Week 7 Assignment: Improved Digit Generation with VAEs

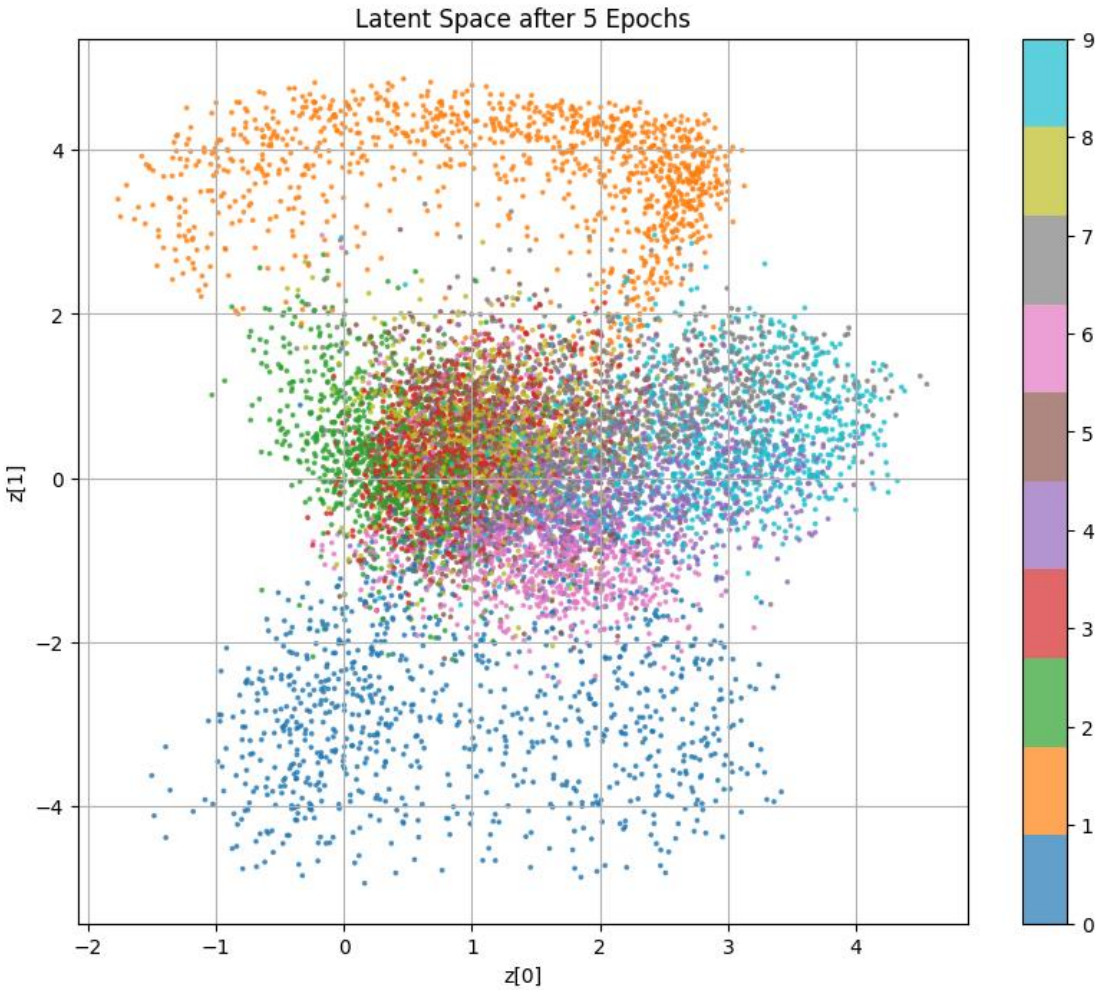
Candidate: Sneha Santha Prabakar

Coding platform: Google Collab

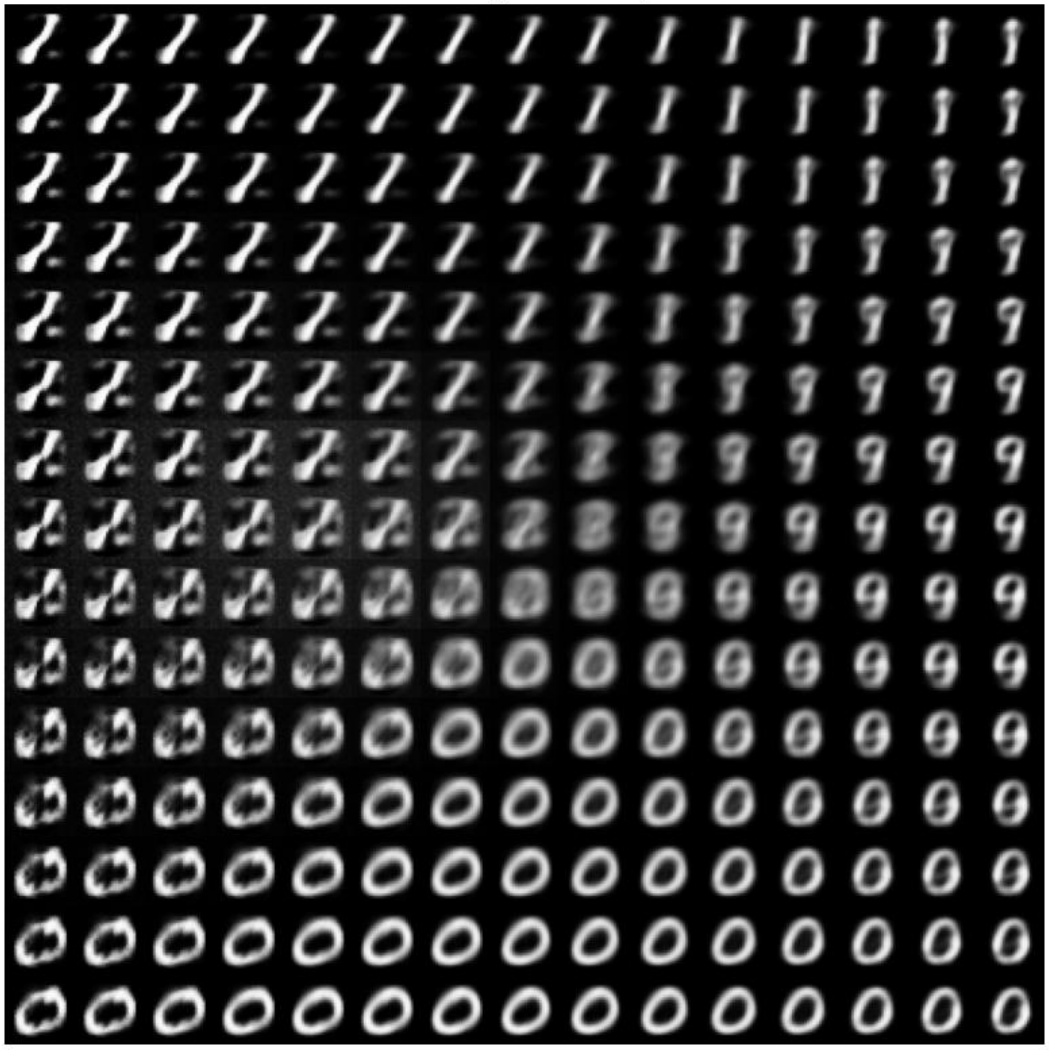
Please Note:

The python notebook included in the submission contains all the code and steps mentioned in the grading criteria. This PDF document only provides the output plots for different epochs and the answer to questions mentioned in the assignment instructions. Please refer to the python notebook for the execution steps.

