

ASSIGNMENT 1
Prof. [Onur Barut](#)


Name: Prathyaksh Nilson

E-mail: pnilsonparamashiva@clarku.edu

PART 1 & 2

The implementation part of 1 and 2 are done one google collab and the code is also pushed to Github. Both the links are available below.

Github Link: <https://github.com/pratx08/Gen-AI/tree/main/Assignment-1>

Collab Link:  Gen AI Assignment 1.ipynb

PART - 3

10. The main differences in training goals between the classification and generation are,
Logistic Regression: It learns the direct boundaries in the classes. The main objective of the model is to minimize the classification error. If we talk about the training then its stable and converges reliably.

GAN: Here it basically learns to model the entire data distribution to generate new samples. The main objective here is to balance between the generator and discriminator. Talking about the training, it's a bit unstable.

Why the generative model training is difficult is because there are two networks that are trained simultaneously which leads to loss of gradients, oscillations and also increases the chances of risk of mode collapse. This also makes the training unpredictable.

11.

Training Logistic Regression with fake data generated by GAN

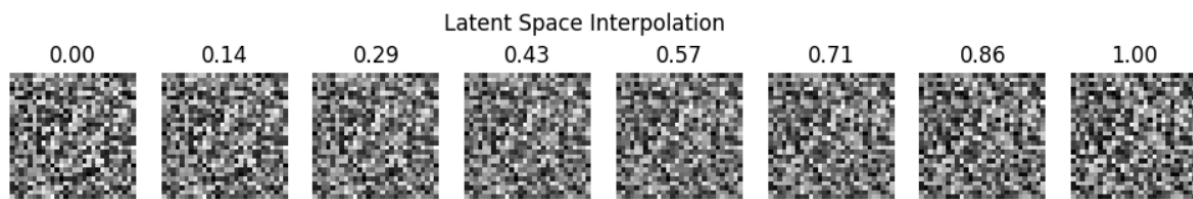
```
LogReg trained on GAN data, tested on real test set  
Accuracy: 0.0816
```

Actual data:

```
Logistic Regression Accuracy: 0.8432
```

When Logistic Regression was trained on real Fashion-MNIST, it reached about 84% accuracy. But when trained only on GAN-generated fake data, accuracy dropped to about 8%, close to random guessing. This shows a large domain gap, where our GAN's synthetic data is not useful as a substitute for real training data.

12.



In the latent space interpolation experiment, we moved step by step between two random noise vectors and generated images. The outputs changed gradually but still looked noisy and unclear, showing that our simple GAN did not learn smooth or meaningful transitions in the latent space.

13.

Went through the citing at <https://arxiv.org/abs/1406.2661>

So as per the experiment from unstable training, oscillating losses, and mode collapse. We observed the same issues in our experiments. This concludes why the GAN is unstable in nature.

In our assignment as well the GAN showed unstable behavior. The losses kept going up and down instead of settling, some generated images looked too noisy, and in the mode collapse test many outputs looked almost the same. These signs match the known instability problems of GAN training

