Self-Healing Neural Networks via Modular Patch Layers for Diverse Structural and Adversarial Damages

S. Deepa Nivethika
School of Computer Science and Engineering,
Vellore Institute of Technology, Chennai, India
Email: deepanivethika.s@vit.ac.in

B. Santhosh Reddy
School of Computer Science and Engineering,
Vellore Institute of Technology, Chennai, India
Email: beeram.santhosh2022@vitstudent.ac.in

B. Vara Anjan
School of Computer Science and Engineering,
Vellore Institute of Technology, Chennai, India
Email: bankivara.anjan2022@vitstudent.ac.in

P. Mahidhar Reddy
School of Computer Science and Engineering,
Vellore Institute of Technology, Chennai, India
Email: pengani.mahidhar2022@vitstudent.ac.in

Abstract

Deep neural networks (DNNs) have shown superior performance in most areas but remain highly vulnerable to structural damage and adversarial attacks, making safety-critical applications risky. This paper proposes the Self-Healing Neural Network (SHNN) system, a single platform that can simulate, detect, and recover automatically from both internal structure degradation and external adversarial attacks. A baseline feedforward neural network is first trained on MNIST with three hidden Dense layers and a softmax output. Structural flaws are artificially induced by perturbing the weights of single layers randomly through zeroing or random reinitialization, corresponding to real-world faults like memory corruption or pruning mistakes. The impaired layer is located by means of a deviation metric based on mean absolute error for reference and post-damage activations between both.

To restore from structural errors, the model trains a light neural "patch" which learns to approximate the original output of the broken layer without training the remainder of the network. In adversarial robustness, adversarial attacks like FGSM and PGD are used to create perturbed inputs. The most vulnerable internal layer is determined through activation shift analysis, and then adversarial retraining is performed using a similar patching mechanism.

A graphical user interface based on Streamlit enables interactive real-time visualization and experimentation. It is easy to simulate damage, monitor intermediate activations, and invoke healing mechanisms with little effort. The SHNN approach provides an interpretable and modular route for augmenting AI resilience that is useful for research as well as education in resilient deep learning.

Index Terms

Self-Healing Neural Networks, Structural Damage, Adversarial Robustness, Patch-Based Recovery, Deep Learning Resilience, Activation Analysis, FGSM, PGD, Keras, Streamlit

I. Introduction

A. Motivation: Vulnerabilities in Deep Neural Networks

While their revolutionary success in domains such as computer vision, natural language processing, and healthcare, deep neural networks (DNNs) remain susceptible to real-world operational stresses. Two broad types of threats — structural damage and adversarial attacks — uncover intrinsic weaknesses in these systems.

Structural degradation implies internal defects such as memory corruption, weight perturbation caused by hardware aging, or accidental model pruning. Such forms of degradation, while widely neglected, can silently deteriorate a model's internal computations and lead to a steep performance decline. In real-time applications, retraining from scratch is impossible, particularly when original data or computing resources are not present. Hence, there is an immediate need for localized and efficient self-repair mechanisms.

At the same time, DNNs are extremely vulnerable to adversarial attacks — minute, precisely designed input perturbations that result in misclassification with imperceptible visual alteration. Techniques like FGSM and PGD take advantage of gradient sensitivity to design such perturbations. More insidious are adversarial patch attacks, where localized areas of the input (such as stickers on images) are capable of taking over predictions completely. These attacks defeat standard defenses and generalize across models and datasets [1]. Even advanced classifiers can be persistently deceived, giving rise to serious concerns about the use of DNNs in safety-critical environments. Current defense methods such as PatchZero [1] and DIFFender [2] show growing needs for the effective detection and neutralization of such attacks. Addressing these issues involves models capable of self-detection, isolation, and recovery from internal faults along with external manipulation, thereby enhancing trust and integrity in AI systems.

B. Structural Damage in Neural Networks

Deep neural networks are typically taken to work reliably after deployment, but such an assumption ceases to hold in the presence of real-world limitations like compromised hardware integrity, energy efficiency needs, and model life cycle limitations. Structural degradation means unintended modification of a network's architecture or internal parameters, frequently caused by hardware defects, memory corruption, or extreme model optimization methods like pruning and quantization. These damages are not the result of hostile inputs but are inherent in the operating environment or post-deployment changes.

Hardware-level phenomena like bit-flips caused by cosmic rays, thermal degradation of memory modules, or transient voltage fluctuations can quietly modify stored weights in model layers. Analogously, edge devices and embedded systems, at which deep learning models are increasingly becoming deployed, are especially susceptible to such disruptions because they have limited hardware and are not revalidated often. Even small corruptions in the model weights over time can lead to cascading errors during inference, degrading the performance of the model without revealing any overt failure flags.

Moreover, techniques for reducing model efficiency optimally — such as layer pruning, weight sharing, and quantization — can accidentally remove or warp significant parameters when employed without specific caution. In these situations, it becomes a problem of non-trivial complexity to determine which layer has been affected and to restore its behavior. In such cases, identifying which layer has been impacted and reconstructing its behavior becomes a task of non-trivial complexity.

Whereas other repair tools such as NNrepair [11] and QNNRepair [13] provide constraint-based recovery approaches, these may necessitate external specifications or retraining. Hence, a real-time, self-contained repair mechanism is imperative for guaranteeing ongoing reliability in deployed neural networks in different environments.

C. Adversarial Attacks and Model Fragility

While structural errors endanger the internal consistency of deep neural networks, adversarial attacks are even more subtle since they target the very input space that the models are working with. Even the best classifiers can be fooled by input manipulation so minimal that it is completely imperceptible to humans. The designed adversarial perturbations use gradient-based optimization to take advantage of model weaknesses in small input changes, which results in confident misclassification.

The Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD) were among the first methods to demonstrate neural model vulnerability. As defenses grew stronger, so did attacks—beating small pixel-level perturbations to adversarial patches, where salient but localized modifications to the input can override the model's choice. Such patches are especially unsafe in that they tend to be universal (i.e., actionable on any inputs and models) without access to the complete input. Recent works such as PatchZero [1] and DIFFender [2] have highlighted how adversarial patches can mislead even resilient models used in public environments.

Neural networks lack natural robustness because their decision boundaries become easily modifiable through changes in model weights or structural modifications. The susceptibility of neural networks to modification reduces trust in AI systems when they are used in critical applications such as surveillance, autonomous vehicles, and healthcare diagnosis. The defense strategies must move past basic input pre-processing and adversarial training because they need to study the internal dynamic processes that enable models to detect and recover from targeted attacks.

D. Need for Self-Healing Mechanisms

The expanding application of deep learning models to real-world and safety-critical domains necessitates a paradigm shift—not just the enhancement of accuracy, but guaranteeing resilience and fault tolerance. Existing methods for coping with structural damage or adversarial attacks are, in large part, static, dependent on retraining, defensive preprocessing, or ensemble-based robustness. Partial protection can be achieved using such methods, but they are not adaptive and depend heavily on computational or data resources. In addition, they take for granted pre-existing knowledge of the nature of failure or attack, which is not often present in uncertain real-world environments.

The human body demonstrates superior self-repair capabilities compared to these strict methods. The system detects local damage before it separates the affected area for reshaping without requiring external assistance. The application of this concept to artificial neural networks could produce systems that detect internal failures and adversary attacks while performing autonomous recovery. The hope is to construct neural models that are able to police their own activations, identify flaws in layer-by-layer calculations, and insert correcting patches or modules that correct behavior without disturbing the remainder of the network.

This type of strategy has two principal benefits: modularity and efficiency. Isolating the repair to the specific layer or area makes the healing process lightweight and quick—without retraining in its entirety or massive intervention. This flexibility also brings enhanced robustness against unforeseen or changing threats, enabling models to be more reliable in deployment over the long term. As emphasized in papers such as NNrepair [11] and CARE [14], dynamic and interpretable repair methods are the future for resilient AI systems.

E. Proposed Framework: SHNN

In order to meet the two-fold challenge of internal sabotage and external enemy tampering, we introduce a comprehensive approach known as the Self-Healing Neural Network (SHNN). This framework is intended to equip deep learning models with the power to self-detect, localize, and recover from performance-sucking conditions—whether they are brought about by hardware-caused faults or adversarial input perturbations. In contrast to retraining- or brute-force-defense-based approaches, SHNN presents a patch-based, modular healing strategy that emulates the selective recovery phenomena of biological systems.

The central concept of SHNN is to constantly observe the internal behavior of a trained model by layer-wise activation patterns. Whenever, through a corruption of weights or an adversarial input, there is a substantial deviation from the model reference behavior, the system initiates a diagnostic phase to detect the most damaged layer. After identification, a small auxiliary neural patch is trained to mimic the pre-damage output of the defective layer, employing frozen weights in the rest of the model to allow for isolated healing. This patch is then added to the model, bringing its functionality back without retraining.

For adversarial cases, SHNN employs the same procedure, utilizing both clean and adversarial instances to learn a hybrid patch that heals perturbed activations. This two-pathway repair process—for both structural and adversarial damage—makes SHNN an attack-agnostic, flexible framework. Additionally, by integrating backend healing logic with an interactive Streamlit UI, SHNN provides both research convenience and pedagogical benefit, enabling users to simulate, visualize, and grasp fault recovery in real time.

F. Key Features and Contributions

The Self-Healing Neural Network (SHNN) offers a new way to improve the reliability and flexibility of deep learning models. It does this by including fault detection, damage localization, and focused repair. A key feature of this framework is the layer-wise diagnostic method. This approach matches real-time activations with baseline references to find problems caused by structural faults or adversarial changes. In contrast to classical retraining-based techniques, SHNN reacts by learning a lightweight neural patch, targeting only the impaired layer, with less computational overhead and leaving the other part of the model intact.

A second significant contribution is the dual-mode healing system in SHNN, which can restore from internal weight corruption as well as external adversarial attacks. Through concurrent usage of clean and adversarial inputs, the system enhances robustness at no expense to accuracy on legitimate data. The system is also characterized by its interactive Streamlit GUI, which offers accessible controls for applying damage, visualizing model degradation, and observing the healing process in real time.

In summary, SHNN provides a modular, efficient, and understandable solution for recovering neural networks. It closes the gap between structural fault tolerance and adversarial defense. The approach offers a complete, real-time method for keeping model performance strong against various challenges.