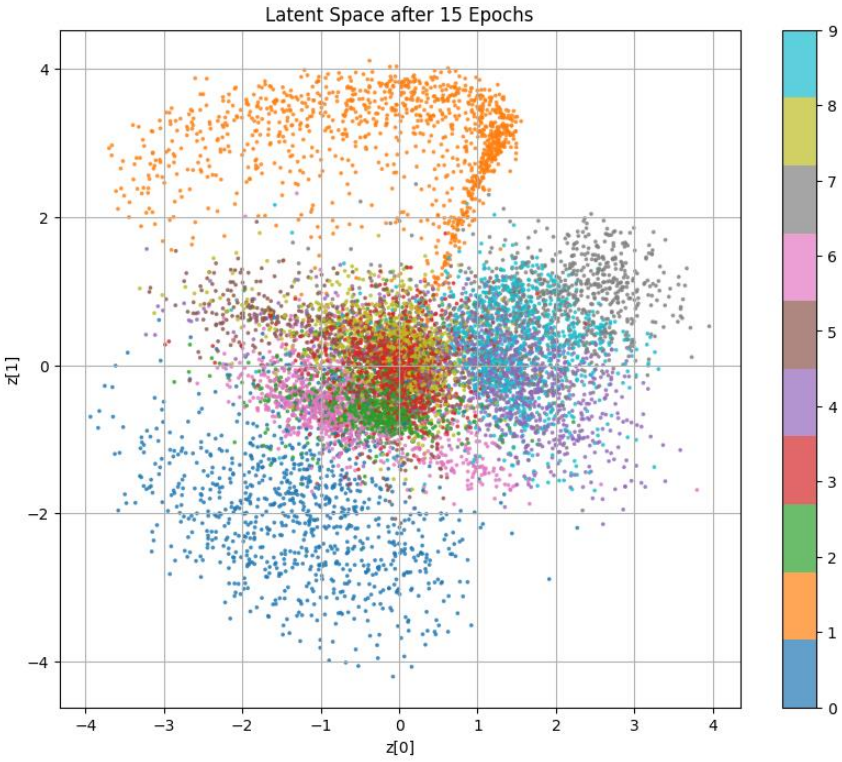
Epochs = 5

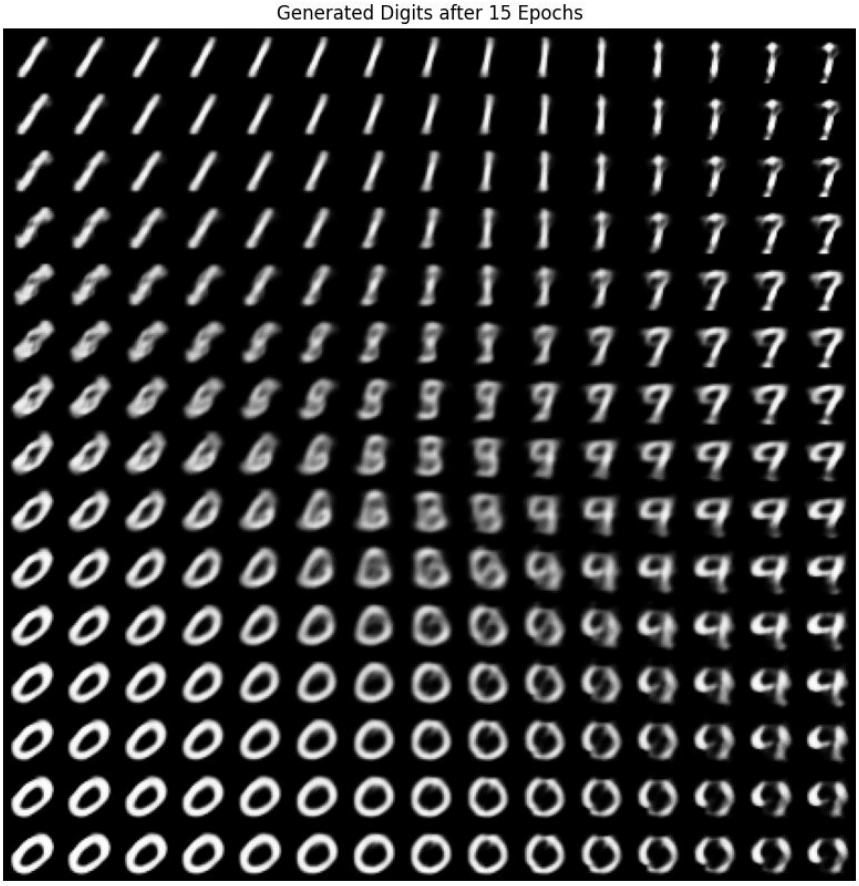


Generated Digits after 5 Epochs

