Network Inversion for Uncertainty-Aware Out-of-Distribution Detection

Shaik Rehna Afroz
22B3932
EE691 R & D Project
IIT Bombay

Problem Statement :

• Classifiers are excellent at distinguishing between in-distribution classes but often struggle when presented with out-of-distribution (OOD) samples, misclassifying them with high confidence

• OOD samples are data points that don't belong to the same distribution as the classifier's training data

• A model trained on MNIST digits (0–9) may incorrectly classify inputs like clothing images (e.g., trousers from FashionMNIST) with high confidence. This misclassification is dangerous in real-world deployments (e.g., medical imaging, autonomous driving) where OOD inputs can lead to catastrophic outcomes

Why Should This Problem Be Solved:

• Safety: OOD inputs in medical, industrial, or autonomous systems can cause fatal decisions

• Generalization: Robust systems should recognize when an input does not belong to the training distribution

• Trust: Users need interpretable confidence estimates to trust predictions

• Solving OOD detection is critical for building safe, interpretable, and deployable AI systems

Challenges :

- Data Distribution Gaps: Real-world OOD samples can be very different from training data

- Confidence Miscalibration: Softmax outputs are often overconfident, even for unfamiliar inputs

Previous approaches:

• The Paper “ Autoinverse: Uncertainty Aware Inversion of Neural Networks” (https://arxiv.org/abs/2208.13780v2) proposes Autoinverse - a highly automated approach for inverting neural network surrogates. This approach seeks inverse solutions in the vicinity of reliable data by taking into account the predictive uncertainty of the surrogate and minimizing it during inversion.

• The paper “Uncertainty-Aware Out-of-Distribution Detection with Gaussian Processes” ”(https://arxiv.org/pdf/2412.20918) proposes a Gaussian-process-based method for out-of-distribution (OOD) detection in deep neural networks (DNNs) that eliminate the need for OOD data during training.

• This approach includes multi-class Gaussian processes (GPs) to quantify uncertainty in unconstrained Softmax scores of a DNN and defines a score function based on the predictive distribution of the multi-class GP to distinguish in-distribution(InD) and OOD samples.

Our approach :

• To address the problem of Out-of-distribution-detection (OOD) detection, Network inversion is used to generate OOD samples from the classifier’s input space. These samples, which deviate from the trained distribution, are assigned to ”garbage” class

• This method uses an n+1 approach, where n is the number of classes present in the training data. Extra ”garbage” class is added to the classifier’s output for ood detection

• To make the classifier familiar with the OOD data, it is trained with an additional ”garbage” class, initially populated by samples generated from random Guassian noise. This helps the classifier learn to recognize and isolate OOD samples

Training Process:

- After each training epoch, inverted samples generated by the conditioned generator are added to the ”garbage” class, further augmenting the dataset with OOD examples.
- This iterative retraining process helps the classifier differentiate between in-distribution and OOD data, improving its robustness.
- The inclusion of the ”garbage” class creates a data imbalance, which is managed using a weighted cross-entropy loss function. This function dynamically adjusts weights to ensure effective learning despite the imbalance.

Network Inversion

The paper ”Network Inversion and its Applications” (https://arxiv.org/pdf/2411.17777) introduces Network Inversion (NI)- a technique that determines the input from a known output and system mapping, providing insights into NN decision-making.

Methodology:

This method involves three key stages:

- **First, the classifier is trained and set to evaluation mode**
- **Next, an untrained generator (with conditioning mechanisms) learns to synthesize diverse input samples that would produce specific target outputs when fed through the classifier.**
- **This conditioned generator effectively inverts the classifier’s mapping - creating input distributions that reliably trigger desired output classifications while maintaining output diversity.**

Key characteristics:

- **Classifier remains fixed during inversion**
- **Generator training is driven by classifier feedback**
- **Conditioning ensures label-consistency in generated samples**
- **Diversity losses prevent mode collapse**

