Software Engineering Department
Braude College

Image denoising using Noise2Noise with Resnet

Project code: 24-2-R-22

Supervisor: Prof. Zeev Volkovich
Advisor: Dr. Renata Avros

Amit Vinograd – Amit.Vinograd@e.braude.ac.il
Omer Ben Shimol – Omer.Haim.Ben.Shimol@e.braude.ac.il

Abstract

This project investigates the denoising of images using the Noise2Noise algorithm in combination with ResNet architecture. Unlike traditional approaches, Noise2Noise learns to denoise images using pairs of noisy images without the need of clean references. The goal of the project is to evaluate whether ResNet, known for its deep residual learning capabilities, can enhance this process, offering improved performance over other networks like U-Net. By training on noisy datasets and comparing the results. We aim to assess whether ResNet can achieve the results we are expecting for.

Keywords
Image denoising, Noise2Noise, ResNet50, deep learning, residual learning, Gaussian noise, Poisson noise, impulse noise, image restoration, convolutional neural networks (CNN).

Github link: https://github.com/AmitVinograd/Image-denoising-using-Noise2Noise-with-Resnet-50

1. Introduction and Background

Image restoration is the process of removing distortions from images. Noise is a common form of distortion that can significantly degrade image quality and can make tasks like object recognition and image analysis being complicated or even impossible.

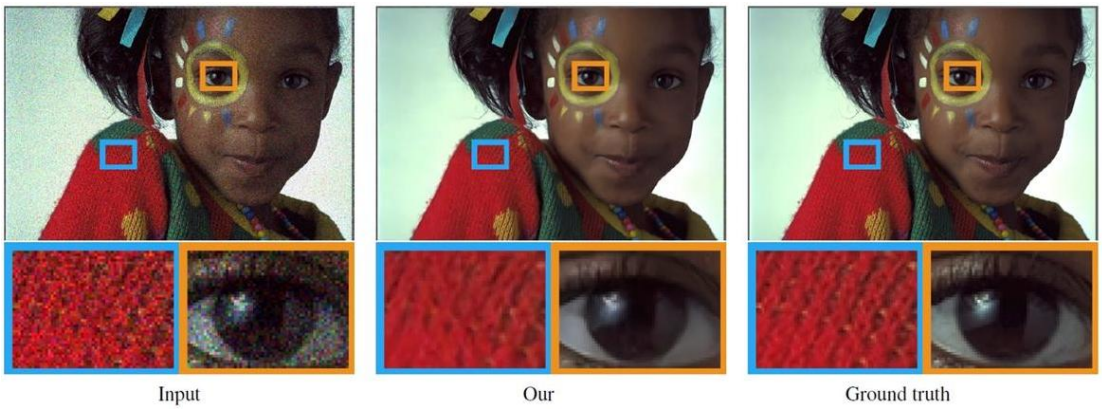
Traditional image denoising methods often rely on prior knowledge of the noise distribution or clean image examples for training. These methods can be expensive or impractical. The Noise2Noise algorithm revolutionized image restoration by learning to remove noise without needing clean data, it makes cleaning images from noises simpler, cheaper and possible.

The Noise2Noise algorithm introduced by Jaakko Lehtinen, Jacob Munkberg and Jon Hasselgren. The algorithm leverages the power of deep learning to remove noise from images without requiring clean data to learn from.

Noise2Noise trains the model on pairs of independently generated noisy images of the same scene. By learning the mapping between these noisy versions, the model can effectively remove noise and reconstruct a clean image.

Noise2Noise can handle various types of noise, including multiplicative Bernoulli noise, which introduces random black pixels causing abrupt intensity transitions, and Poisson noise, common in low-light conditions leading to statistical fluctuations in pixel intensities. It also addresses additive Gaussian noise, characterized by random values from a Gaussian distribution, and random-valued impulse noise, where corrupted pixels get random intensity values. Additionally, it tackles textual noise, which consists of textual overlays degrading image quality. By utilizing deep learning, Noise2Noise provides an innovative and efficient solution to denoise

images, making image restoration more accessible and effective without relying heavily on prior knowledge or clean data examples.



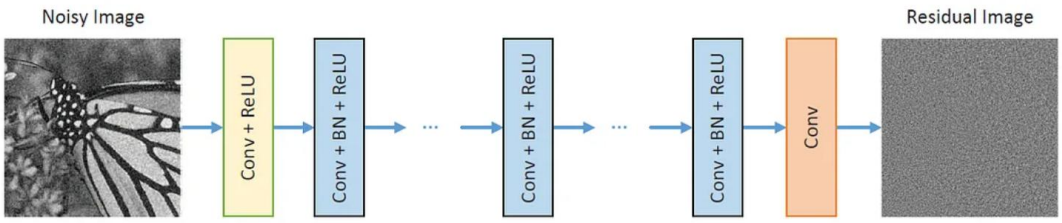
[4] Image restoration by denoising it using Noise2Noise algorithm.

2. Related work

Recent studies in image denoising have introduced several effective methods. DnCNN addresses Gaussian noise reduction using convolutional layers [1]. RedNet employs a residual encoder-decoder structure with pyramid supervision to enhance noise pattern recognition [2]. BM3D utilizes block-matching and collaborative filtering to achieve robust denoising [3]. U-Net, originally designed for segmentation, has shown promise in denoising by training on noisy-cleansed image pairs [4]. These methods pave the way for investigating how ResNet50, known for its deep residual learning capabilities, can further improve image denoising performance.

2.1 DnCNN

DnCNN (Denoising Convolutional Neural Network)[1], one of the first neural networks specifically designed for removing noise from images. This network showed high accuracy in removing Gaussian noise.
By combining convolution with ReLU, DnCNN can gradually separate the image structure from the noisy observation through the hidden layers.



DnCNN model diagram. [10]

In the training phase, the model receives noise-free images and noisy images, and after the noisy image passes through the network, the loss function measures the pixel differences between the clean image and the restored image.
The main problem is that the training phase uses only clean images. Noisy images contain a lot of noise, simulating the noise that appears in real MRI images.
Models trained on noisy images learn to deal with these noises, and therefore will be less sensitive to changes in imaging conditions.
The performance of the model will be more stable when it is applied to images taken with different noises.

2.2 Residual Encoder-Decoder Networks (RedNet) [7]

RedNet utilizes a residual encoder-decoder architecture for RGB-D semantic segmentation. The network's encoder compresses the image to capture essential features, while the decoder reconstructs the image, maintaining important details through residual connections. This design helps the network learn noise patterns and differentiate between noisy and clean images, resulting in improved generalization and performance.

2.2.1 Pyramid Supervision

Pyramid supervision is a training method used in RedNet that applies supervised learning at multiple layers of the decoder. This approach optimizes the network by addressing the gradient vanishing problem, ensuring that the network learns effectively at different levels of detail. This technique enhances the model's ability to generalize to various types of noise and maintain stable performance.

2.2.2 Depth Information Fusion

Depth information fusion in RedNet involves combining features extracted from both RGB and depth images. The network processes RGB and depth images separately, then fuses their features over several layers. This method leverages depth cues to improve the accuracy and robustness of semantic segmentation in indoor scenes.

2.3 BM3D

BM3D (Block-Matching and 3D Filtering) is a classic image denoising method that while not using a neural network, has inspired many modern neural network-based denoising algorithms. It was developed by researchers from Tampere University of Technology and is based on grouping similar 2D image blocks into 3D data arrays and applying collaborative filtering.

The method involves three main steps:

Block matching: Similar blocks in the noisy image are identified and grouped together.

Collaborative filtering: 3D transformation and hard thresholding are applied to the grouped blocks to reduce noise.

Aggregation: The filtered blocks are aggregated back into the image to produce the denoised result.

BM3D remains one of the most effective traditional methods for image denoising and serves as a benchmark for evaluating the performance of modern denoising algorithms. Its principles of block matching and collaborative filtering have influenced neural network-based approaches, like learning effective features and reducing noise through grouping and aggregation techniques.

2.4 U-net

U-Net represents a specialized convolutional neural network architecture designed primarily for image segmentation tasks, notable for its effectiveness in handling variable lighting and blur conditions.

Originally developed for biomedical image segmentation, U-Net architectures excel in scenarios where the output requires the same spatial resolution as the input. This capability makes them highly suitable for tasks such as creating segmentation masks and performing image processing tasks like super-resolution or colorization.

The U-Net architecture used for image denoising, as described in the article "Noise2Noise: Learning Image Restoration without Clean Data," consists of an encoder and a decoder. The encoder compresses the input image into a lower-dimensional representation by progressively extracting essential features through convolutional layers followed by pooling operations, which reduce the spatial dimensions. Conversely, the decoder reconstructs the image to its original size through a series of upsampling operations, typically using transposed convolutions, thereby restoring the spatial resolution lost during downsampling. A key feature of U-Net is the use of skip connections, which directly connect corresponding layers in the encoder and decoder. These connections transfer high-resolution features from the encoder to the decoder, aiding in the accurate reconstruction of the image.

The unique aspect of the Noise2Noise training approach is that the network is trained using pairs of noisy images instead of clean ones. This method leverages synthetic noise, such as Gaussian or Poisson noise, added to the original images to simulate real-world conditions. By learning to map from a noisy input image to another noisy version, the network effectively denoises images during inference without the need for clean training data.

For a set of corrupted observations $\hat{x}_i$ the goal is to find the clean signal y_i Use a loss function $L(F_\theta(\hat{x}_i), y_i)$ where F_θ is the model with parameters θ. The loss is minimized to learn the mapping from corrupted to clean images. Even if the targets are corrupted, if the noise has zero mean, the expectation of the corrupted targets equals the clean target. This allows the model to learn from corrupted data effectively.

The training process utilizes the L2 loss function, which measures the average squared difference between the estimated values and the actual value. The L2 loss function is used because it is effective at penalizing larger errors more than smaller ones, making it suitable for image restoration tasks. By minimizing the L2 loss, the neural network learns to produce outputs that are as close as possible to the clean target images, even when trained with noisy inputs. The L2 loss function is represented as $L_2 = L(z, y) = (z - y)^2$.

Additionally, the L1 loss function, which minimizes the absolute differences, is mentioned as an alternative for specific tasks like text removal where it performs better than the L2 loss function. The L1 loss function, represented as $L(z, y) = |z - y|$, aims to find the median of the data distribution. This loss is effective in reducing the impact of outliers, making it suitable for scenarios where fewer than 50% of the pixels are corrupted

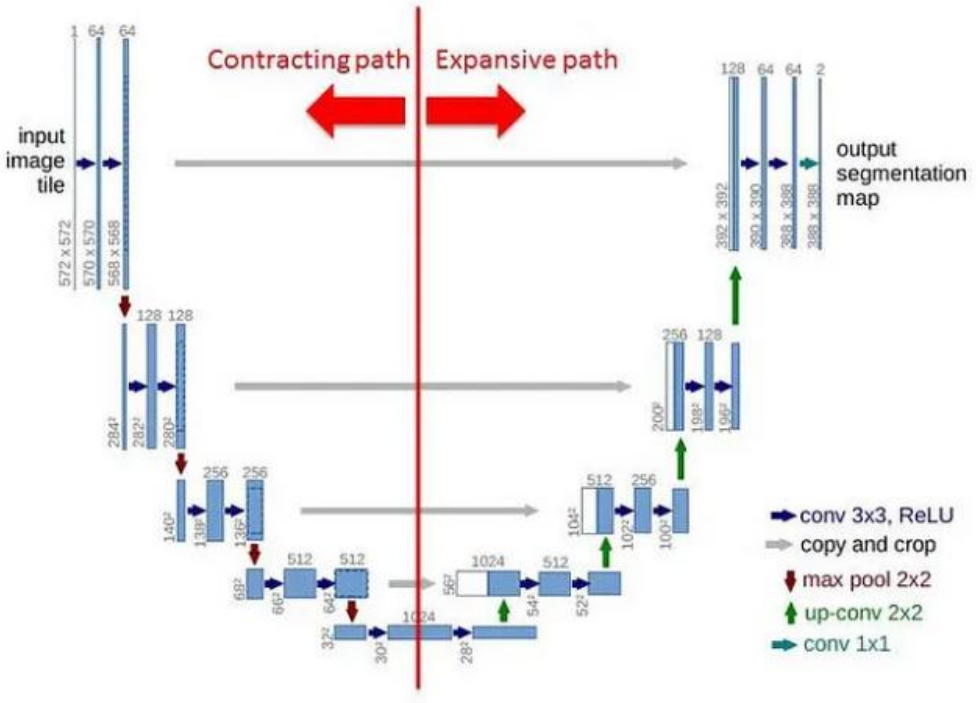
Input ($p \approx 0.25$)	L_2	L_1	Clean targets
17.12 dB	26.89 dB	35.75 dB	35.82 dB

[4] Denoising example using U-Net, using only L1 and L2 functions.

