**Link to the GitHub:** [**https://github.com/JasonMullen/Task-4-A.I./tree/main**](https://github.com/JasonMullen/Task-4-A.I./tree/main)

**Figure1:**

**A**: A sample Variational Auto-encoder. The VAE contains one encoder and one decoder part. Encoder starts from x,h and ends in z=(σ + μ). [(σ + μ) learns latent representation or key features of the images]. Decoder starts from z=(σ + μ) to h2 and ends in x2. Decoder utilizes learned important representation from z=(σ + μ) and tries to regenerate the image in x2.

**B**: A sample normal auto-encoder. A normal autoencoder contains only a fully connected layer z instead of a pair of layers (σ + μ) to learn the hidden representation.

In this project we will use the MNIST computer vision digit dataset and experiment with auto-encoders such as Variational Auto-encoder(VAE) and simple autoencoder. The Jupyter notebook provided contains a VAE architecture and process of downloading the dataset.

**Task 4 Subtasks:**

**1.** Learn latent features from the digit dataset. Use the model given in reference [1]. Perform the following tasks:

**a.** (\*\*) The given model has a three layer architecture for each encoder and decoder part. Can you modify the architecture into a four layer format. In this task, you need to convert encoder part into (x, h1, h2, z=(σ + μ) )= (784\*400\*200\*20) and decoder part into (z=(σ + μ), h3, h4, x2 )= (20, 200, 400, 784). Finally you need to compare the results based on their:

**i.** Optimal loss after the model is fully trained, and

**ii.** Visually inspecting the output they generate using the images they generate and reconstruct. You can use plot\_generation() and plot\_reconstruction() function from the notebook.

Based on optimal loss and visual inspection write down your opinion which model is better and also try to give an explanation why a model is giving good performance over another.

**b.** (\*) Take the base three layer architecture and check the performance of the model for six different configuration, where h\_dim and z\_dim is changed into following patterns: [(400,20), (400, 10), (400, 30), (300,20), (300, 30), (300,30)](Note: First one is the base architecture). Perform the same type of comparison you have done in task **a** using optimal loss of the model and visual inspection and write down your opinion which model is better and also try to give an explanation why a model is giving good performance over another.

**c.** (\*\*) Take the base three layer architecture and convert it into a normal auto-encoder (figure 1.b)[2], e.g. replace z\_dim such a way that it will be single layer. As noraml autoencoder and variational auto-encoder have very different way of loss calculation, you need to modify loss function too. Now Perform the same type of comparison you have done in task **a** using optimal loss of the model and visual inspection and down your opinion which model is better and also try to give an explanation why a model is giving good performance over another.

**Analysis:**

Data preparation: The code uses the MNIST dataset, which consists of handwritten digit images. The dataset is loaded using torchvision.datasets.MNIST and transformed into tensors using torchvision.transforms.ToTensor(). A DataLoader is created for both the training and test sets.

VAE model definition: The VAE class defines the architecture of the VAE model, with separate encoder and decoder components. The encoder maps input images to a latent space, while the decoder reconstructs images from the latent space. The VAE model can be modified to have different numbers of layers and dimensions for the hidden and latent layers.

Training the model: The VAE model is trained using the Adam optimizer and a combination of reconstruction loss (binary cross-entropy) and KL divergence for the loss function. The code trains the VAE model for a predefined number of epochs, printing the reconstruction loss and KL divergence for each step.

Model comparison: The code compares the performance of different VAE architectures, including 3-layer and 4-layer models, as well as models with different hidden and latent layer dimensions. The comparison is done based on the optimal loss after the models are fully trained and by visually inspecting the generated and reconstructed images using plot\_generation() and plot\_reconstruction() functions.

Autoencoder implementation: In addition to VAE models, the code also includes the implementation of a 3-layer autoencoder as a non-variational counterpart. The autoencoder model is trained using a different loss function, focusing solely on the reconstruction loss. The performance of the autoencoder is compared with the VAE models based on optimal loss and visual inspection of generated and reconstructed images.