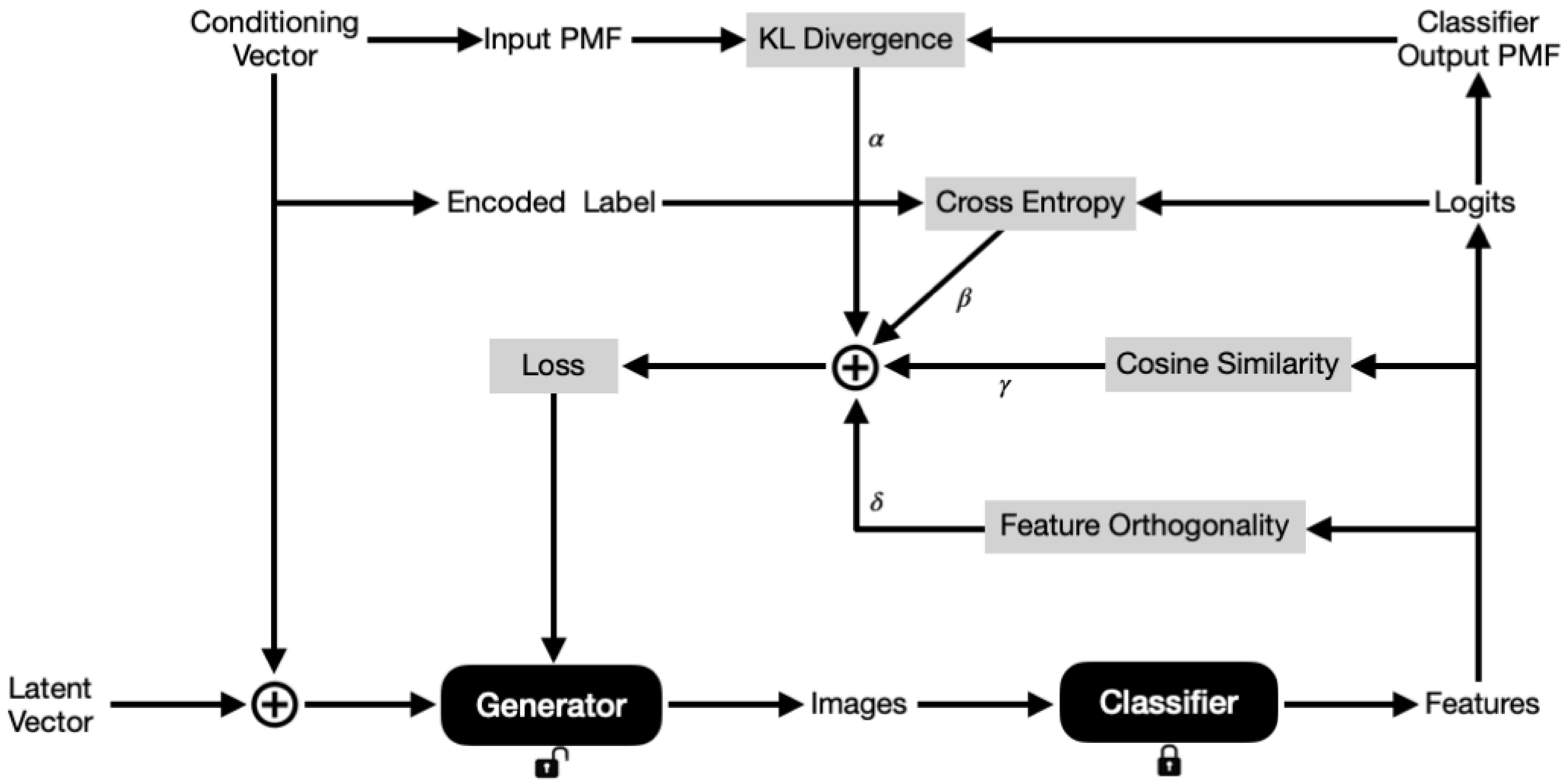


Figure 1. Proposed Approach to Network Inversion

Network Inversion and its Applications
(https://arxiv.org/pdf/2411.17777)

$$\mathcal{L}_{\text{Inv}} = \alpha \cdot \mathcal{L}_{\text{KL}} + \beta \cdot \mathcal{L}_{\text{CE}} + \gamma \cdot \mathcal{L}_{\text{Cosine}} + \delta \cdot \mathcal{L}_{\text{Ortho}}$$

