Flax [9]:** For implementing the neural network and training logic.
- **Matplotlib [10] and Pandas [11]:** For data visualization and management.

References

[1] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[2] A. Gretton, K. Borgwardt, M. J. Rasch, B. Scholkopf, and A. J. Smola, "A Kernel Method for the Two-Sample Problem," May 2008.

[3] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," Dec. 2022.

[4] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, Mar. 1951.

[5] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, "Jupyter Notebooks – a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Schmidt, Eds. IOS Press, 2016, pp. 87–90.

[6] M. Wouts, "Mwouts/jupytext," Aug. 2025.

[7] "Microsoft/vscode," Microsoft, Jul. 2025.

[8] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: Composable transformations of Python+NumPy programs," 2018.

[9] J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, and M. van Zee, "Flax: A neural network library and ecosystem for JAX," 2024.

[10] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[11] The pandas development team, "Pandas-dev/pandas: Pandas."

Appendix

TABLE II
HYPERPARAMETERS USED FOR TRAINING THE VAE AND AE MODELS.

Hyperparameter	**VAE**	**AE**
Latent dimension	32	10
Encoder architecture	[1000, 1000, 1000, 500]	[1000, 1000, 500]
Decoder architecture	[500, 1000, 1000, 1000]	[500, 1000, 1000]
Optimizer	Adam(lr=0.001)	Adam(lr=0.001)
Minibatch size	512	64
Number of epochs	1000	500
Regularization weight	0.65	0.1
Regularization parameters	NA	sigma=$[0.5, 1, 3, 5]$
Dropout	Yes (p=0.1)	Yes (p=0.1)
Layer norm	No	No
Latent space noise	No	Yes ($\sigma = 0.2$)

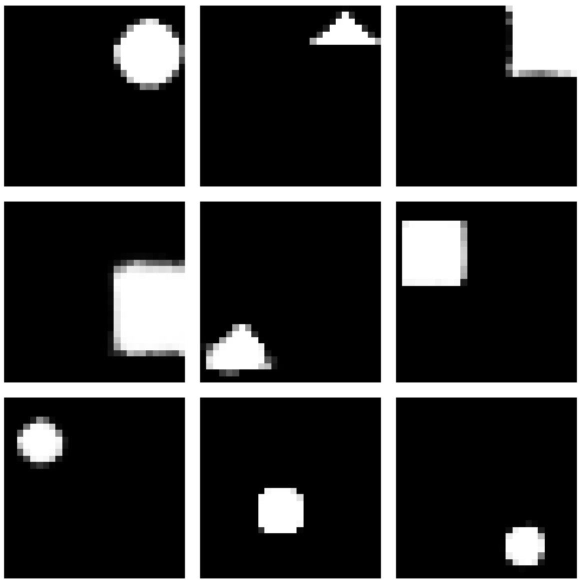

Fig. 1. Samples generated from the trained VAE model. The model is able to generate a variety of shapes, sizes, and positions.

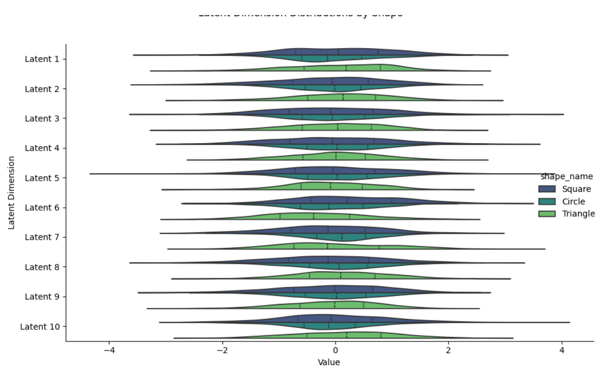

Fig. 2. Distribution of latent dimensions grouped by shape for the AE model. No strong correlation between shape and any latent dimension is visible.

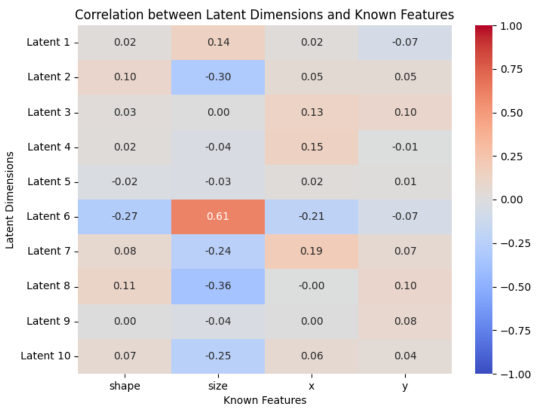

Fig. 3. Correlation between known features and latent dimensions for the AE model. Some strong correlation is visible, particularly for y position.

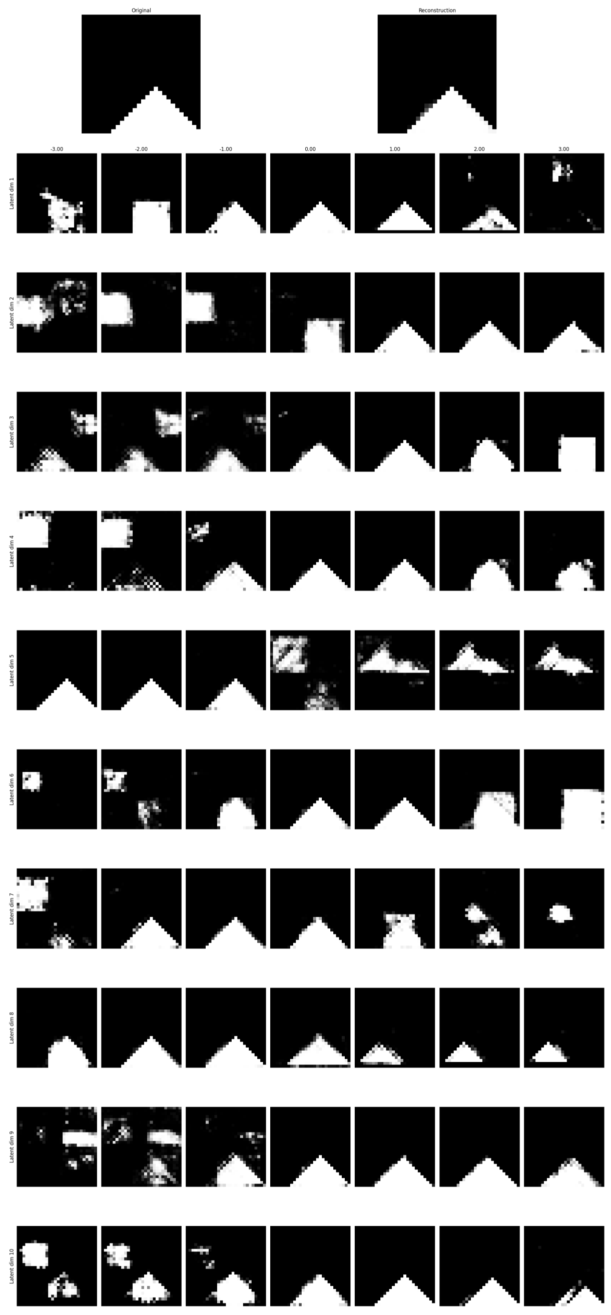

Fig. 4. Latent space traversals for the AE model. Each row shows the effect of varying one latent dimension from -3 to 3 while keeping the others constant.