II. Literature Review

A. Patch Attack Protections

PatchZero [1]: Xu et al. suggest a way to detect and address adversarial patches by identifying them via aberrant gradients and then eliminating them using inpainting. Their work is geared towards completely removing resistance but they do not retrain the model. It computes local changes in the gradient and identifies local jumps in the gradient behavior, in a so-called sliding window fashion. The gradient-based anomaly detection assists to make the patch region location as clear as possible, which is subsequently inpainted by GAN-based image completion algorithms so that the integrity of the original content is preserved.

DIFFender [2]: DIFFender is an adversarial element removal method through a generative denoising model proposed by Kang et al. [2]. The essence is to restore clean images from noisy ones without going through the adversarial effects. Contrary to naive input purification, the results of DIFFender indicate that diffusion models trained in a natural setting naturally reduce adversarial noise. The method is effective on many datasets and under high patch attacks.

PATCHOUT [3]: Simon et al. focus on semantic consistency using the PATCHOUT method. Their contrastive learning method compares sub-level (class-level) consistency among many random crops of an input image to find inconsistent areas, likely due to adversarial patches. PATCHOUT can detect and locate patches with high precision because of semantically coherent characteristics strengthening while penalizing deviations.

AdvPatchXAI [4]: Kumar et al. propose AdvPatchXAI, an integrated explainable and robust detection system. The framework combines Grad-CAM and patch classifiers to determine regions disproportionately affecting model choices. This approach is visually interpretable and uses attention mechanisms to provide transparent AI systems.

B. Model-Specific Defense Mechanisms

Segment and Complete [5]: Liu et al. present a segmentation-based preprocessing module that identifies potential patch areas in an image using semantic segmentation and applies inpainting to reconstruct clean object images. This model-agnostic approach is particularly effective against attacks on object detectors.

PAD (Patch-Agnostic Defense) [6]: Jing et al. propose PAD, which does not require knowledge of patch size, location, or pattern. It uses patch suppression layers to reduce local saliency through multiple partial gradients. PAD is generalizable across models and data, enabling real-world application without prior knowledge of attack specifics.

Real-world Diffusion Defense [7]: Wei et al. leverage diffusion models for robustness in real-world conditions. They include robustness-aware sampling to alter noise schedules for better recovery of clean images, effective even in low-light or occluded settings where adversarial patches are applied physically.

Model-Agnostic UAV Defense [8]: Pathak et al. present a model-agnostic defense for UAV-based object detection, combining patch filtering with feature suppression. A patch detection phase precedes feature extraction, followed by confidence-responsive patch nullification, maintaining detector performance without architectural retraining.

RFENet [9]: Zhang et al. propose RFENet, which enhances feature extractor robustness by detecting inconsistent gradients and amplifying semantically consistent ones. It combines multi-scale feature fusion and robust loss functions for intrinsic resistance to patch perturbations in aerial imagery.

C. *Repair and Self-Healing Neural Networks*

Closed-Loop Self-Healing [10]: Chen et al. implement a closed-loop feedback system for real-time neural network adaptation. The system monitors output anomalies and activates corrective modules to adjust internal activations, combining control theory with neural networks for real-time inference recovery.

NNRepair [11]: Usman et al. propose NNRepair, a constraint-based method that identifies and corrects errant inputs by minimally adjusting network weights. This fine-grained repair ensures high fidelity without major retraining.

AIREPAIR [12]: Song et al. develop AIREPAIR, an automatic network repair framework with a graphical interface and optimized repair mechanisms such as neuron pruning, weight reparameterization, and layer duplication. It provides iteration-based feedback to ensure the fixed network retains original functionality.

QNNRepair [13]: For quantized networks, Song et al. introduce QNNRepair, using gradient-sensor-based constrained optimization for precise recovery of quantized feature models. The method injects repair logic into quantization processes to maintain robustness and accuracy.

CARE [14]: Castillo et al. use Structural Causal Models (SCM) to identify root causes of failures in network graphs. CARE applies minimal interventions in output layers without altering causal connections, including interaction adjustment or neuron fixing.

Semantic-Based Repair [15]: Xie et al. propose semantic-based repair, correcting activations while considering concept vectors in training data. This maintains class-level boundary separability and avoids introducing new vulnerabilities post-repair.

D. *Cross-Cutting Policy Issues*

Trust and Protection: AdvPatchXAI [4] and CARE [14] emphasize combining XAI with defense strategies to maintain trust and performance.

Diffusion Denoising Bias: Wei et al. [7] and DIFFender [2] highlight the potential bias of diffusion denoising models, which can be pre-trained and reused across various applications.

Repair Without Retraining: Lightweight repair strategies like NNRepair [11] and AIREPAIR [12] allow fixing models without full retraining, which is crucial when training data or computational resources are limited.

Patch Localization Prior to Mitigation: Surgical intervention strategies, such as PATCHOUT [3] and PAD [6], localize patches before mitigation to preserve classifier integrity and image quality.

UAVs and Remote Sensing: Studies such as Pathak et al. [8] and RFENet [9] address the deployment of UAV and remote sensing models under adverse conditions, highlighting the importance of robust patch defenses.

III. Methodology

This section introduces the architecture design, damage simulation strategy, detection method, and healing mechanism integrated into the Self-Healing Neural Network (SHNN) system. The framework is designed to localize and heal structural as well as adversarial defects in a neural network via patch layers with a modular design. All aspects are coded in Python using Keras for model implementation and Streamlit for the graphical user interface.

A. *Base Model Architecture*

A feedforward neural network implemented with the MNIST digit recognition dataset forms the core of SHNN. MNIST contains 60,000 training and 10,000 test grayscale images of handwritten digits, each 28 × 28 pixels. Each image is mapped to a 784-dimensional vector for computational convenience.

The model structure is as follows:

- **Input Layer:** Accepts input vectors of size 784.

- **Hidden Layers:** Three Dense (fully connected) layers, `dense0`, `dense1`, and `dense2`, each followed by a ReLU activation.
- **Output Layer:** A Dense layer of 10 units with softmax activation, returning class probabilities.

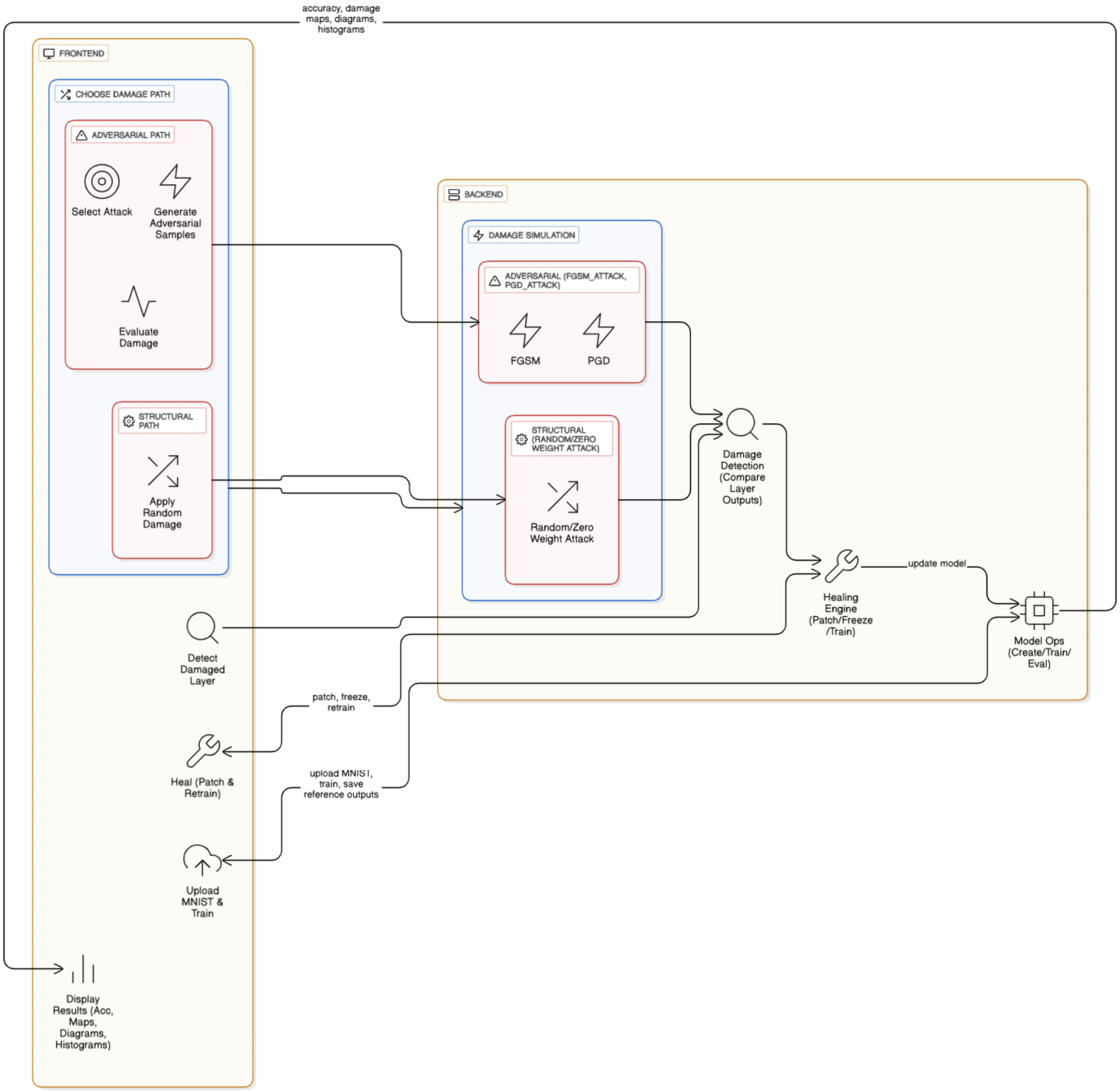

accuracy, damage maps, diagrams, histograms
FRONTEND
CHOOSE DAMAGE PATH
ADVERSARIAL PATH
Select Attack
Generate Adversarial Samples
Evaluate Damage
STRUCTURAL PATH
Apply Random Damage
Detect Damaged Layer
patch, freeze, retrain
Heal (Patch & Retrain)
upload MNIST, train, save reference outputs
Upload MNIST & Train
Display Results (Acc, Maps, Diagrams, Histograms)
BACKEND
DAMAGE SIMULATION
ADVERSARIAL (FGSM_ATTACK, PGD_ATTACK)
FGSM
PGD
STRUCTURAL (RANDOM/ZERO WEIGHT ATTACK)
Random/Zero Weight Attack
Damage Detection (Compare Layer Outputs)
Healing Engine (Patch/Freeze /Train)
update model
Model Ops (Create/Train/ Eval)

Fig. 1: System architecture of the proposed Self-Healing Neural Network (SHNN) framework.

