SRE(Supervised Research Exposition) Neural Network Inversion

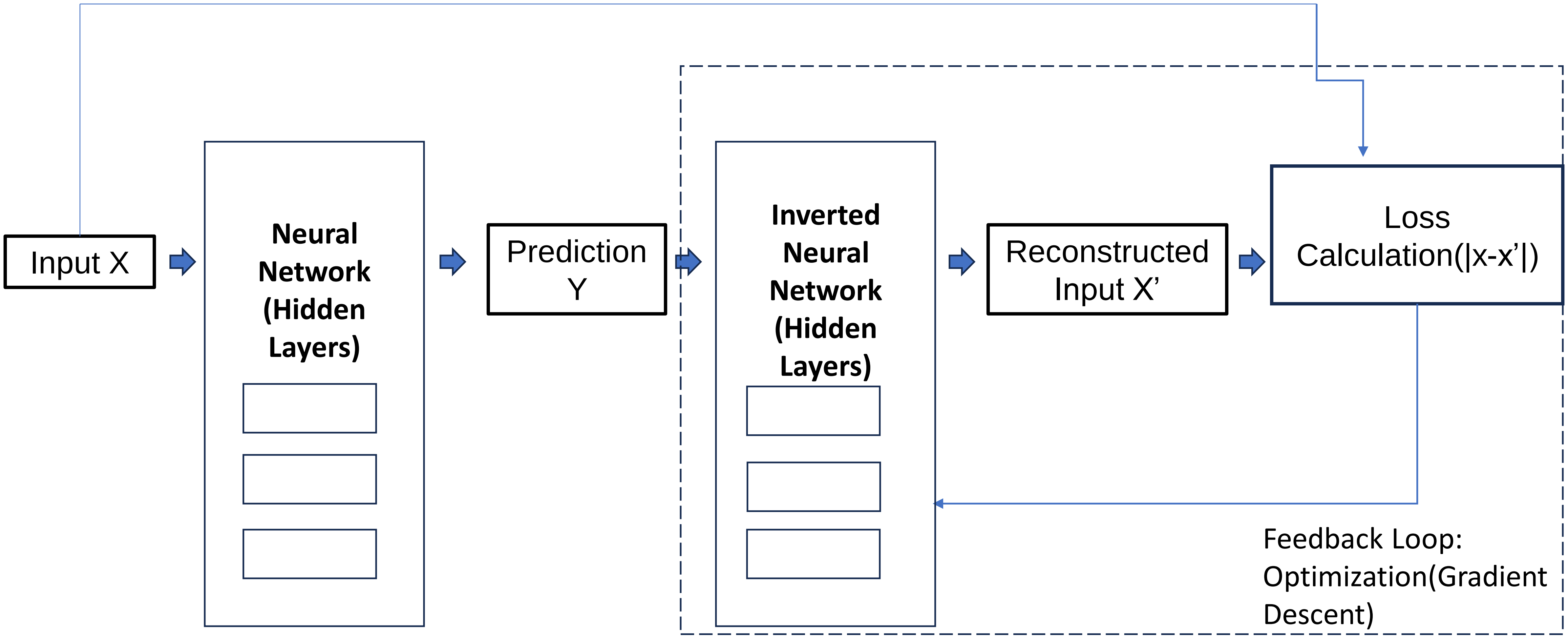
Akshat Taparia(210110014)

Supervised Research Exposition:Presentation

Guide : Prof. Amit Sethi
Department of Electrical Engineering
Co-Guide: Pirzada Suhail
Department of Electrical Engineering

What is Neural Network Inversion?

Neural Network Inversion is the process of reconstructing input data (such as images or signals) from the output or intermediate features of a trained neural network. Essentially, given the output or activations, the goal is to infer what input could have produced those activations.



