TRANSFORMING ADVERSARIAL SAMPLES TO CLEAN SAMPLES

K Rithwin 220537

Ayush Meena 220268

Ritesh Baviskar 220286

Aaditi Agrawal 220006

Piyush Patil 220759

Aditya Jagdale 220470

***Abstract*—Deep neural networks have demonstrated remarkable success across various domains, but remain vulnerable to adversarial attacks—carefully crafted perturbations that can deceive classifiers while being imperceptible to human observers. While numerous defense mechanisms have been proposed, existing approaches often face critical limitations: they either focus exclusively on detection or purification, exhibit poor generalization across different attack types, or significantly degrade performance on clean inputs. To address these challenges, we present a comprehensive and unified defense framework that seamlessly integrates detection and purification within a single robust pipeline while maintaining high accuracy on unperturbed inputs.**

Our novel architecture consists of two key components working in tandem. First, we introduce an advanced dual-manifold adversarial detector that leverages Bayesian uncertainty estimation through Monte Carlo dropout. This innovative approach models distinct manifolds for both clean and adversarial features, enabling reliable identification of malicious inputs without relying on traditional kernel density estimation methods and their associated sensitivity to bandwidth parameters. Second, we develop a High-Level Feature Guided Denoiser (HFGD), a sophisticated denoising autoencoder trained across multiple attack vectors (including FGSM, PGD, and BIM variants) that performs reconstruction in feature space rather than pixel space. This design choice, combined with perceptual loss functions, allows for more effective preservation of semantic content during the purification process. Through extensive experimentation and analysis, we demonstrate that our approach establishes new standards in adversarial robustness, effectively bridging the gap between detection and mitigation while setting a benchmark for generalizable defense mechanisms in deep learning systems.

***Index Terms*—Adversarial defense, uncertainty estimation, deep learning security, denoising autoencoder, manifold learning, robust machine learning, adversarial purification.**

I. Introduction

Deep neural networks (DNNs) are machine learning techniques that impose a hierarchical architecture consisting of multiple layers of nonlinear processing units. In practice, DNNs achieve state-of-the-art performance for a variety of generative and discriminative learning tasks from domains including image processing, speech recognition, drug discovery and genomics.

Although DNNs are known to be robust to noisy inputs, they have been shown to be vulnerable to specially-crafted adversarial samples. These samples are constructed by taking a normal sample and perturbing it, either at once or iteratively, in a direction that maximizes the chance of misclassification.

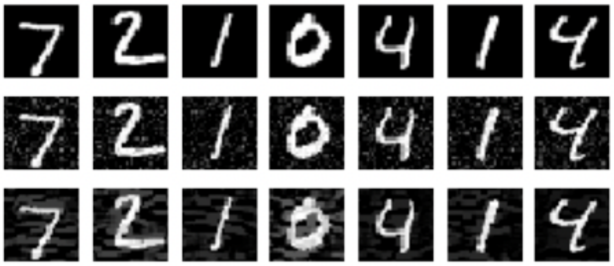

Fig. 1: Examples of normal (top), noisy (middle) and adversarial (bottom) MNIST samples for a convnet. Adversarial samples were crafted via the Basic Iterative Method and fool the model into misclassifying 100% of the time.

Fig. 1 shows some examples of adversarial MNIST images alongside noisy images of equivalent perturbation size. Adversarial attacks which require only small perturbations to the original inputs can induce high-efficacy DNNs to misclassify at a high rate. Some adversarial samples can also induce a DNN to output a specific target class. The vulnerability of DNNs to such adversarial attacks highlights important security and performance implications for these models. Consequently, significant effort is ongoing to understand and explain adversarial samples and to design defenses against them.

II. Related Work

Recent research in adversarial defense has focused on two primary strategies: detection of adversarial examples and purification through denoising. This section analyzes key approaches and their limitations while contextualizing our contributions.

A. Detection-Based Approaches

1) Experimental Detector: Feature Squeezing: [4] We implemented the detector from *Xu et al.* (2017) that identifies adversarial examples through input complexity reduction. Key components include:

- **Pre-trained CNN**: LeNet (MNIST) and CNN (CIFAR-10) for classification
- **Squeezing Transformations**:
  - Bit-depth reduction (1-bit MNIST/5-bit CIFAR)
  - 2×2 median filtering

- Non-local means denoising (CIFAR-specific)
- **Anomaly Scoring**: Maximum L1 distance between original and squeezed predictions

Implementation Details:

- Thresholding: 5% FPR baseline on validation data
- Attack Performance: Effective against BIM but limited against C&W/DeepFool

Exclusion Rationale:

- Architectural Constraints: Input transformations incompatible with our purification pipeline
- Attack Generalization: Insufficient coverage against modern adversarial variants

2) Feature Artifact: [1]

Feature Artifact Detection (Feinman et al.): Leverages Bayesian uncertainty estimates and feature density analysis in DNN hidden layers. Demonstrates 88% detection accuracy on CIFAR-10 against FGSM attacks. However, performance degrades significantly against adaptive attacks (e.g., C&W) and requires full access to model internals, limiting practical deployment.

3) Artifact-Based Detection (Feinman et al., 2017): This attack-agnostic detector identifies adversarial samples through two complementary feature spaces:

- **Bayesian Uncertainty:** Utilizes dropout variational inference to estimate epistemic uncertainty:

$$\mathbb{V}[y|x] = \frac{1}{T}\sum_{t=1}^{T}\hat{y}_t^2 - \left(\frac{1}{T}\sum_{t=1}^{T}\hat{y}_t\right)^2$$

  where T Monte Carlo dropout passes generate predictions $\hat{y}_t$ [4].
- **Deep Feature Density:** Computes kernel density estimates in the penultimate layer's activation space:

$$\log p(z) \propto -\frac{1}{2\sigma^2}||z-\mu||^2 + C$$

  where z denotes test sample activations and μ the training class mean [4].