where $\mathcal{L}_{\text{KL}}$ is the KL Divergence loss, $\mathcal{L}_{\text{CE}}$ is the Cross Entropy loss, $\mathcal{L}_{\text{Cosine}}$ is the Cosine Similarity loss, and $\mathcal{L}_{\text{Ortho}}$ is the Feature Orthogonality loss. The hyperparameters $\alpha, \beta, \gamma, \delta$ control the contribution of each individual loss term defined as:

$$\mathcal{L}_{\text{KL}} = D_{\text{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

$$\mathcal{L}_{\text{CE}} = -\sum_i y_i \log(\hat{y}_i)$$

$$\mathcal{L}_{\text{Cosine}} = \frac{1}{N(N-1)} \sum_{i \neq j} \cos(\theta_{ij})$$

$$\mathcal{L}_{\text{Ortho}} = \frac{1}{N^2} \sum_{i,j} (G_{ij} - \delta_{ij})^2$$

Network Inversion and its Applications
(https://arxiv.org/pdf/2411.17777)

Uncertainty Estimation Method :

- **To quantify the model's uncertainty on a given input, a normalized deviation from the uniform distribution is used over class probabilities. The uncertainty estimate lies in the range [0, 1], where values closer to 1 indicate higher uncertainty.**

- **Uncertainty estimate is given by:** $\mathcal{U}(x) = 1 - \frac{a}{b}$

- **Classifier logits z dimension: [B×K] for a batch of B samples and K classes:**

- **Softmax probabilities:**

$$p = \mathrm{Softmax}(z), \quad p \in \mathbb{R}^{B\times K}$$

$$a = \sum_{i=1}^{K} (p_i - u_i)^2$$

- **Uniform distribution over K classes:**

$$u = \left[\frac{1}{K}, \frac{1}{K}, \ldots, \frac{1}{K}\right] \in \mathbb{R}^K$$

$$b = \sum_{i=1}^{K} (\hat{y}_i - u_i)^2, \quad \text{where } \hat{y} = \text{one-hot}(\arg\max(p))$$

- **This normalizes the deviation of the model's predicted distribution from the uniform baseline. A value of U(x) close to 0 implies high confidence, whereas values near 1 reflect a distribution close to uniform (i.e., maximum uncertainty).**

- MNIST dataset is used for training and testing was done on FMNIST, CIFAR10, SVHN
- Several experiments were conducted, during which the code was iteratively debugged and refined, hyperparameters were tuned. The experiments detailed here were performed using the finalized and corrected implementation.

Results:

Experiment -1

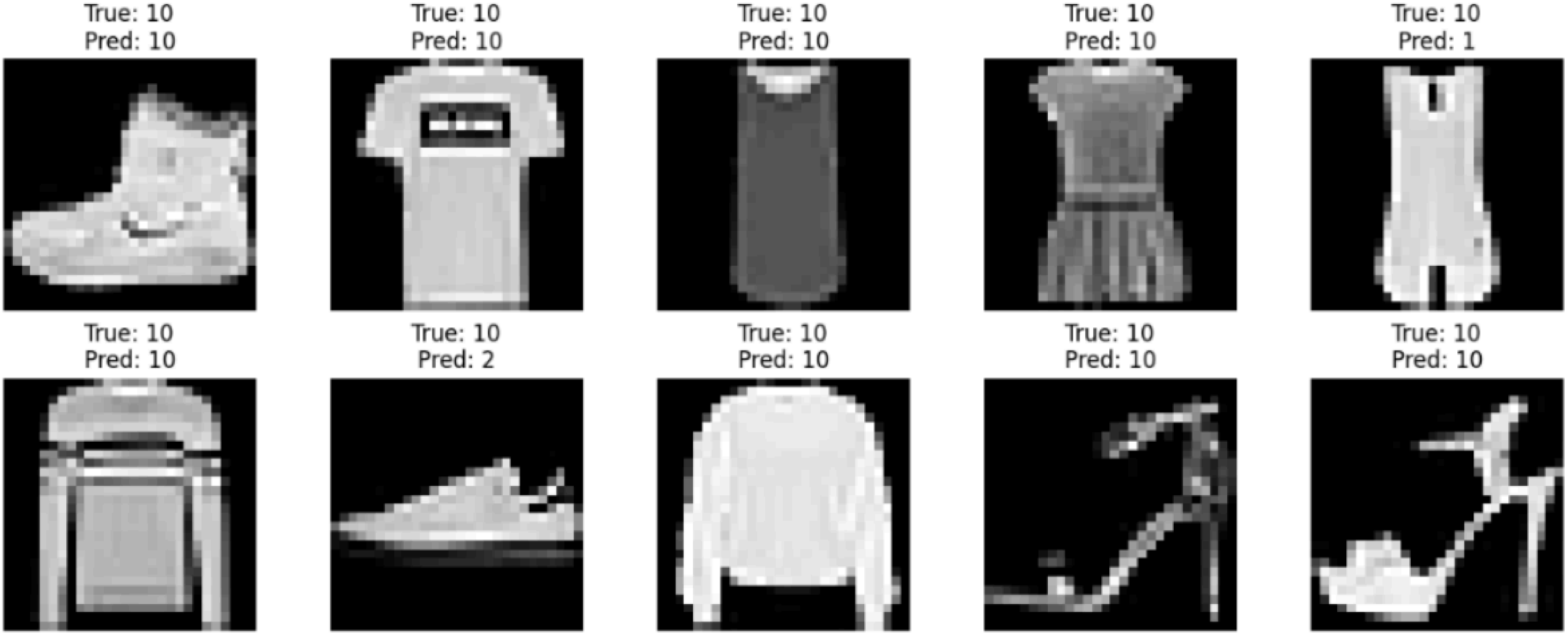
Hyperparameters and Settings

- Inversion Loss Weights: $\alpha = 500$, $\beta = 500$, $\gamma = 6000$, $\delta = 10$
- Latent Dimension: 100, Batch Size: 64
- Number of Classes: 11 (MNIST + Garbage class)
- Generator Steps per Epoch: 250
- Number of samples generated in each step = 1000
- Number of Epochs: 10

Dataset	Before Inversion (%)	After Inversion (%)
Fashion-MNIST (FMNIST)	28.81	71.89
CIFAR-10	87.3	99.99
SVHN	87.71	100

Epoch	Avg Uncertainty
1	0.9783
2	0.9778
3	0.9474
4	0.9719
5	0.9134
6	0.8922
7	0.9389
8	0.9459
9	0.9090
10	0.8736

- **Predictions from EXP-1**



Experiment -2

Hyperparameters and Settings

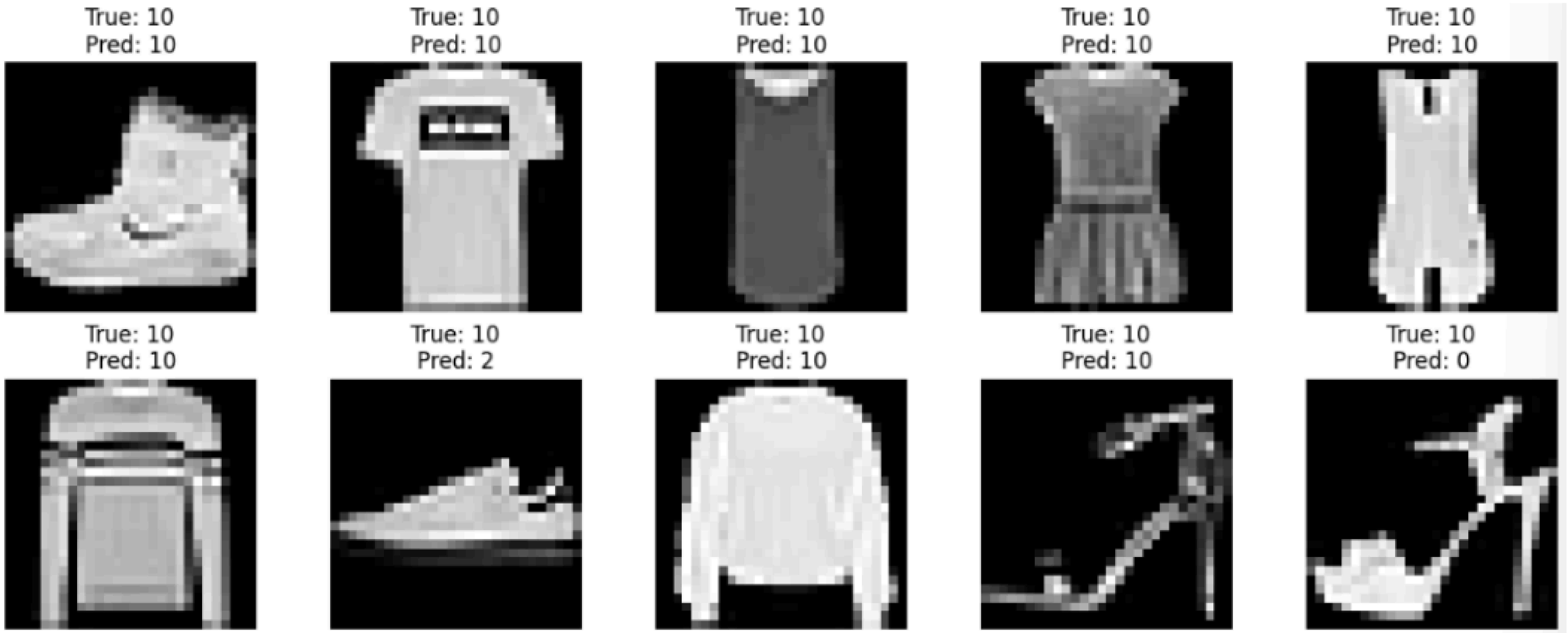
- **Inversion Loss Weights: α = 500, β = 500, γ = 6000, δ = 10**
- **Latent Dimension: 100, Batch Size: 64**
- **Number of Classes: 11 (MNIST + Garbage class)**
- **Generator Steps per Epoch: 250**
- **Number of samples generated in each step = 1000**
- **Number of Epochs: 20**