We essentially continue this process until the reconstructed input x’ becomes a better approximation of x(input)

Summary of Work Done

Experiments conducted

- **Loss Ablations:** Analysed the impact of losses like cross entropy loss, cosine similarity, orthogonality and KL divergence on IA
- **Hyperparameter Studies:** Tuned label smoothing, learning rate, batch size and loss coefficients for better inversion accuracy
- **OOD Detection:** Generating some gaussian noises in garbage class and iteratively trained classifier and generator

Datasets of Interest/Codes Modified

- MNIST/Fashion MNIST
- CIFAR10/SVHN
- OOD Detection

Literature Review(main references)

- (Network Inversion and Application Thereof-CVPR)
- (Out of Distribution Detection Based on Deep Learning :A Review-MDPI)

Mapping the Inversion Process

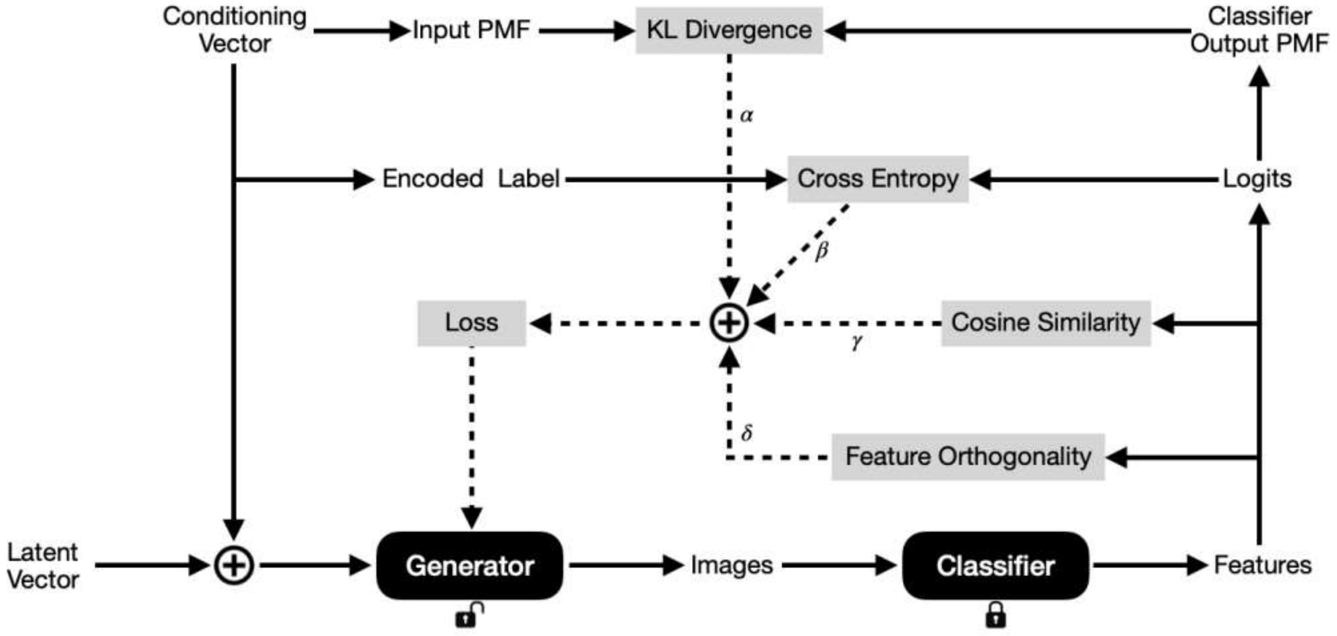


Figure 1: Schematic Representation of the Inversion Process

Applications of Neural Network Inversion include(as discussed here):

- Training Data Reconstruction
- Adversial Data Generation
- Fine-Tuning Classifiers

- **Classifier and Generator**:
  The classifier is pre-trained and kept in evaluation mode during the inversion process. The generator takes a latent vector and a conditioning vector to create images that the classifier should label correctly

- **Conditioning Mechanism**:
  Instead of using simple class labels, vector conditioning is applied to encourage greater diversity in the generated images.
  The conditioning vector is randomly generated and encoded to indirectly represent the desired label

- **Loss Functions**(discussed in depth ahead)

- **Inversion Process**:
  The generator is refined iteratively, using classifier feedback to minimize the inversion loss, producing diverse and accurate images that represent the network's decision-making patterns

Reference: [AAAI2 (1).pdf] (Network Inversion and Applications Thereof)