Fig. 2: Two-stage detection framework: (1) Extract uncertainty/density features, (2) Train SVM classifier on normal vs adversarial feature distributions [4]

Key Properties:

- Achieves 85-92% ROC-AUC across MNIST/CIFAR-10 against FGSM, JSMA, and C&W attacks
- Requires no prior knowledge of attack methodology
- Compatible with any DNN architecture supporting dropout

Implementation Considerations:

- Density estimation requires stored training class means
- Dropout rate (typically 0.5) impacts uncertainty calibration
- Kernel bandwidth selection critical for detection performance

B. Purification-Based Approaches

[3]

- **Adversarial Purification using Denoising AutoEncoders (Kalaria et al., 2022)** is a defense method using Denoising Autoencoders to iteratively purify adversarial examples by reducing reconstruction error, aligning inputs with the clean data distribution. Unlike standard DAEs, it preserves semantic features and is non-differentiable, improving robustness against adaptive attacks. Tested on MNIST, CIFAR-10, and ImageNet, it outperforms methods like MagNet and Defense-GAN, especially under strong gray-box and adaptive scenarios, while highlighting a trade-off between purification strength and clean accuracy.
- **High-Level Guided Denoiser (Liao et al.) [3][4]:** Introduces feature-guided denoising using perceptual losses from pretrained models. Achieves 64.37% accuracy on CIFAR-10 under combined FGSM+PGD attacks, but requires training on 210K adversarial examples and fails to detect attacks before purification. The method also exhibits 4% accuracy drop on clean images.

C. Limitations of Existing Works

- Detection methods lack purification capabilities
- Denoisers introduce error amplification effects
- Performance degradation on complex datasets
- Separate pipelines for detection and purification
- High computational overhead calculations(in terms of time taken)

D. Our Contributions

Our unified framework addresses these limitations through:

- **Joint Detection-Purification:** Integrated pipeline using feature discrepancy analysis for detection and residual learning denoiser for purification
- **Error Suppression:** Bottleneck architecture with skip connections prevents error amplification
- **Cross-Domain Robustness:** Generalizes to unseen attacks through adversarial feature alignment

III. Adversarial Attacks

Adversarial attacks aim to slightly modify input data in a way that causes a machine learning model—especially deep neural networks—to make incorrect predictions. These perturbations are often imperceptible to humans but significantly impact the model's output. Below we describe some of the most commonly used attack techniques.

Epsilon (ϵ) is a parameter used in adversarial attacks that defines the maximum amount by which each pixel (or feature) in the input can be perturbed, typically measured in terms of the ℓ_∞ norm. In other words, it sets the upper bound on how much the adversarial noise can alter the input while generating adversarial examples.

A higher epsilon value means larger perturbations are allowed, which increases the strength of the attack and makes it more likely that the model will be fooled into making incorrect predictions. Conversely, a lower epsilon restricts the perturbation, resulting in weaker attacks that are less likely to deceive the model. Thus, epsilon is directly proportional to attack strength: as epsilon increases, the attack becomes more effective at fooling the model.

A. *Fast Gradient Sign Method (FGSM)*

The Fast Gradient Sign Method (FGSM), proposed by Goodfellow et al., is one of the earliest and most efficient white-box adversarial attacks. It uses the gradient of the loss function with respect to the input to create a perturbed example in a single step.

Given a model with parameters θ, input image x, true label y, and loss function $J(\theta, x, y)$, FGSM generates the adversarial example x_{adv} as:

$$x_{\text{adv}} = x + \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y))$$

where:

- ϵ controls the perturbation strength.
- $\text{sign}(\cdot)$ returns the sign of each gradient component.
- $\nabla_x J(\theta, x, y)$ is the gradient of the loss with respect to the input.

FGSM is fast and effective, but since it uses only a single gradient step, it may not always find the most powerful adversarial examples.

B. *Projected Gradient Descent (PGD)*

Projected Gradient Descent (PGD) is an iterative variant of FGSM and is considered a strong first-order adversary. It performs multiple steps of gradient ascent to maximize the loss, while ensuring the perturbation remains within a predefined ϵ-ball around the original input.

The PGD update rule is:

$$x^{t+1} = \Pi_{\mathcal{B}_\epsilon(x)} \left(x^t + \alpha \cdot \text{sign}(\nabla_x J(\theta, x^t, y)) \right)$$

where:

- α is the step size.
- $\Pi_{\mathcal{B}_\epsilon(x)}(\cdot)$ is the projection operator that clips the perturbed input to stay within the ℓ_∞ ball of radius ϵ around the original input x.
- t denotes the iteration number.

PGD is widely regarded as the most robust baseline attack, and models that withstand PGD are often considered adversarially robust.

C. *Basic Iterative Method (BIM)*

The Basic Iterative Method (BIM), also known as Iterative FGSM or I-FGSM, is another iterative extension of FGSM. It applies FGSM multiple times with a small step size and optionally clips the perturbations after each step to ensure they remain within the allowed range.

The update rule is:

$$x^{t+1} = \text{Clip}_{x,\epsilon} \left(x^t + \alpha \cdot \text{sign}(\nabla_x J(\theta, x^t, y)) \right)$$

where:

- α is a small step size.
- $\text{Clip}_{x,\epsilon}(\cdot)$ ensures that x^{t+1} stays within an ϵ-neighborhood of the original input x, and within valid pixel bounds (e.g., [0,1]).

BIM features two primary variants with distinct optimization strategies:

- **BIM-A (Early-Stopping Variant):** Executes iterative perturbations until either:
  - The model misclassifies the adversarial example
  - A predefined maximum iteration count N is reached

  The attack stops immediately upon successful misclassification, making it computationally efficient. The update rule is:

  $$x^{(t+1)} = \text{clip}_\epsilon \left(x^{(t)} + \alpha \cdot \text{sign}(\nabla_x \mathcal{L}(x^{(t)}, y_{\text{true}})) \right)$$

  where $\alpha = \epsilon/T$ (typically $T = 10$) ensures controlled perturbation growth [4].
- **BIM-B (Complete Iteration Variant):** Performs the full sequence of N iterations regardless of early success, often producing stronger adversaries. It applies:

  $$x^{(t+1)} = \text{clip}_\epsilon \left(x^{(t)} + \alpha \cdot \text{sign}(\nabla_x \mathcal{L}(x^{(t)}, y_{\text{true}})) \right)$$

  for all $t \in \{1, ..., N\}$, without early termination. This exhaustive approach typically requires $\alpha = 1.25\epsilon/N$ to ensure perturbation bounds [3][4].