The L0 loss function measures the number of non-zero differences between the estimated values and the actual values. Unlike the L2 loss function, which penalizes larger errors more than smaller ones, the L0 loss function focuses solely on the count of errors, regardless of their magnitude. This makes the L0 loss particularly effective in applications where sparsity is important, such as feature selection or image compression tasks. By minimizing the L0 loss, the neural network learns to produce outputs with the fewest possible errors, promoting sparsity in the model's parameters. The L0 loss function is mathematically represented as $L_0 = L(z, y) = \sum 1(z \neq y)$ where 1is the indicator function. Additionally, the L0 loss function is advantageous in scenarios where the goal is to identify a few significant features or pixels, as it directly penalizes the presence of errors rather than their magnitude. This property makes it suitable for tasks like anomaly detection or certain types of denoising where the focus is on detecting the occurrence of noise rather than its intensity. The L0 loss function thus promotes both sparsity and robustness in the model's output, making it an effective choice for applications requiring precise identification of relevant features.

[4] Denoising example using U-Net, using L0, L1 and L2 functions.

2.4.1.4 U-NET Architecture



[7] Schematic sketch of the U-net architecture

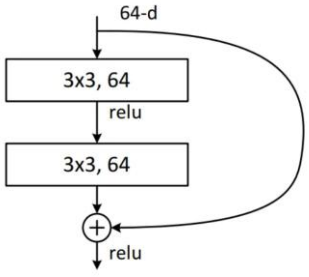
Pooling layers (POOL): Reduce spatial dimensions to enhance computational efficiency in the encoder pathway.
Upsampling layers (UPSAMPLE): Increase spatial dimensions in the decoder pathway to reconstruct the output.
Concatenation (CONCAT): At each decoding step, the network concatenates corresponding feature maps from the encoder, preserving detailed information for improved segmentation accuracy.
Encoder convolutional layers (ENC_CONV 3x3): Utilize 3x3 filters to extract features.

Decoder convolutional layers (DEC_CONV): Refine features following concatenation to ensure a detailed and contextually rich output.

2.5 ResNet-50

ResNet-50 is an advanced convolutional neural network architecture developed by Microsoft Research in 2015**.**
ResNet-50 represents a specific version of the Residual Network architecture, featuring 50 layers. This architecture has become popular in deep learning due to its ability to train very deep networks without suffering from the vanishing gradient problem, because of the introduction of residual connections.
ResNet-50 has proven to be highly effective in image segmentation, object detection, and image denoising. The key innovation in ResNet-50 is the use of residual blocks, which include shortcut connections that bypass one or more layers, allowing gradients to flow directly through the network. This enables the construction of much deeper networks, significantly improving performance and training efficiency.
In the context of image denoising, ResNet-50 can be adapted for effective image restoration tasks. The architecture consists of multiple residual blocks, each containing convolutional layers, batch normalization, and ReLU activation functions. These blocks are designed to learn the residual mapping instead of directly learning the clean image. This approach enhances the network's ability to generalize and maintain high performance even when trained with only noisy data.

A typical ResNet-50 architecture for image denoising begins with an initial convolutional layer that processes the input image, followed by a series of residual blocks. Each block includes shortcut connections that add the input of the block to its output, allowing gradients to flow more easily through the network during training. The final layers typically include a global average pooling layer and a fully connected layer that outputs the restored image.



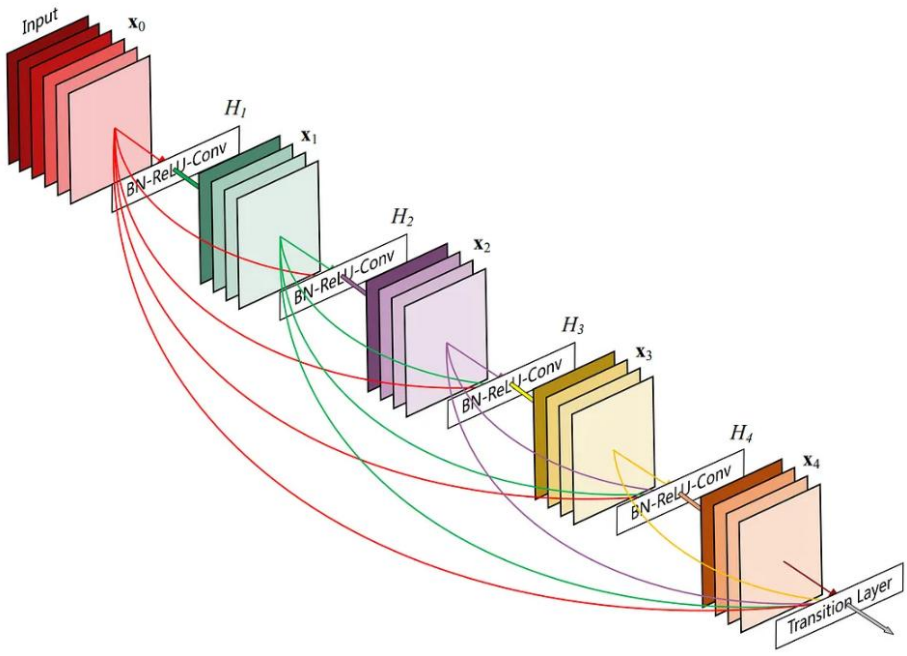
[9] A residual block scheme

The resulting output of the block, which includes both the transformed input and the original input, is then passed on to the next block or layer.

2.5.1 Architecture

Convolutional layer: This layer is responsible for extracting features from the image using convolutional filters.

Batch normalization layer: This layer is responsible for stabilizing the distribution of the values in the convolution layer, which makes training the network easier.
ReLU function: This function is responsible for turning negative values into zero.



[6] Residual blocks diagram.

2.6 ResNet-101

ResNet-101 is a deep convolutional neural network. It is part of the Residual Network family and is distinguished by its depth, containing 101 layers. This architecture effectively addresses the vanishing gradient problem, which often hinders the training of deep neural networks, by employing residual connections.
Known for its versatility, ResNet-101 excels in tasks like image segmentation, object detection, and image denoising. The defining feature of this architecture is its use of residual blocks, which include shortcut pathways that bypass specific layers. These pathways allow gradients to propagate directly through the network, enabling deeper architectures to achieve improved performance and more efficient training.
For image denoising applications, ResNet-101 can be customized to perform image restoration effectively. Its structure consists of multiple residual blocks, each incorporating convolutional layers, batch normalization, and ReLU activations. Instead of reconstructing clean images directly, the network focuses on learning the residuals—the difference between noisy and clean images—thereby enhancing its ability to generalize and perform well even when trained with limited noisy data.
A typical implementation of ResNet-101 for denoising begins with an initial convolutional layer to preprocess the input image, followed by a series of residual

blocks. These blocks leverage shortcut connections that add the input to the block's output, ensuring smooth gradient flow during training. The architecture typically concludes with a global average pooling layer and a fully connected layer, which generate the final restored image.

2.7 ResNet-152

ResNet-152, developed by Microsoft Research is one of the deeper architectures in the Residual Network family, containing 152 layers. It addresses the challenges of training very deep neural networks, particularly the vanishing gradient problem, by introducing residual connections. These connections simplify optimization and enhance performance, even in extremely deep models.
ResNet-152 is widely recognized for its effectiveness in various computer vision tasks, such as image classification, segmentation, object detection, and image denoising. Its hallmark feature is the residual block, which uses shortcut connections to bypass one or more layers. This design ensures that gradients can flow smoothly across the network, facilitating the training of deeper models and achieving superior results.
In the context of image denoising, ResNet-152 can be adapted to perform image restoration tasks with high efficiency. Its architecture includes numerous residual blocks, each composed of convolutional layers, batch normalization, and ReLU activations. The network is designed to focus on learning the residual mapping rather than directly reconstructing the clean image, which enhances its ability to generalize and maintain performance even when trained on noisy data alone.
A standard ResNet-152 architecture for image denoising starts with an initial convolutional layer to process the input image, followed by a deep series of residual blocks. These blocks incorporate shortcut connections, allowing the network to maintain smooth gradient flow and efficient training. The architecture typically concludes with a global average pooling layer and a fully connected layer, which outputs the restored image.

2.8 Loss Functions

2.8.1 L0 Loss - (Zero-Norm Loss) [12]:

- Measures the number of non-zero differences between predicted and true values.
- Not commonly used due to its discontinuous nature, making it hard to optimize with gradient-based methods.
- Encourages sparsity in the model, often used in feature selection.
- The L0 loss function is mathematically represented as $L_0 = L(z, y) = \sum 1(z \neq y)$ where '1' is the indicator function.

2.8.2 L1 Loss - (Mean Absolute Error - MAE) [13]:

- Measures the average absolute differences between predicted and true values.

- More robust to outliers compared to L2 loss.
- Encourages sparsity in the model weights, making it useful for creating simpler models.
- The L1 loss function, represented as $L(z, y) = |z - y|$

2.8.3 L2 Loss - (Mean Squared Error – MSE) [11]:

- Measures the average squared differences between predicted and true values.
- Penalizes larger errors more heavily than L1 loss.
- Commonly used in regression tasks due to its smooth and continuous nature, making it easier to optimize with gradient-based methods.
- The L2 loss function is represented as $L_2 = L(z, y) = (z - y)^2$.

2.9 U-Net and ResNet comparison:

Purpose:
- U-Net: Designed for image segmentation, producing pixel-wise predictions.
- ResNet: A deep residual network for image classification feature extraction, often used in object recognition tasks.

Architecture:
- U-Net: Has an encoder-decoder structure with skip connections that link the encoder to the decoder to recover fine details.
- ResNet: Built from residual blocks with shortcut connections that help train very deep networks by learning residuals.

Skip Connections:
- U-Net: Used for transferring spatial information between the encoder and decoder.
- ResNet: Used to bypass layers and improve gradient flow in deep networks.

Output:
- U-Net: Outputs a segmentation mask (same size as input image).
- ResNet: Outputs class probabilities for classification.

For Noise2Noise project, U-Net is suited for pixel-wise tasks, while ResNet may need modification to handle denoising or similar tasks effectively.

