Informed Repair of Deep Neural Networks

Akshat Adsule
University of California, Davis
aadsule@ucdavis.edu

Darroll Saddi
University of California, Davis
dwsaddi@ucdavis.edu

Suyash Goel
University of California, Davis
sngoel@ucdavis.edu

1 Introduction

Deep neural networks (DNNs) have become increasingly prevalent in modern applications. DNNs have seen use in virtually every field from air traffic control to self-driving vehicles. However, these models are not infallible and do produce mistakes, which could prove disastrous given the model's application.

Recent research [**? ? ?**] has explored methods of repairing DNNs once incorrect inputs are identified. Repair techniques have the goal of correcting the model's behavior on a specified set of inputs, often referred to as the repair set. The goal of many existing techniques is to adjust network weights and biases while satisfying the conditions of: (i) provable, (ii) generalizing, (iii) architecture-preserving, (iv) scalable, and (v) local repair. In other words, these techniques typically aim to minimally adjust the network's parameters to correct its behavior on the specified repair set, often while providing formal guarantees on the outcome for those inputs or related input regions. While methods that satisfy some or even all of these conditions exist, most require user intervention to select the repair set and which layers of the network should be affected. Thus, in practice, applying these repair methods reveals significant ambiguities that can affect the quality and efficiency of the repair.

Take, for example, APRNN, the method proposed in [**?**] that offers provable, architecture-preserving repair over specified input regions (V-polytopes). APRNN achieves all the previously stated conditions (i-v), and works by provably repairing the network's weights and biases up to a chosen layer. This involves modifying network weights starting primarily at that chosen layer and adjusting biases in subsequent layers.

While effective, APRNN includes critical ambiguities in practice, mainly as a result of the user needing to select the starting layer or affected layers for weight and bias adjustment. The paper itself doesn't prescribe how to choose affected layer, and this choice can significantly impact the repair's effectiveness, efficiency, and efficacy. The choice of repair set, which defines the repair polytope, is also left to the user – fundamentally determining the target behavior of the repair. The composition and scope of this set influence the repair outcome and how well the fix generalizes to similar, unseen inputs. This situation is not ideal because these choices can be arbitrary and greatly impact the quality of the repair.

This paper aims to investigate and develop heuristics to guide the DNN repair process, specifically addressing the ambiguities highlighted above in methods like APRNN. By providing data-driven or structurally-informed ways to make these choices, we seek to enable more efficient and informed repairs of DNNs. We propose to explore and evaluate various heuristics, including but not limited to:

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Layer Selection Heuristics:

Activation-Based Selecting the start layer based on metrics calculated across the repair set, such as the layer exhibiting the highest average activation magnitude or the highest variance in activations.

Gradient/Sensitivity-Based Identifying layers where parameters show the most sensitivity (e.g., largest gradient norms) with respect to the inputs in the repair set, indicating layers most influential on the incorrect output.

Change-Based (for Adversarial Inputs) Selecting the layer whose activations or feature representations changed most drastically between the original and adversarial inputs in the repair set.

Feature-Similarity Based Choosing a layer where the internal representations of the inputs within the repair set are most similar, suggesting a point of unified processing relevant to the required fix.

Layer Type/Position Simple heuristics such as always choosing the first fully-connected layer after convolutional blocks, or the penultimate layer.

Brute Force Exhaustively evaluating all layers and selecting the one that yields the best repair outcome, though this is computationally expensive.

Repair Set Analysis

Diversity Evaluating the diversity of the repair set, such as the number of unique classes or the distribution of inputs across the input space.

Concentration Analyzing the concentration of points defining the polytope, such as the number of points needed to define a convex hull or the dimensionality of the convex hull.

Size Considering the size of the repair set, including the number of points and the dimensionality of the input space.

This project enhances AI trustworthiness by making the crucial process of model repair—itself a method for increasing trust after failures—more efficient and informed. Current repair techniques can involve arbitrary choices, leading to unpredictable outcomes. By developing heuristics to guide decisions within the repair process, such as selecting the optimal network layer for modification, we enable more systematic, reliable, and effective correction of identified flaws. This ultimately increases confidence in the robustness and safety of AI systems by ensuring that necessary fixes are applied more predictably and with a better understanding of their potential impact.