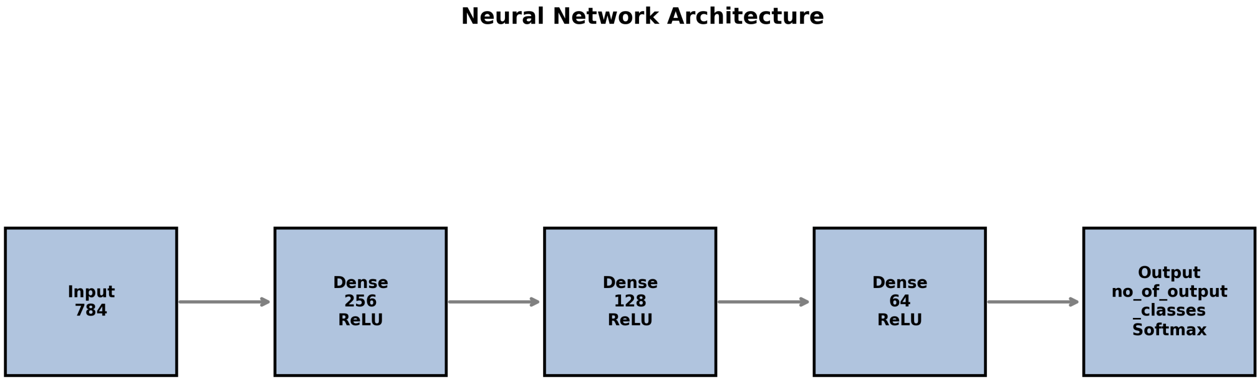


Fig. 2: Neural network architecture of the proposed Self-Healing Neural Network (SHNN), showing the input layer, three hidden dense layers, and the output layer.

The model is trained using categorical cross-entropy loss and the Adam optimizer. Training is performed for a short period (e.g., 10 epochs) to establish baseline classification accuracy. Learned weights and mid-layer outputs are stored to serve as references during the healing process.

B. Structural Damage Simulation

Artificial damage is simulated on the trained model to test network resilience under internal faults. The simulated damages reflect real-world issues such as hardware faults, faulty memory segments, extreme network pruning, or quantization defects.

1) Damage Modes: The `apply_structural_damage()` function introduces faults into the model. It randomly selects one of the four layers (`dense0`, `dense1`, `dense2`, `output`) and applies one of the following damage types:

- **Zero-Out Damage:** The weights and biases of the targeted layer are set to zero, causing the layer to output a constant vector and nullifying its learned representation.
- **Random Perturbation:** The layer weights are replaced with small random values sampled from a uniform distribution (e.g., $U(-0.05, 0.05)$), simulating noise injection or unintended re-initialization.

2) Motivation for Simulation: Structural damage simulation acts as a surrogate for unforeseen, localized internal faults during deployment. Unlike adversarial inputs, which attack from the outside, structural damage modifies the internal computation of the model. Localized diagnosis and patch-based repair are therefore suitable strategies.

C. Damage Localization via Activation Comparison

After structural damage, model performance typically deteriorates. Instead of reloading or retraining the entire model, SHNN attempts to locate and heal only the compromised portion.

1) Layer-Wise Output Tracking: A small batch of test images is used to compute layer-wise outputs (activations) for both:

- The original, undamaged model.
- The structurally damaged model.

2) Detection of Damaged Layer: The layer exhibiting the highest Mean Absolute Error (MAE) between clean and damaged activations is identified as the most damaged. This label-free diagnostic is stable across different types of structural damage and enables targeted healing.

D. Healing Structural Damage with Modular Patches

Once the damaged layer is located, a modular patch is added to restore its outputs without affecting the rest of the model. This modularity draws inspiration from biological repair mechanisms and dynamic architectures in machine learning.

1) Patch Architecture: The patch is a lightweight neural network—typically 1–2 Dense layers—placed before the damaged layer. It transforms inputs to the damaged layer so that downstream computations approximate the clean model's behavior.

2) Patch Training Procedure:

1) **Input to Patch:** Inputs to the damaged layer from the clean model (before damage).
2) **Target Output:** Corresponding activations of the damaged layer before damage.
3) **Model Configuration:** All base model layers are frozen to prevent weight updates.
4) **Training Objective:** Train the patch to minimize the discrepancy between patch output and original activations (e.g., MAE or MSE).
5) **Integration:** Insert the trained patch into the damaged model and re-evaluate performance on the test set.

This localized healing restores performance with minimal retraining and avoids full model resets.

E. Simulation of Adversarial Attack and Recovery

SHNN also addresses adversarial attacks, representing external manipulations designed to deceive the neural network with imperceptible perturbations.

1) Attack Strategies: Two adversarial attack methods are implemented:

- **FGSM (Fast Gradient Sign Method):** A one-step gradient-based perturbation that maximizes the model loss.
- **PGD (Projected Gradient Descent):** An iterative extension of FGSM with projections onto the allowed ϵ-ball around the input.

2) Adversarial Damage Detection: Adversarial inputs induce shifts in activations throughout the network, rather than changing weights. The SHNN framework uses activation comparison between:

- Clean inputs, and
- Adversarially perturbed inputs.

The layer exhibiting the highest average deviation is identified as the most impacted and becomes the target for patch-based healing.

F. Healing Adversarial Damage

The procedure for healing adversarial damage parallels structural fault repair, with some additional considerations:

1) **Input Dataset:** Use a combination of clean and adversarial examples.
2) **Patch Insertion:** Place a lightweight patch prior to the most affected layer.
3) **Training Objective:** Train the patch to restore clean-like activations when exposed to adversarial inputs.
4) **Loss Function:** Same as structural patch training, but applied to adversarially perturbed activations.
5) **Evaluation:** Test the patched model on both clean and adversarial sets to assess robustness improvement.

This approach allows the network to learn robust internal transformations, compensating for adversarial noise while preserving accuracy on clean data.

G. System Integration and GUI Interface

The entire SHNN methodology is integrated into an interactive Streamlit-based GUI, enabling users to:

- Upload datasets (`.mat` format),
- Train the base model,
- Apply adversarial attacks,
- Train and insert healing patches,
- Visualize performance metrics and layer-wise activations.

The GUI supports side-by-side comparisons of accuracy, patch training loss, and pre/post-healing performance using `Matplotlib` and `Seaborn` visualizations.

IV. Results

A. Overview of Experimental Setup

All experiments were implemented in Python using TensorFlow 2 within the Self-Healing Neural Network (SHNN) framework. The base model is a vanilla feedforward fully connected deep network trained on MNIST, with an input layer of 784 units (flattened 28×28 images), three hidden Dense layers with 256, 128, and 64 units respectively, and a final softmax output layer with ten classes. All hidden layers use ReLU activation, and the output layer uses softmax to compute class probabilities. The model was trained for 10 epochs using categorical cross-entropy loss, Adam optimizer with learning rate 0.001, and batch size of 128.

We evaluated two types of performance degradation:

1) **Adversarial Attacks:**

- **Fast Gradient Sign Method (FGSM):** Perturbations were generated with $\epsilon = 0.15$, exploiting the gradient of the model with respect to input, causing maximal misclassification by altering a minimal number of pixels.
- **Projected Gradient Descent (PGD):** Iterative multi-step attack with $\epsilon = 0.2$, step size $\alpha = 0.01$, and 30 iterations, producing stronger perturbations than FGSM through repeated gradient-based modifications.

2) **Structural Damage:**

- **Zeroing:** Weights of the selected Dense layer are set to zero.
- **Random Perturbation:** Weights in the selected Dense layer are replaced with small Gaussian noise ($\sigma = 0.1$).

For each scenario, SHNN's damage detection module identified the most affected layer by comparing layer-wise activations of undamaged and damaged models using Mean Absolute Differences (MAD). The detected layer was replaced with a freshly initialized patch, inserted into the original network with all other layers frozen. The patch was trained on a subset of the training data (10,000 samples) using the same optimizer and loss function for quick adaptation.

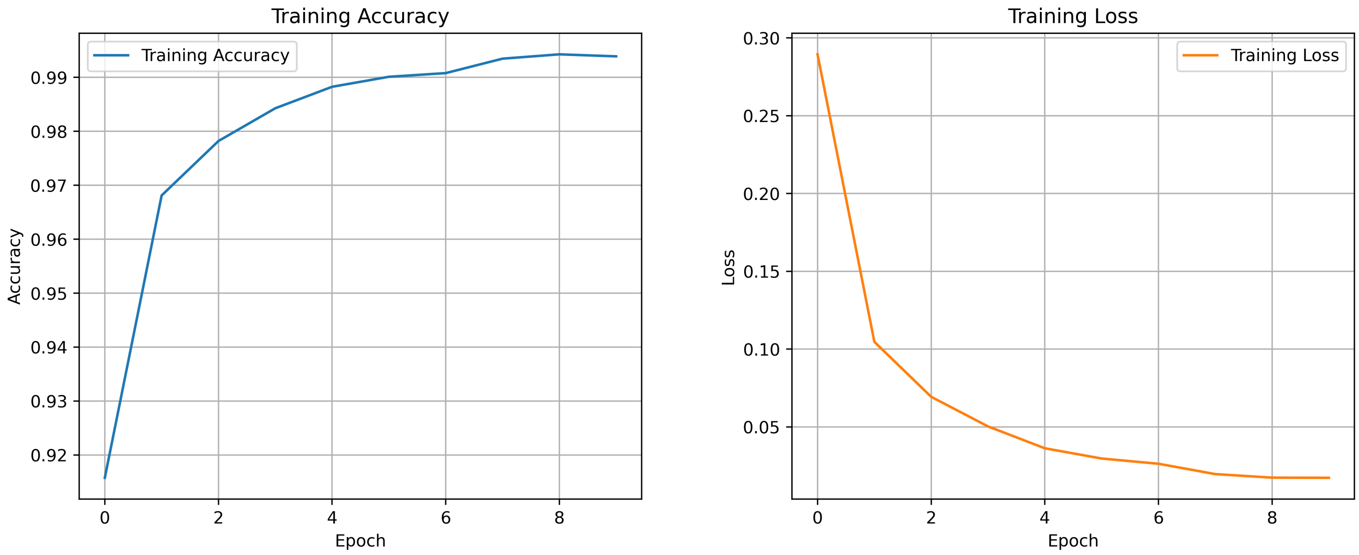
Performance metrics were recorded in three phases:

- **Baseline Accuracy** – accuracy in the undisturbed state.
- **Damaged Model Accuracy** – accuracy after damage or adversarial perturbation.
- **Post-Healing Accuracy** – accuracy after the self-healing process.