IV. Proposed Methodology

Our proposed adversarial defense framework is a two-stage system consisting of a Detector Module followed by a Purifier Module. The system is designed to first identify potentially adversarial inputs and then selectively purify them to restore their clean representations before passing them to a downstream classifier. This selective mechanism ensures minimal performance degradation on clean samples and effective defense against various attacks such as FGSM, PGD, and BIM.

A. *Detector Module*

In this section, we describe the proposed adversarial detector, which leverages both manifold-based statistical distances and predictive uncertainty. The aim is to distinguish adversarial examples from clean inputs using a low-dimensional, interpretable feature set, without significantly impacting inference speed or requiring access to gradients of the classifier. The detector is designed to be modular and agnostic to specific attack types.[2]

1) Deep Feature Extraction: The detection pipeline begins by extracting semantic features from a pretrained classifier. Let $f : \mathbb{R}^d \to \mathbb{R}^k$ denote the mapping from input space to the feature space obtained from the penultimate layer of the classifier. These features retain class-discriminative information even in the presence of adversarial perturbations.

Given an input image x, we compute:

$$\mathbf{z} = f(x)$$

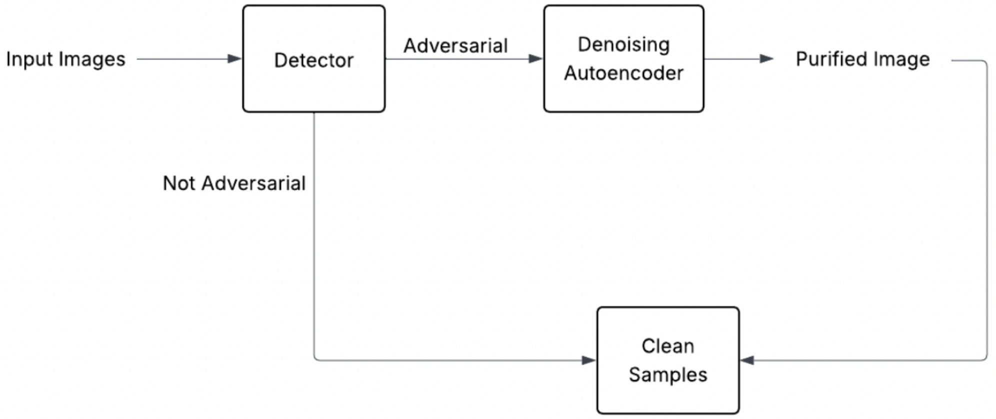

Fig. 3: Overall Model Architecture

The extracted features $\mathbf{z}$ are then used to compute class-wise distances and train the detector. Importantly, this process is non-intrusive and does not require retraining or modifying the classifier.

2) Dual-Manifold Distance Estimation: The core of our detection method involves modeling the clean and adversarial class manifolds in the deep feature space and measuring deviations from these reference distributions.

For each class $c \in \mathcal{C}$, we estimate two multivariate Gaussian distributions:

- A distribution from clean samples correctly predicted as class c,
- A distribution from adversarial samples that are confidently predicted as class c.

The clean class-conditional manifold is modeled as:

$$\boldsymbol{\mu}_c^{\text{clean}} = \frac{1}{N_c} \sum_{i:y_i=c} f(x_i),$$

$$\boldsymbol{\Sigma}_c^{\text{clean}} = \frac{1}{N_c - 1} \sum_{i:y_i=c} (f(x_i) - \boldsymbol{\mu}_c^{\text{clean}})(f(x_i) - \boldsymbol{\mu}_c^{\text{clean}})^\top$$

Similarly, we compute $\boldsymbol{\mu}_c^{\text{adv}}$ and $\boldsymbol{\Sigma}_c^{\text{adv}}$ for the adversarial samples.

We define the Mahalanobis distances to the clean and adversarial manifolds as:

$$D_{\text{clean}}(x, c) = \sqrt{(f(x) - \boldsymbol{\mu}_c^{\text{clean}})^\top (\boldsymbol{\Sigma}_c^{\text{clean}})^{-1} (f(x) - \boldsymbol{\mu}_c^{\text{clean}})}$$

$$D_{\text{adv}}(x, c) = \sqrt{(f(x) - \boldsymbol{\mu}_c^{\text{adv}})^\top (\boldsymbol{\Sigma}_c^{\text{adv}})^{-1} (f(x) - \boldsymbol{\mu}_c^{\text{adv}})}$$

The dual-manifold score is computed as:

$$S_{\text{dual}}(x) = D_{\text{clean}}(x, \hat{y}) - D_{\text{adv}}(x, \hat{y})$$

where $\hat{y}$ is the predicted label of x from the classifier.

3) Uncertainty Estimation (Optional): Adversarial examples often increase predictive uncertainty. We quantify this using Monte Carlo (MC) Dropout, where dropout is activated at test time to produce multiple stochastic forward passes.

For an input x, we perform T stochastic passes:

$$\{\hat{\mathbf{p}}_t\}_{t=1}^T, \quad \text{where } \hat{\mathbf{p}}_t = \text{Softmax}(f_{\text{clf}}^{\text{drop}}(x))$$

From this we calculate:

$$\bar{\mathbf{p}} = \frac{1}{T} \sum_{t=1}^{T} \hat{\mathbf{p}}_t$$

$$S_{\text{unc}}(x) = \sum_{k=1}^{|\mathcal{C}|} \left(\frac{1}{T} \sum_{t=1}^{T} (\hat{p}_{t,k} - \bar{p}_k)^2 \right)$$

This predictive variance serves as the uncertainty score. **However, if the classifier does not have dropout layers, this uncertainty-based feature cannot be computed. In such cases, only $S_{\text{dual}}(x)$ is used, which may result in lower ROC-AUC values.**

4) Detector Construction: We form a feature vector for detection as follows:

$$\mathbf{s}(x) = \begin{cases} [S_{\text{dual}}(x), S_{\text{unc}}(x)] & \text{if uncertainty is available} \\ [S_{\text{dual}}(x)] & \text{otherwise} \end{cases}$$

We then train a logistic regression classifier $h : \mathbb{R}^d \to \{0, 1\}$, where 0 indicates a clean input and 1 indicates an adversarial one.