Out-of-Distribution(OOD) Detection

Out-of-Distribution (OOD) detection focuses on identifying whether an input sample differs from the distribution the model was trained on. This is critical for real-world applications like intrusion detection, fraud prevention, and health monitoring. Deep learning methods are central to OOD detection research, which classifies methods into three categories: supervised, semi-supervised, and unsupervised approaches

Unsupervised

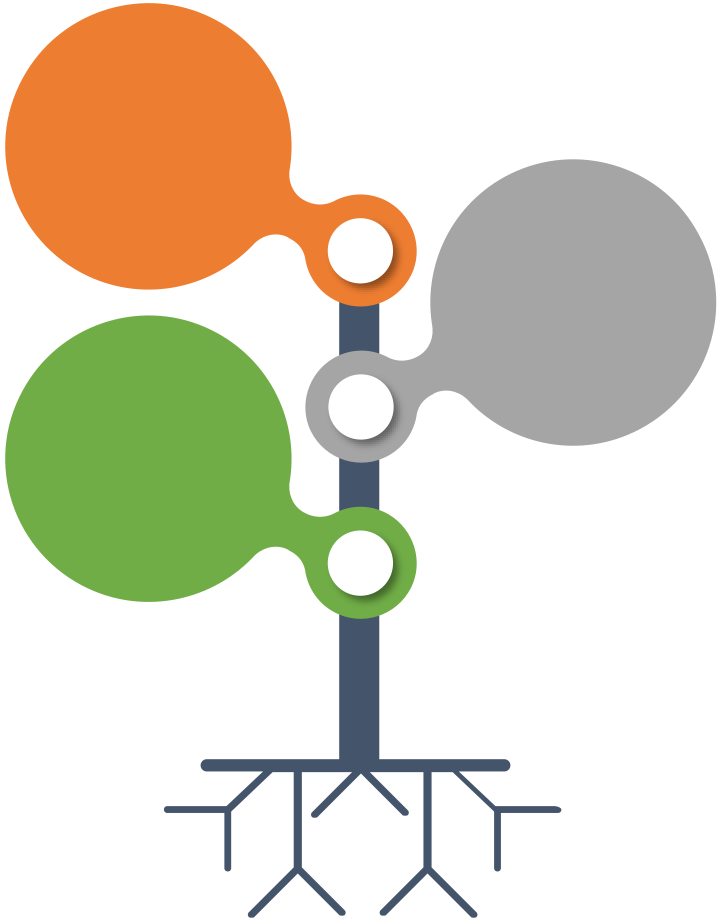
No labeled data is used, and techniques like clustering for anomaly detection are employed to find OOD samples

Application: In **network security** to detect unknown cyberattacks without requiring labeled OOD data

Supervised

Rely on labeled data and include model-based, distance-based, and density-based techniques

Application: In **Fraud detection systems** to flag transactions by learning from labeled fraudulent and legitimate cases



Semi-Supervised

Part of the training data is labeled, and these methods use techniques like autoencoders to identify OOD data based on reconstruction errors

Application: In **medical imaging**, an autoencoder trained on normal scans could detect abnormalities (OOD samples) by failing to accurately reconstruct scans showing rare diseases or unseen conditions

Logits: Raw, unnormalized outputs of a neural network's final layer. They are converted into probabilities using an activation function like softmax.

OOD Detection Models

$$\text{softmax}(logit) = \frac{e^{logit}}{\sum_{i=1}^{n} e^{logit_i}}$$

SMOOD

Description: Uses softmax scores from a pre-trained model for OOD detection, lower softmax score indicates OOD

How it works: Maximum softmax probability is used. If below a threshold, it's likely OOD

Stengths: Simple, no model modifications needed

Weaknesses: Overconfident as high softmax scores may incorrectly classify OOD samples as ID

LC(Linear Classifier)

Description: Applies linear separability for OOD detection based on class boundaries

How it works: OOD samples are far from class boundaries

Stengths: Simple, computationally efficient

Weaknesses: Struggles with complex datasets or high-dimensional features

ODIN (Out-of-Distribution Detector for Neural Networks)