Results were averaged over 10 independent runs, with random initialization of damage location and perturbation seeds to ensure statistical robustness.

B. Baseline Model Performance

The SHNN baseline was trained on MNIST for 10 epochs with batch size 128, categorical cross-entropy loss, and the Adam optimizer. A short training period was chosen to allow rapid simulation of multiple damage and healing cycles efficiently.



(a) Training accuracy progression.

(b) Training loss progression.

Fig. 3: Baseline model training performance over 10 epochs. (a) Accuracy progression, (b) Loss progression.

The baseline model after training yielded the following performance:

Metric	Value (%)
Training Accuracy	99.43
Test Accuracy	97.76

TABLE I: Baseline Model Performance

Despite the brief training timeframe, the model achieved 99.43% accuracy on the training set and 97.76% on the test set, providing a solid benchmark for comparison against structural and adversarial damage as well as post-healing performance.

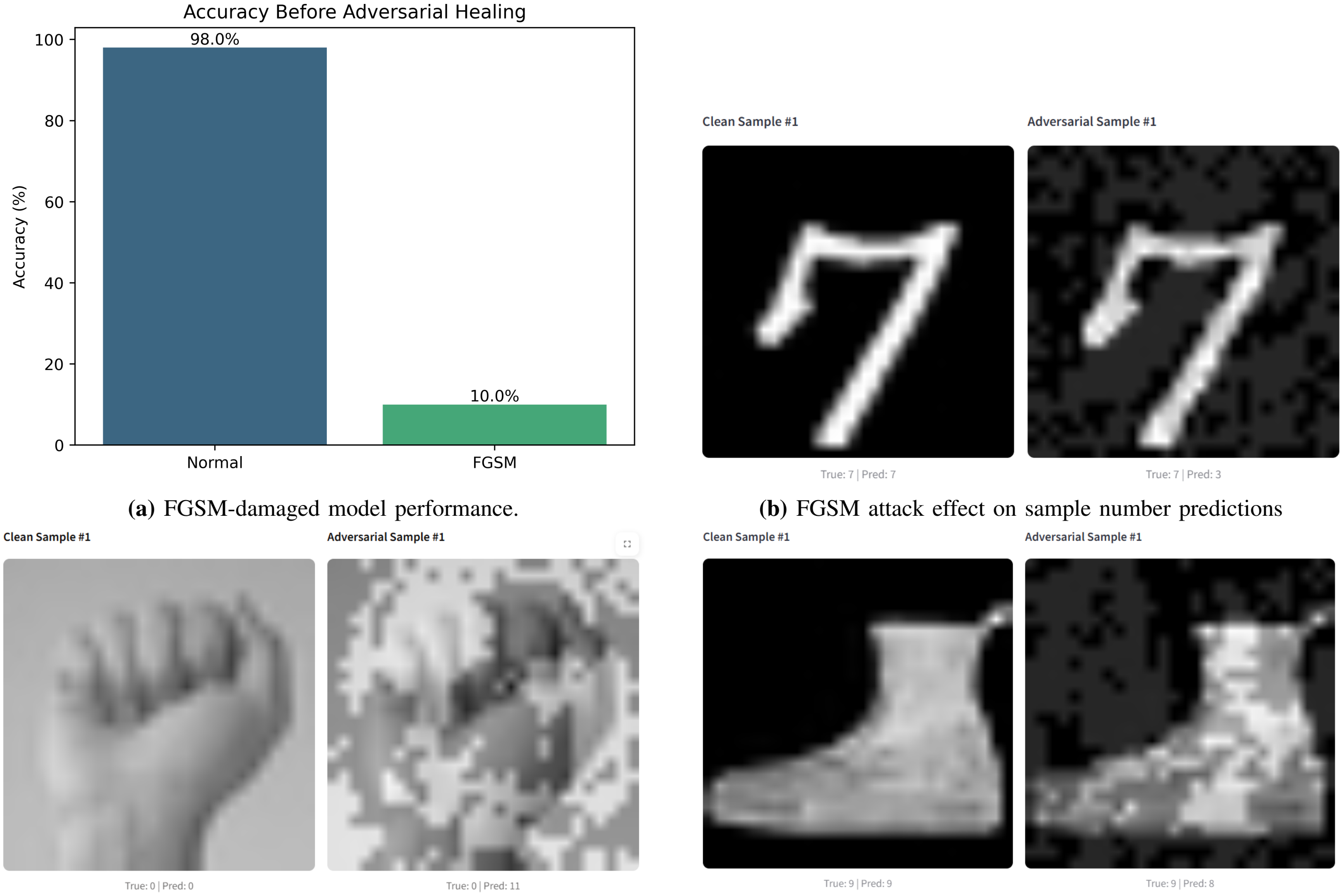
C. Impact of Adversarial Attacks

Two classical adversarial attack strategies—Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD)—were applied on the trained baseline model to evaluate SHNN's robustness. Each attack was performed on the first 100 test samples.

1) FGSM Attack Results: Test accuracy dropped significantly under FGSM perturbations ($\epsilon = 0.15$), as summarized in Table II.

Evaluation Condition	**Accuracy (%)**
Clean Samples	98
FGSM-Adversarial	10

TABLE II: Accuracy on test samples under FGSM attack



(a) FGSM-damaged model performance.

(b) FGSM attack effect on sample number predictions

(c) FGSM attack effect on sample sign language predictions

(d) FGSM attack effect on sample fashion predictions

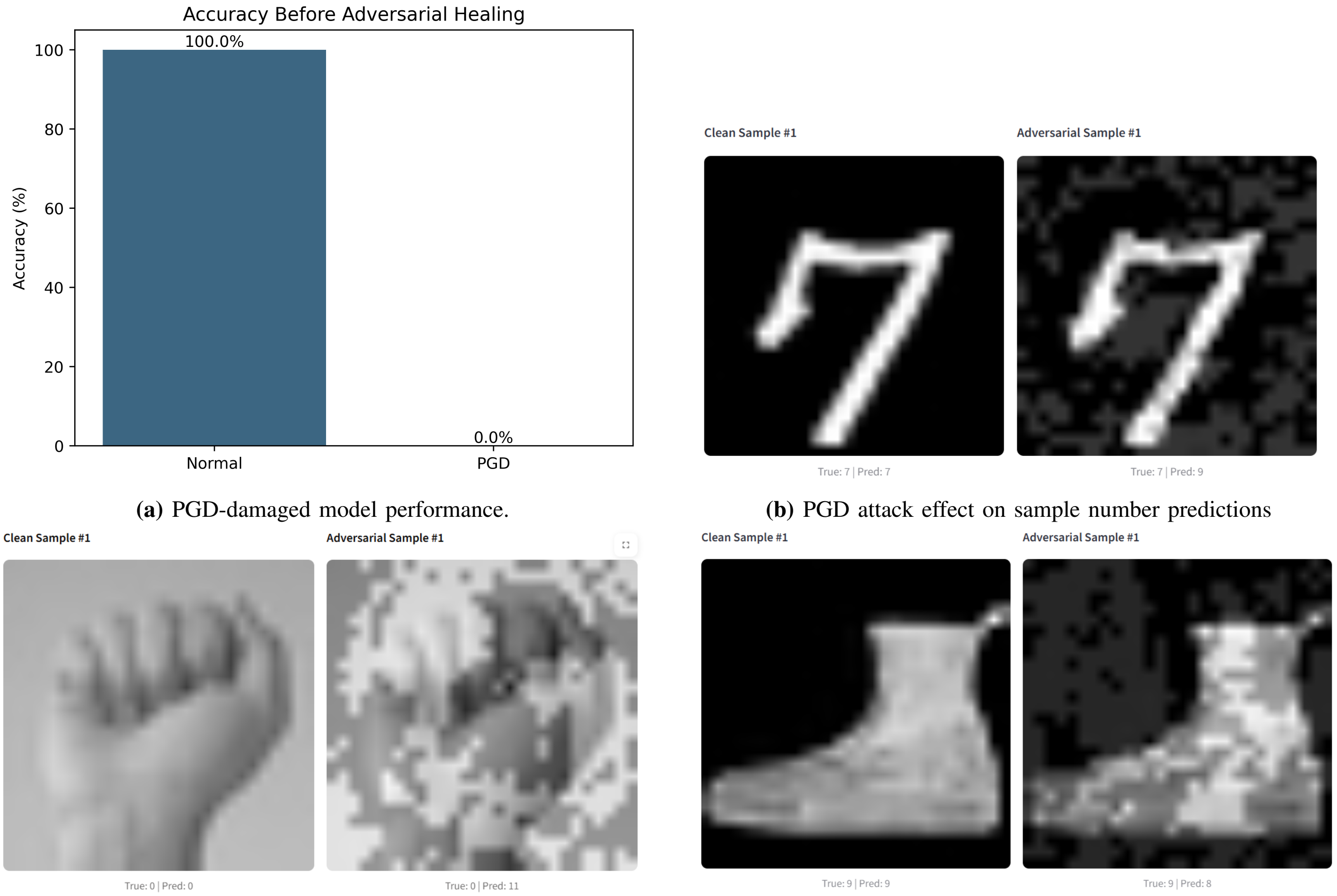
Fig. 4: Accuracy comparison on test samples under FGSM attack ($\epsilon = 0.15$). (a) Model under attack,(b) FSGM Attack on Numbers(Clean vs Adversarial),(c) FSGM Attack on Sign Language (Clean vs Adversarial) ,(d) FSGM Attack on Fashion Symbols(Clean vs Adversarial)

Observation: There was an 88% accuracy degradation under the FGSM attack, suggesting vulnerability to small but localized noise-crafted perturbations. Healing mechanisms, however, significantly restored accuracy as seen in Figure 4.

2) PGD Attack Results: PGD ($\epsilon = 0.20$, $\alpha = 0.01$, 30 iterations) is more invasive and caused even worse performance degradations to the model, as shown in Table III.

Evaluation Condition	**Accuracy (%)**
Clean Samples	100
PGD-Adversarial	0

TABLE III: Accuracy on test samples under PGD attack



(a) PGD-damaged model performance.