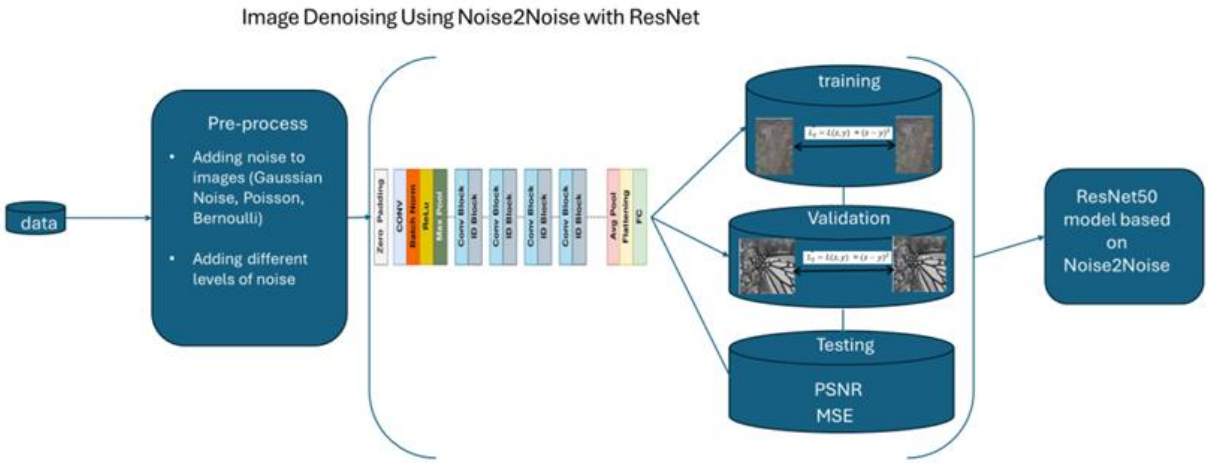
3. Process

The model is based on Noise2Noise algorithm architecture. Our research check if there is more efficient way to denoise images in faster and more accurate. It also checks if there is a way to improve denoising capabilities across various types of noise. We will use images from few datasets to train our model, by using pairs of noisy images. In our research we will explore hyperparameters to optimize the performance of the model. To test the model's accuracy we will use loss functions. Our model will be based on ResNet instead of U-NET. By using Noise2Noise model based on ResNet, we aim to develop an enhanced version of Noise2Noise that has better denoising performance across a wide range of noise types. Our approach ensures that the algorithm remains both effective and efficient.

3.1. Pre-processing

Since the aim is to train ResNet with noisy images, the noisy data itself needs to be carefully prepared. This includes ensuring that the noise is representative of the real-world scenarios where the model will be deployed. The data should include a variety of noise types and levels to improve the model's robustness and generalization capabilities.
These pre-processing steps are crucial for preparing the data, ensuring that it is consistent, and well-suited for training the ResNet model in the Noise2Noise framework. This approach ultimately aims to enable effective noise removal from images through the model's learning process.

4. Model Diagram



This diagram shows the steps in using ResNet for the Noise2Noise algorithm. The approach begins with adding various types of noise to the data, then using the modified ResNet architecture, which is tailored to image denoising. The training process involves predicting clean images from noisy pairs, and the model's performance is assessed using metrics like PSNR and MSE. This adaptation leverages the strengths of ResNet's residual learning to effectively handle the image denoising.

5. Research Process:

We began our project by conducting an in-depth exploration of the Noise2Noise algorithm architecture and its typical applications in image denoising. Traditionally, this architecture has employed U-Net as its backbone for feature extraction and

reconstruction, so we made research about U-Net. However, our goal was to investigate whether replacing U-Net with ResNet, so we had to gather information about ResNet and its advantages. ResNet is a model with superior feature extraction capabilities, that might improve denoising performance using Noise2Noise.

The Caltech101 dataset was selected for this study as it provides a diverse set of images that would test the model's robustness to different noise types. The dataset contains 101 distinct object categories, which ensures that our experiments generalize well across varied image content. Our dataset preprocessing pipeline involved resizing all images to 128x128 dimensions to save resources while maintaining sufficient resolution for denoising tasks.

For implementing the model, we chose Python due to its robust ecosystem of machine learning libraries. These libraries significantly streamlined our work, providing pre-built modules and efficient tools for neural network development. TensorFlow, a Python libarary played a crucial role by enabling the integration with our dataset and accelerating our workflow. To address the computational demands of training the model, we opted to run our project on Google Colab. Google Colab provided several key advantages, including cloud-based accessibility, which allowed us to work from any location without being tied to a specific machine.

Additionally, we utilized a Google Colab Pro Plus account, which granted us access to 52GB of RAM and powerful hardware resources such as CPUs and GPUs. These resources made it possible to train our model efficiently and cost-effectively, eliminating the need to invest in high-end computing equipment. The availability of these resources ensured that our machine learning model, which would have been too resource-intensive for standard PCs, could run smoothly and without limitations.

Data Preparation

We began by splitting the dataset into training and testing sets. Each image was normalized by scaling pixel values between 0 and 1. Noise was then added to the dataset using three different noise types: Gaussian, Poisson, and Bernoulli. Parameters for each noise type were carefully chosen to simulate realistic scenarios:

- **Gaussian Noise:** We used a noise factor of 0.3, simulating random fluctuations in pixel intensity.
- **Poisson Noise:** Images were scaled with a factor of 30 to simulate photon noise common in low-light photography.
- **Bernoulli Noise:** A probability of 0.05 was used for random binary pixel corruption, representing salt-and-pepper noise.

These noisy datasets were generated in pairs. This pairing is fundamental to Noise2Noise training, as the model learns to map noisy inputs to noisy targets.

Model Architecture

To adapt ResNet for the Noise2Noise task, we modified its structure to function as an encoder-decoder model. The encoder consisted of convolutional layers from ResNet, while the decoder comprised upsampling layers and additional convolutional layers to reconstruct the denoised image. This architecture is a significant deviation from the standard Noise2Noise approach that relies on U-Net.

Our ResNet-based architecture was designed with computational efficiency in mind. The input resolution was reduced to 128x128 to accommodate memory constraints, and the batch size was set to 16. The loss function was Mean Squared Error (MSE). The choice of MSE was motivated by its effectiveness in penalizing deviations in pixel intensities.

Training and Evaluation

We trained the model for 50 epochs for each noise type, monitoring loss and mean absolute error (MAE) during training and validation. Early stopping was not employed, as we wanted to observe the full learning curve for research purposes. Metrics were recorded for each noise type to compare their impact on model performance.

Visualization played a critical role in our evaluation. We plotted the training and validation loss curves for each noise type, which revealed the model's ability to generalize to unseen data. Additionally, we displayed sample outputs, showing the original image, noisy input, and denoised output for qualitative analysis.

Challenges and Adjustments

One challenge we encountered was the dataset we chose. At the beginning of this process, we tried to use different datasets, these datasets contained small number of images or very low resolution images. This was not enough to train the model good enough to denoise images well.

Another challenge was managing memory usage during training. By reducing input dimensions and batch size, we ensured the experiments could be conducted within available hardware constraints without compromising the integrity of the study.

6. Dataset:

As part of our project, we looked for a dataset that could effectively train our Noise2Noise model with ResNet instead of U-Net for image denoising. We needed a dataset with a wide variety of images to challenge the model and help it learn to handle different types of noise patterns. After exploring several datasets, we decided to use the Caltech101 dataset. This dataset contains over 9,000 images from 101 object categories. The images are labeled and cover a wide range of content, providing the diversity we needed for our experiments.

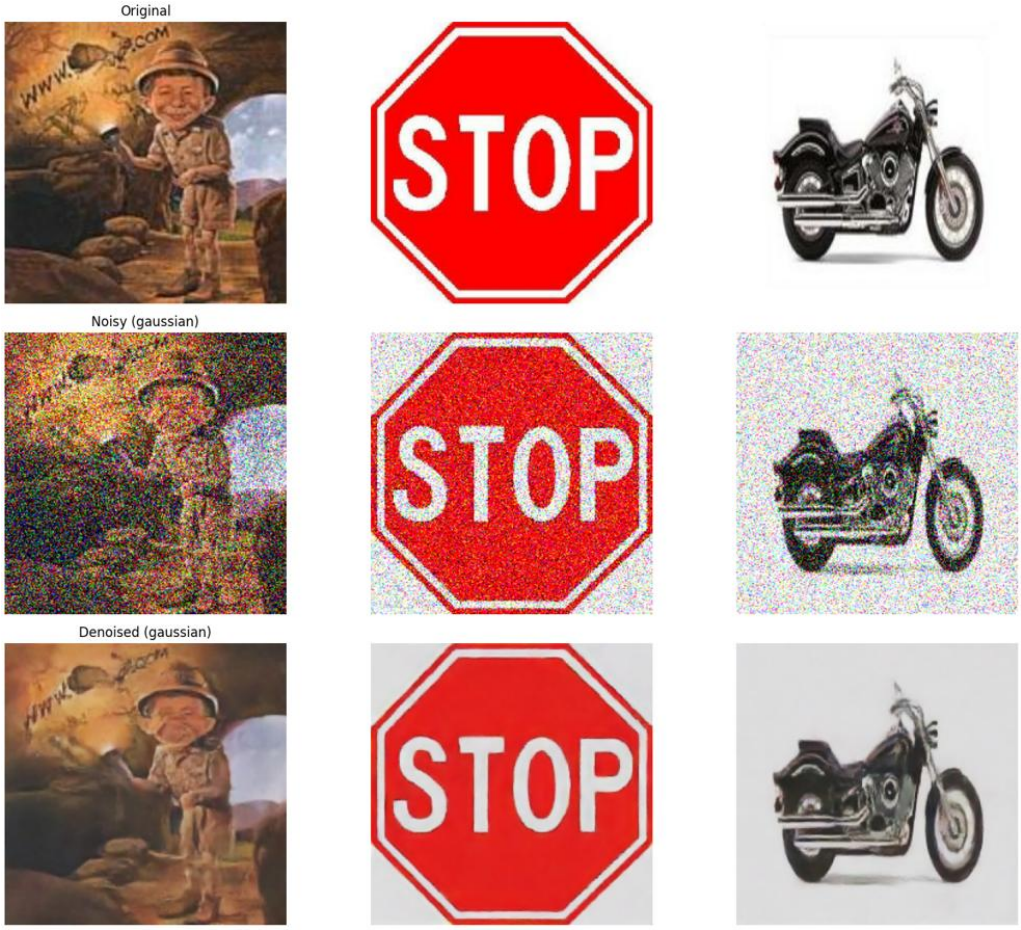
Using the Caltech101 dataset allowed us to apply and study noise on images with various textures, shapes, and patterns, which helped us train the model for real-world noise scenarios. Before training, we resized the images to smaller dimensions to make the training process faster and more efficient while still keeping the important details needed for denoising. The combination of the dataset's diversity and its manageable size made it a great choice for showing how our modified Noise2Noise framework works.

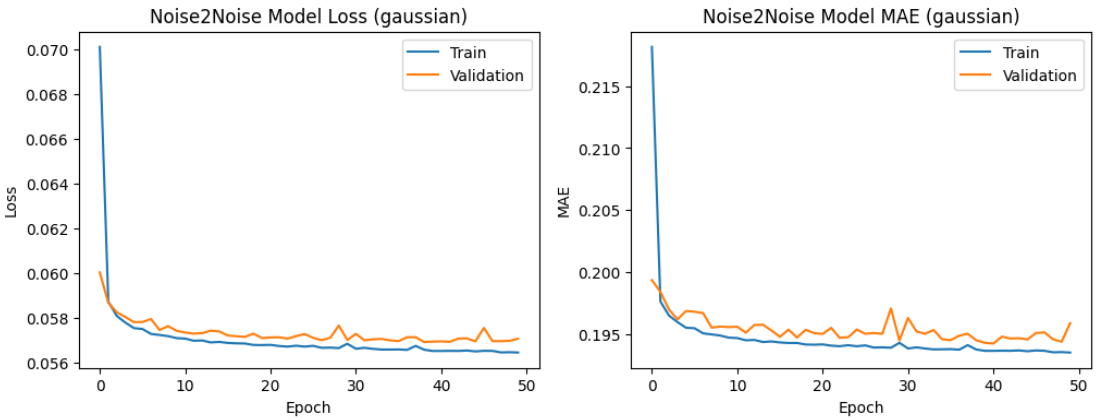
7. Results:

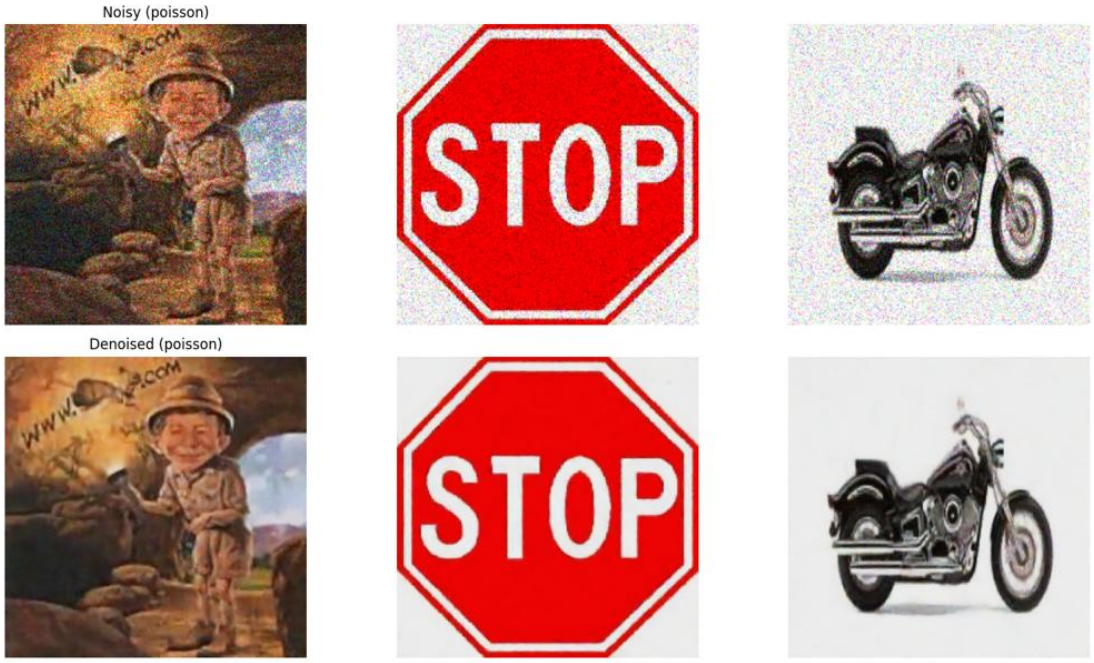
The images below are the results of Noise2Noise with different CNNs.

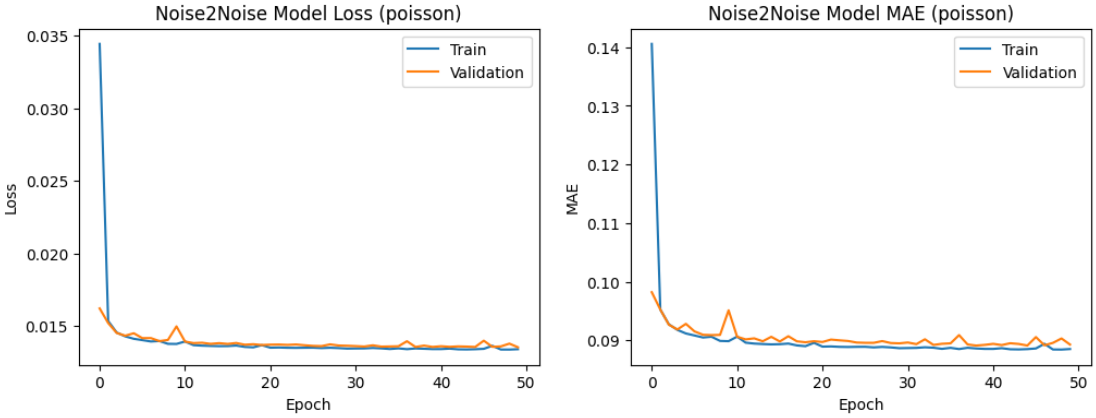
We ran the Noise2Noise with U-Net, Resnet50, Resnet101 and Resnet152.

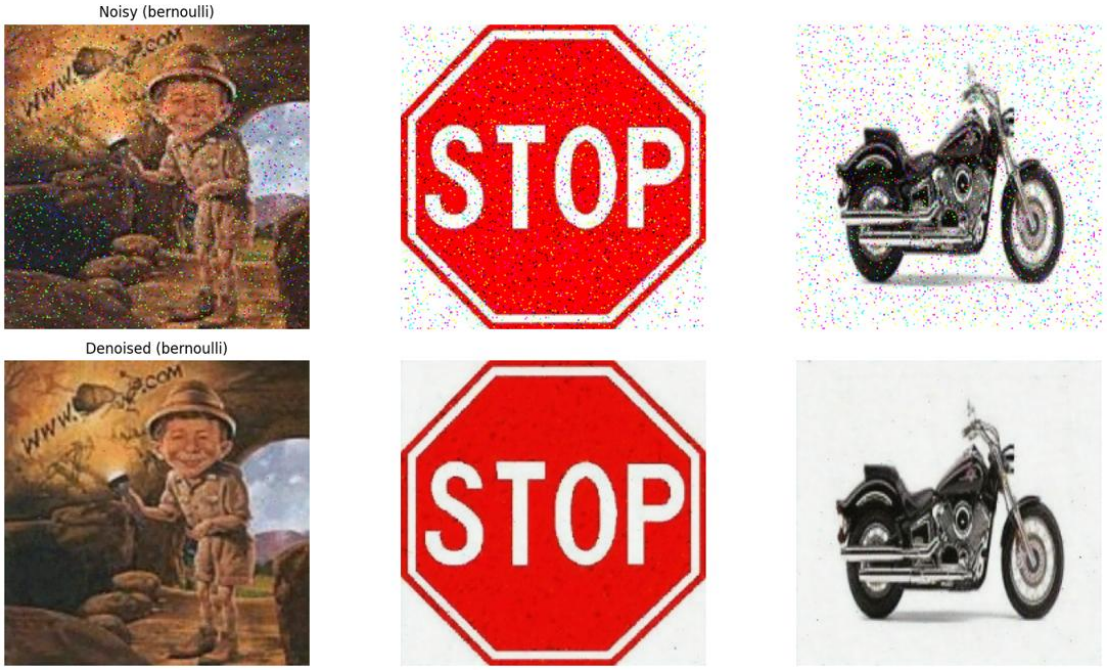
U-Net:

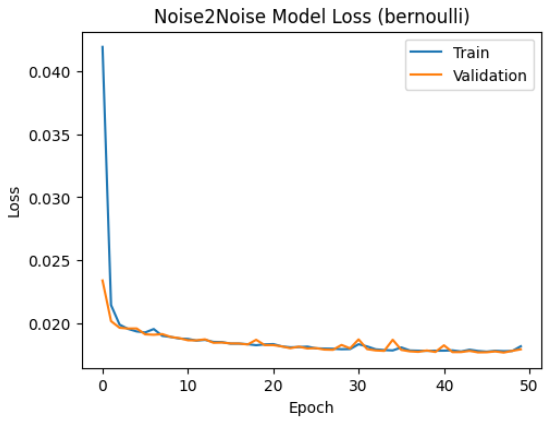


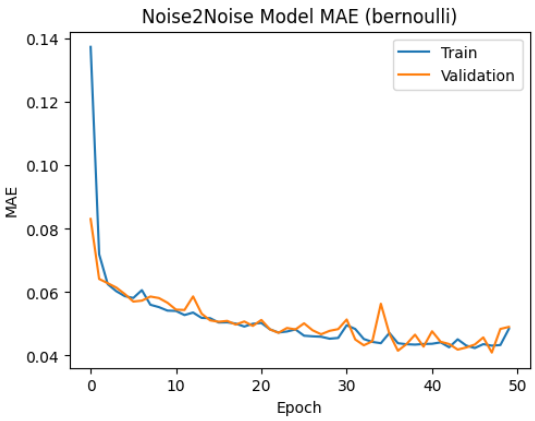



Noisy (poisson)
Denoised (poisson)
STOP
STOP


Noise2Noise Model Loss (poisson)
Train
Validation
Loss
Epoch
0.035
0.030
0.025
0.020
0.015
0
10
20
30
40
50
Noise2Noise Model MAE (poisson)
Train
Validation
MAE
Epoch
0.14
0.13
0.12
0.11
0.10
0.09

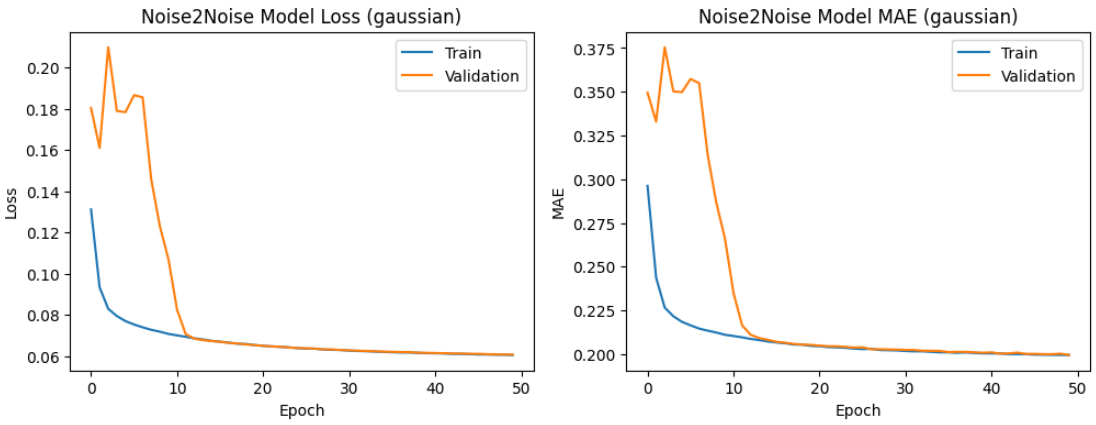

Noisy (bernoulli)
Denoised (bernoulli)
STOP
STOP

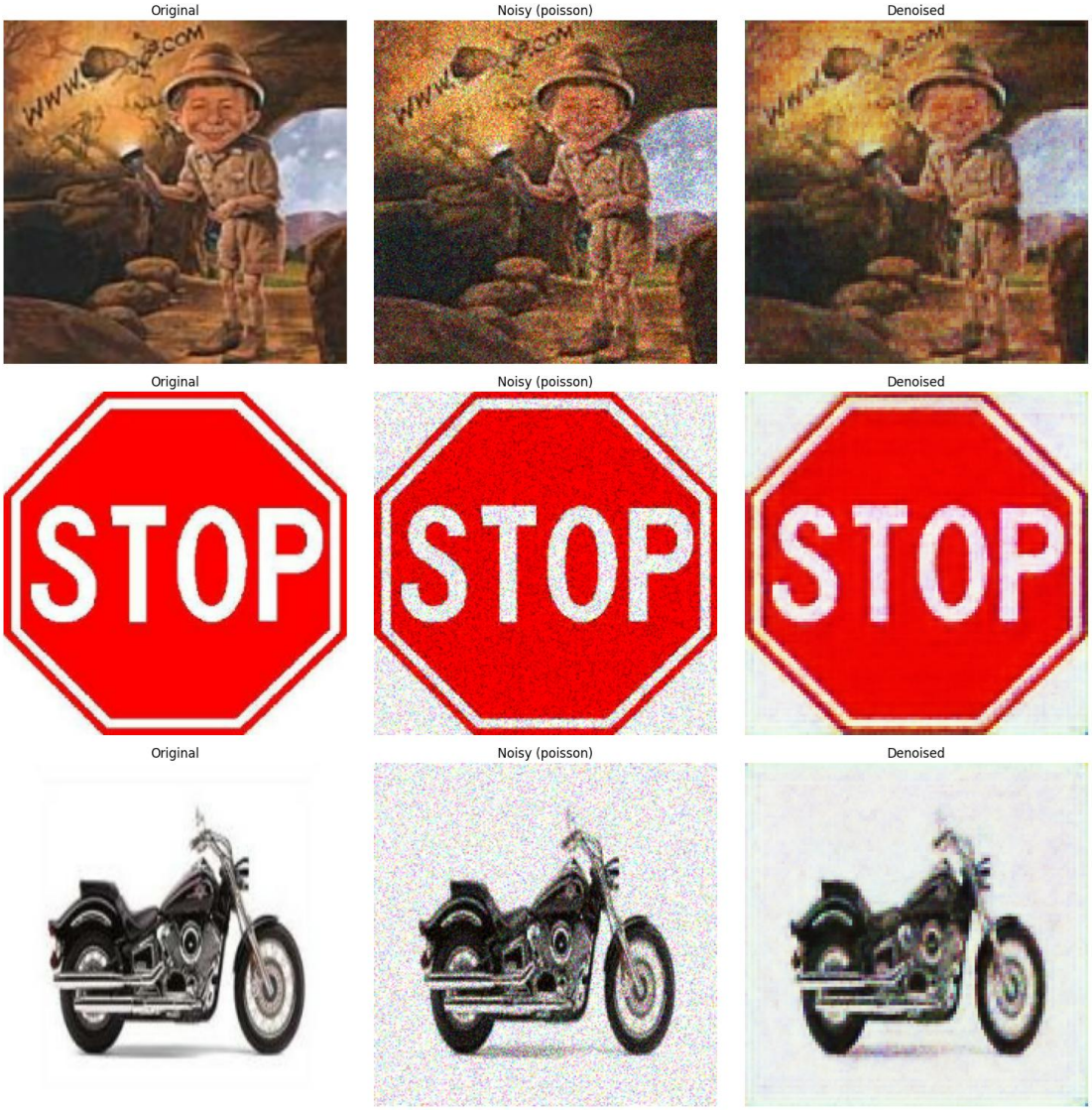

Noise2Noise Model Loss (bernoulli)
Train
Validation
Loss
Epoch
0.040
0.035
0.030
0.025
0.020
0
10
20
30
40
50