Description: Enhances SMOOD by adding temperature scaling and input preprocessing

How it works: Temperature scaling smooths logits; input perturbations help differentiate OOD samples

Stengths: Better performance and generalization compared to SMOOD

Weaknesses: Requires tuning parameters; slightly slower due to preprocessing

OODL (Out-of-Distribution detection using Layer output)

Description: Uses features from intermediate layers to detect OOD samples

How it works: Finds the most discriminative layer and applies a classifier to separate ID from OOD

Stengths: State-of-the-art performance due to deeper feature extraction

Weaknesses: More time-consuming and computationally expensive

Reference: Out-of-Distribution_OOD_Detection_Based_on_Deep_Le.pdf

OOD Detection using Neural Network Inversion-Methodology

01 Dataset Preparation

Prepares the dataset for training while setting a baseline for in-distribution (ID) data

02 Add Garbage Class

Random noise images added initially to it to simulate OOD samples in order to train the model to recognize non-MNIST inputs as distinct

03 Define Class Weights

Computes weights for each class including garbage class to ensure that the model learns to balance ID and OOD detection

04 Initialize Models

Define Classifier and Generator models with proper weight initialization. The Classifier learns to differentiate OOD from ID classes, and the Generator supports inversion tasks

05 Define Loss Function

Use Weighted Cross-Entropy Loss with class weights. Enhances the Classifier's ability to recognize OOD classes by accounting for class imbalance.

06 Training Loop

Train the **Classifier** to distinguish ID from OOD data, while the **Generator** refines OOD sample generation through inversion

07 Repeat

Update dataset with generated samples and retrain Classifier. Iterative refinement improves the Classifier's ability to detect OOD inputs consistently.

https://colab.research.google.com/drive/1uURugUxzd7A8LJcun076FQu4_5k8oeq6?usp=sharing

Hyperparameters of Interest

Hyper-parameter	Explanation
Label Smoothing	Softens hard class labels to prevent overconfidence, improving stability, diversity, and generalization in inversion images
Learning Rate	Controls the step size for model updates; too high can cause instability, too low may slow convergence
Weight Decay	Penalizes large weights to prevent overfitting, promoting smoother and more generalizable inversion results
Batch Size	Larger batch sizes stabilize training and accelerate convergence but may require tuning to balance between performance and memory constraints
L1 regularization	Adds sparsity by penalizing the sum of absolute weights, reducing noise and improving model interpretability