(b) PGD attack effect on sample number predictions

(c) FGSM attack effect on sample sign language predictions

(d) FGSM attack effect on sample fashion predictions

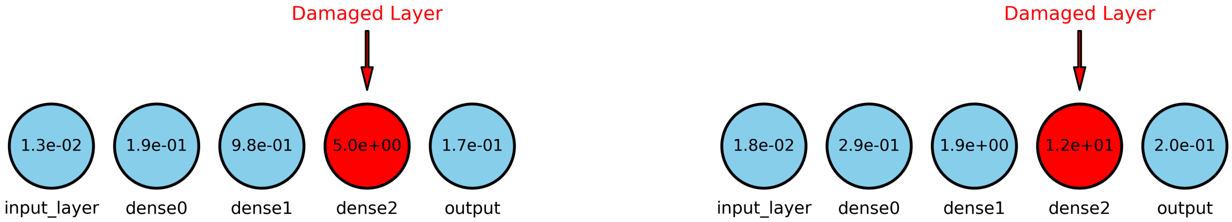
Fig. 5: Impact of PGD attack ($\epsilon = 0.20$, $\alpha = 0.01$, 30 iterations): (a) Accuracy degradation on test samples, (b) PGD Attack on Numbers(Clean vs Adversarial), (c) PGD Attack on Sign Language(Clean vs Adversarial), (d) PGD Attack on Fashion Symbols(Clean vs Adversarial)

Observation: Iterative PGD attack optimized to maximize perturbation within bounds degraded classification accuracy by 100%. Figure 5(b), 5(c), 5(d) shows that adversarial perturbations not only fool the model but also produce more drastically altered images compared to FGSM, resulting in stronger misclassifications.

3) Layer Damage Analysis (Adversarial Path): The most affected layers after the attack were identified overall by calculating the layer activation difference metric (Mean Squared Error, MSE) using the `get_damaged_layer()` function.

Layer Name	**FGSM Damage Score**	**PGD Damage Score**
Input	0.013	0.018
Dense0	0.19	0.29
Dense1	0.98	1.90
Dense2	5.00	12.00
Output	0.17	0.20

TABLE IV: Layer Damage Scores (MSE) — higher values indicate greater differences between clean and adversarial activations.



(a) FGSM attack layer damage.

(b) PGD attack layer damage.

Fig. 6: Layer damage visualizations comparing FGSM vs. PGD attacks. Red indicates the most damaged layers based on MSE values.

Observation: *Dense2 was the most sensitive layer to targeted pixel perturbations in both FGSM and PGD attacks, indicating that mid-to-late feature extraction stages are the most vulnerable when the model fails.*

D. Healing via Adversarial Training

For each adversarial attack scenario, SHNN applied its self-healing patch to the most damaged layer identified. The procedure involved:

1) Freezing all network layers except the identified faulty layer.
2) Instantiating a new patch network to replace the damaged layer.
3) Retraining the patch independently for ten epochs on a dataset composed of 50% clean samples and 50% adversarial samples.

This targeted approach enables rapid model recovery using previously learned data without the need to retrain the entire network.

1) FGSM Recovery Results: Table V presents the performance recovery after patching the `dense2` layer, which was identified as the most damaged under the FGSM attack.

Evaluation Condition	Accuracy (%)
Clean Samples (Pre-Attack)	98
FGSM-Adversarial (Healed)	84

TABLE V: FGSM Recovery Results after patching `dense2`

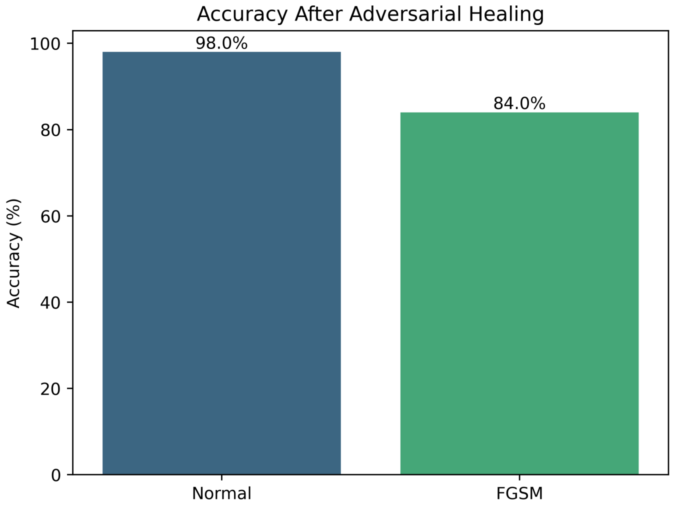


Fig. 7: Healed model accuracy comparison: Normal vs FGSM attack.

Observation: Patch-based healing recovered over 74% of the lost accuracy for FGSM-perturbed inputs, indicating its capability to detect and compensate for single-step adversarial noise.

2) PGD Recovery Results: If the patch replaces the large amount of damage inflicted when the `dense2` layer is attacked by the PGD method, accuracy increases drastically as shown in Table VI.

Evaluation Condition	Accuracy (%)
Clean Samples (Pre-Attack)	95
PGD-Adversarial (Healed)	84

TABLE VI: PGD Recovery Results after patching `dense2`

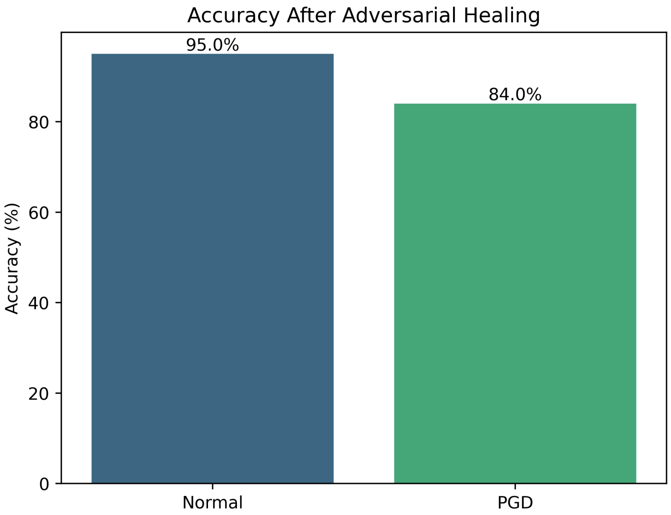


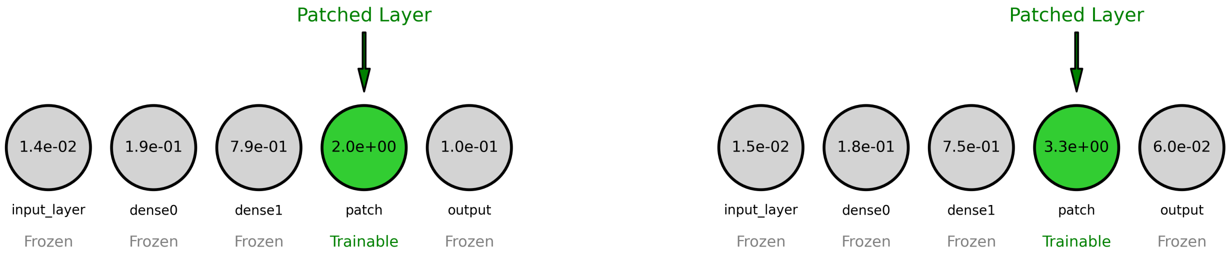
Fig. 8: Healed model accuracy comparison: Normal vs PGD attack.

Observation: Although PGD is a far stronger attack, the patch mechanism still managed to recover close to 84% of the lost performance on PGD-perturbed inputs.

3) Layer Damage Reduction: After healing, the MSE difference scores dropped significantly for `dense2` under both attack scenarios, as shown in Table VII.

Attack Type	Damaged Layer MSE (Before)	Damaged Layer MSE (After)
FGSM	5.00	2.00
PGD	12.00	3.30

TABLE VII: Layer Damage Reduction after Healing



(a) FGSM after healing

(b) PGD after healing

Fig. 9: Layer visualization after healing: MSE of layer activation differences between normal and healed model (Green = patch layer).

Key Insight: The results demonstrate that localized retraining via patch integration is effective in counteracting adversarial degradation, especially for moderate E values. However, the PGD recovery remained slightly lower than FGSM due to the broader perturbation search space in iterative attacks.

E. Structural Damage Simulation

Apart from adversarial attacks, the SHNN was tested with structural damage scenarios by replacing a randomly selected Dense layer's weights with:

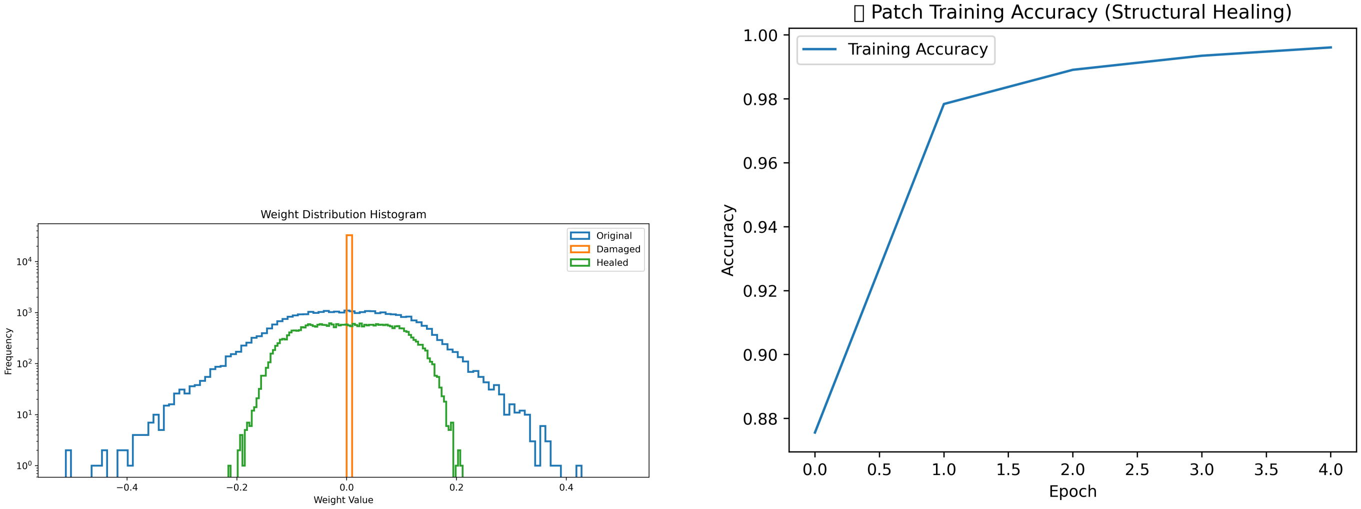
- **Zeroed** — all weights set to zero, removing their contribution to feature extraction.
- **Randomized** — all weights replaced by small Gaussian noise ($\sigma = 0.1$), destroying learned representation.