The analysis of the code shows that the 4-layer VAE model performs slightly better than the 3-layer model in terms of optimal loss. For 3-layer VAE models with different hidden and latent layer dimensions, models with larger hidden layers and more compact latent spaces perform better. When comparing the VAE and autoencoder models, the autoencoder has a lower optimal loss but produces less realistic generated images.

Overall, the code provides a comprehensive analysis of the impact of architecture and hyperparameter choices on the performance of VAE models. The results can be used to guide the selection of appropriate model architectures and hyperparameters for future applications of VAEs in image generation, reconstruction, and unsupervised learning tasks.

**Report:**

Title: A Comparative Study of Variational Autoencoders (VAEs) with Different Architectures and Hyperparameters

1. Introduction

Variational Autoencoders (VAEs) are a powerful class of deep generative models that have gained significant attention recently. They have been used for various applications such as image generation, image reconstruction, and unsupervised learning. This project aims to investigate the impact of different architectures and hyperparameters on the performance of VAEs. Specifically, we will compare the models based on their optimal loss and visual inspection of the generated and reconstructed images. In this study, we consider the following configurations:

* 3-layer VAE (baseline) with h\_dim=400 and z\_dim=20
* 4-layer VAE with h\_dim=400, h2\_dim=200, and z\_dim=20
* 3-layer VAEs with different combinations of h\_dim and z\_dim [(400,20), (400, 10), (400, 30), (300,20), (300, 30), (300,30)]
* A 3-layer autoencoder (AE) as a non-variational counterpart

1. Methodology

We begin by implementing the baseline 3-layer VAE model and training it using the MNIST dataset. We then modify the architecture to create a 4-layer VAE model and a 3-layer autoencoder. We also create six different 3-layer VAE models with various h\_dim and z\_dim configurations. To compare the performance of these models, we use the following criteria:

* Optimal loss after the model is fully trained
* Visually inspecting the generated images using the plot\_generation() function
* Visually inspecting the reconstructed images using the plot\_reconstruction() function

1. Results

3.1 Comparison of 3-layer and 4-layer VAEs

The optimal loss of the 4-layer VAE model is slightly lower than that of the 3-layer model, indicating better performance in reconstruction and KL divergence. Visually, the generated and reconstructed images produced by the 4-layer VAE are of comparable quality to those of the 3-layer model. Furthermore, the additional hidden layer in the 4-layer VAE provides a more expressive model that can learn a better representation of the data.

3.2 Comparison of 3-layer VAEs with Different h\_dim and z\_dim Configurations

Upon comparing the six different 3-layer VAE models, we observe that the models with higher h\_dim and lower z\_dim generally perform better in an optimal loss. This suggests that a more extensive hidden layer and a more compact latent space improve the model's ability to learn the data distribution effectively. Visually, the generated and reconstructed images for these models are of similar quality, with no significant differences in appearance.

3.3 Comparison of 3-layer VAE and 3-layer Autoencoder

When comparing the 3-layer VAE to the 3-layer autoencoder, we notice that the optimal loss for the autoencoder is lower than that of the VAE. This can be attributed to the autoencoder not having a KL divergence term in its loss function, focusing solely on reconstruction. Regarding visual inspection, the autoencoder-generated images appear blurrier and less realistic compared to the VAE. The reconstructed images, however, are of comparable quality.

1. Conclusion

In conclusion, our comparative study of VAEs with different architectures and hyperparameters has revealed some interesting insights. The 4-layer VAE model performs slightly better than the 3-layer model, suggesting that adding more layers can improve the model's expressiveness. In addition, 3-layer VAEs with larger h\_dim

**References:**

1. <https://gxithub.com/dataflowr/notebooks/blob/master/HW3/VAE_clustering_empty.ipynb>

2. <https://www.analyticsvidhya.com/blog/2021/06/complete-guide-on-how-to-use-autoencoders-in-python/>

3. \_\_Source: I used Chat GPT to help with with assignment\_\_