2 Experimental Setup

The experiments are designed to answer the primary research question: *How do different heuristics for layer selection (e.g., activation-based, gradient-based) and repair set analysis (e.g., diversity, concentration) influence the effectiveness, generalization, locality, and efficiency of DNN repair techniques like APRNN?* We aim to determine which heuristics provide the most significant improvements over arbitrary or naive selection strategies.

2.1 Models & Datasets

2.1.1 Selected Architectures

We aim to identify heuristics for a wide variety of model types and repair types. We choose models that are frequenty used in most machine learning tasks. These include:

Multi-Level Perceptrons (MLPs) MLPs are foundational neural networks with hidden layers, fully connected neurons, and non-linear activations used widely for classification and regression tasks.

Convolutional Neural Networks (CNNs) CNNs excel at processing grid-like data, especially images, by using convolutional layers to learn spatial feature hierarchies. CNNs are most used for vision tasks such as classification, object detection, and segmentation.

Vision Transformers (ViTs) ViTs apply the Transformer architecture to images by treating them as sequences of patches and using self-attention to capture global context. They have similar applications to CNNs and are used (as the name suggests) for vision tasks.

Language Transformers Similar to ViTs, Language Transformers leverage self-attention to understand contextual relationships between words in text. Language transforms are used in NLP tasks like machine translation, text summarization, and question answering.

2.1.2 Selected Models

For experimentation purposes, we choose the following pre-trained models for each aforementioned model architecture. We picked models that are relativly well-established for their respective tasks. We also purposefully choose smaller models for ease of experimentation. However, we still expect our results to apply to larger and more complex models.

Our chosen models are:

Clustering MLP By implementing a simple supervised clustering problem, we can create a simple MLP model that is trained to classify points in a 2D space into two clusters. We will use the `make_moons` function from `sklearn.datasets` to generate a dataset, where an MLP will be insufficiently trained on this dataset to classify the points into two clusters. By either increasing the complexity of the dataset (e.g. the number of points, noise), or by not training the MLP enough, we can apply our heuristics to see if any significant improvements can be made, or if there are any significant differences between the heuristics. By training an MLP on a supervised (known labels) clustering task, we can create a simple MLP model that is trained to classify points in a 2D space into two clusters.

SqueezeNet [?] SqueezeNet is a CNN architecture designed for high efficiency by minimizing parameters, employing dimensionality reduction and late downsampling, achieving AlexNet-level accuracy with significantly fewer parameters.

MobileViT-XX-Small [?] MobileViT-XX-Small is a highly efficient hybrid vision model that combines the local feature learning of CNNs with the global context capabilities of Vision Transformers through a specialized MobileViT block, making it ideal for mobile and resource-constrained environments.

DistilBERT [?] DistilBERT is a smaller, faster, and lighter version of BERT achieved through knowledge distillation, significantly reducing its parameter count and inference time.

2.1.3 Selected Datasets

We will use the following datasets for our experimentation that correspond to the selected pre-trained models.

ImageNet [?] ImageNet is a large-scale image dataset with over 14 million labeled images across 20,000 categories, widely used for training and evaluating computer vision models. ImageNet is used by both SqueezeNet and MobileViT-XX-Small.

GLUE [?] The General Language Understanding Evaluation (GLUE) benchmark is a collection of nine diverse NLP tasks designed to evaluate the performance of models on various language understanding challenges. GLUE is used by DistilBERT.

2.1.4 Summary

The following table summarizes the models and datasets we use in our experimentation.

Architecture	Model	Dataset
MLPs	!TODO	!TODO
CNNs	SqueezeNet [?]	ImageNet [?]
Vision Transformers	MobileViT-XX-Small [?]	ImageNet [?]
Language Transformers	DistilBERT [?]	GLUE [?]

2.2 Evaluation Metrics

We will evaluate the heuristics based on the following metrics:

Repair Success Rate The percentage of inputs in the repair set for which the model produces the correct output after repair.

Generalization The model's performance on unseen inputs similar to those in the repair set, measured by accuracy or other relevant metrics.

Locality The extent to which the repair minimally affects the model's behavior on inputs outside the repair set, often quantified by changes in the model's predictions or parameter values.

Efficiency The computational cost of the repair process, including time and resources required.

Scalability The ability of the repair method to handle larger models or repair sets without significant degradation in performance or efficiency.