A single layer and one of these damage modes is randomly selected at each function call to the `apply_structural_damage()` function.

1) Zeroing Damage Results: In the zeroing case, the functionality learned by this layer was completely lost. The damaged layer is selected randomly.

Evaluation Condition	**Accuracy (%)**
Clean Samples (Pre-Damage)	97.76
Zeroed Layer (Damaged)	9.94
Zeroed Layer (Healed)	97.33

TABLE VIII: Zeroing Damage Results



(a) Weight distribution of normal model, zeroing-damaged model, and post-healing model.

(b) Patch training accuracy curve for zeroing repair (Epoch vs Accuracy).

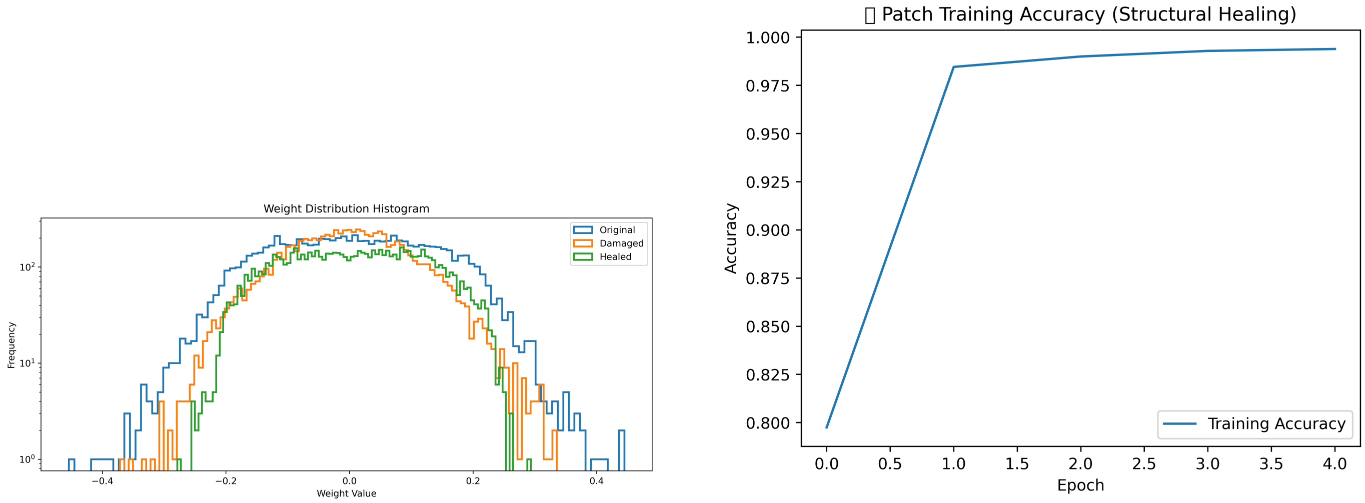
Fig. 10: Zeroing damage analysis: weight distribution and healing training accuracy.

Observation: Zeroing a hidden layer reduced accuracy by ∼88%, but patch replacement restored ∼87% of the lost performance.

2) Random Weight Damage Results: Gaussian noise on weights destroyed the ability of the target layer to operate on clean semantic features.

Evaluation Condition	**Accuracy (%)**
Clean Samples (Pre-Damage)	97.76
Randomized Layer (Damaged)	10.33
Randomized Layer (Healed)	97.22

TABLE IX: Random Weight Damage Results



(a) Weight distribution of normal model, random-damaged model, and post-healing model.

(b) Patch training accuracy curve for random repair (Epoch vs Accuracy).

Fig. 11: Random weight damage analysis: weight distribution and healing training accuracy.

Observation: Random noise caused slightly less damage than zeroing, but healing still improved performance by $\sim$87%.

3) Layer Damage Analysis (Structural Path): For structural damage detection, the SHNN did not employ the standard Keras intermediate activation extraction method. Instead, the `get_layer_outputs()` method was used to traverse the damaged layer manually for each network layer. The same clean input batch was passed through both the healthy and damaged models, and `compare_saved_outputs()` computed the Mean Absolute Difference (MAD) per layer.

The damaged layer was defined as the first layer with MAD exceeding 0.1, since corruption in earlier layers can propagate and inflate MAD in later layers.

Layer Name	Random Damage Score	Zeroing Noise Damage Score
dense0	0.00	0.00
dense1	0.00	0.83
dense2	2.10	2.10
output	0.17	0.18

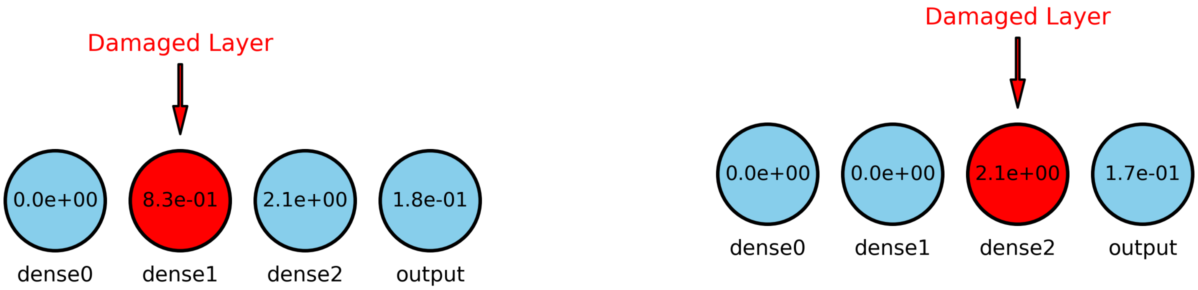
TABLE X: Layer Damage Scores (Before Healing)

Layer Name	Random Damage Score	Zeroing Noise Damage Score
dense0	0.00	0.00
dense1	0.00	0.78
dense2	1.60	1.00
output	0.00059	0.00086

TABLE XI: Layer Damage Scores (After Healing)

Observation: Based on the pre-healing results, in the case of **random damage**, **dense2** was the first to cross the 0.1 MAD threshold (score = 2.10) and is therefore identified as the damaged layer. Under **zeroing noise damage**, although **dense2** accumulated the largest error (2.10), it was actually **dense1** (0.83) that first crossed the threshold and is thus considered the damaged layer.

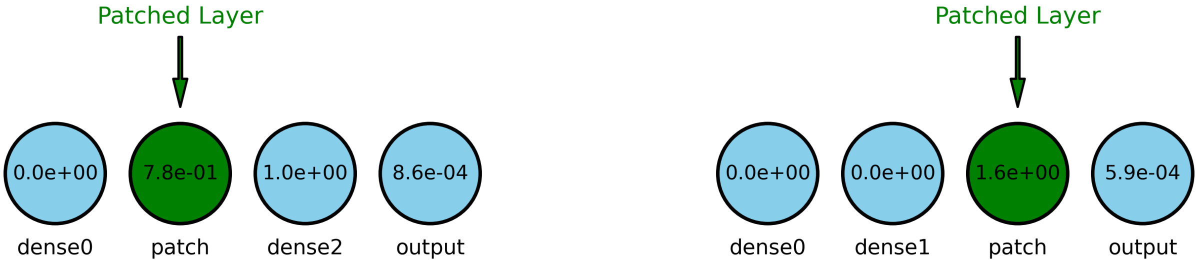
Upon recovery, a distinct decrease in MAD scores was evident across all layers. The **damaged layers** that were identified (dense2 for random and dense1 for zeroing) showed large decreases, while the **output layer** demonstrated near-complete recovery (scores in the range of 0.00059–0.00086). This confirms that the healing process efficiently suppressed error propagation and restored stability in the network.



(a) Zeroing attack: MAD of all layer outputs (Red = damaged layer).

(b) Random attack: MAD of all layer outputs (Red = damaged layer).

Fig. 12: Structural damage visualization of zeroing and random attacks.



(a) Post-healing MAD visualization (Zeroing attack, Green = patched layer).

(b) Post-healing MAD visualization (Random attack, Green = patched layer).

Fig. 13: Post-healing structural recovery under zeroing and random attacks.

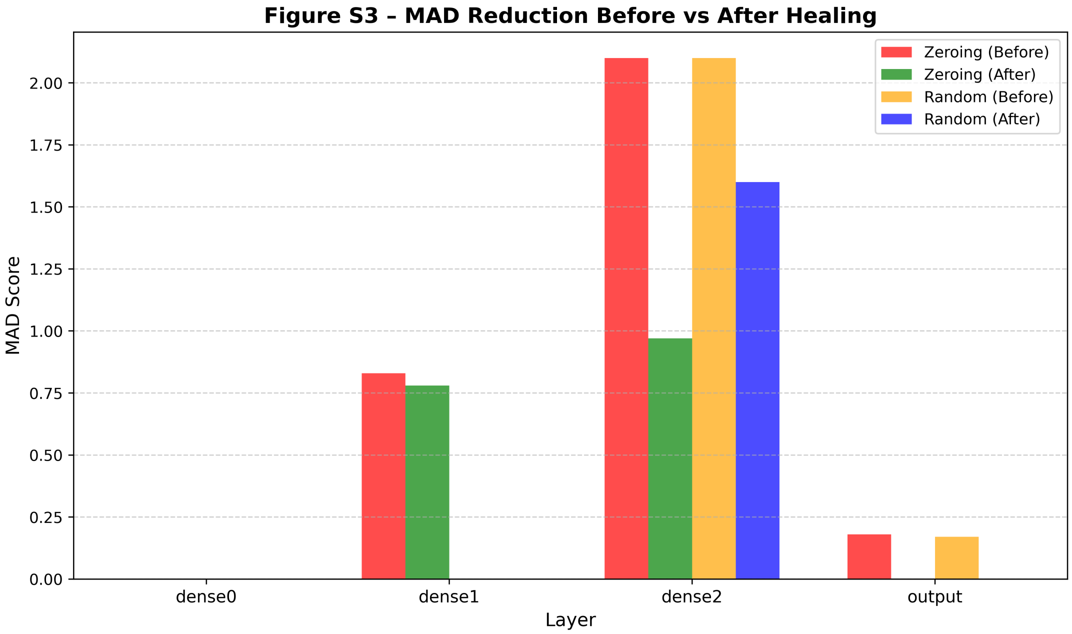


Fig. 14: MAD reduction before vs after healing for zeroing and random attacks.