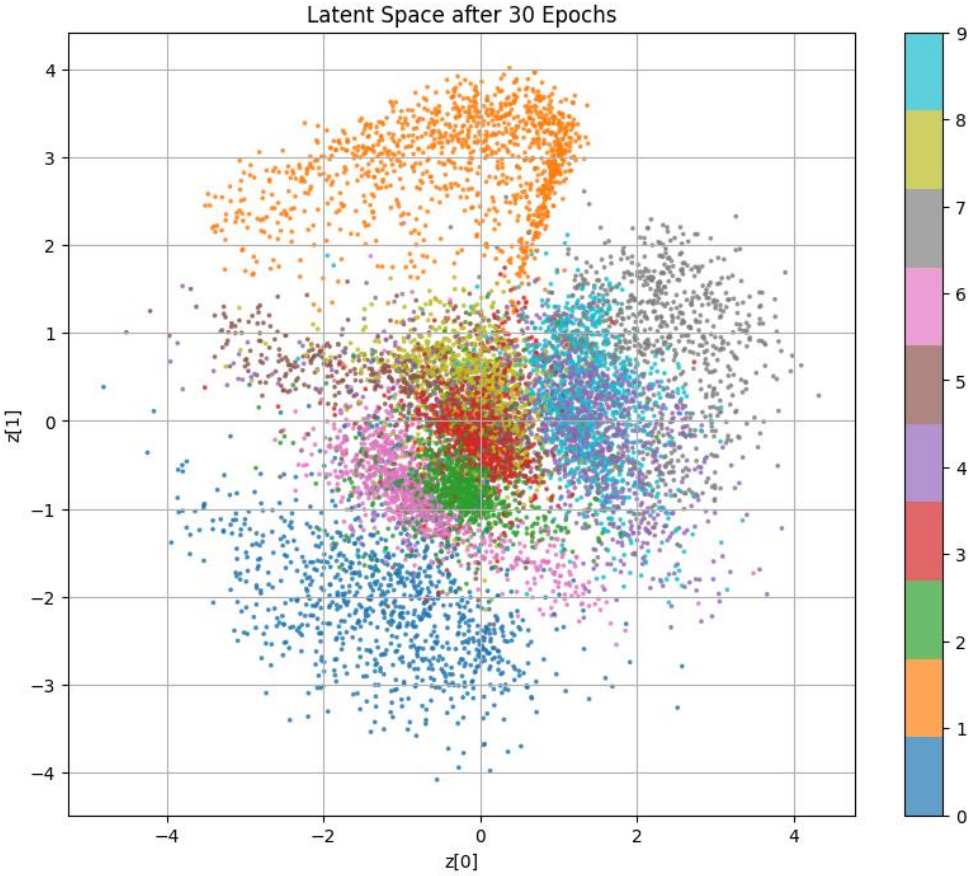


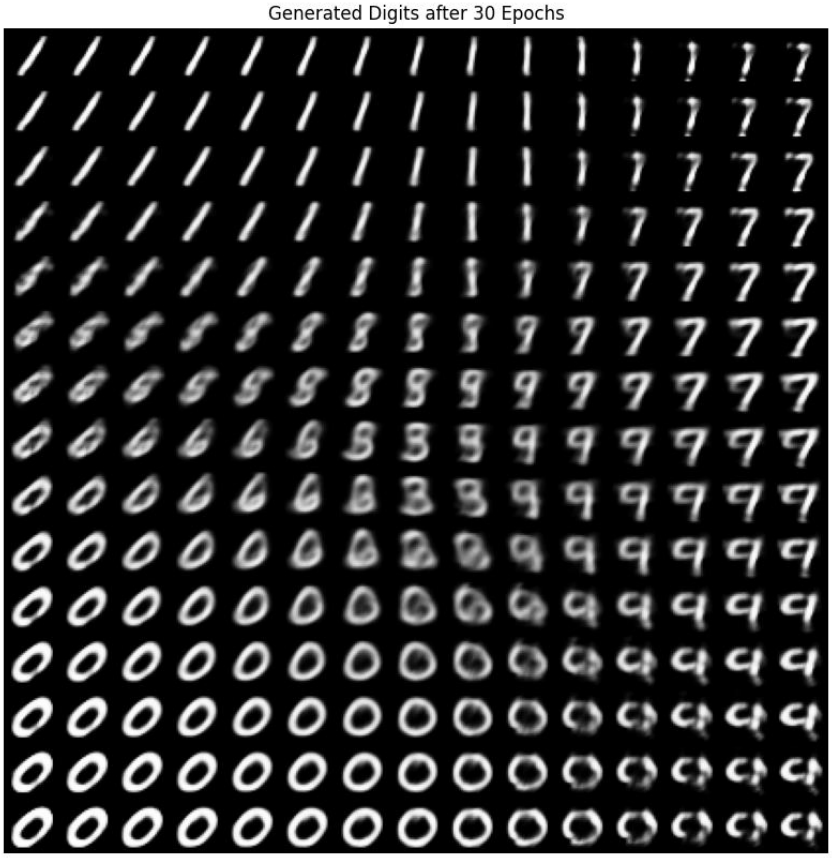
Epochs = 15:


Latent Space after 15 Epochs
z[0]
z[1]


