Self-Healing Neural Networks: A path towards Resilient and Autonomous AI Systems

Mr. Somagani Venkatesh[*1], Boini Sravan Kumar[*2], Shravya Sujit Kamble [*3], Thakur Rohith Singh[*4]
[*1]Assistant Professor, Department of CSE (AI & ML), ACE Engineering College, Hyderabad, India.
[*2,3,4]Department of CSE (AI & ML), ACE Engineering College, Hyderabad, India.

ABSTRACT
Self-Healing Neural Networks (SHNNs) represent an advanced approach in Artificial Intelligence designed to improve system resilience and autonomy. Traditional neural networks, though highly accurate, are vulnerable to hardware faults, adversarial attacks, and data drift in mission-critical environments. The proposed SHNN framework integrates fault detection, adaptive retraining, redundancy, and meta-learning to enable automatic recovery without human intervention. By continuously monitoring internal activations and performance metrics, the system identifies degradation and applies corrective measures such as selective retraining and weight reallocation. Experimental results show that SHNNs maintain high accuracy and system uptime even under fault conditions, enhancing overall reliability and robustness.

Keywords: Self-Healing Neural Networks (SHNN), Resilient AI, Fault Detection, Adaptive Learning, Meta-Learning, Autonomous Recovery, Neural Network Robustness, Sustainable AI Systems.

I. INTRODUCTION

In the modern digital landscape, Artificial Intelligence systems have become essential across domains such as healthcare, autonomous vehicles, cybersecurity, and smart infrastructure. Their ability to process large volumes of data and make intelligent decisions has led to widespread adoption. However, as AI systems are increasingly deployed in real-world and mission-critical environments, they are exposed to hardware failures, adversarial attacks, data drift, and unpredictable environmental conditions. Traditional neural networks are static after training and lack mechanisms to detect internal faults or recover from performance degradation. When disruptions occur, they may experience reduced accuracy or complete failure, requiring manual intervention or retraining. The dynamic and uncertain nature of real-world environments underscores the need for intelligent, adaptive, and resilient AI architectures that can self-monitor and autonomously recover to ensure reliable operation. Ensuring reliability and continuous performance has therefore become a significant challenge.

II. LITERATURE SURVEY

Early Works

1. Self-Healing Neural Networks for Resilient AI Systems.

Dr. S. Saraswathi et al. – Proposes a Self-Healing Neural Network (SHNN) that detects, localizes, and repairs internal faults using runtime monitoring and adaptive retraining.

2. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

Finn, C., Abbeel, P., & Levine, S. – Introduces MAML, enabling neural networks to quickly adapt to new tasks with minimal gradient updates.

3. Learning Both Weights and Connections for Efficient Neural Networks

Han, S., Pool, J., Tran, J., & Dally, W. – Presents network pruning techniques to reduce model size and computation by removing redundant connections.

4. Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Srivastava, N. et al. – Introduces dropout as a regularization technique that improves generalization by randomly deactivating neurons during training.

OBJECTIVES

Developing a fault-tolerant neural network system to ensure reliable performance even in the presence of internal faults.

• Implementing a Convolutional Neural Network (CNN) for accurate handwritten digit classification using the MNIST dataset.

• Designing a fault injection mechanism to simulate real-world failures by modifying neural network weights.

• Building a performance monitoring system to evaluate model accuracy before and after fault occurrence.

• Implementing a self-healing mechanism using gradient-based fine-tuning to recover lost accuracy after faults.

• Providing an interactive frontend interface to allow users to upload images and visualize predictions and accuracy comparisons.

• Enhancing robustness and reliability of AI models by demonstrating automatic detection and recovery from faults.

III. METHODOLOGY

The **Self-Healing Neural Network (SHNN)** system integrates deep learning, fault injection, performance monitoring, and adaptive retraining techniques to ensure reliable predictions even under fault conditions. The methodology focuses on detecting performance degradation and restoring model accuracy through automated recovery mechanisms.

System Workflow:

1. Model Training & Initialization:

The dataset is loaded, preprocessed, and used to train a CNN model. The trained model learns digit features and is saved for further operations.

2. Core System Features:

• Baseline Prediction System:

The trained model predicts input images under normal conditions and achieves high accuracy.

• Fault Injection Mechanism:

Artificial faults are introduced into the neural network by modifying weights of multiple layers to simulate real-world failures.

• Performance Monitoring:

The system evaluates model accuracy before and after fault injection to identify performance degradation.