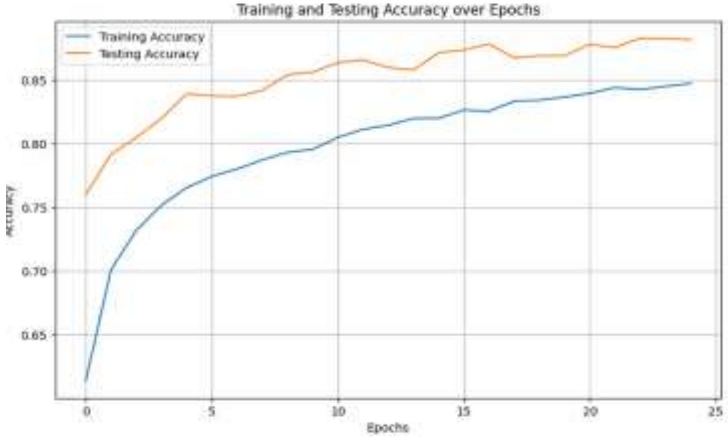
Description of Datasets used

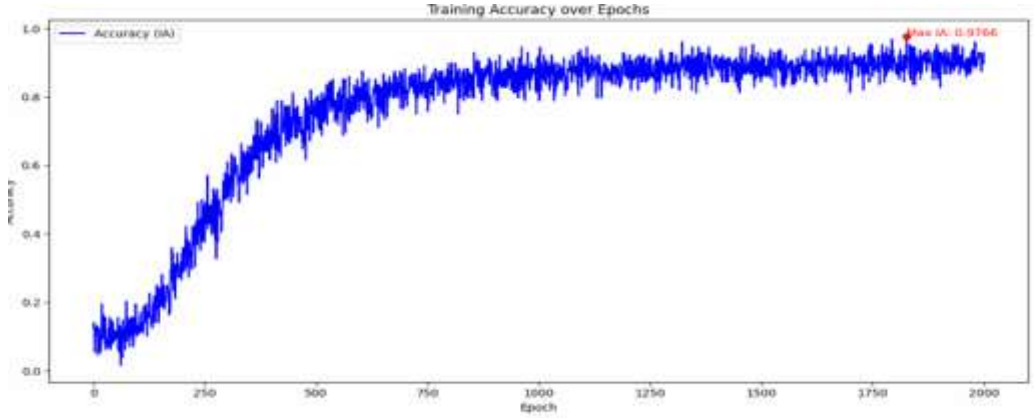
Feature	**MNIST/Fashion MNIST**	**CIFAR-10/CIFAR-100**	**SVHN**
Images	28x28 pixels, grayscale	32x32 pixels, RGB	32x32 pixels, RGB (color images)
Number of Classes	10 (digits/fashion categories)	10 (CIFAR-10), 100 (CIFAR-100)	10 (digits 0-9)
Channels	1 (grayscale)	3 (RGB - color images)	3 (RGB - color images)
Total Images	70,000	60,000	600,000+ (73,257 train, 26,032 test, 531,131 extra)
Data Complexity	Simple	Complex (real-world objects)	Real-world digits in varying backgrounds, different lighting conditions
Colour Information	None	Yes	Yes
Size of Dataset	Smaller	Moderate	Large (more challenging due to the real-world, varied settings)

Loss(commented out)	**Effect on Inversion Accuracy(numericals achieved on MNIST, trend same for all datasets, dealing with 2000 epochs)**	**Insights**	**Possible Explanation**
Cross Entropy Loss	**Critical for achieving high IA**; without it, IA drops significantly (≈ 0.28)	Without cross-entropy, **generated images become unclear and mismatched with their true labels**, leading to **blurry or incorrect class reconstructions**	Cross-entropy ensures that the generated images are **correctly classified**, which is essential for image inversion. Removing it leads to **poor label matching**
Feature Orthogonality Loss	**Slower convergence** and **reduced max IA (≈ 0.89)** when removed	Without orthogonality loss, **images show less diversity** and feature redundancy, causing **repetitive patterns** in the inversion, reducing visual quality	Orthogonality loss ensures diverse and distinct feature representations, improving the expressiveness and quality of the inverted images
KL Divergence	**Improved IA (≈ 0.8) after fewer epochs** (600-650); **rapid initial improvement** in inversion accuracy	Removing KL divergence may lead to **more varied images** that converge faster but are **less aligned with the target distribution**, sometimes causing artifacts	KL divergence helps match output distribution with target distribution. When removed, the network focuses more on feature alignment, improving initial IA but leading to variability in output
Cosine Similarity Loss	**Faster convergence to higher IA (≈ 0.8)** with a **steeper slope**; rapid increase in accuracy (≈ 500 epochs)	**Sharper, more distinct image features** are generated more quickly, but there's **less constraint on feature alignment**, which cause small inconsistencies	Cosine similarity encourages feature alignment, promoting smoother gradients. Without it, the network focuses more on accuracy but may generate slightly misaligned features

Results on Fashion MNIST dataset

Metric	Value/Reasoning
Best Classifier Test Accuracy	0.882
Learning Rate	0.001 to 0.003
Batch Size	2048
Label Smoothing	0.01
Data Augmentations	Removed to improve performance/reduce classifier training time
Loss Functions	Cross-Entropy (Critical), Cosine Similarity (Enhancing clarity) Orthogonality (Adjusted), KL Divergence (Reduced) L1 Regularization (Reduced)
Weight Decay, L1 regularization	1e-4,1e-4(without regularization not much difference in IA)