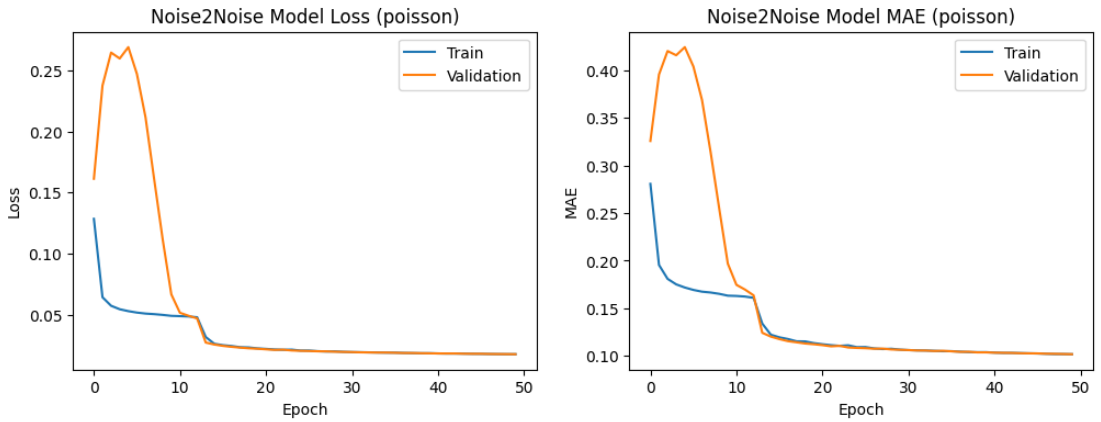

Noise2Noise Model MAE (bernoulli)
Train
Validation
MAE
Epoch
0.14
0.12
0.10
0.08
0.06
0.04
0
10
20
30
40
50

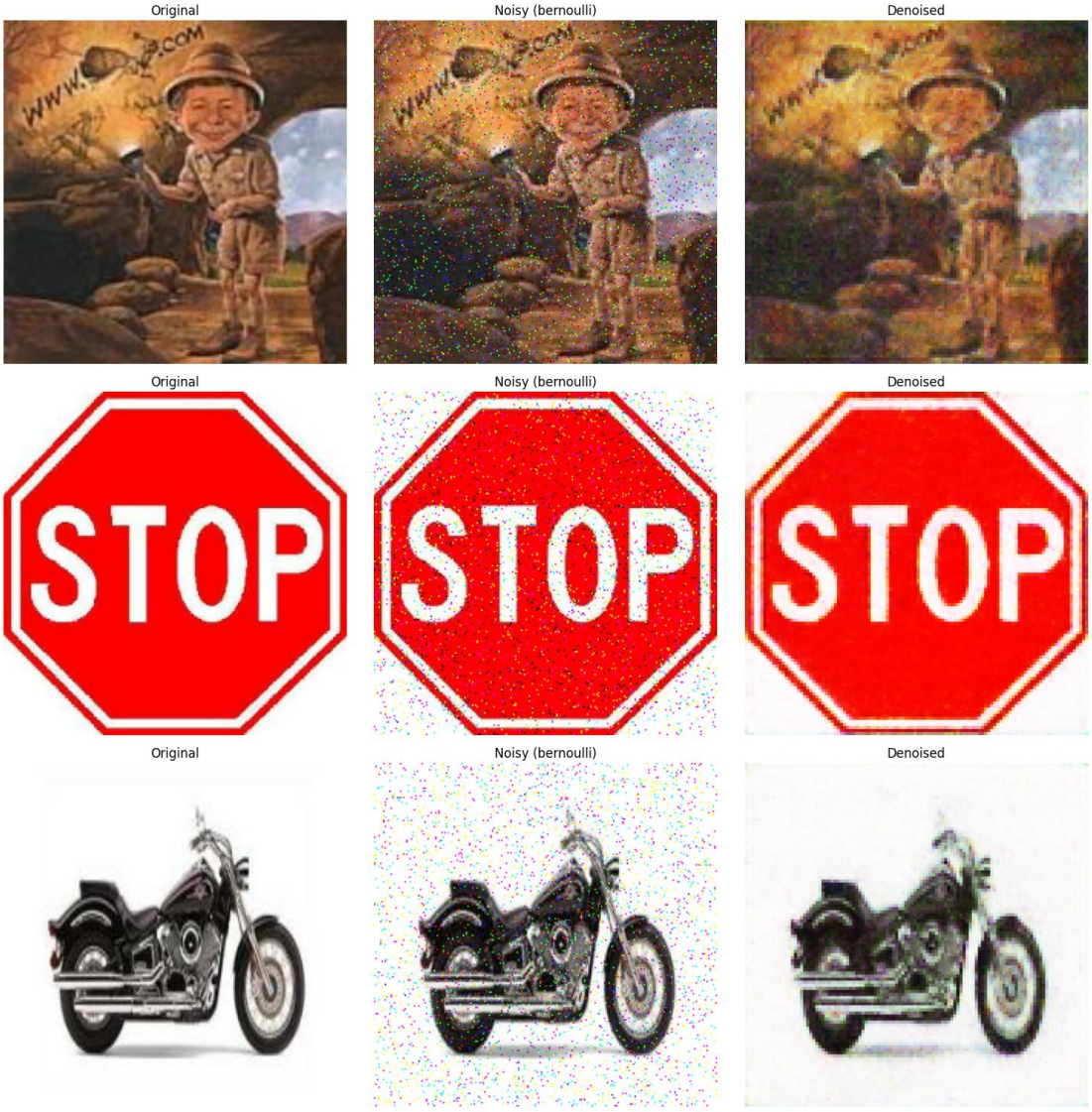
Resnet50:

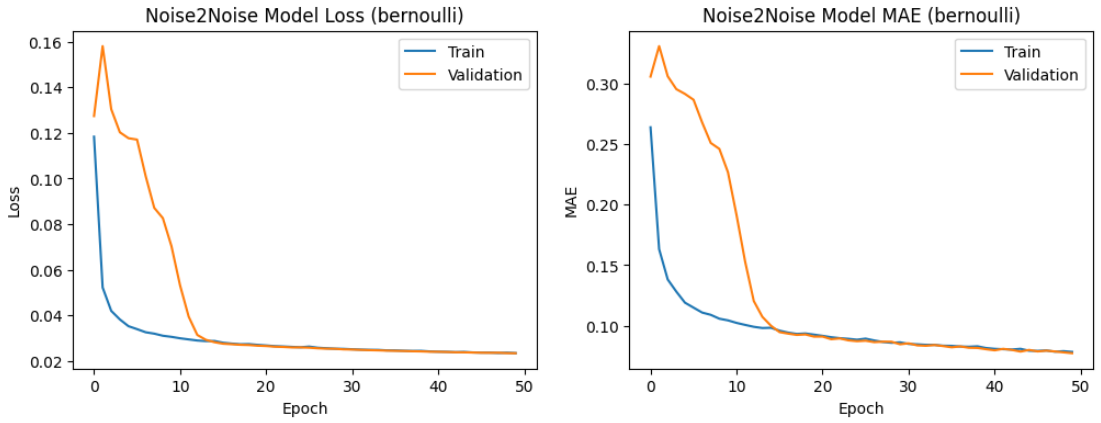

Original
Noisy (gaussian)
Denoised
Original
Noisy (gaussian)
Denoised
Original
Noisy (gaussian)
Denoised


Noise2Noise Model Loss (gaussian)
Train
Validation
Loss
Epoch
Noise2Noise Model MAE (gaussian)
Train
Validation
MAE
Epoch

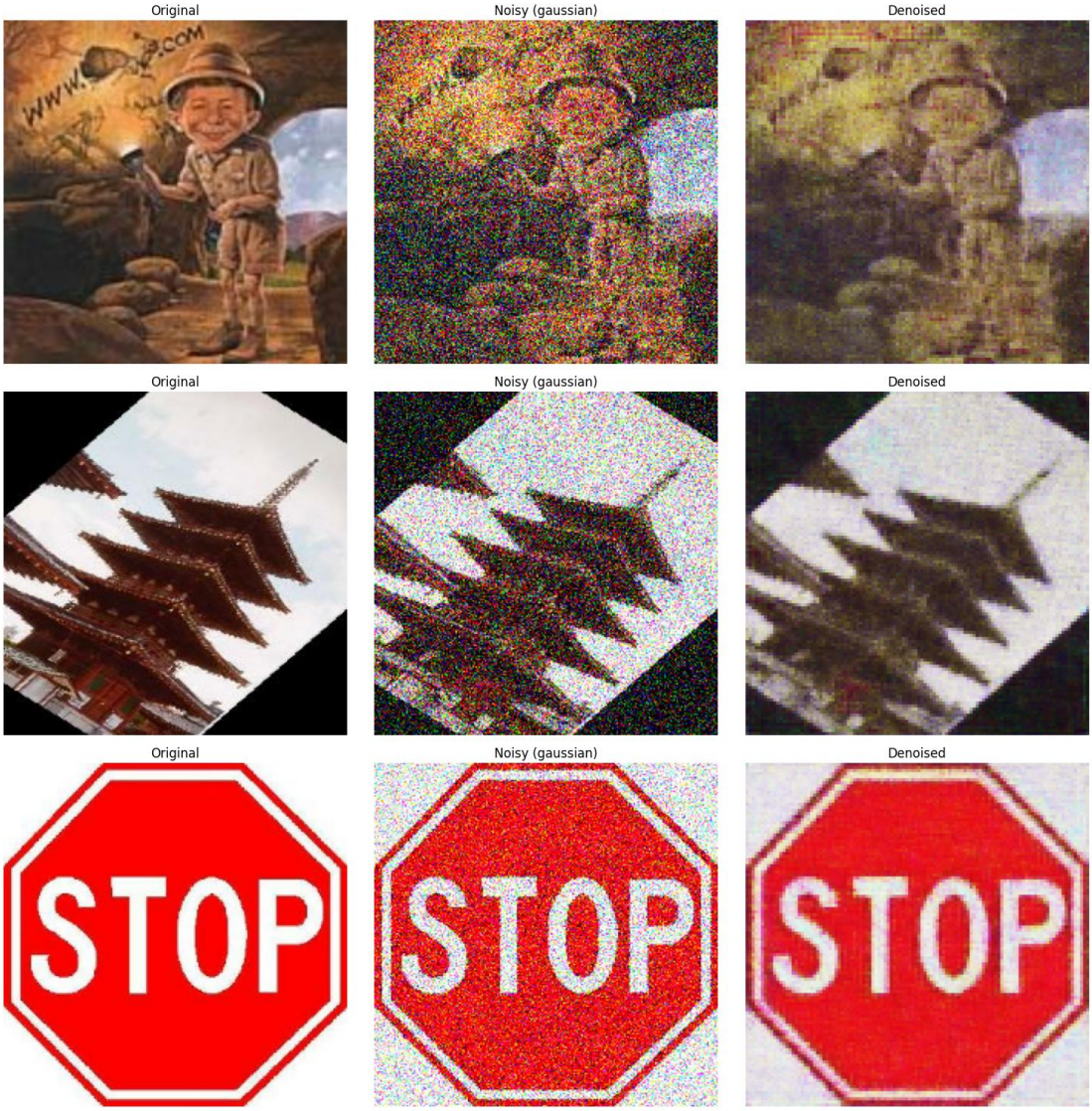

Original
Noisy (poisson)
Denoised
Original
Noisy (poisson)
Denoised
Original
Noisy (poisson)
Denoised