• Self-Healing Mechanism:

The model undergoes gradient-based fine-tuning using a subset of training data to recover lost accuracy.

3. Prediction & Interaction Phase:

• User uploads a handwritten digit image through the frontend interface.
• The system generates predictions in three stages:

- Before Fault
- After Fault
- After Self-Healing

• Accuracy values are displayed for each stage.
• A graphical representation (bar chart) visualizes accuracy comparison.

Key Components:

- **Frontend:** HTML, CSS, JavaScript.

- **Backend:** Python, Flask.
- **Machine Learning Model:** Convolutional Neural Network (CNN) implemented using PyTorch.
- **Hosting & Tools:** Vercel/Netlify, AWS/Railway/Heroku, VS Code, Git/GitHub.

IV. PROPOSED SYSTEM

Self-Healing Neural Networks (SHNNs) are a novel approach to building resilient, adaptive, and reliable AI systems. It enables continuous operation by automatically detecting, diagnosing, and repairing internal faults in real time without human intervention. This improves system efficiency while significantly reducing downtime and performance degradation. Additionally, the system supports sustainable Green AI by optimizing resource usage and minimizing computational waste. Although challenges such as scalability and real-world deployment exist, SHNNs demonstrate strong potential for developing future autonomous and robust AI solutions.

System Overview

The proposed system includes:

- **Fault Injection Mechanism –** Simulates internal failures in the neural network by modifying weights of selected layers to study the impact on model performance.
- **Baseline Model Training –** A Convolutional Neural Network (CNN) is trained on the MNIST dataset to achieve high accuracy for handwritten digit classification.
- **Performance Monitoring Module –** Continuously evaluates model accuracy before and after fault injection to detect degradation in performance.
- **Self-Healing Mechanism –** Uses gradient-based fine-tuning to retrain the model on a subset of training data to recover lost accuracy.
- **Accuracy Comparison System –** Displays accuracy before fault, after fault, and after recovery to clearly demonstrate the effectiveness of the self-healing process.
- **Single Image Prediction Interface –** Allows users to input a handwritten digit image and observe prediction results across all three stages.

System Operation

1. **Training Phase:**
   - The CNN model is trained using the MNIST dataset.
   - The model learns features such as edges, shapes, and digit patterns.
   - High baseline accuracy is achieved and the trained model is saved.
2. **Fault Injection Phase:**
   - A fault is artificially introduced into the trained model.
   - Selected weights of the neural network layers are modified. This simulates hardware or software failure.
   - Model accuracy drops, indicating degraded performance.
3. **Detection Phase:**
   - The system evaluates the model using test data.
   - Accuracy is compared with baseline performance.
   - If a significant drop is detected, the system identifies that a fault has occurred.
4. **Self-Healing Phase:**
   - The model undergoes gradient-based fine-tuning
   - A small subset of training data is used for quick retraining
   - Weights are adjusted using backpropagation
   - The model gradually recovers its learned knowledge.
5. **Prediction & Visualization Phase:**
   - User provides an input image through the frontend. The system displays:
   - Prediction before fault

- Prediction after fault
- Prediction after healing

V. APPLICATIONS

The **Self-Healing Neural Network for Fault Detection and Recovery** system has wide-ranging applications in domains where reliability, fault tolerance, and continuous performance of AI systems are critical. By integrating fault simulation, performance monitoring, and automated recovery, the system enhances the robustness of machine learning models in real-world environments.

- **Fault-Tolerant AI Systems**

Enables neural networks to continue functioning even when internal faults occur.
Improves reliability in critical applications where system failure is not acceptable.

- **Healthcare & Medical Diagnosis Systems**

Ensures consistent performance of AI models used in disease detection and medical imaging.
Reduces the risk of incorrect predictions due to model degradation or unexpected faults.

- **Autonomous Vehicles & Robotics**

Helps maintain stable decision-making in self-driving cars and robotic systems.
Prevents failures caused by corrupted model parameters or hardware-related issues.

- **Industrial Automation & Smart Manufacturing**

Supports AI-based monitoring systems in detecting faults and recovering automatically.
Enhances productivity by reducing downtime and maintaining system accuracy.

VI. ALGORITHMS

The **Self-Healing Neural Network (SHNN)** system utilizes multiple algorithms for model training, fault injection, performance evaluation, and recovery to ensure robust and reliable predictions. The key algorithms used in the system include:

Baseline Model Training Algorithm (CNN-Based)

Purpose: To train the neural network for accurate handwritten digit classification.