Training data consists of:

- Clean images,
- Gaussian-noised images (for robustness),
- Adversarial examples from multiple attack types (FGSM, PGD, BIM, etc.).

The binary cross-entropy loss is minimized:

$$\mathcal{L}_{\text{det}} = -\sum_i y_i \log(h(\mathbf{s}(x_i))) + (1 - y_i) \log(1 - h(\mathbf{s}(x_i)))$$

The detector is evaluated on a held-out test set using the Receiver Operating Characteristic Area Under Curve (ROC-AUC) score, which captures its ability to distinguish clean and adversarial examples effectively.

Dual Manifold detection performs better on stronger attacks because these attacks introduce significant perturbations that push adversarial examples far from the clean data manifold. This results in clear divergence in the feature or latent space, making it easier for the dual manifold approach to distinguish between clean and adversarial inputs. In contrast, weaker attacks generate only slight perturbations that keep the adversarial examples close to the clean manifold. As a result, the separation between clean and adversarial features becomes less distinct, reducing the effectiveness of the dual manifold method and leading to lower ROC-AUC scores. Essentially, the success of the dual manifold approach depends on how far the adversarial input deviates from the learned clean manifold, which is much more noticeable in stronger attacks.

B. Purifier Module

The DUNET architecture extends the classic U-Net by incorporating lateral connections between its encoder (feed-forward) and decoder (feedback) paths. A basic unit C was defined as a 3×3 convolution followed by batch normalization [**batchnorm**] and a ReLU activation, and C_k was defined as a sequence of k such units. In the encoder, the input image was processed through five successive blocks—one C_2 followed by four C_3 blocks—in which only the first convolution of each C_3 employed a 2×2 stride (all other layers used unit stride). This configuration gradually reduced spatial resolution while increasing feature depth, yielding a hierarchy of feature maps (see the upper branch of Figure 3).

In the decoder, four blocks—three C_3 followed by one C_2—were interleaved with a final 1×1 convolution. Each block first upsampled its incoming feedback signal via bilinear interpolation to match the spatial dimensions of its corresponding encoder output, then concatenated these two tensors and processed them through the appropriate C_k unit. As the decoder progressed, spatial resolution was progressively restored. The decoder's final output passed through a 1×1 convolution to produce a tensor interpreted as the negative of the estimated noise ($-\widehat{dx}$); the denoised image was obtained by summing this negative noise with the original input.

Although DUNET was originally trained using pixel-wise reconstruction error, adversarial perturbations were observed to be amplified by deep networks even after denoising, thereby distorting high-level feature responses and leading to misclassification. To address this, a High-level Representation Guided Denoiser (HGD) was employed, replacing the pixel-level loss with a feature-reconstruction loss computed on activations of a pre-trained target network. Denoting by $f_\ell(\cdot)$ the activations at layer ℓ of the target model, the loss was formulated as

$$L = \left\| f_\ell(\widehat{x}) - f_\ell(x) \right\|_1.$$

C. Our Classifier

To evaluate the effectiveness of adversarial purification, we use two standard classifiers trained on the MNIST and CIFAR-10 datasets. These models are employed to measure accuracy on clean, adversarial, and purified inputs.

1) **MNIST Classifier:**
A lightweight convolutional neural network consisting of two convolutional layers, each followed by max-pooling and dropout, and subsequently two fully connected layers. This architecture is optimized for the task of handwritten digit recognition.
2) **CIFAR-10 Classifier:**
A deeper convolutional model comprising multiple convolutional blocks with batch normalization, max-pooling, and dropout layers, followed by three fully connected layers. This architecture is designed to capture complex patterns present in natural images.

In addition to their primary classification role, these networks also serve as feature extractors during the training of the purification model. Intermediate representations from specific internal layers are used to guide the purification process through a feature-based loss function, thereby promoting semantic alignment between clean and purified samples. The available feature extraction points include:

- **Conv:** Output from the final convolutional block before entering the dense layers.
- **Flat:** The flattened feature tensor just before the first fully connected layer (used in MNIST).
- **Pre-FC2:** The activation vector after the first fully connected layer and ReLU, but before the second FC layer.
- **Pre-FC3:** The feature vector prior to the final classification layer (used in CIFAR-10).
- **Logits:** The raw, unnormalized output from the final layer.

For our experiments, we utilize the **Pre-FC2** features as the target representation for MNIST, and the **Pre-FC3** features for CIFAR-10. By incorporating these intermediate features into the training loss, the purification model is encouraged to produce outputs that are not only visually similar to clean inputs in pixel space but also semantically aligned in the internal representation space of the respective classifiers.

V. Experimental setup

All experiments were conducted using the following hardware and software configurations:

- **Hardware:** NVIDIA GeForce GTX 1080 Ti
- **Deep Learning Framework:** PyTorch 1.12 with CUDA 11.6 acceleration
- **Supporting Libraries:** Core Libraries:
  - TorchVision 0.16.1
  - TorchAttacks 3.4.0
  - NumPy 1.26.0
  - scikit-learn 1.3.2

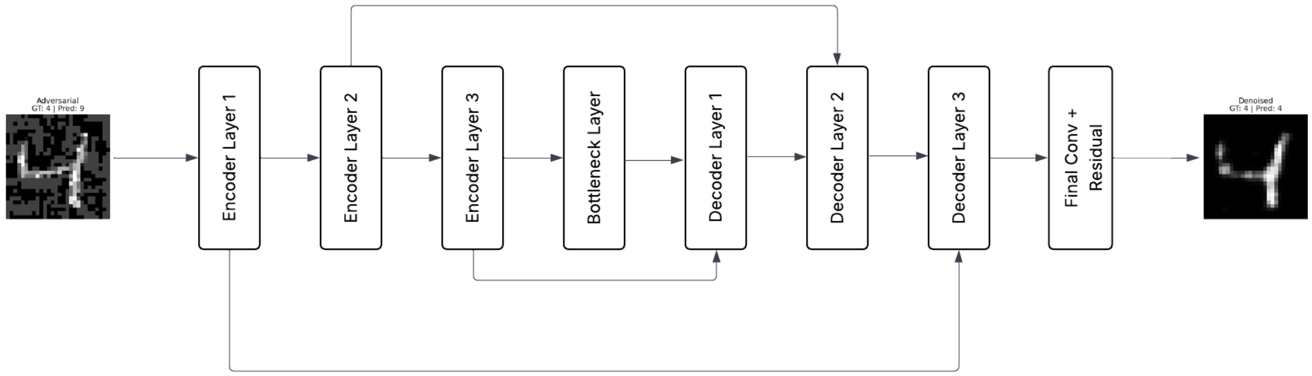

Fig. 4: Purifier Architecture.