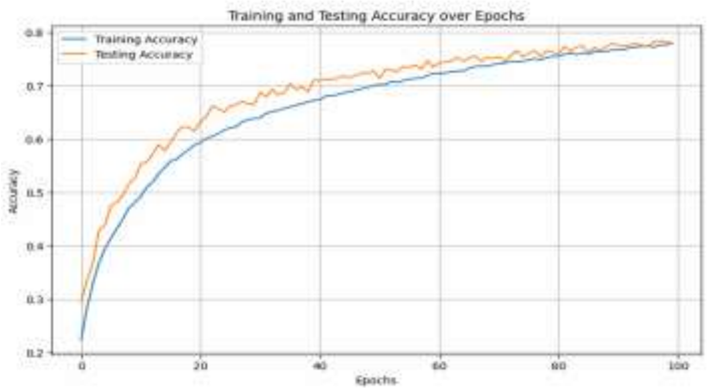


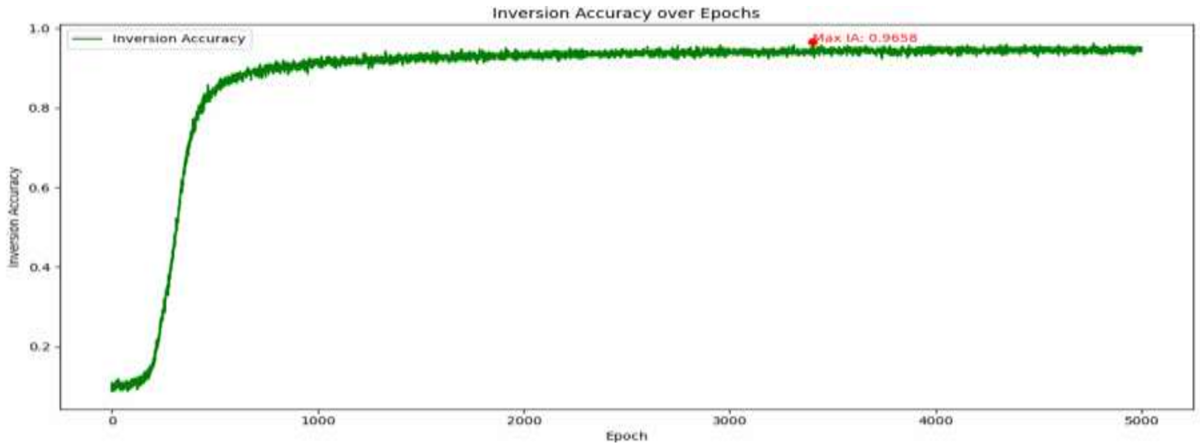


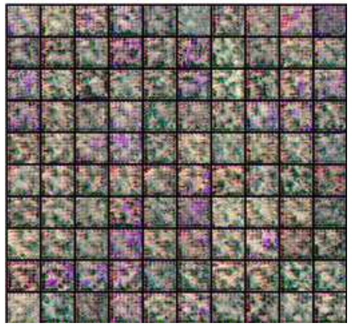
Without cosine loss

Results on CIFAR10 dataset

Metric	Value/Reasoning
Best Classifier Test Accuracy	0.7922 (not as high as grayscale datasets like MNIST)
Learning Rate	0.003-0.005
Batch Size	2048 (1024 was also tried with some changes in IA)
Label Smoothing	0.01
Weight Decay, L1 regularization	1e-4, 1e-4
Loss Functions	Cross-Entropy (Essential), Cosine Similarity (Feature alignment) Orthogonality (Feature diversity), KL Divergence (Reduced) L1 Regularization (Reduced)[Best images with coefficient for cross entropy around 6k-10k and that of cross entropy around 1k]







Insights of OOD Detection using INN

$L\infty$ perturbation is commonly used in adversarial attacks for its pixel-level control, ensuring subtle changes. In contrast, on adding L1 and L2 perturbations, we see more effective global detection OOD shifts, as they evenly distribute changes across the image to capture broader distributional variations.

OOD Dataset	**Using Softmax (SMOOD)**	**LC**	**ODIN(temperature scaling and perturbations)**
Noisy(like Gaussian, Uniform)	Baseline	Best	Lower than LC
Structured(like LSUN)	Baseline	Lower than ODIN	Best

Performance across OOD datasets: ODIN excels with structured OOD using temperature scaling/perturbations, but struggles with noisy data where LC performs better due to effective linear separation.