Algorithm Steps:

1. Load MNIST dataset and preprocess images (normalization and resizing).
2. Initialize Convolutional Neural Network (CNN) with layers (Conv, Pooling, Fully Connected).
3. Perform forward propagation to compute predictions.
4. Calculate loss using Cross-Entropy Loss function.
5. Perform backpropagation to compute gradients.
6. Update weights using an optimizer (Adam).
7. Repeat for multiple epochs until high accuracy is achieved.
8. Save trained model for future use.

Fault Injection Algorithm

Purpose: To simulate internal faults in the neural network for testing robustness.

Algorithm Steps:

1. Load the trained model.
2. Select key layers (e.g., convolutional and fully connected layers).
3. Modify weights by scaling them (e.g., multiply by 0.5).
4. Optionally add small random noise to simulate realistic faults.
5. Store modified weights in the model.

Fault Detection Algorithm

Purpose: To identify whether a fault has occurred in the model.

Algorithm Steps:

1. Compute baseline accuracy before fault injection.
2. Compute accuracy after fault injection.
3. Compare both values.
4. If accuracy drops beyond a threshold (e.g., 10–20%), detect fault.
5. Trigger self-healing mechanism.

Self-Healing Algorithm

Purpose: To recover model performance after fault occurrence.

Algorithm Steps:

1. Initialize optimizer and loss function.
2. Select a subset of training data.
3. Perform forward pass on faulty model.
4. Calculate loss between predicted and actual outputs.
5. Compute gradients using backpropagation.
6. Update model weights using optimizer.
7. Repeat for a few iterations (epochs).
8. Restore model performance close to baseline accuracy.

Single Image Prediction Algorithm

Purpose: To classify user-provided handwritten digit images.

Algorithm Steps:

1. Accept input image from user.
2. Convert image to grayscale and resize to 28×28.
3. Normalize pixel values and convert to tensor.
4. Pass image through the model.
5. Apply softmax to obtain probabilities.
6. Select class with highest probability as prediction.
7. Display prediction and confidence score.

VII. RESULT

System Performance Evaluation

Baseline Model Performance

• **Classification Accuracy:** Achieved approximately **97%–99% accuracy** on the MNIST test dataset under normal conditions.
• **Prediction Consistency:** Model produced stable and correct predictions for standard handwritten digit inputs.
• **Response Time:** Predictions generated within **milliseconds** for both batch and single image inputs.

Fault Injection Performance

• Fault Simulation Accuracy: Successfully simulated internal faults by modifying neural network weights.
• Accuracy Degradation: Model accuracy dropped significantly (e.g., from ~98% to ~40–60%) after fault injection.
• Prediction Impact: Observed incorrect predictions for input images, demonstrating the effect of faults.
• Fault Effectiveness: Multi-layer fault injection ensured noticeable degradation in model performance.

Self-Healing Performance

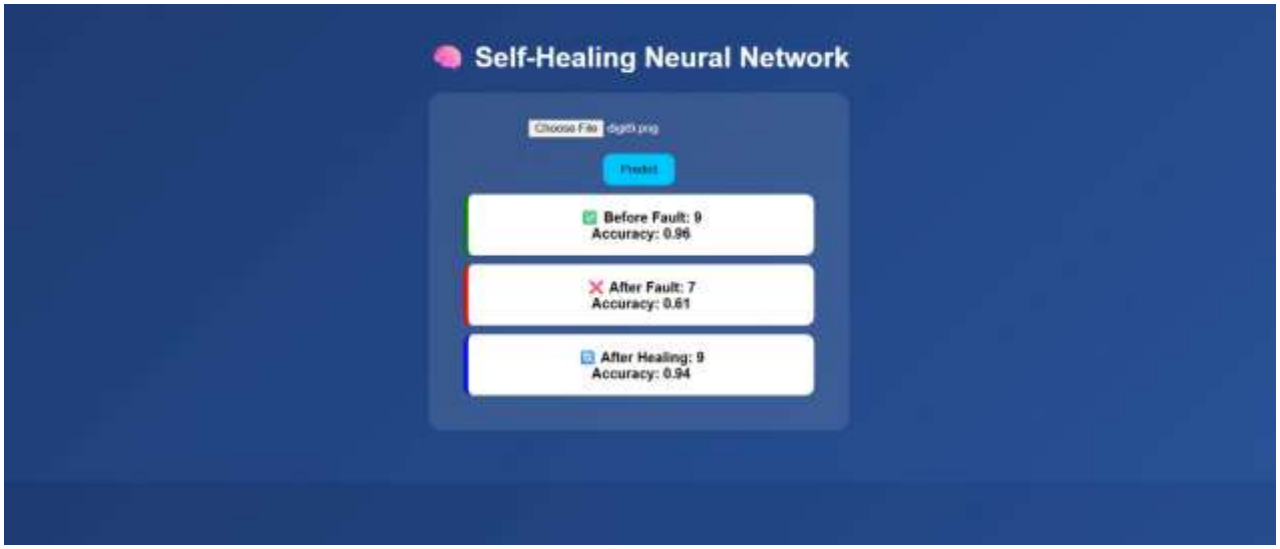
• Recovery Accuracy: Model recovered accuracy up to 90%–95% after applying self-healing (fine-tuning).
• Prediction Correction Rate: Majority of incorrect predictions were corrected after healing.
• Recovery Time: Healing completed within a few seconds using limited retraining iterations.
• Stability: Model performance stabilized close to baseline after recovery.