VI. Ablation Studies

A. Denoising Autoencoder Architecture

The ConvDenoiser is a convolutional autoencoder designed to reconstruct clean images from noisy or adversarial inputs. This model serves as our foundational approach to understanding the process of image denoising.

The architecture consists of an encoder and a decoder. The encoder progressively downsamples the input image using convolutional layers with ReLU activations and max pooling operations. This compresses the image into a lower-dimensional representation. The decoder then upsamples this compressed representation using transposed convolutional layers (ConvTranspose2D) with ReLU activations, ultimately reconstructing the image to its original dimensions.

The structure of the model is as follows:

- **Encoder:** The input image of size 28×28 passes through several convolutional layers with increasing filter sizes, followed by max pooling layers to reduce spatial dimensions.
- **Bottleneck:** The most compressed representation of the image is obtained after the final pooling layer.
- **Decoder:** The compressed representation is upsampled using transposed convolutional layers, gradually restoring the image to its original size.
- **Output:** The final output is a denoised image that closely resembles the clean ground truth.

The following operations are used throughout the architecture:

- **Convolution + ReLU:** Extracts features and introduces non-linearity.
- **MaxPooling2D (2x2):** Reduces spatial dimensions and helps retain important features.
- **ConvTranspose2D + ReLU:** Upsamples the feature maps to reconstruct the image.

This autoencoder framework provides a robust baseline for learning how to remove noise from images using convolutional neural networks.

B. Loss Functions

In this section, we explore two distinct loss functions for training our denoising autoencoder architecture: pixel-wise loss and feature-wise loss. Both approaches aim to optimize the reconstruction quality of adversarially perturbed images, but they operate on different principles and yield varying results across datasets and attack types.

1) Pixel-wise Loss: We use mean squared error (MSE) between the clean input and denoised image, which quantifies the average squared difference between corresponding pixel values:

$$\mathcal{L}_{\text{pixel}} = \|x_{\text{clean}} - x_{\text{denoised}}\|_2^2$$

Pixel-wise loss functions focus on minimizing the direct difference between individual pixel intensities of the clean reference image and our model's denoised output. This approach is computationally efficient and provides a straightforward optimization objective. By minimizing the squared Euclidean distance between pixel values, the model learns to produce outputs that closely match the clean images in terms of low-level details. However, this loss function may not explicitly preserve higher-level semantic information or perceptual quality, as it treats each pixel independently without considering structural relationships.

2) Feature-wise Loss: To preserve higher-level semantics, we use a loss calculated over feature activations from a pretrained model (our custom classifier):

$$\mathcal{L}_{\text{feature}} = \sum_{l} \|\phi_l(x_{\text{clean}}) - \phi_l(x_{\text{denoised}})\|$$

Feature-wise loss leverages the internal representations from a pre-trained neural network to capture perceptual and semantic characteristics of images. In this formulation, ϕ_l represents the activation function at layer l of the pre-trained network. By computing the squared Euclidean distance between feature representations of clean and denoised images across multiple layers, the model optimizes for similarity in higher-level abstractions rather than just pixel values. This approach encourages the preservation of structural and semantic content, potentially producing reconstructions that better maintain classification accuracy even when pixel-perfect reconstruction is not achieved.

3) Experimental Results: We evaluated both loss functions on the MNIST and CIFAR-10 datasets under various adversarial attack scenarios. The following tables present our findings, where we measure the classification accuracy on clean images, adversarially perturbed images, and images after passing through our denoising autoencoder.

TABLE I: Feature-Loss Performance on MNIST Dataset

Attack	**Clean Accuracy**	**Adv. Accuracy**	**Denoised Accuracy**
FGSM	0.9950	0.1959	0.8394
PGD	0.9950	0.0079	0.9116
BIM-A	0.9950	0.0005	0.8705
BIM-B	0.9950	0.0004	0.9087

TABLE II: Feature-Loss Performance on CIFAR-10 Dataset

Attack	**Clean Accuracy**	**Adversarial Accuracy**	**Denoised Accuracy**
FGSM	0.8036	0.0338	0.2733
PGD	0.8036	0.0001	0.2514
BIM-A	0.8036	0.0001	0.2702
BIM-B	0.8036	0.0001	0.2217

TABLE III: Pixel-Loss Performance on MNIST Dataset

Attack	**Clean Accuracy**	**Adversarial Accuracy**	**Denoised Accuracy**
FGSM	0.9950	0.1959	0.8858
PGD	0.9950	0.0079	0.8411
BIM	0.9950	0.0005	0.8006

C. High-Level Guided Denoiser

To enhance adversarial robustness, we introduce a high-level guided denoiser that leverages intermediate features from a pretrained classifier. Rather than depending solely on pixel-level similarity, our approach incorporates feature-level losses computed from internal layers of the classifier. This encourages the purification process to align semantically with clean inputs.

We conducted experiments using various feature layers—*Conv*, *Flat*, *Pre-FC2*, *Logits*and *Pre-FC3*—to assess their effectiveness in guiding purification. These layers represent different semantic depths, and our results indicate that high-level features lead to better semantic preservation, especially against strong adversarial perturbations.

Figures 5, 6, 7, and 8 illustrate the denoised outputs produced by our high-level guided denoiser under different adversarial attacks (FGSM, PGD, BIM-A, BIM-B). Each figure corresponds to a specific attack, and within each figure, the columns show the denoising results guided by different classifier feature layers—ranging from low-level (*Conv*) to high-level (*Pre-FC2*, *Logits*, *Pre-FC3*). These visualizations demonstrate how the choice of feature guidance influences the visual quality and semantic fidelity of the denoised images across attack types.