Epoch	Avg Uncertainty
1	0.980230
2	0.962530
3	0.652213
4	0.918750
5	0.983479
6	0.962047
7	0.954969
8	0.944361
9	0.940250
10	0.912971

Dataset	Before Inversion (%)	After Inversion (%)
Fashion-MNIST (FMNIST)	28.81	87.9
CIFAR-10	87.3	99.998
SVHN	87.71	100

Epoch	Avg Uncertainty
11	0.916835
12	0.869079
13	0.880915
14	0.894120
15	0.936265
16	0.882743
17	0.894302
18	0.910122
19	0.874536
20	0.869838

- **Predictions from EXP-2**



Experiment -3

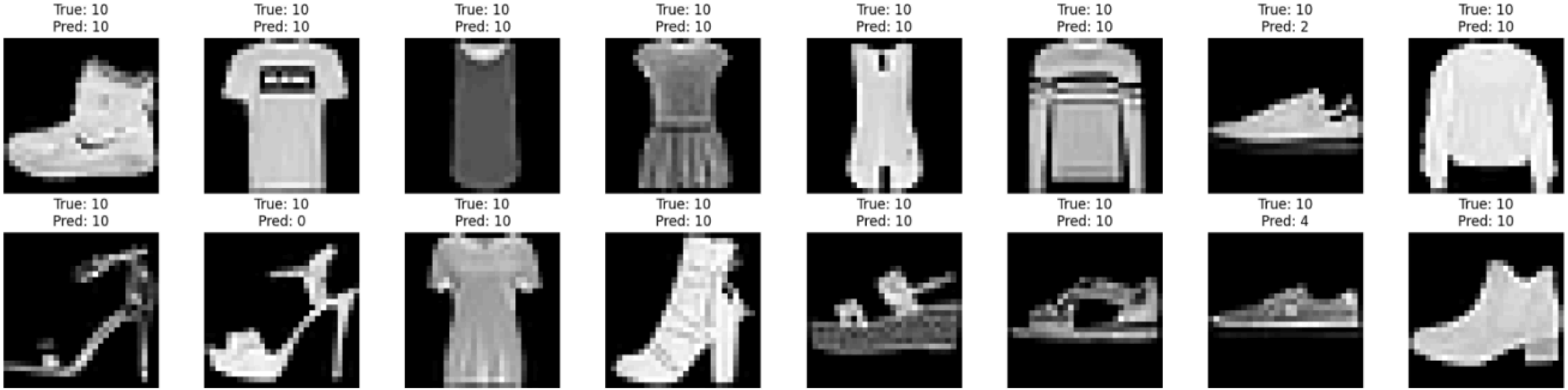
Hyperparameters and Settings :