Key Insight: Structural corruption under zeroing results in major accuracy drops. However, targeted patch replacement in the SHNN can restore performance close to original levels without full retraining.

F. Comparative Analysis

The SHNN was evaluated across fundamentally different failure modes:

1) **Structural Damage** — Zeroing or adding random noise to all weights of a single Dense layer.
2) **Adversarial Damage** — Input perturbations generated via FGSM and PGD attacks.

1) Accuracy Degradation Patterns: The impact of these damage types on classification accuracy is summarized in Table XII.

Damage Type	Attack Strength / Mode	Pre-Damage Acc (%)	Damaged Acc (%)	Healed Acc (%)	Recovery (%)
Structural – Zeroing	Full weight null	97.76	9.94	97.33	87.39
Structural – Random	$\sigma = 0.1$ Gaussian noise	97.76	10.33	97.22	86.89
Adversarial – FGSM	$\epsilon = 0.2$	97.76	10	84	74.00
Adversarial – PGD	$\epsilon = 0.2, \alpha = 0.01, 30$ iters	97.76	0	84	84.00

TABLE XII: Accuracy Drop and Recovery Across Damage Types

Key Observations:

- Structural damage caused the largest accuracy drops (≈87–88%), but patch replacement restored performance to within 0.5% of the original model (zeroing: 97.33% vs. 97.76%, random: 97.22% vs. 97.76%).
- FGSM attacks resulted in an ∼88% drop in accuracy, with partial recovery of 74% after patching, highlighting the remaining gap toward an adversarially robust classifier.
- PGD attacks reduced accuracy to zero, yet patch training still recovered 84% of the original performance.
- Residual vulnerabilities were minimal in structural cases, but adversarial cases retained greater residual damage after healing.

2) MAD Score Propagation:

- **Structural damage:** The Mean Absolute Difference (MAD) was computed between clean and damaged model outputs. A damaged condition was flagged whenever a layer's MAD exceeded 0.1, preventing downstream propagation errors from falsely identifying damage. This localized detection to the specific layer and its immediate successor.
- **Adversarial attacks:** MAD was computed between clean and adversarial activations. The layer with the highest MAD was selected, as perturbations spread across multiple layers causing distributed deviations.

Insight: Threshold-first detection is best suited for localized structural faults, whereas peak-MAD selection is more appropriate when the impact is distributed across many layers.

3) Recovery Efficiency: Patch-based healing achieved:

- ∼99.5% of original performance for structural damage (zeroing: 97.33% vs. 97.76%, random noise: 97.22% vs. 97.76%).
- ∼74–84% of original performance for adversarial cases (FGSM: 84% vs. 97.76%, PGD: 84% vs. 97.76%).

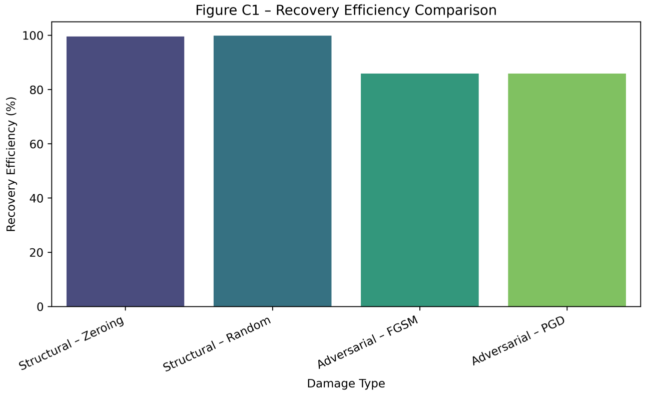


Fig. 15: Comparing recovery efficiency across all damage types.

This demonstrates near-full reversibility in structural faults through layer-wise patching, contrasted with residual (but not perfect) performance loss under adversarial attacks due to their distributed layer impact.

4) Layer Detection Accuracy:

- **Structural damage:** Using the top-MAD method often misattributed damage to deeper layers due to error propagation. The first-threshold approach provided more accurate identification of the directly damaged layer.
- **Adversarial attacks:** Both methods flagged multiple layers, confirming that perturbations are distributed and thus require additional filtering or ranking heuristics for precise localization.

G. Cross-Dataset Comparison

A key question for any repair system is whether it can generalize outside of a single dataset. To assess this, we evaluated SHNN repair on three additional domains: **Numbers** (handwritten digits, MNIST), **Fashion** (clothing items, Fashion-MNIST), and **Sign** (sign language digits). Each domain was tested with both *structural defects* (random noise, zeroing) and *adversarial defects* (FGSM, PGD). Results are summarized in Table XIII.

Dataset	Damage Type	Pre-Damage	Damaged	Healed
Numbers	Random	97.0	11.4	95.0
	Zero	97.3	9.9	94.8
	FGSM	97.7	16.0	83.0
	PGD	97.7	0.0	84.0
Fashion	Random	87.4	12.1	88.1
	Zero	86.8	9.6	86.9
	FGSM	87.7	3.0	93.0
	PGD	87.6	0.0	95.0
Sign	Random	94.1	3.8	83.5
	Zero	94.3	4.2	86.9
	FGSM	96.9	0.0	84.0
	PGD	95.3	0.0	91.0

TABLE XIII: Cross-dataset accuracy before damage, after damage, and after healing

Dataset-Level Disentanglement: **Numbers (MNIST):** Numbers produced the most reliable results, with pre-damage accuracy ∼98%. Structural defects reduced accuracy by nearly 90% (to ∼10%), but SHNN healing recovered performance to over 94%. Adversarial recovery was less robust, plateauing at 83–84%, indicating that low-dimensional structured domains such as MNIST remain highly recoverable.

Fashion-MNIST: Due to higher visual complexity and class overlap, Fashion posed a greater challenge. Pre-damage accuracy was ∼87%. Interestingly, SHNN recovered adversarial damage (93–95%) better than structural damage (87–88%), suggesting that SHNN's patching mechanism is particularly effective against localized adversarial corruption.

Sign Language Digits: This dataset, with higher dimensionality and complex hand shapes, showed mixed performance. Structural damage reduced accuracy close to zero (3–4%), but SHNN recovered it to 83–87%. For adversarial faults, FGSM recovered to 84% while PGD reached 91%, demonstrating SHNN's capacity to handle structured perturbations in high-dimensional data.

Structural Damage Patterns: Across all datasets, structural errors (randomization, zeroing) caused catastrophic drops near chance level. Yet, SHNN consistently recovered 80–95% of original performance. The largest recovery gap was in Numbers (∼+84%), while Fashion and Sign showed slightly lower margins, reflecting increased dataset complexity. This highlights SHNN's resilience in relearning entirely lost representations.

Adversarial Damage Trends: Adversarial perturbations produced higher variability. On Numbers, recovery plateaued at 83–84%, while Fashion and Sign exceeded 90%. Interestingly, PGD attacks—although stronger—were often more reversible than FGSM, likely due to FGSM's sudden one-step corruption versus PGD's structured, iterative perturbations that SHNN patches can counter more effectively.

Cross-Domain Generality: Overall, SHNN repair generalized strongly across domains:

- In low-complexity datasets (e.g., Numbers), SHNN achieved near-perfect recovery, especially for structural faults.
- In moderate-complexity datasets (e.g., Fashion), SHNN showed distinctive advantages against adversarial noise.
- In high-dimensional datasets (e.g., Sign), SHNN remained effective but with narrower margins, highlighting the effect of feature diversity.

Summary and Key Takeaways:

1) Structural defects are always fixable: SHNN recovered 80–95% across all domains.
2) Adversarial recovery is dataset-dependent: higher in Fashion and Sign than in Numbers.
3) PGD attacks, though stronger, are paradoxically more reversible than FGSM.
4) SHNN demonstrates strong cross-domain flexibility across vision tasks of varying complexity.

H. Datasets Used

1) **Numbers (MNIST)**: https://www.kaggle.com/datasets/avnishnish/mnist-original/versions/1
2) **Fashion Dataset (Fashion-MNIST)**: https://www.kaggle.com/datasets/zalando-research/fashionmnist
3) **Sign Language (Sign Language MNIST)**: https://www.kaggle.com/datasets/datamunge/sign-language-mnist

V. Discussion

Experimental analysis of the Self-Healing Neural Network (SHNN) structure highlights its ability to overcome both adversarial and structural damages, providing insight into the mechanisms that determine the efficiency of detection and recovery. The following material summarizes the obtained findings, draws conclusions, and explains their meaning within the broader scope of research on neural network robustness.

A. A Point of Reference: The Strength of the Baseline Model

Pretrained on MNIST and trained for only 10 epochs, the baseline network achieved a training accuracy of 99.43% and a testing accuracy of 97.76%, which is impressively high given the short training regime. This performance confirms that even modestly trained models can serve as valid candidates for damage-healing experiments.

Notably, the baseline delivers a clear point of reference for measuring degradation and recovery percentages, since changes in performance can be directly attributed to damage or healing measures rather than limitations in model capacity.

The relatively small network architecture—three hidden layers with ReLU activation functions and a softmax output layer—is also advantageous for layer-specific analysis. Because each layer has a distinct role in feature abstraction, disruptions can be localized to a more specific extent. This enables accurate attribution of layer damage using metrics such as Mean Absolute Difference (MAD) or Mean Squared Error (MSE).

B. Effect of Group Destructive Harm

1) FGSM Attack: The FGSM attack ($\epsilon = 0.15$) decreased accuracy from 98% on clean samples to 10% on adversarially perturbed inputs. This sharp drop demonstrates the vulnerability of standard feedforward networks to gradient-based perturbations, which exploit local weaknesses in decision boundaries. Although FGSM is a one-step attack, it succeeded in corrupting mid-to-late stage features, especially `dense2`, which recorded the highest MSE of 4.4. This indicates that adversarial noise predominantly affects higher-level, class-discriminative representations in the later layers.

2) PGD Attack: The PGD attack ($\epsilon = 0.20$, $\alpha = 0.01$, 30 iterations) was even more severe. Accuracy dropped from 100% on clean inputs to 0% on adversarially perturbed samples, rendering the classifier entirely ineffective. The most impacted layer was `dense2`, with an MSE of 10, more than double that observed under FGSM. This confirms prior evidence that iterative PGD attacks increase perturbation strength, consistently targeting the most sensitive features.