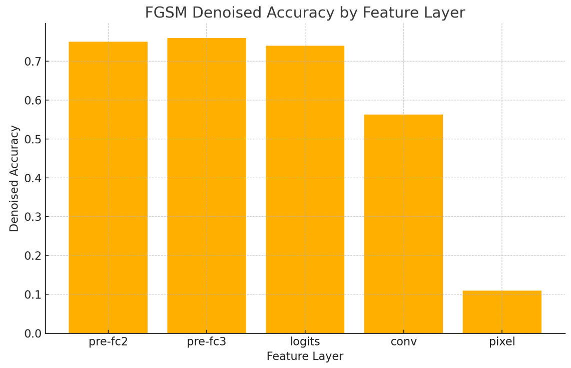

Fig. 5: Purification result using high-level feature loss on FGSM adversarial input.

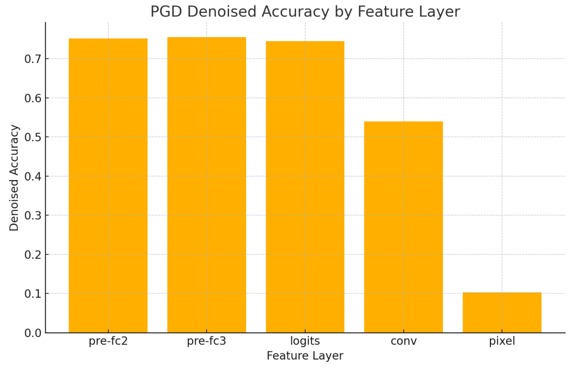

Fig. 6: Purification result using high-level feature loss on PGD adversarial input.

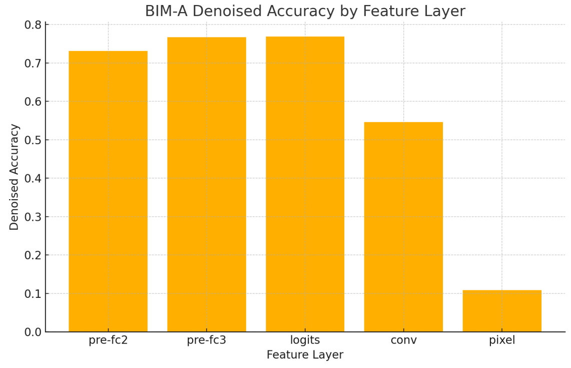

Fig. 7: Purification result using high-level feature loss on BIM-A adversarial input.

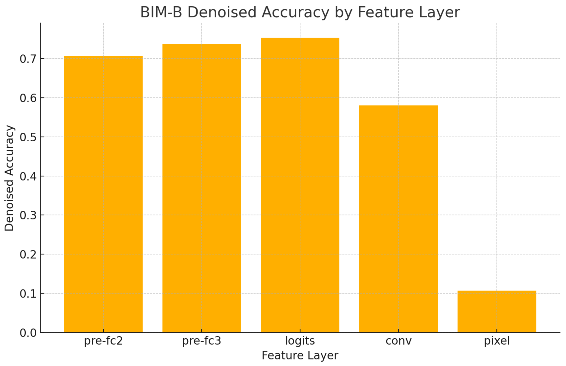

Fig. 8: Purification result using high-level feature loss on BIM-B adversarial input.

TABLE IV: Performance of High-Level Guided Denoiser on MNIST Under Different Adversarial Attacks (Pre-FC2)

Attack Type	Clean Accuracy	Adversarial Accuracy	Denoised Accuracy
FGSM	0.9864	0.1706	0.9853
PGD	0.9864	0.0027	0.9819
BIM-A	0.9864	0.1416	0.9852
BIM-B	0.9864	0.1308	0.9803

TABLE V: Performance of High-Level Guided Denoiser on CIFAR-10 Under Different Adversarial Attacks (Pre-FC3)

Attack Type	Clean Accuracy	Adversarial Accuracy	Denoised Accuracy
FGSM	0.8036	0.0338	0.7596
PGD	0.8036	0.0001	0.7552
BIM-A	0.8036	0.0001	0.7671
BIM-B	0.8036	0.0001	0.7370

D. Analysis and Discussion

1) Analysis I: Baseline Denoiser Limitations: Our initial denoiser architecture, though functional on MNIST, performed poorly on the CIFAR-10 dataset. The network is a straightforward encoder-decoder model without any skip connections. This lack of skip connections causes the network to lose vital spatial information during downsampling. As a result, the decoder struggles to reconstruct sharp or semantically consistent outputs, especially when handling complex textures and color variations in CIFAR-10 images.

Moreover, the low-capacity design, particularly in the encoder for CIFAR-10, limits its ability to capture rich feature hierarchies. Without architectural elements like residual connections or attention mechanisms, the model cannot preserve fine-grained details that are crucial for successful purification. Consequently, its denoised outputs tend to be blurry and less class-discriminative, making them ineffective against strong adversarial perturbations.

2) Analysis II: Feature-Guided Training Dynamics: When trained with a purely pixel-wise reconstruction objective (e.g., mean-squared error on RGB values), the denoiser is tasked only with minimizing raw L^2 distance to the clean image. In the presence of adversarial perturbations—small but strategically placed tweaks that leave Euclidean differences minimal—this encourages the network to average or blur away fine-grained structure, thereby eliminating the very patterns the classifier relies upon. As a result, the downstream model receives inputs devoid of the sharp edges, contrast boundaries, and texture cues it has learned to detect, and its performance collapses.

Recasting the loss at the level of early convolutional activations remedies this by aligning the denoiser's output to the classifier's own low-level feature maps. Because those feature maps capture local gradients, edge orientations, and basic textural motifs, matching them ensures that the denoised images retain the mid-level structures critical for recognition. In effect, the denoiser learns not merely to *look like* the original in pixel space, but to preserve the internal representations that form the foundation of subsequent layers' semantic abstractions.