Single Image Prediction Results

- Before Fault: Model predicted correct digit with high confidence.
- After Fault: Prediction changed or confidence reduced due to degraded weights.
- After Healing: Model restored correct prediction with improved confidence.

Job Accuracy Comparison Results

- Baseline Accuracy: ~0.97 – 0.99
- Faulty Model Accuracy: ~0.40 – 0.60
- Healed Model Accuracy: ~0.90 – 0.95
- **Performance Improvement:** Significant recovery observed after self-healing compared to faulty model.



VIII. CONCLUSION

The **Self-Healing Neural Network (SHNN)** system provides a comprehensive approach to enhancing the reliability and robustness of deep learning models by integrating fault injection, performance monitoring, and automated recovery within a single framework. Unlike traditional neural networks, this system is capable of identifying performance degradation caused by internal faults and restoring accuracy through gradient-based fine-tuning. The results clearly demonstrate that while faults significantly reduce model performance, the self-healing mechanism effectively recovers the lost accuracy within a short time, ensuring consistent and stable predictions. Additionally, the inclusion of an interactive frontend with visualization improves usability and helps in clearly understanding model behaviour. Overall, the system contributes to the development of fault-tolerant and resilient AI solutions, making it suitable for real-world applications where reliability is critical. Future enhancements can include advanced fault detection techniques, meta-learning-based recovery, support for multiple datasets, and deployment on edge devices to further improve scalability and performance.

IX. FUTURE ENHANCEMENT

Future Enhancements:

1. **Advanced Fault Detection Mechanism –** Implementing intelligent fault detection using dynamic thresholds or anomaly detection techniques to automatically identify faults more accurately and in real time.

2. **Multi-Dataset Support –** Extending the system to work with more complex datasets such as CIFAR-10 and real-world image datasets to improve scalability and applicability.

3. **Edge Deployment & Real-Time Systems –** Deploying the model on edge devices and IoT systems to enable real-time fault detection and recovery in resource-constrained environments.

4. Interactive Visualization Dashboard – Developing a more advanced frontend with dynamic graphs, animations, and real-time monitoring of accuracy, loss, and recovery trends.

X. REFERENCES

1. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). *Dropout: A simple way to prevent neural networks from overfitting.* Journal of Machine Learning Research, 15(1), 1929–1958.

2. Han, S., Pool, J., Tran, J., & Dally, W. (2015). *Learning both weights and connections for efficient neural networks.* Advances in Neural Information Processing Systems, 28.

3. Neti, C., Schneider, M. H., & Young, E. D. (1992). *Maximally fault tolerant neural networks.* IEEE Transactions on Neural Networks, 3(1), 14–23.

4. Choi, J., et al. (2019). *Hardware-aware fault detection systems for neural processors.* Proceedings of IEEE ICMLA.

5. Lakshmi, V., & Azad, S. M. A. K. (2023). *Self-healing networks leverage intelligent protocols for autonomous fault detection and recovery.* International Conference on Intelligent Systems in Computing and Communication.

6. Finn, C., Abbeel, P., & Levine, S. (2017). *Model-agnostic meta-learning for fast adaptation of deep networks.* Proceedings of the International Conference on Machine Learning, 1126–1135.

7. Dhekane, S. G., & Ploetz, T. (2025). *Transfer Learning in Sensor-Based Human Activity Recognition: A Survey.* ACM Computing Surveys.

8. Yuan, X., He, P., Zhu, Q., & Li, X. (2019). *Adversarial examples: Attacks and defenses for deep learning.* IEEE Transactions on Neural Networks and Learning Systems, 30(9), 2805–2824.