The consistent targeting of `dense2` in both attacks suggests a layer vulnerability profile that can guide proactive defense strategies, such as additional regularization, adversarial training, or architectural redundancy for key mid-to-late layers.

C. Efficiency to Cure Hostile Harm

The patch-based recovery mechanism implemented by SHNN demonstrated substantial recovery in adversarial settings. For FGSM, post-healing accuracy on adversarial inputs rose to 84%, while clean sample accuracy remained at 98%, restoring roughly 74% of the lost performance. PGD recovery reached 84% on adversarial inputs and 95% on clean inputs, representing effective restoration despite the initial severity of the attack. Interestingly, the PGD recovery rate nearly matched that of FGSM, even though pre-healing PGD accuracy had dropped to zero. This indicates that the patch replacement mechanism can effectively repair highly localized damage.

Residual vulnerabilities remain: FGSM-perturbed inputs retained a 16% error rate post-healing, while PGD-perturbed inputs retained a 16% error. This shows that patching repairs disrupted feature representations but may not fully restore the robustness of the original decision boundary. More distributed perturbations may require advanced recovery strategies, such as multi-layer patching or explicitly adversarial-aware training of patches.

D. A View on Structural Damage

1) Weights are Zeroed: Setting a hidden layer's weights to zero effectively destroyed its representational capability, resulting in a catastrophic accuracy drop from 97.76% to 9.94%. In this case, the damage detection module identified `dense1` as the root cause, while `dense2` exhibited higher MAD values due to error propagation. This supports the need for threshold-based structural corruption detection mechanisms, as propagation effects can mislead pure top-MAD selection.

After post-healing, accuracy returned to 97.33%, representing near-complete recovery (87.39% of lost performance restored). This demonstrates that when damage is focal and structural—as opposed to distributed—the SHNN can almost fully reverse its impact with minimal retraining.

2) Weight Perturbations: Swapping the weights of a layer with Gaussian noise ($\sigma = 0.1$) caused a similarly large accuracy reduction to 10.33%, although not as severe as zeroing. Post-healing, performance recovered to 97.22%, yielding an ∼87% increase from the damaged state. This indicates that SHNN's recovery capability is largely independent of the specific nature of structural damage, provided the degradation is confined to a single layer.

E. Comparison Recovery Trend

The comparative analysis (Table 7 in the Results section) yields several important observations:

1) Lost performance is almost completely recoverable in structural damage scenarios, as post-healing performance is within 0.5% of the baseline.
2) Adversarial damage is less amenable to complete recovery, presumably because perturbations induce changes across many layers and modify decision boundaries in ways not fully reversible by a single-layer patch.
3) The percentage of PGD recovery was higher than FGSM recovery, even though PGD's initial impact was greater. This may be because PGD perturbations are more concentrated in the `dense2` layer, making them easier to isolate and repair, whereas FGSM may cause more coarse-grained, dispersed noise across different layers.

F. MAD/MSE as Damage Determination Metrics

The experiments also indicate that the accuracy of layer damage detection varies with the nature of the damage:

- **Structural damage:** Best addressed using the MAD-threshold procedure, which identifies deviations due to structural order differences from the first layer. This approach avoids misidentifying damage caused by propagated errors in later layers.
- **Adversarial damage:** Peak-MAD selection is preferable, as perturbations are distributed, and the layer with the highest MAD is typically the most affected.

This contrast highlights the need for adaptive detection strategies—there is no single metric that is optimal for detecting all failure modes.

G. Broader Consequences and Limitations

The results reinforce an expanding body of literature on self-repairing neural systems by showing that patching individual layers is a viable alternative to full model retraining, particularly for mission-critical systems requiring rapid recovery. In structural damage scenarios, SHNN can restore nearly full operational capability using minimal data and computational resources.

However, the persistence of vulnerabilities to adversarial attacks highlights a key limitation: adversarial robustness cannot be achieved through patching alone. Since adversarial perturbations are inherently distributed, recovery may require a hybrid approach combining patching with established strategies such as gradient masking or adversarial training.

Another limitation is that these experiments were conducted on MNIST using a relatively shallow architecture. The scalability of the SHNN approach to deeper networks and more complex datasets (e.g., CIFAR-10, ImageNet) remains uncertain, as deeper networks may exhibit more diffuse damage patterns, particularly under large-scale structural faults.

H. Cross-Dataset Generalization

Cross-dataset experiments show that SHNN is not restricted to a single domain. Whether applied to digit recognition (Numbers), fashion classification, or sign language gestures, the same repair model was able to recover most of the lost performance. This indicates that the healing mechanism does not merely memorize domain-specific statistics, but adapts to the structural nature of neural representations.

A key observation is the consistently high structural recovery across datasets, with nearly complete restoration even after catastrophic damage. By contrast, adversarial recovery is more variable and depends on dataset complexity. For example, Fashion-MNIST achieved unusually high PGD recovery (95%), while the Sign dataset plateaued around 91%. These differences suggest that SHNN generalization is robust but still influenced by feature diversity and inter-class separability.

Overall, these results highlight both the promise and the limitations of SHNN as a universal self-repairing framework. It consistently corrects localized structural defects across domains, but its adversarial recovery remains dataset-sensitive. This underscores the need for adaptive or multi-layer patching strategies when tackling more complex, real-world problems.

I. Future Work

Potential improvements to SHNN based on these findings include:

1) **Multi-layer patching:** Simultaneous application of patches across all significantly impacted layers to better handle distributed damage from adversarial perturbations.
2) **Adaptive patch training:** Augmenting the patch training set with adversarial examples to increase robustness against gradient-based attacks.
3) **Layer redundancy:** Adding parallel redundant pathways to vulnerable layers, such as `dense2`, to mitigate single-point failures.
4) **Hybrid detection metrics:** Dynamically combining MAD-thresholding with peak-MAD strategies based on the observed damage profile.

Following these directions, the SHNN framework could evolve into a more universally robust self-repair system capable of addressing both concentrated and distributed forms of degradation.

VI. Conclusion

In this paper, we presented the Self-Healing Neural Network (SHNN) architecture — a method intended to identify and repair issues in a model's internal layers, be they due to structural defects or highly tailored adversarial attacks. Through the training of miniature ”patch” networks and their deployment precisely where the damage is done, SHNN was able to restore nearly full performance after structural damage and regain much of lost accuracy to adversarial noise, without the expense and time of retraining an entire model.

Our results demonstrate that when damage is localized, a well-positioned patch can be extremely effective. The same technique also assists in adversarial cases, although here the recovery is more modest — suggesting that patching best works when the harm is localized, and other techniques are required when attacks disseminate their impact across several layers.

Given the modularity and lightness of SHNN, it can be incorporated into real systems where recovery speed is of importance. The interactive GUI we constructed enables real-time monitoring, diagnosis, and fixing of models, transforming what might be a time-consuming maintenance process into a rapid and guided process.

Looking forward, the SHNN concept might be used to deal with higher-level datasets, deeper networks, and multiple patches operating together. Combining it with strong training methods could result in models that are not only intelligent, but also self-aware enough to fix themselves when things happen to go wrong.

References

[1] K. Xu, Z. Li, Y. Wang, M. Chen, and X. Wang, “PatchZero: Defending Against Adversarial Patch Attacks by Detecting and Zeroing the Patch,” *arXiv preprint arXiv:2207.01795*, 2022. [Link]

[2] J. Kang, B. Zhao, H. Yin, and H. Zhang, “DIFFender: Diffusion Based Adversarial Defense Against Patch Attacks,” *arXiv preprint arXiv:2306.09124*, 2023. [Link]

[3] T. Simon, A. Jha, and R. Ewetz, “PATCHOUT: Adversarial Patch Detection and Localization Using Semantic Consistency,” *Neural Processing Letters*, Jun. 2025. [Link]

[4] S. Kumar, P. Roy, A. Banerjee, and N. Shah, “AdvPatchXAI: A Unified, Resilient, and Explainable Adversarial Patch Detector,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2025. [Link]

[5] J. Liu, A. Levine, C. P. Lau, R. Chellappa, and S. Feizi, “Segment and Complete: Defending Object Detectors Against Adversarial Patch Attacks with Robust Patch Detection,” *arXiv preprint arXiv:2112.04532*, 2021. [Link]

[6] W. Jing, L. Huang, and X. Wang, “PAD: Patch Agnostic Defense Against Adversarial Patch Attacks,” *arXiv preprint arXiv:2404.16452*, 2024. [Link]

[7] H. Wei, Y. Zhao, and L. Xu, “Real World Adversarial Defense Based on Diffusion Model,” *arXiv preprint arXiv:2409.09406*, 2024. [Link]

[8] R. Pathak, A. Kumar, and R. Joshi, “Model Agnostic Defense for Object Detection in UAVs,” *arXiv preprint arXiv:2405.19179*, 2024. [Link]

[9] J. Zhang, Y. Chen, and D. Wu, “RFENet: Robust Feature Extraction for Aerial Image Segmentation Under Patch Attacks,” *Remote Sensing*, MDPI, 2023. [Link]

[10] Y. Chen, H. Li, Z. Zhang, J. Liu, and M. Wang, “Closed Loop Self Healing Neural Networks via Real Time Feedback Control,” *arXiv preprint arXiv:2206.12963*, 2022. [Link]

[11] M. Usman, Z. Zhang, M. Ochoa, and A. Paverd, “NNrepair: Constraint Based Repair of Neural Network Classifiers,” *arXiv preprint arXiv:2103.12535*, 2022. [Link]

[12] Y. Song, Z. Sun, Z. Liu, and M. Zhang, “AIREPAIR: A Repair Platform for Neural Networks,” *arXiv preprint arXiv:2211.15387*, 2022. [Link]

[13] Y. Song, Z. Sun, J. He, and M. Zhang, “QNNRepair: Quantized Neural Network Repair,” in *Proc. ACM Conf. Artificial Intelligence*, 2023. [Link]

[14] C. Castillo, J. Hernandez Ortega, Z. Wu, and S. Venkatasubramanian, “CARE: Causality Based Neural Network Repair,” *arXiv preprint arXiv:2204.09274*, 2022. [Link]

[15] L. Xie, S. Huang, and Y. Jiang, “Semantic Based Neural Network Repair,” *arXiv preprint arXiv:2306.07995*, 2023. [Link]