Pushing the perceptual loss still deeper—onto activations immediately preceding fully connected layers or directly onto the classifier's logits—further refines the objective toward class-discriminative content. By enforcing agreement in those high-level embeddings or decision scores, the denoiser is compelled to restore global shape information, part relationships, and category-specific textures that the classifier exploits for its final prediction. In this manner, the hierarchy of losses—from raw pixels through convolutional features to fully connected features and logits—shifts the denoiser's focus from superficial fidelity toward the exact semantic constructs the classifier uses, yielding reconstructions that best support accurate inference.

3) Analysis III: Training Curves on MNIST: We analyze the training and validation losses during adversarial training using Pre-FC2 feature loss for BIM-A and BIM-B attacks. As observed in Figures 9 and 10, the training loss steadily decreases over epochs, indicating that the model continues to fit the training data. However, after around 15–20 epochs, the validation loss begins to plateau and fluctuate, showing no consistent improvement despite the drop in training loss. This divergence between training and validation curves suggests the model is beginning to overfit—learning patterns specific to the training data rather than generalizable features. The lack of validation improvement signals diminishing returns from further training, underscoring the need for regularization or early stopping.

E. Dataset

Our experiments utilize two widely adopted benchmark datasets in the field of adversarial machine learning: **MNIST** and **CIFAR-10**. Both datasets are chosen for their prevalence in adversarial robustness research and their suitability for evaluating detection and purification methods across varying complexity levels.

- **MNIST:** The MNIST dataset consists of 70,000 grayscale images of handwritten digits (0-9), each of size 28×28 pixels. The dataset is split into 60,000 training images and 10,000 test images. MNIST is a standard benchmark for evaluating the performance of image classification models and adversarial defense techniques due to its simplicity and well-understood structure.

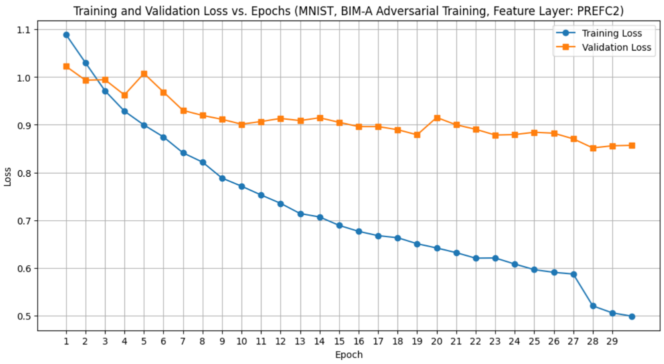

Fig. 9: Training and validation loss over epochs on MNIST (BIM-A) using Pre-FC2 feature loss.

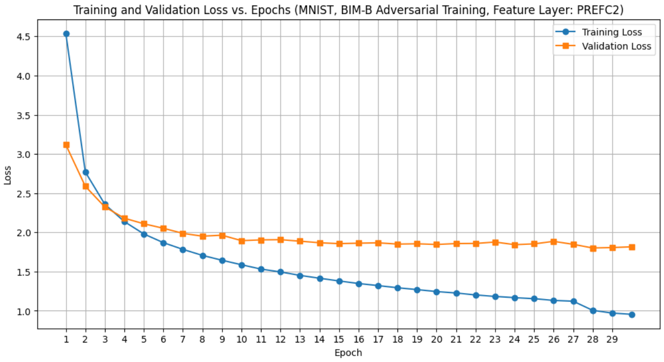

Fig. 10: Training and validation loss over epochs on MNIST (BIM-B) using Pre-FC2 feature loss.

text

- **CIFAR-10:** The CIFAR-10 dataset comprises 60,000 color images of size 32 × 32 pixels, categorized into 10 distinct object classes (such as airplanes, cars, birds, etc.). It is divided into 50,000 training images and 10,000 test images. CIFAR-10 presents a more challenging scenario for adversarial detection and purification due to its higher visual complexity and diversity of classes.

Dataset summary:

TABLE VI: Summary of Datasets Used

Dataset	Number of Classes	Training Set Size	Test Set Size
MNIST	10	60,000	10,000
CIFAR-10	10	50,000	10,000

These datasets are extensively used in adversarial machine learning research, enabling reproducibility and comparison with prior work. In our project, they serve as the foundation for generating adversarial examples (using attacks such as FGSM, PGD, and BIM), training detection modules, and evaluating the effectiveness of purification architectures

VII. Results

FGSM (single-step attack) applies a bounded, linearized perturbation (max-) to cross the decision boundary. Its perturbations are subtle and noise-like, causing adversarial examples to lie close to the natural data manifold. This leads to significant overlap in feature-space statistics (e.g., confidence scores, dropout variance), making detection challenging.

BIM-A is typically a low- version of BIM (or the early iterations). The perturbations might be just enough to shift KDE density, but not enough to shift the feature representation far from both clean and adversarial manifolds, making them harder to detect using the Dual Manifold approach.

TABLE VII: Detector ROC-AUC Scores on MNIST Dataset

Detector	FGSM	PGD	BIM-A	BIM-B
KDE-Based Detection	0.8678	0.6600	0.7881	0.8219
Dual Manifold	0.8261	0.9447	0.9223	0.8213
KDE-Based Detection (Dropout)	0.9057	-	0.9723	0.8206
Dual Manifold (Dropout)	**0.9785**	**0.9861**	**0.9466**	**0.9459**

Observation: The Dual Manifold method with dropout consistently outperforms all other detectors, achieving near-perfect ROC-AUC on strong attacks like PGD and BIM-A.

BIM-A is typically a low- version of BIM (or the early iterations). The perturbations might be just enough to shift KDE density, but not enough to shift the feature representation far from both clean and adversarial manifolds, making them harder to detect using the Dual Manifold approach.

TABLE VIII: Detector ROC-AUC Scores on CIFAR10 Dataset

Detector	FGSM	PGD	BIM-A	BIM-B
KDE-Based Detection (Dropout)	0.7223	-	0.8105	0.9441
Dual Manifold (Dropout)	**0.7901**	**0.9882**	**0.9223**	**0.9770**

Observation: Similar trends are observed for CIFAR10, with Dual Manifold (Dropout) showing robustness across all attack types.

