

Adaptive Noise-Type Routing for CNN Denoising

Team: Zachary De Aguiar, Alexander Green , William Moulton, and Zachary Wood

Motivation and Problem Statement

Real-world images are often degraded by diverse noise types such as Gaussian, salt-and-pepper, Poisson, or compression artifacts.

Traditional CNN-based denoisers are typically trained for a single noise model, which limits their robustness and generalization. A unified model struggles to adapt to different or mixed noise conditions. This project addresses the problem by developing a noise-type adaptive CNN that identifies the noise class and routes the input through specialized denoising branches.

Research Objectives

1. Develop a noise-type adaptive CNN framework that predicts the noise type present in an input image and applies a specialized denoising branch tuned for that type.
2. Evaluate the denoised images by comparing directly to the corresponding clean ground-truth images (original, noise-free dataset images)
3. Test the model's performance across multiple noise types (Gaussian, salt-and-pepper, Poisson, and potential motion blur or JPEG artifacts) and mixed-noise settings.
4. Analyze whether adaptive routing improves reconstruction quality compared to a single unified model when evaluated against clean ground-truth images.

State of the Art, and your Key Novelty

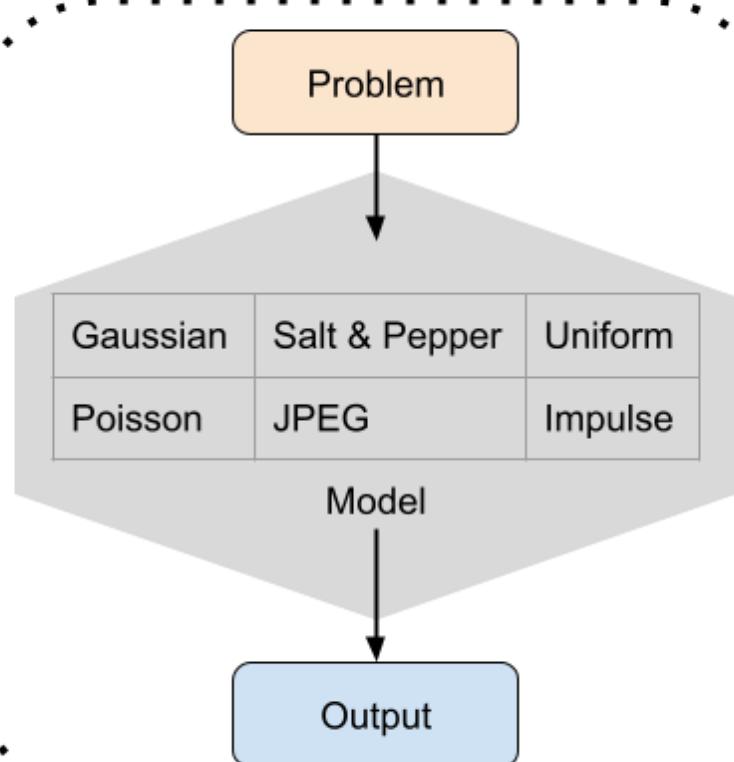
1. **Noise-type awareness:** Directly measures the denoiser's ability to reconstruct clean images, eliminating reliance on comparisons with other models.
2. **Adaptive multi-branch design:** Dynamically routes inputs to noise-type-specific modules or blends outputs, optimized purely for reconstruction quality. This places a focus on generalization without depending on a single fixed-architecture denoiser.