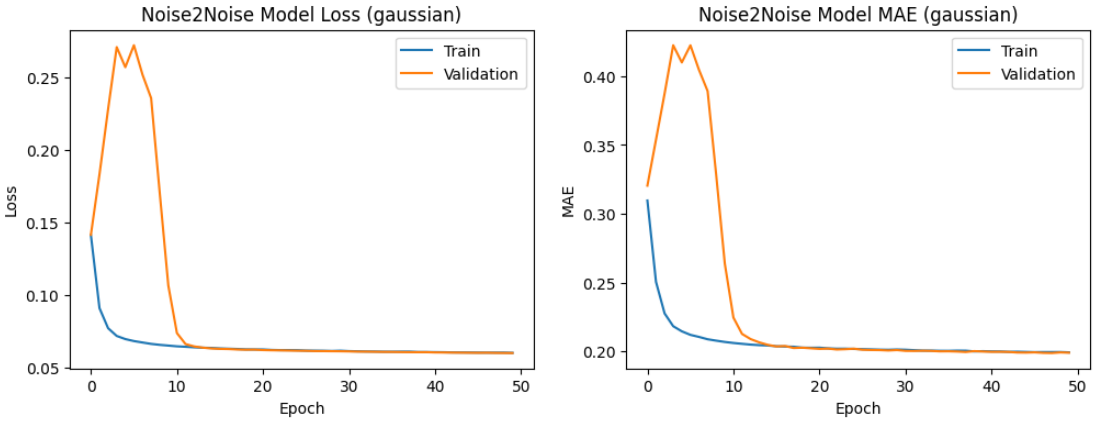

Noise2Noise Model Loss (poisson)
Train
Validation
Loss
Epoch
Noise2Noise Model MAE (poisson)
Train
Validation
MAE
Epoch

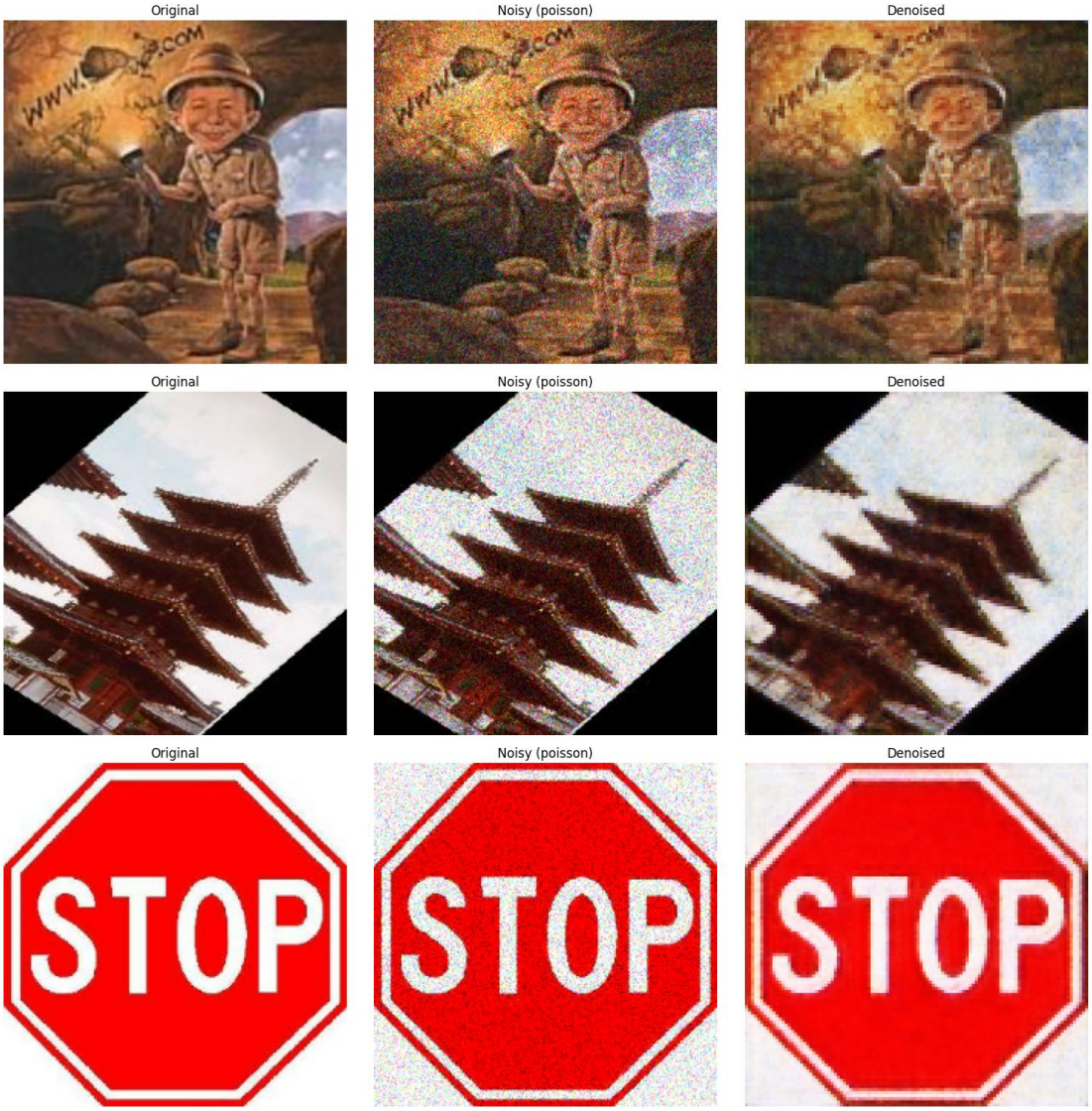

Original
Noisy (bernoulli)
Denoised
Original
Noisy (bernoulli)
Denoised
STOP
STOP
STOP
Original
Noisy (bernoulli)
Denoised

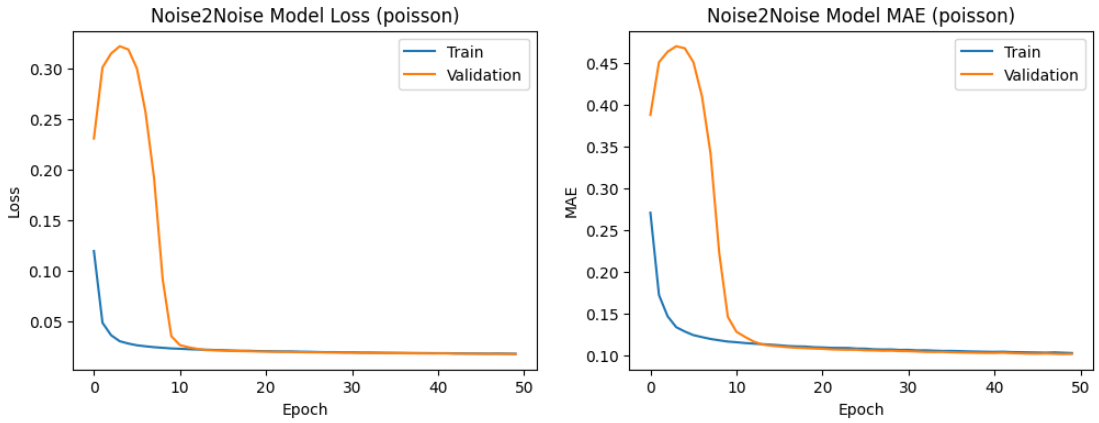

Noise2Noise Model Loss (bernoulli)
Train
Validation
Loss
Epoch
Noise2Noise Model MAE (bernoulli)
Train
Validation
MAE
Epoch

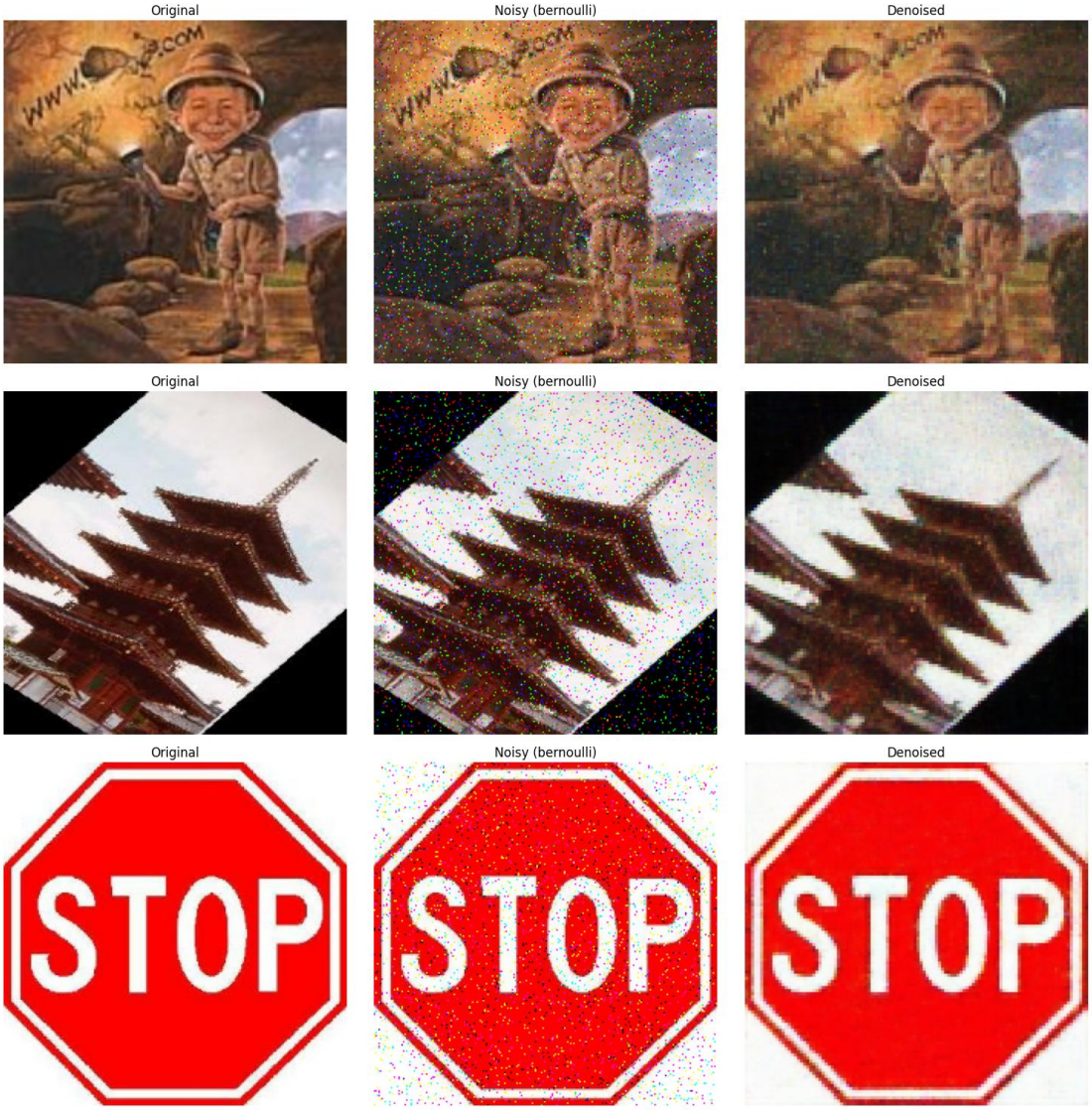
Resnet101:

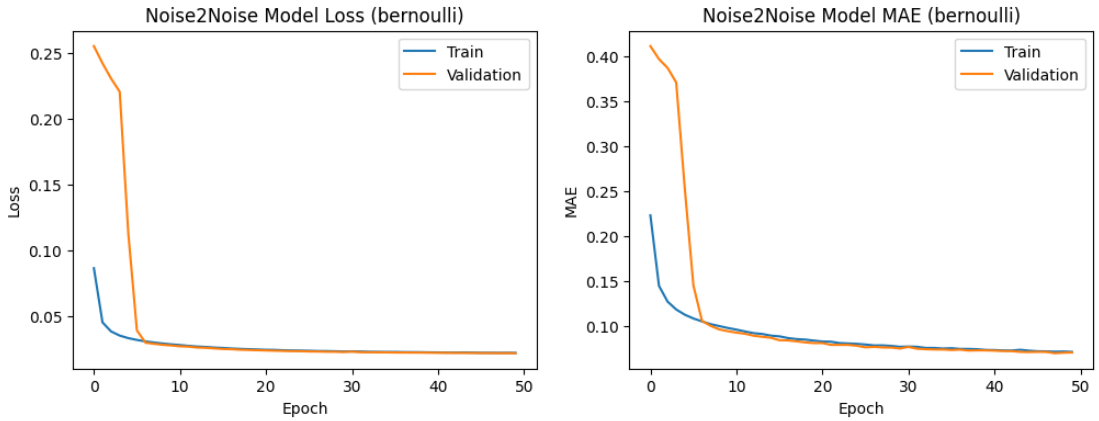

Original
Noisy (gaussian)
Denoised
Original
Noisy (gaussian)
Denoised
Original
Noisy (gaussian)
Denoised
STOP
STOP
STOP


Noise2Noise Model Loss (gaussian)
Train
Validation
Loss
Epoch
Noise2Noise Model MAE (gaussian)
Train
Validation
MAE
Epoch

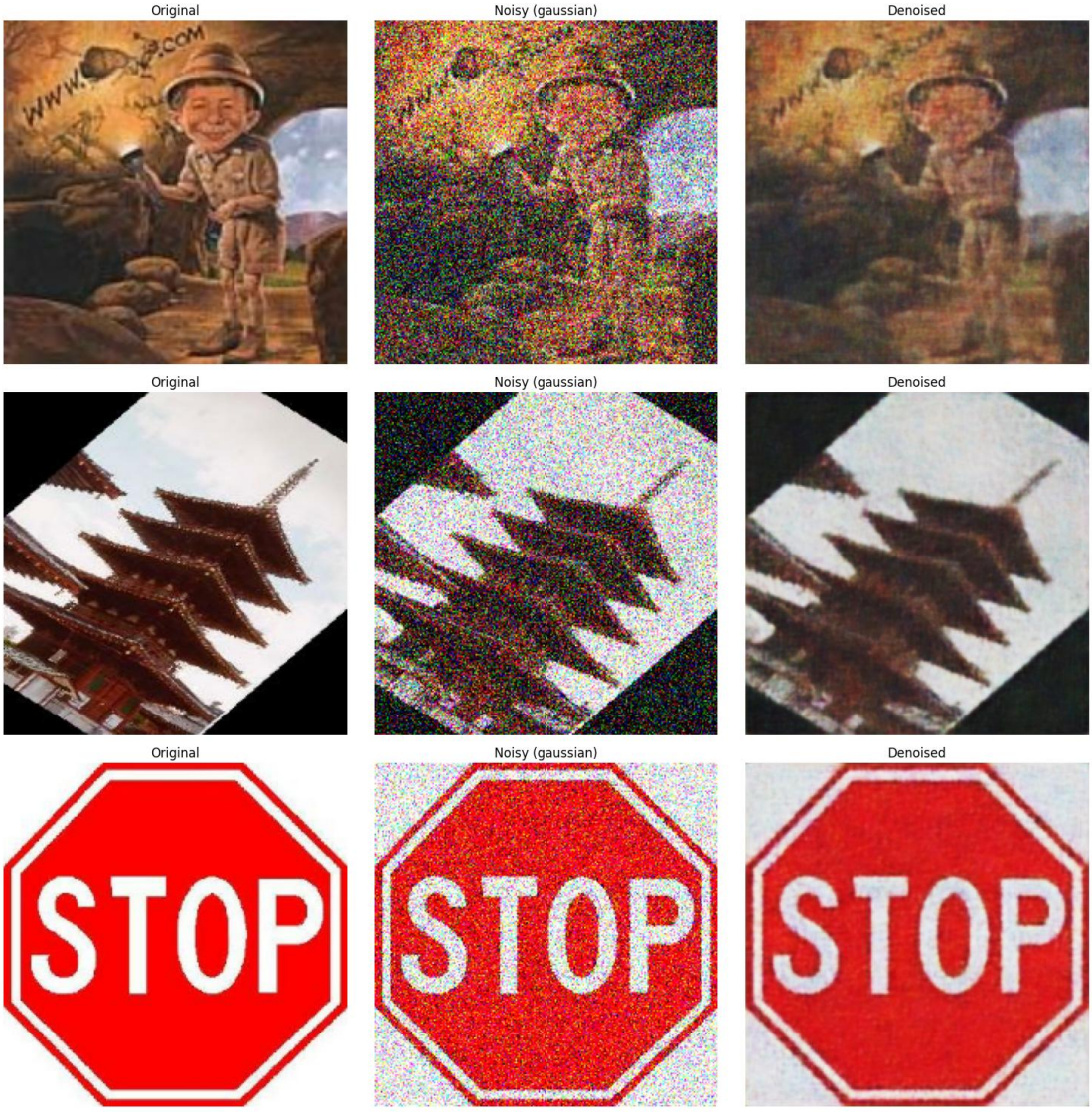

Original
Noisy (poisson)
Denoised
Original
Noisy (poisson)
Denoised
Original
Noisy (poisson)
Denoised
STOP
STOP
STOP

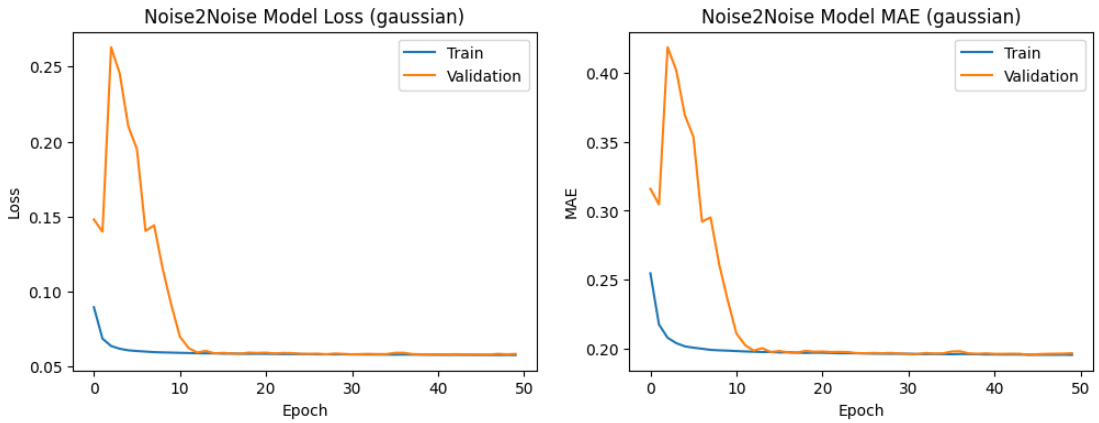

Noise2Noise Model Loss (poisson)
Train
Validation
Loss
Epoch
Noise2Noise Model MAE (poisson)
Train
Validation
MAE
Epoch

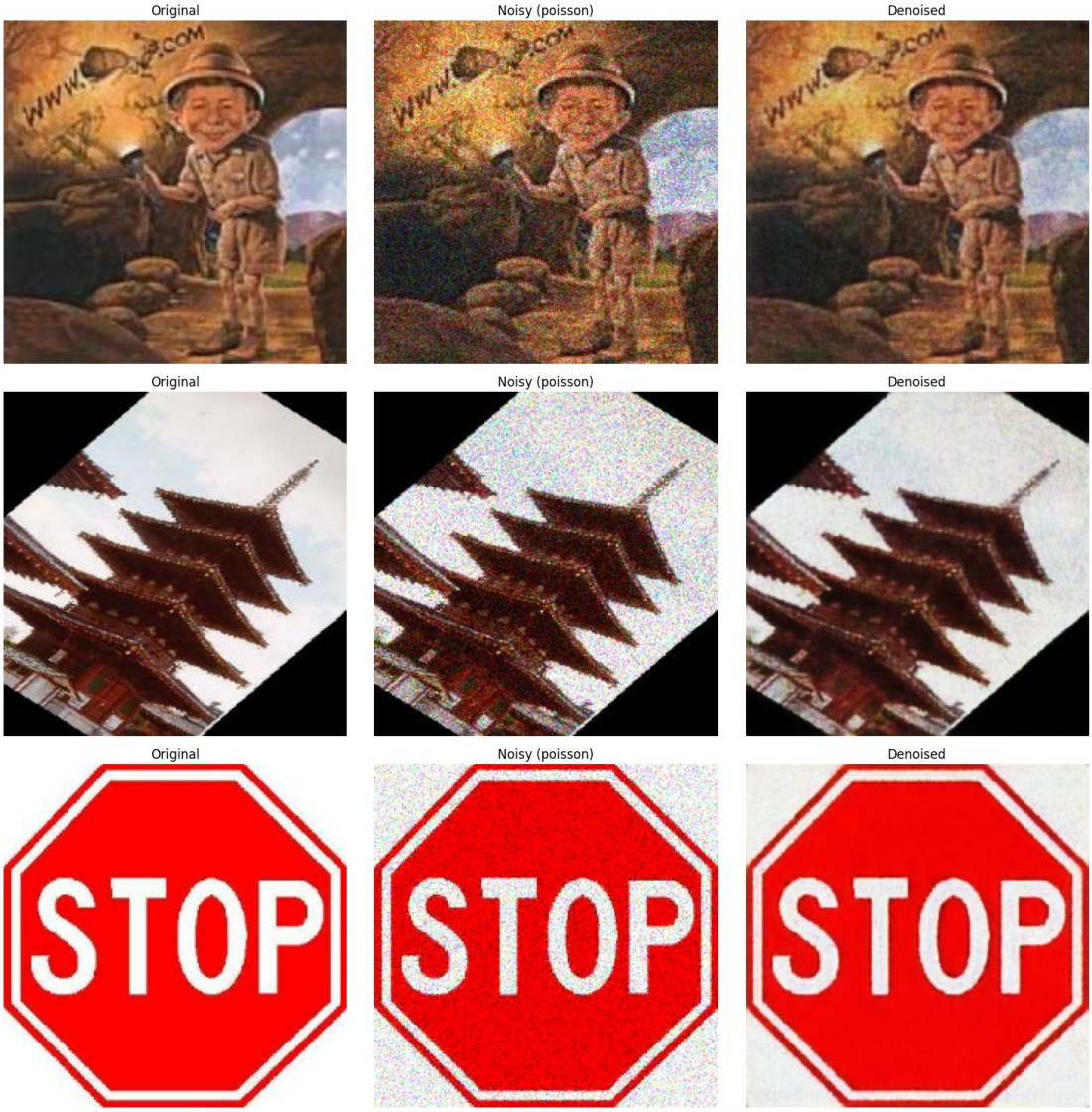

Original
Noisy (bernoulli)
Denoised
Original
Noisy (bernoulli)
Denoised
Original
Noisy (bernoulli)
Denoised
STOP
STOP
STOP

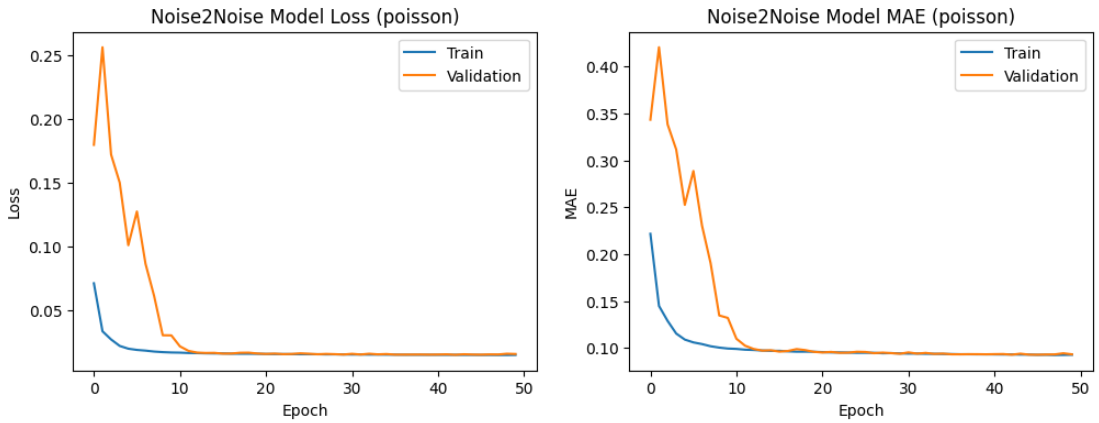

Noise2Noise Model Loss (bernoulli)
Train
Validation
Loss
Epoch
Noise2Noise Model MAE (bernoulli)
Train
Validation
MAE
Epoch

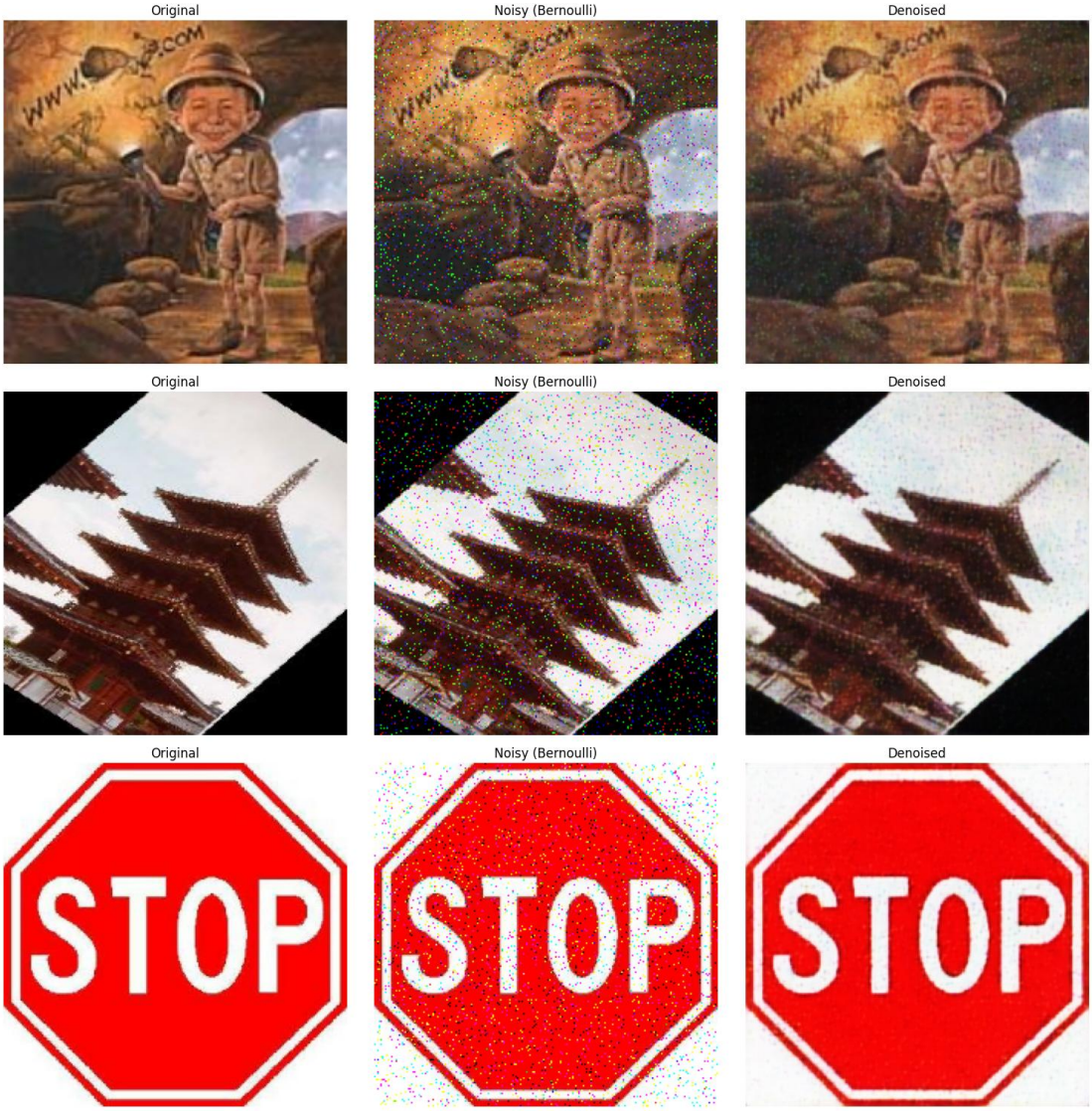
Resnet152:

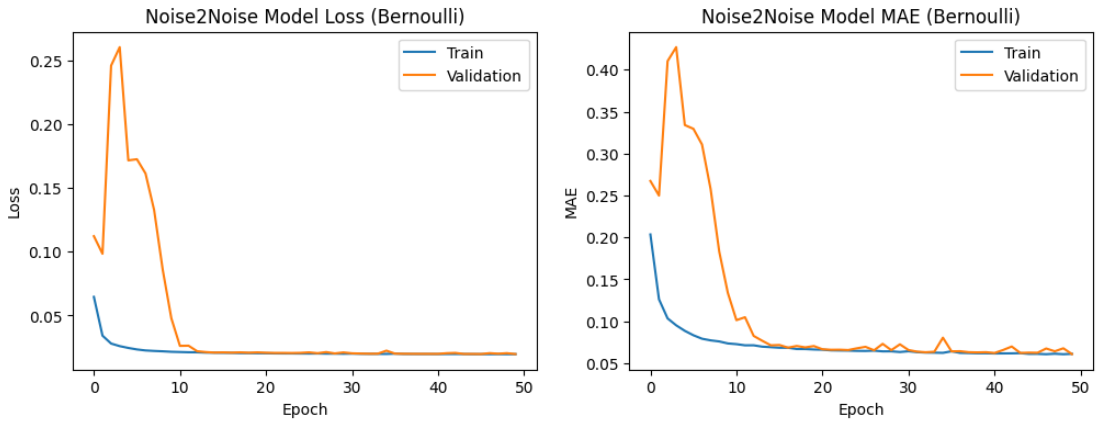

Original
Noisy (gaussian)
Denoised
Original
Noisy (gaussian)
Denoised
Original
Noisy (gaussian)
Denoised
STOP
STOP
STOP


Noise2Noise Model Loss (gaussian)
Train
Validation
Loss
Epoch
Noise2Noise Model MAE (gaussian)
Train
Validation
MAE
Epoch


Original
Noisy (poisson)
Denoised
Original
Noisy (poisson)
Denoised
Original
Noisy (poisson)
Denoised
STOP
STOP
STOP


Noise2Noise Model Loss (poisson)
Train
Validation
Loss
Epoch
Noise2Noise Model MAE (poisson)
Train
Validation
MAE
Epoch


Original
Noisy (Bernoulli)
Denoised
Original
Noisy (Bernoulli)
Denoised
Original
Noisy (Bernoulli)
Denoised
STOP
STOP
STOP


Noise2Noise Model Loss (Bernoulli)
Train
Validation
Loss
Epoch
Noise2Noise Model MAE (Bernoulli)
Train
Validation
MAE
Epoch

Loss Table:

	U-Net	ResNet-50	ResNet-101	ResNet-152
Gaussian	**0.0077**	0.0106	0.0109	0.0079
Poisson	**0.0019**	0.0061	0.0062	0.0040
Bernoulli	**0.0010**	0.0060	0.0046	0.0027

MAE Table:

	U-Net	ResNet-50	ResNet-101	ResNet-152
Gaussian	0.0722	0.0822	0.0830	**0.0715**
Poisson	**0.0325**	0.0564	0.0569	0.0485
Bernoulli	**0.0248**	0.0531	0.0463	0.0380

The results shows that Noise2Noise with U-Net works better than Noise2Noise with ResNet. From the tables above, it is clear that U-Net outperforms all versions of ResNet (ResNet-50, ResNet-101, and ResNet-152) for denoising under Gaussian, Poisson, and Bernoulli noises. U-Net achieves lower loss values and lower MAE in most cases, which means it produces cleaner, more accurate images. While ResNet-152 tends to do better than ResNet-50 and ResNet-101, it still cannot match U-Net's overall performance, especially when dealing with Poisson and Bernoulli noise. These results suggest that U-Net's encoder–decoder structure and skip connections help preserve important details that are often lost when using ResNet architectures.

8. Conclusions:

The main goal of our project was to find a way to enhance and improve Noise2Noise algorithm by replacing U-Net with ResNet. We studied ResNet and U-Net models, understood their differences to see if it is possible. We compared U-Net with different ResNet architectures to see if ResNet could outperform U-Net in the Noise2Noise denoising task. We tried ResNet-50 first, and when it didn't perform as well as we expected, we moved on to ResNet-101 and ResNet-152. Even though the deeper ResNets got closer in some cases, none of them could consistently match or exceed U-Net's results across all types of noise. It seems that U-Net's encoder–decoder structure and skip connections help preserve important spatial details better than the ResNet, which are primarily designed for classification. Therefore, based on our experiments, U-Net remains the stronger choice for Noise2Noise denoising.

9. References:

[1] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang 1
Beyond a Gaussian Denoiser: Residual Learning of
Deep CNN for Image Denoising
[2] Jindong Jiang, Lunan Zheng, Fei Luo, and Zhijun Zhang RedNet: Residual Encoder-Decoder Network for
indoor RGB-D Semantic Segmentation
[3] Marc Lebrun An Analysis and Implementation of the BM3D Image
Denoising Method
[4] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. Noise2Noise: Learning Image Restoration without Clean Data.
[5] Christopher Thomas BSc Hons. MIAP U-Nets with ResNet Encoders and cross connections
[6] Recognition and Mapping of Landslide Using a Fully Convolutional DenseNet and Influencing Factors Xiao Gao, Tao Chen , Senior Member, IEEE, Ruiqing Niu, and Antonio Plaza , Fellow, IEEE
[7] Schematic sketch of the U-net architecture used, going sequentially... | Download Scientific Diagram (researchgate.net)
[8] Peak signal-to-noise ratio - Wikipedia
[9] Deep Residual Learning for Image Recognition. Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun.
[10] META-OPTIMIZATION OF DEEP CNN FOR IMAGE DENOISING USING LSTM. Basit O. Alawode, Motaz Alfarraj. King Fahd University of Petroleum and Minerals, Electrical Engineering Department, Dhahran, Saudi Arabia.
[11] Mean squared error - Wikipedia
[12] Loss function - Wikipedia
[13] Mean absolute error - Wikipedia