The Dual Manifold method, depending on its implementation, might be particularly good at detecting PGD attacks due to how it captures differences between clean and adversarial distributions. Since PGD involves iterative perturbations, its features may be more distinct from clean examples compared to other attacks like FGSM or BIM. As a result, your detector may have become sensitive to these specific perturbations, leading to a higher ROC-AUC score.

TABLE IX: High-Level Guided Denoiser with Feature-Loss: MNIST

Attack	Clean Accuracy	Adversarial Accuracy	Denoised Accuracy
FGSM	0.9864	0.0338	0.9853
PGD	0.9864	0.0001	0.9819
BIM-A	0.9864	0.0001	0.9852
BIM-B	0.9864	0.0001	0.9803

Observation: Feature-loss based denoising is highly effective for MNIST, recovering near-original accuracy across all attacks.

Observation: Denoising improves performance under all attacks, though there is a slight drop from clean accuracy, especially under BIM-B.

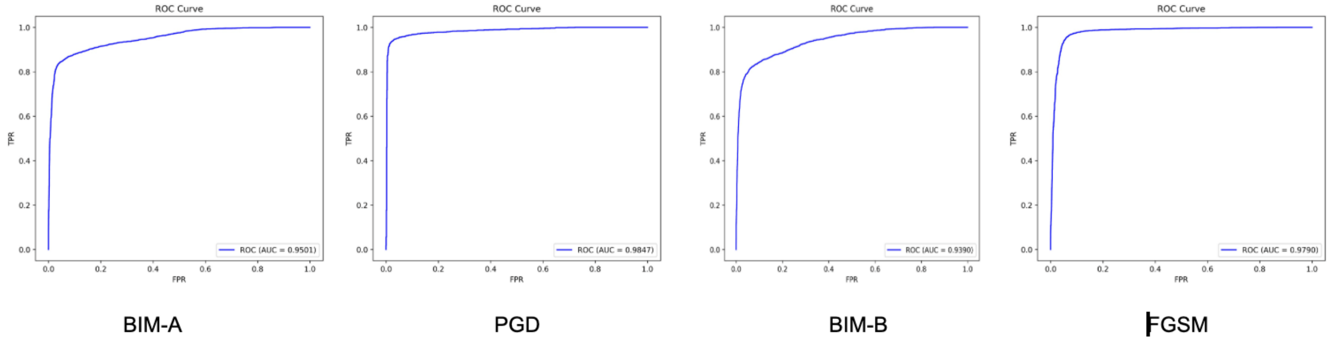

Fig. 11: ROC-AUC curves for the dual-manifold adversarial detector on CIFAR-10 under different attack types: BIM-A, PGD, BIM-B, and FGSM.

TABLE X: High-Level Guided Denoiser with Feature-Loss: CIFAR10

Attack	Clean Accuracy	Adversarial Accuracy	Denoised Accuracy
FGSM	0.8036	0.0338	0.7596
PGD	0.8036	0.0001	0.7552
BIM-A	0.8036	0.0001	0.7671
BIM-B	0.8036	0.0001	0.7370

TABLE XI: High-Level Guided Denoiser with Pixel-Loss: CIFAR10

Attack	Clean Accuracy	Adversarial Accuracy	Denoised Accuracy
FGSM	0.8036	0.0338	0.1094
PGD	0.8036	0.0001	0.1029
BIM-A	0.8036	0.0001	0.1081
BIM-B	0.8036	0.0001	0.1067

Observation: Pixel-loss based denoising fails to recover performance on CIFAR10, highlighting the importance of semantically meaningful loss functions.

TABLE XII: Detector accuracy and threshold values for different adversarial attacks on CIFAR-10

Attack Type	Detector Accuracy (%)	Threshold Value
FGSM	76.65	0.317
PGD	96.90	0.474
BIM-A	75.45	0.371
BIM-B	94.55	0.543

We evaluated the detector using 1,000 adversarial and 1,000 clean samples, measuring the accuracy based on threshold values specific to each attack, as reported in Table XII.

Figure 11 shows the ROC-AUC curves for the adversarial detector across multiple attack types. The consistently high AUC values confirm the effectiveness of the detector in distinguishing adversarial inputs from clean data.

VIII. Discussion and Future work

For future work, we aim to enhance the robustness of our detector by removing the reliance on dropout layers, thereby exploring alternative architectures such as ResNet. Additionally, we plan to extend the generalization capability of our model to effectively handle a wider range of adversarial attacks. By incorporating diverse attack strategies during training and evaluation, we aim to build a more resilient and universally applicable defense mechanism against adversarial threats.Furthermore, we will explore domain adaptation techniques to ensure the model remains effective across different datasets and real-world scenarios.

IX. Conclusion

We presented a unified adversarial defense framework that integrates Bayesian uncertainty-based detection with high-level feature guided denoising. This joint approach enables robust identification and purification of adversarial examples while preserving semantic fidelity. Experiments on MNIST and CIFAR-10 demonstrate strong performance across multiple attack types, significantly improving adversarial accuracy without compromising clean input performance. [1]

D. Kalaria, D. Vasisht, Y. Sharma, and A. Singla, "Towards Adversarial Purification using Denoising AutoEncoders," *arXiv preprint arXiv:2202.03410*, 2022.

F. Liao, M. Liang, Y. Dong, T. Pang, J. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided denoiser," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 1778–1787.

Whole bibliography

[1] Reuben Feinman et al. *Detecting Adversarial Samples from Artifacts*. 2017. arXiv: 1703.00410 [stat.ML]. URL: https://arxiv.org/abs/1703.00410.

[2] Susmit Jha et al. "Detecting Adversarial Examples Using Data Manifolds". In: *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*. 2018, pp. 547–552. DOI: 10.1109/MILCOM.2018.8599691.

[3] Fangzhou Liao et al. *Defense against Adversarial Attacks Using High-Level Representation Guided Denoiser*. 2018. arXiv: 1712.02976 [cs.CV]. URL: https://arxiv.org/abs/1712.02976.

[4] Weilin Xu, David Evans, and Yanjun Qi. “Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks”. In: *Proceedings 2018 Network and Distributed System Security Symposium*. NDSS 2018. Internet Society, 2018. DOI: 10.14722/ndss.2018.23198. URL: http://dx.doi.org/10.14722/ndss.2018.23198.