- **Inversion Loss Weights:**
$\alpha = 500$, $\beta = 500$, $\gamma = 6000$, $\delta = 10$
- **Latent Dimension: 100, Batch Size: 64**
- **Number of Classes: 11 (MNIST + Garbage class)**
- **Generator Steps per Epoch: 250**
- **Number of samples generated in each step = 1000**
- **Number of Epochs: 50**

Epoch	Avg UE	Epoch	Avg UE	Epoch	Avg UE	Epoch	Avg UE	Epoch	Avg UE
1	0.855	11	0.000	21	0.800	31	0.822	41	0.879
2	0.935	12	0.000	22	0.792	32	0.847	42	0.808
3	0.964	13	0.958	23	0.815	33	0.883	43	0.802
4	0.690	14	0.960	24	0.867	34	0.859	44	0.703
5	0.130	15	0.949	25	0.824	35	0.878	45	0.828
6	0.002	16	0.962	26	0.851	36	0.786	46	0.829
7	0.005	17	0.946	27	0.839	37	0.865	47	0.856
8	0.685	18	0.927	28	0.870	38	0.877	48	0.829
9	0.081	19	0.890	29	0.832	39	0.837	49	0.606
10	0.002	20	0.844	30	0.836	40	0.793	50	0.676

Dataset	**Before Inversion (%)**	**After Inversion (%)**
Fashion-MNIST (FMNIST)	30.97	73.96
CIFAR-10	98.8	100
SVHN	99.73	100
MNIST	98.66	98.44

- **Predictions from EXP-3**



Experiment -4

Hyperparameters and Settings :

- **Inversion Loss Weights:**
α = 1000, β = 2000, γ = 6000, δ = 100
- **Latent Dimension: 100,**
Batch Size: 64
- **Number of Classes: 11 (MNIST + Garbage class)**
- **Generator Steps per Epoch: 250**
- **Number of samples generated in each step = 1000**
- **Number of Epochs: 30**

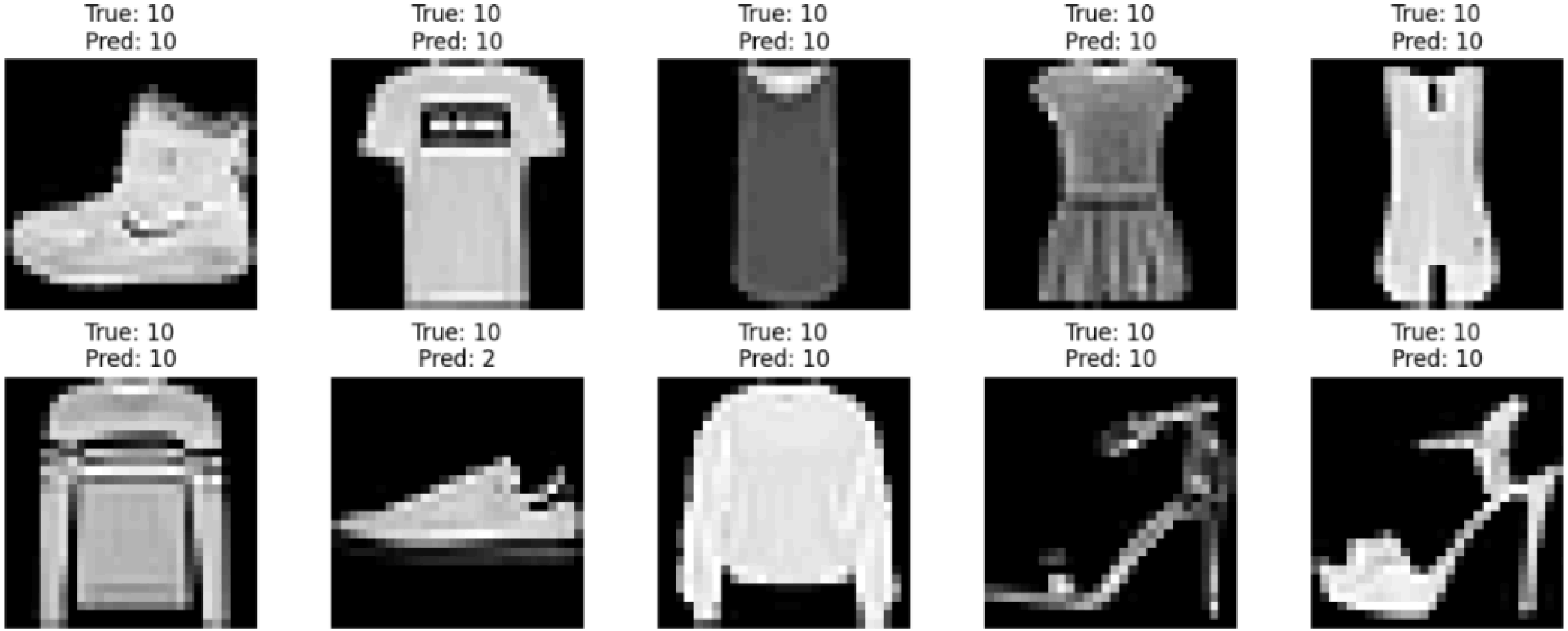
Epoch	Avg UE
1	0.9797
2	0.4496
3	0.9593
4	0.9410
5	0.9677
6	0.9521
7	0.9242
8	0.9330
9	0.8938
10	0.8741