PART - 4

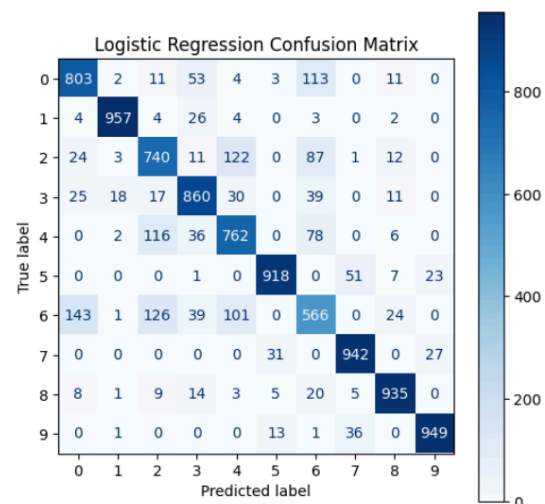
14. Visualisations

Logistic regression

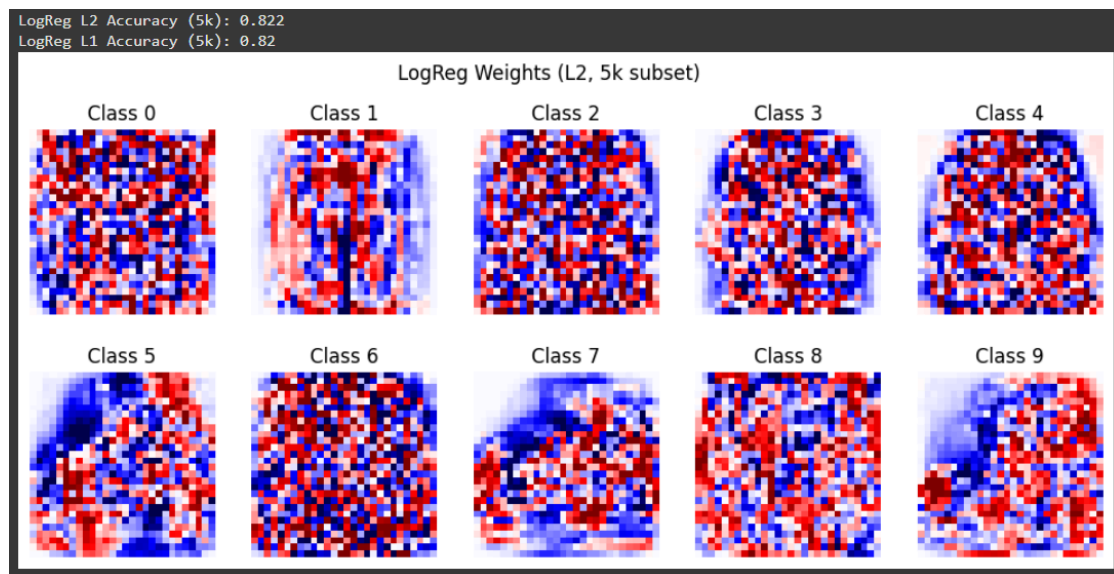
- Confusion Matrix

Logistic Regression Accuracy: 0.8432

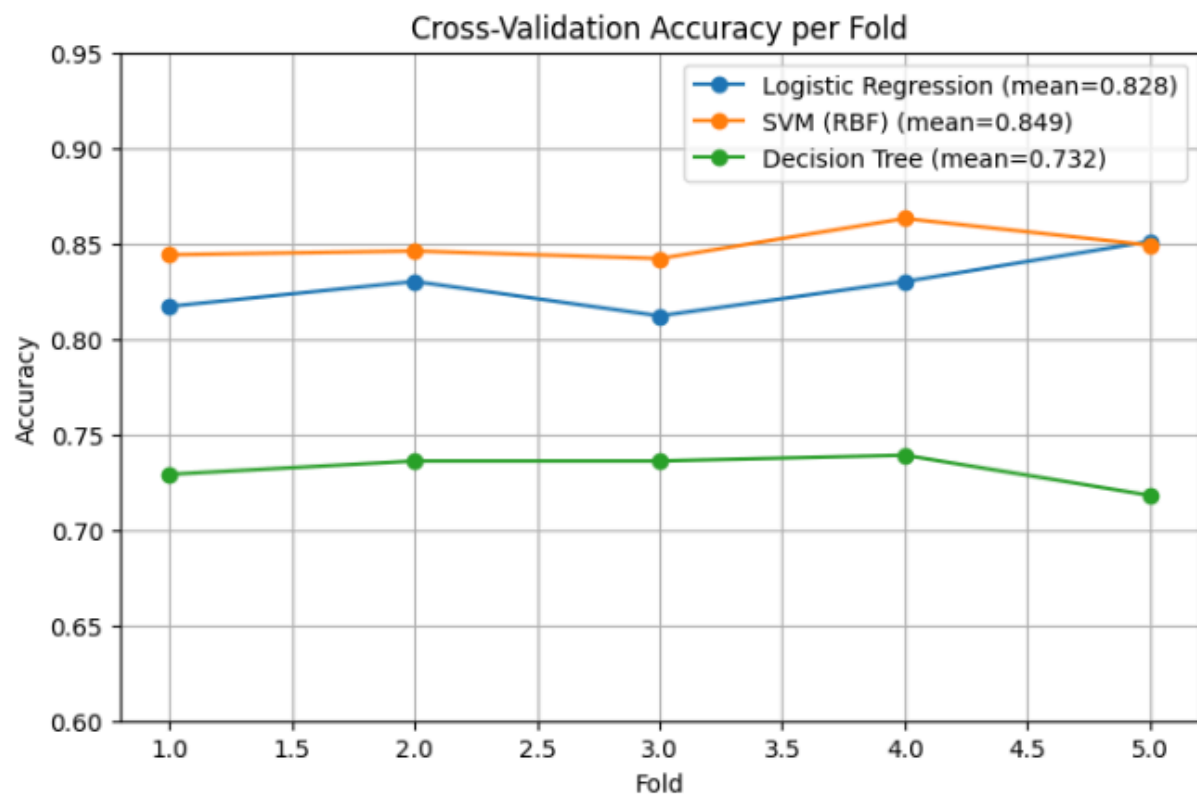
Classification Report:				
	precision	recall	f1-score	support
0	0.80	0.80	0.80	1000
1	0.97	0.96	0.96	1000
2	0.72	0.74	0.73	1000
3	0.83	0.86	0.84	1000
4	0.74	0.76	0.75	1000
5	0.95	0.92	0.93	1000
6	0.62	0.57	0.59	1000
7	0.91	0.94	0.93	1000
8	0.93	0.94	0.93	1000
9	0.95	0.95	0.95	1000
accuracy			0.84	10000
macro avg	0.84	0.84	0.84	10000
weighted avg	0.84	0.84	0.84	10000