Generated Digits after 15 Epochs

Epochs = 30:


Latent Space after 30 Epochs
z[0]
z[1]


Generated Digits after 30 Epochs

Question 1: Briefly describe how the latent space has changed with the number of epochs (Approx. 100 words)

At 5 epochs, the latent space was highly entangled, with digits like ‘3’, ‘5’, and ‘8’ overlapping in a dense central cluster. The model had not yet learned to separate digit classes. By 15 epochs, some structuring began to emerge: digits such as ‘0’, ‘1’, and ‘6’ formed more compact groupings, although overlap was still common - for example, ‘4’ and ‘9’ were often mixed. At 30 epochs, the clusters became tighter for frequently seen digits like ‘0’, ‘1’, and ‘7’, showing improved but still partial separation. Less frequent or harder-to-distinguish digits like ‘2’ and ‘5’ remained intermixed with others.

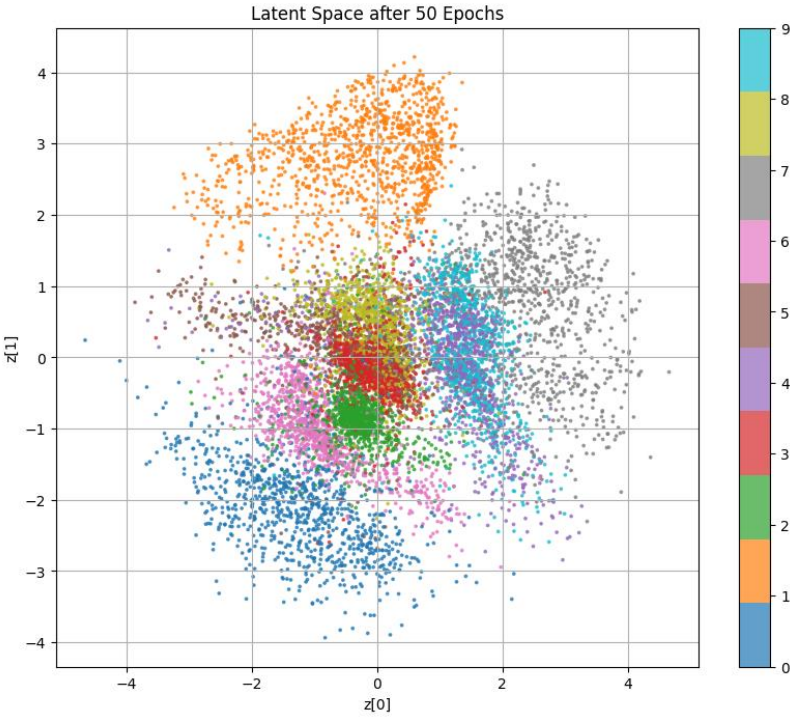
Question 2: Briefly describe how the quality of generated digits has changed with the number of epochs (Approx. 100 words)

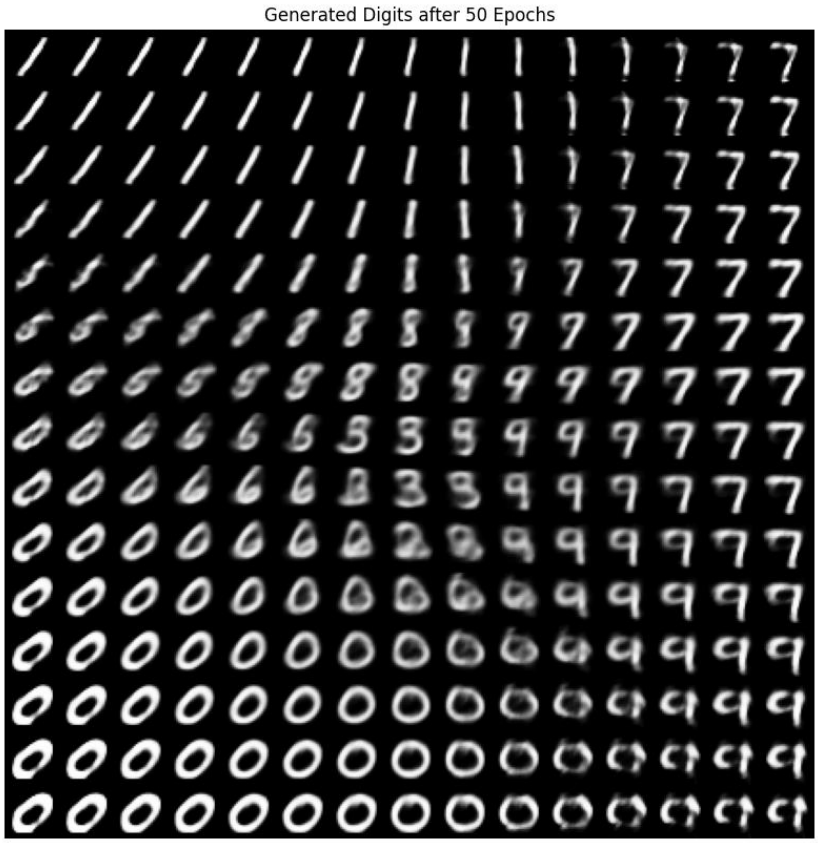
At 5 epochs, the generated digits were blurry and often unrecognizable, with many resembling incomplete strokes or generic vertical bars - for example, many samples resembled distorted versions of ‘1’ or ‘7’. By 15 epochs, accuracy improved: digits like ‘0’, ‘1’, and ‘8’ appeared clearer and more distinguishable. However, diversity remained limited - digits like ‘3’, ‘4’, and ‘2’ were rarely generated or poorly differentiated from ‘5’ or ‘9’. At 30 epochs, accuracy for digits like ‘0’ and ‘9’ was high, with clean loops and consistent shapes. Yet, the model still lacked variety, often generating repetitive digits, indicating mode collapse and limited generalization.

Question 3: What would happen if you try 50 or 100 epochs? Does the model get better or worse?

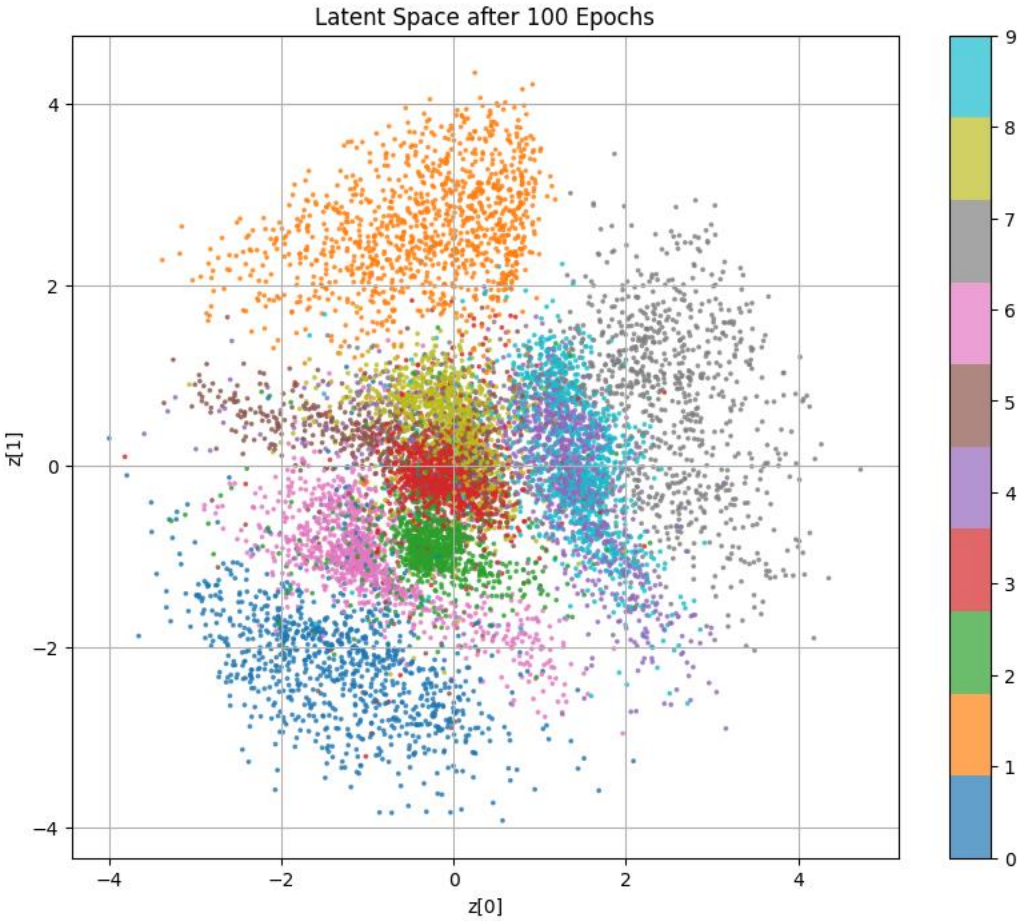
Given below is the visualization of latent space and the generated digits for epochs = 50 and 100, followed by the explanation:

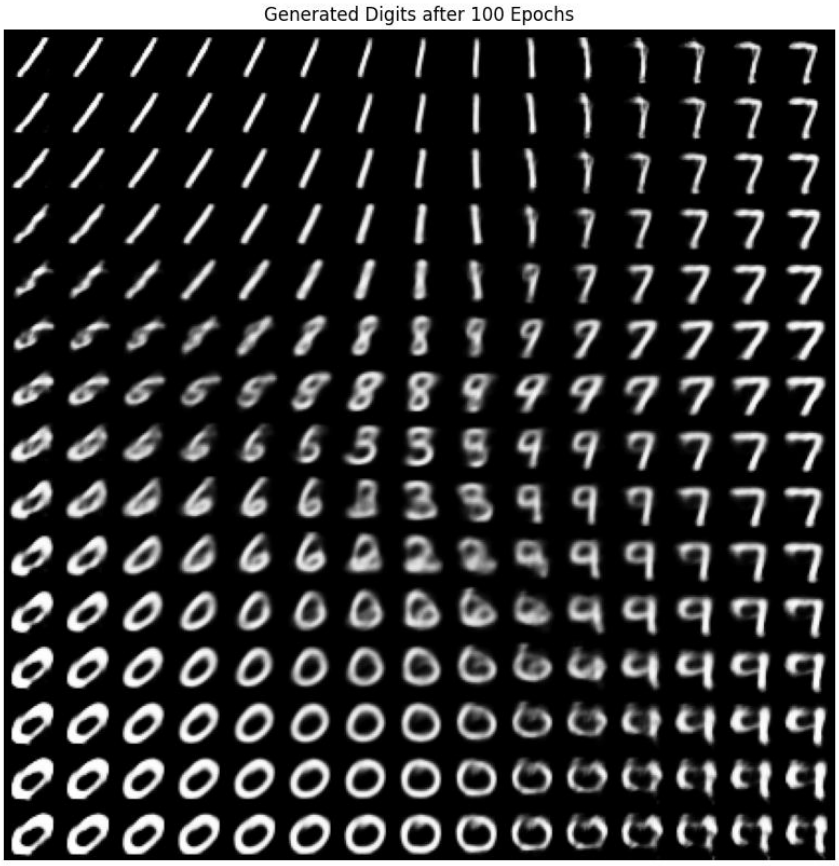
Epoch = 50


Latent Space after 50 Epochs
z[0]
z[1]


Generated Digits after 50 Epochs

Epoch = 100


Latent Space after 100 Epochs
z[0]
z[1]
4
2
0
−2
−4
9
8
7
6
5
4
3
2
1
0


Generated Digits after 100 Epochs

Training the model for 50 and 100 epochs leads to **improved digit clarity** and **greater reconstruction accuracy**, particularly for digits the model was already learning well - such as ‘0’, ‘1’, ‘7’, and ‘9’. This improvement occurs because additional epochs give the model more time to minimize reconstruction loss and fine-tune its weights, allowing it to more confidently generate well-learned patterns. For example, by epoch 100, the loops in digits like ‘0’ and ‘9’ become smoother, and vertical strokes in digits like ‘1’ and ‘7’ appear more consistent and noise-free.

However, **this improvement comes with trade-offs**. The model starts to **overfit** to certain digit types that it sees more frequently or finds easier to reconstruct. As a result, less frequent or structurally complex digits like ‘3’, ‘4’, ‘6’, and ‘5’ are either poorly represented or completely missing from the generated grid. This phenomenon, known as **mode collapse**, reflects a narrowing of the model’s generative diversity - many latent vectors begin mapping to the same output classes, especially ‘1’ and ‘0’.

Moreover, the **2D latent space itself becomes a bottleneck**. With limited capacity to encode complex variations among 10 digit classes, the model defaults to confidently producing high-frequency digits while ignoring nuanced representations of less common ones. While clarity improves with extended training, overall performance worsens in terms of **diversity** and **class differentiation**. Thus, the model gets better in a narrow sense (accuracy of a few digits) but worse in its ability to represent the full MNIST dataset fairly and evenly.