Epoch	Avg UE
11	0.8933
12	0.9046
13	0.9034
14	0.8863
15	0.9111
16	0.8856
17	0.8941
18	0.8893
19	0.9023
20	0.8657

Epoch	Avg UE
21	0.8897
22	0.8886
23	0.8872
24	0.8503
25	0.8862
26	0.8690
27	0.9208
28	0.8304
29	0.8712
30	0.8955

Dataset	Before Inversion (%)	After Inversion (%)
Fashion-MNIST (FMNIST)	25.48	80.54
CIFAR-10	95.72	100
SVHN	96.9	100

- **Predictions from EXP-4**



Summary and Analysis:

Experiment	Epochs	FMNIST (Post-Inversion)	α, β	γ, δ
Exp-1	10	71.89%	500, 500	6000, 10
Exp-2	20	**87.9%**	500, 500	6000, 10
Exp-3	50	73.96%	500, 500	6000, 10
Exp-4	30	80.54%	1000, 2000	6000, 100

- **Across all experiments, network inversion consistently improved test accuracy on OOD datasets (Fashion-MNIST, CIFAR-10, SVHN).**
- **Longer training (Exp-2 with 20 epochs) led to the best FMNIST accuracy (71.89% for 10 epochs to 87.9% for 20 epochs), showing that the model benefits from extended generator refinement.**
- **But further increasing the number of epochs did not show improvement in the performance(87.9% for 20 epochs to 73.96% for 50 epochs)**
- **Experiment-4, with adjusted inversion loss weights (α = 1000, β = 2000, γ = 6000, δ = 100), outperforms Exp-1 and Exp-3 despite running only 30 epochs—highlighting the importance of tuning loss terms for better generator-classifier synergy.**

- **The uncertainty estimates across epochs reveal how both generator and classifier evolve over time:**

- **Early Epochs (1–3):**
  - **High uncertainty (e.g., UE 0.85–0.98) was observed, signaling initial classifier confusion due to noisy or poorly conditioned samples from the generator.**

  - **In some cases (e.g., Exp-2 Epoch 3), sharp dips in uncertainty likely indicate mode collapse or unstable generator output.**

- **Mid Epochs (10–20):**
  - **Uncertainty becomes more stable and gradually declines, suggesting:**
    - **Improved generator conditioning, producing more class-consistent samples.**
    - **Classifier adaptation, with increasing confidence on synthetic inputs.**

- **Late Epochs (20–50 in Exp-3, up to 30 in Exp-4):**
  - **The system enters a stable phase:**
    - **UE stabilizes between 0.79–0.88, indicating a healthy balance between exploration (novel inputs) and exploitation (confidence).**
    - **Some fluctuations (e.g., spikes in Exp-4 Epoch 27) reflect natural generator variations but don’t degrade performance.**

THANK YOU