- Weight Heatmaps

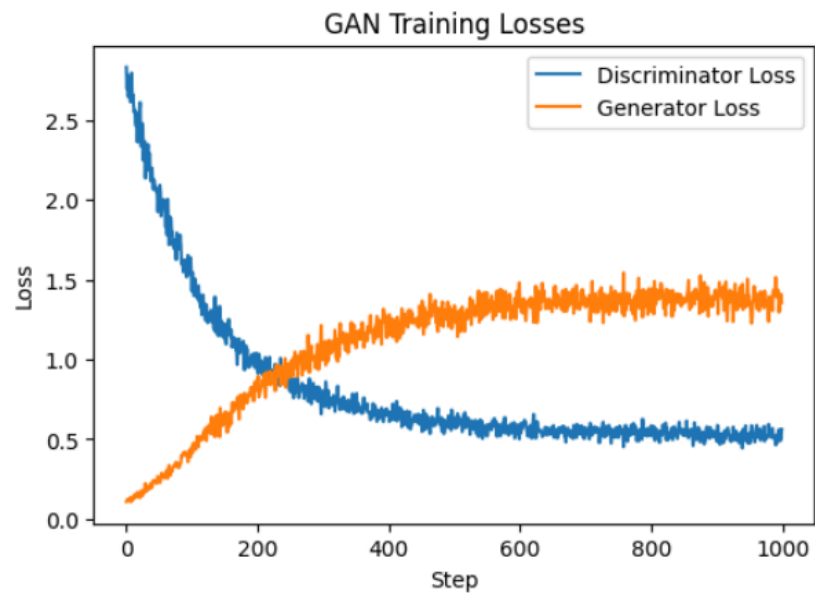


- Cross Validation Accuracy Per fold

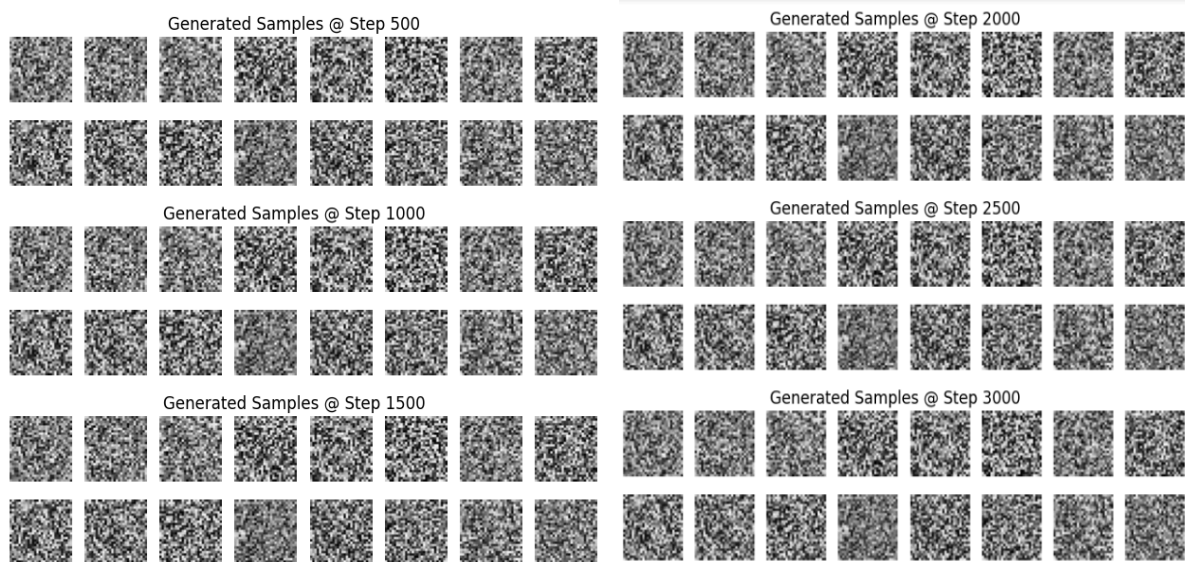


GAN

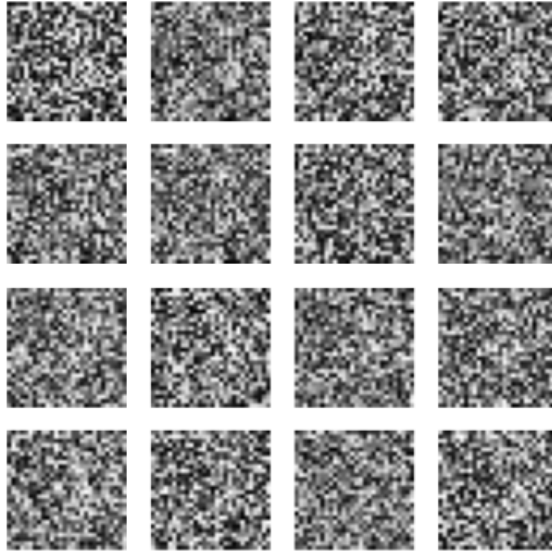
- Loss Curves



- GAN outputs over Epochs



- Mode collapse generated samples



15. Comparative Tables

Model	Training Time (approx.)	Accuracy / Output Quality	Convergence Speed	Qualitative Results
Logistic Regression	~3 mins	0.8432 (test accuracy)	Fast, stable	Easy to train, interpretable weights, limited by linearity
SVM (RBF)	~1 minute	0.8745 (test accuracy, 20k subset)	Medium	High accuracy, sensitive to kernel & hyperparameters, slower than Logistic Regression
Decision Tree	~50 seconds	0.766 (test accuracy, 20k subset)	Fast	Very interpretable, prone to overfitting without pruning
GAN (basic)	~1 minutes (5000 steps)	No accuracy metric; outputs visually improve over epochs	Slow & unstable	Generates realistic samples; unstable training; prone to mode collapse

Note: SVM, Decision Tree and GAN were trained on a **20k training subset**, while Logistic Regression used the **60k samples**.

So approximately for the 60k dataset,

Logistic: ~3 mins

SVM: ~15- 20 mins

Decision Tree: ~30 - 40 secs

GAN: ~10 mins

16. Critical Reflections

Logistic Regression was easier to understand and play around. When I took the whole dataset I was able to run within 3 mins but on the other hand when I tried GAN it took much more significant time sometimes over 20 mins to give the output for the entire dataset. Later on I had to reduce the size to get the output. Overall, Logistic Regression was easier to debug and implement.

17. How does each model handle complexity?