Impact, Outcomes, Achievements

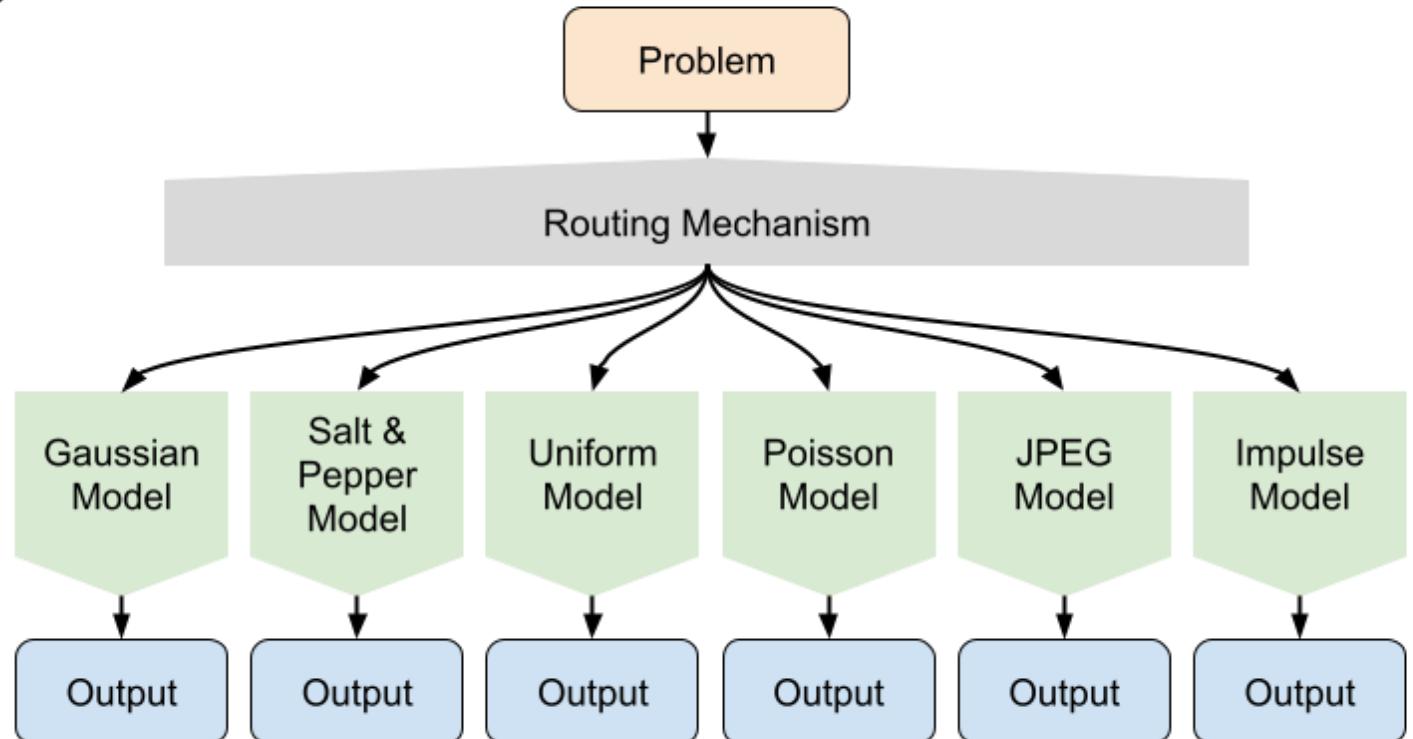
1. **Impact:** Shows that noise-type adaptive CNNs can more effectively recover clean images under varied noise conditions and provides a transparent, ground-truth-based evaluation framework for denoising research.
2. **Outcomes:** A trained adaptive denoising system evaluated on image datasets with multiple synthetic noise types; results include metrics and visual comparisons to original clean images.
3. **Achievements:** End-to-end implementation of a noise-type classifier with adaptive denoising branches, creation of a reproducible benchmark with quantitative and qualitative results.

Approach Comparison

Comprehensive Model



Routing Model



The goal of our project is to help optimize multi-problem AI models by analyzing two different approaches.

1. A very well trained **Comprehensive Model** for dealing with all problems.
2. A **Routing Model** that determines the type of problem and then redirects the user to a specialized AI model.

Action Plan

Phase 1 (Sep 12 – Oct 1):

Goal: Setup and prototype

- Set up environment and dataset loaders (MNIST, CIFAR-10/100, STL-10, etc)
- Implement synthetic noise generators (Gaussian, salt-and-pepper, Poisson, etc)
- Train baseline denoisers and begin implementation of noise-type classifier with basic routing

Deliverable: Prototype pipeline with sample denoised outputs

COMPLETED

Phase 2 (Oct 2 – Nov 6):

Goal: Build full adaptive model and run experiments

- Implement multi-branch adaptive CNN with routing/blending mechanism
- Train model across multiple noise types and datasets
- Evaluate denoised images against clean ground-truth

Deliverable: Trained adaptive model and experimental results

COMPLETED

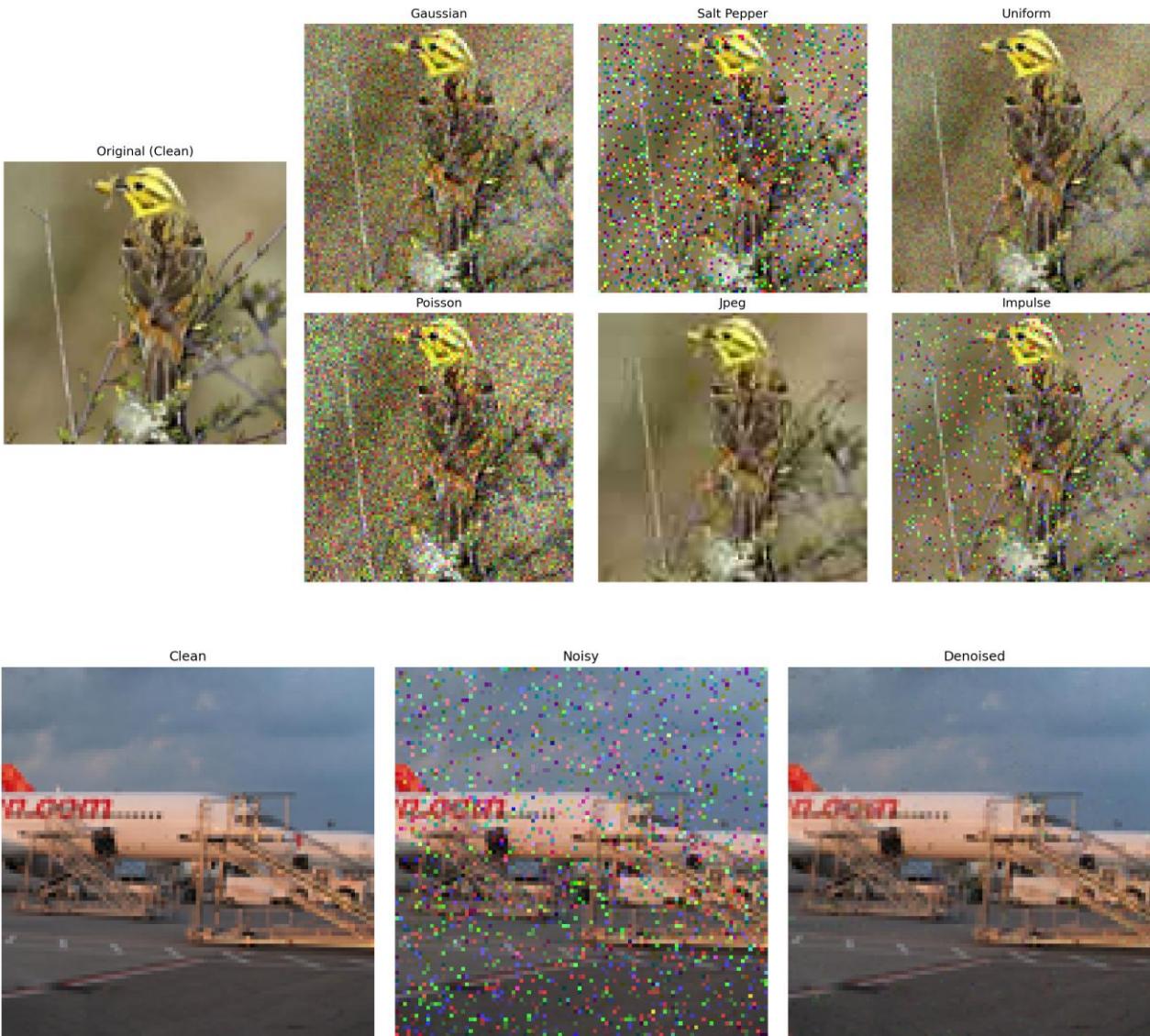
Phase 3 (Nov 7 – Dec 2):

Goal: Analyze results, write report, and prepare presentation

- Analyze quantitative & qualitative results; generate figures (confusion matrix, error curves, etc)
- Write final technical report/paper with methods, experiments, results, and discussion
- Finalize code repository and pre-trained models (README.md, etc)
- Prep presentation

Deliverable: Complete report, reproducible code, and presentation ready

Phase 1 Results So Far



Our first 2 goals of phase 1 were to set up dataset loaders for MNIST, CIFAR-10/100, STL-10, and then to implement synthetic noise generators.

The noise generator implements the following noise types:

- Gaussian
- Salt and Pepper
- Uniform
- Poisson
- JPEG Compression
- Impulse

Our final goal for phase 1 was to train a baseline denoiser and begin the implementation of noise-type classifier for basic routing.

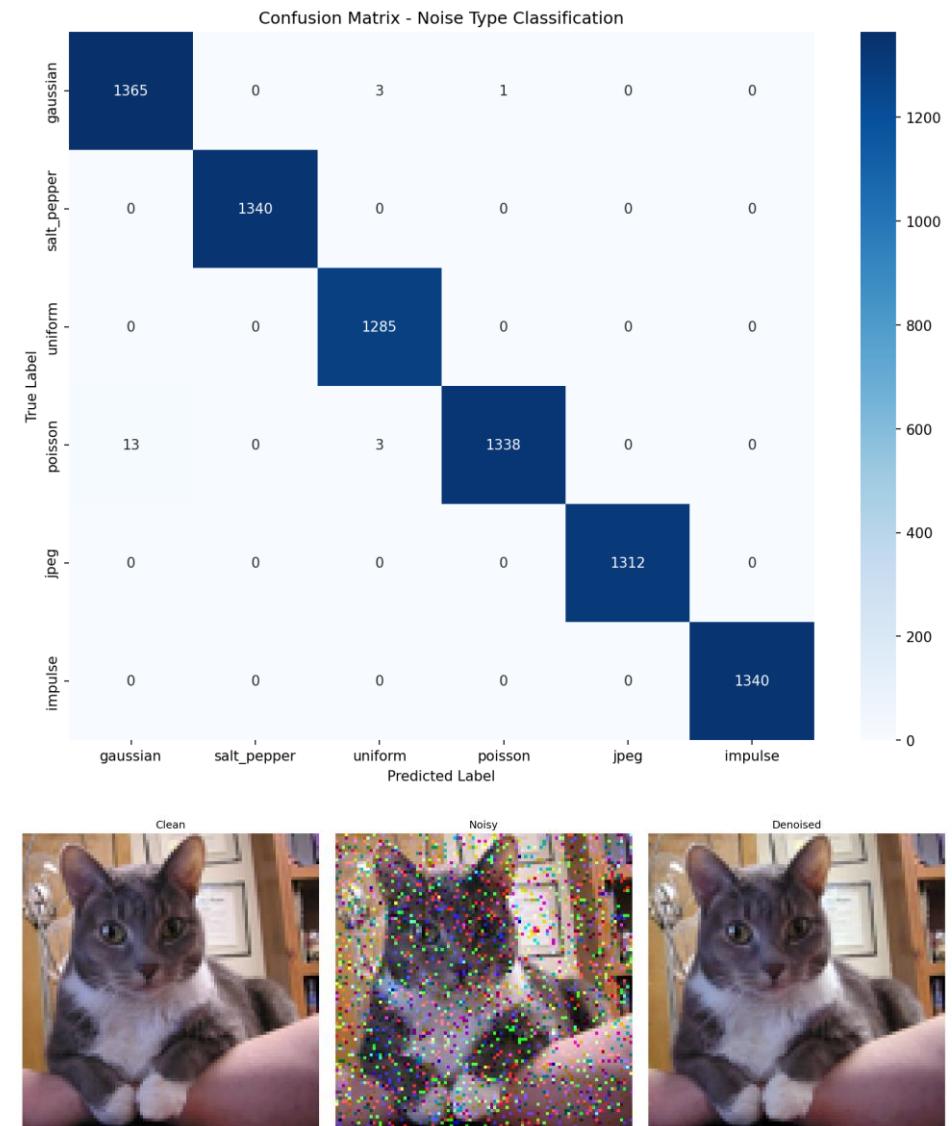
We created a new more rigorous composite loss function:

$$\text{MSE (35\%)} + \text{MAE (35\%)} + \text{SSIM (20\%)} + \text{Gradient (10\%)}$$

Phase 2 Results So Far

For Phase 2, our first goal was to implement multi-branch adaptive CNN with routing mechanism. The confusion matrix on the right shows its accuracy. We then created a unique model, modified to improve performance for each noise type. After, we trained them all against all datasets.

Gaussian	Dilated convolutions for multi-scale context, optimized for smooth noise removal.
Salt and Pepper	Spatial attention mechanisms to identify and correct sparse extreme values.
Uniform	Wider channels with dropout regularization for robustness against uniform distribution.
Poisson	Instance normalization and dual branches for signal-dependent noise handling.
JPEG Compression	Deblocking layers and edge-preserving blocks for compression artifact removal.
Impulse	Corruption detection mask with dilated convolutions for selective pixel replacement.



Phase 3 Path Forward

Noise Type	Loss	PSNR (dB)	SSIM	Count
Gaussian	0.0400	29.54	0.9859	1339
Salt & Pepper	0.0091	35.36	0.9974	1354
Uniform	0.0240	32.84	0.9937	1332
Poisson	0.0644	24.63	0.9599	1338
JPEG	0.0448	27.95	0.9806	1323
Impulse	0.0100	35.99	0.9971	1314
Overall	0.0321	31.05	0.9858	8000

Routing Accuracy: 99.66%

Table 4. Routing Model Performance by Noise Type on STL-10 Dataset

Quantitative and qualitative analysis is underway. We've identified that three models exceed our target loss threshold of 0.03: Gaussian (0.040), Poisson (0.064), and JPEG (0.045).

Next steps:

1. Tune these models to reduce loss below 0.03
2. Finalize the report
3. Clean up code repository for readability