- **Logistic Regression:** Good for simple patterns. Struggles if the data is very complicated.
- **SVM (RBF):** Can handle harder patterns, but becomes slow when the dataset is big.
- **Decision Tree:** Can deal with complicated rules, but remembers too much instead of learning the main idea.
- **GAN:** Built for complex tasks for example like creating images, but the problem is it is unstable to train.

When would each be clearly preferred?

- **Logistic Regression:** When you want a model quickly and if it needs to be simple.
- **SVM (RBF):** When the data is small or medium size and you need good accuracy.
- **Decision Tree:** When you want clear rules that are easy to understand and fast results.
- **GAN:** When the task is to *create* new data, like new images, art, or designs and it is not encouraged for normal classification.

I made use of Chatgpt to learn the course and understand and implement the same. Here are some of my prompts given during the 9th of September, 2025.

I have the requirements mentioned above and want to learn and do the assignment. Could you please step by step explain and help in doing the assignment below. I have some sample code for reference as well.

```
# Plot the first 10 digits
fig, axes = plt.subplots(2, 5, figsize=(8, 3))
for i, ax in enumerate(axes.flat):
    ax.imshow(X[i].reshape(8, 8), cmap='gray')
    ax.set_title(f'Label: {y[i]}')
    ax.axis('off')
plt.suptitle('Sample digits from the dataset')
plt.tight_layout()
plt.show()
```

Could you explain what the code is doing and what question it answers in the Part 1

Great question — this snippet is part of **Part 1 – Discriminative Model** (the Logistic Regression side). Let's break it down: