**Software Engineering Department**
**Braude College**

# Image denoising using Noise2Noise with Resnet-50
**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**

## Table of Contents

**Abstract** ………………………………………………………………………. 3

1. **Introduction and Background** …………………………………….……… 3

2. **Related Work** ...…………………………………………………………… 4

   2.1 DnCNN ……………………………………………………………..….... 4

   2.2 RedNet (Residual Encoder-Decoder) ……………………………………… 5

   2.3 BM3D ……………………………………………………………………… 5

   2.4 U-Net ……………………………………………………………………. 6

   2.5 ResNet50 ………………………………………………………………... 9

   2.6 Loss Functions …………………………………………………………. 10

      2.6.1 L0 Loss - (Zero-Norm Loss) …………………………………………... 10

      2.6.2 L1 Loss - (Mean Absolute Error - MAE) ……………………………….. 10

      2.6.3 L2 Loss - (Mean Squared Error – MSE) ………………………………... 11

   2.7 Unet And Resnet50 comparison …………………………………………. 11

3. **Expected Achievements** ………………………………………….…... 12

4. **Process** ……………………………………………………………………… 12

   4.1 Pre-processing ……………………………………………………………… 12

5. **Model Diagrams** ……………………………………………………...…. 13

   5.1. Training, Validation and Testing processes …………...…………….………… 13

6. **Testing Plan** ……………………………………………………………... 14

7. **References** ……………………………………………………………...….. 16

**Abstract**

This project investigates the denoising of images using the Noise2Noise algorithm in combination with ResNet-50 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-50, 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-50 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

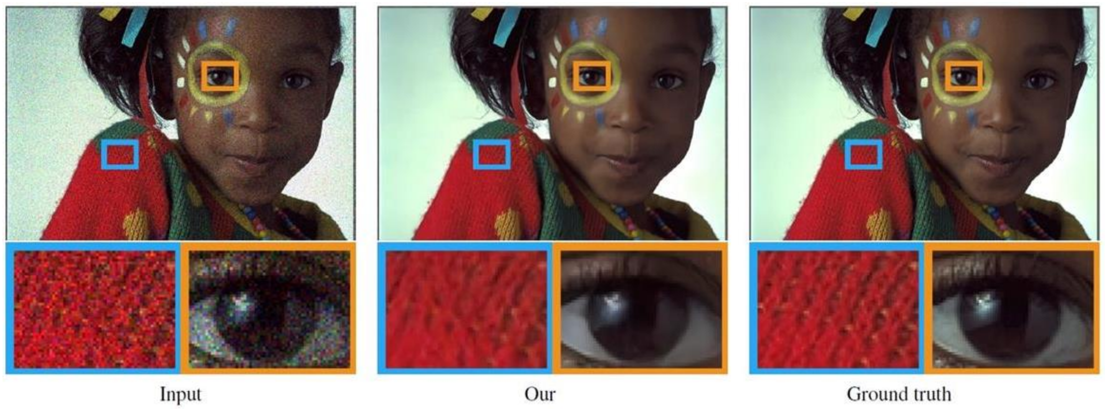
Image restoration is the process of removing distortions from images. Noise is a common form of distortion that can significantly degrade image quality and can make tasks like object recognition and image analysis being complicated or even impossible.

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.



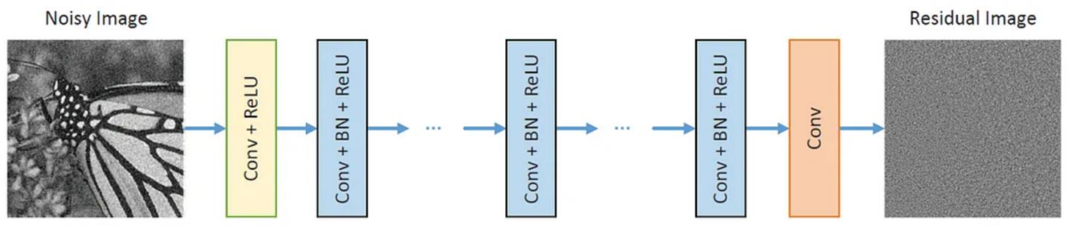
[4] Image restoration by denoising it using Noise2Noise algorithm.

## 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:

**<u>Block matching</u>**: Similar blocks in the noisy image are identified and grouped together.

**<u>Collaborative filtering</u>**: 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.

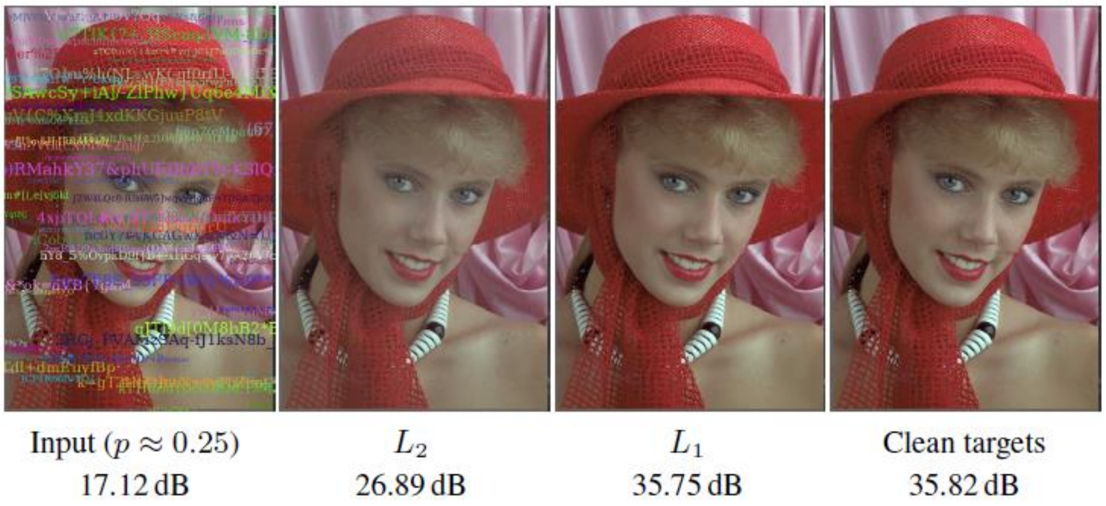
The U-Net architecture used for image denoising, as described in the article "Noise2Noise: Learning Image Restoration without Clean Data," consists of an encoder and a decoder. The encoder compresses the input image into a lower-dimensional representation by progressively extracting essential features 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 $F_\theta$ 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.

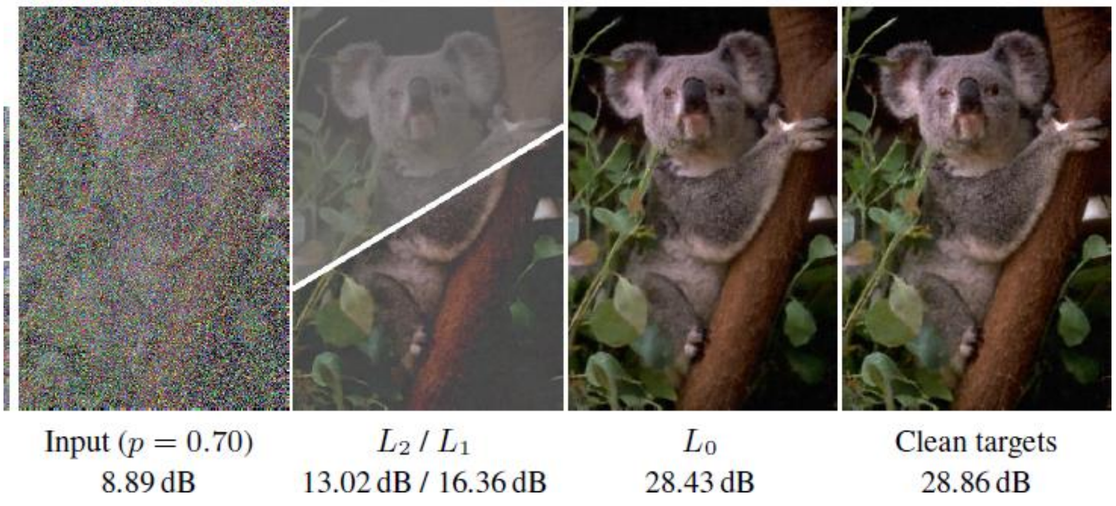
The training process utilizes the L2 loss function, which measures the average squared difference between the estimated values and the actual value. The L2 loss function is used because it is effective at penalizing larger errors more than smaller ones, making it suitable for image restoration tasks. 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



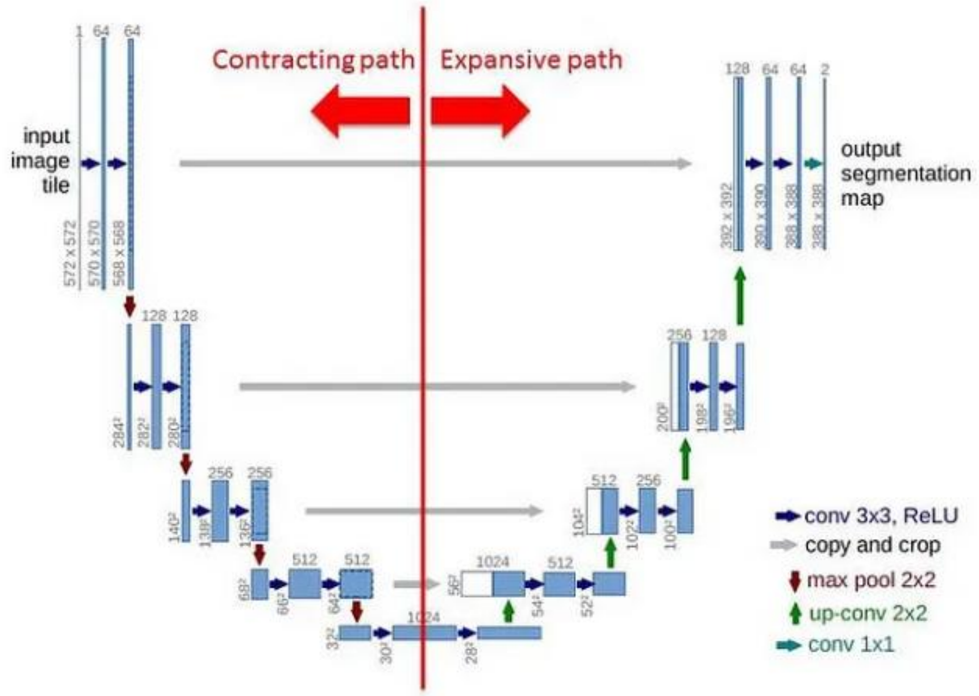
[4] Denoising example using U-Net, using only L1 and L2 functions.

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

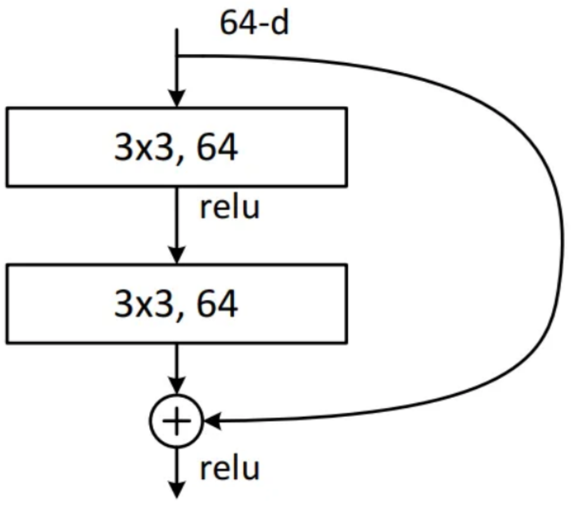
**Pooling layers (POOL):** Reduce spatial dimensions to enhance computational efficiency in the encoder pathway.
**Upsampling layers (UPSAMPLE):** Increase spatial dimensions in the decoder pathway to reconstruct the output.
**Concatenation (CONCAT):** At each decoding step, the network concatenates 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.

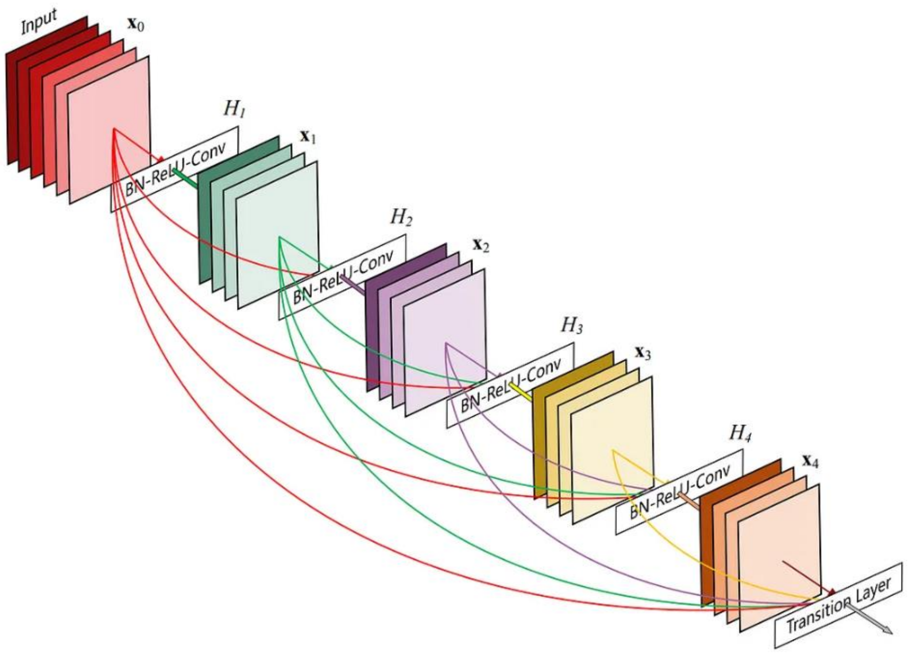


[9] A residual block scheme

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 Loss Functions

### 2.6.1 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.

### 2.6.2 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 $L(z, y) = |z - y|$

### 2.6.3 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 $L_2 = L(z, y) = (z - y)^2$ .

## 2.7 U-Net and ResNet50 comparison:

**Purpose:**
- U-Net: Designed for image segmentation, producing pixel-wise predictions.
- ResNet50: 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.
- ResNet50: 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.
- ResNet50: Used to bypass layers and improve gradient flow in deep networks.

**Output:**
- U-Net: Outputs a segmentation mask (same size as input image).
- ResNet50: Outputs class probabilities for classification.

For Noise2Noise project, U-Net is suited for pixel-wise tasks, while ResNet50 may need modification to handle denoising or similar tasks effectively.

## 3. Expected Achievements

The project aims to explore a new Noise2Noise architecture by using ResNet-50 instead of the traditional U-Net. We believe that ResNet50 might perform better because of its ability to handle complex image features.
We plan to test and evaluate this approach on various image datasets to see if it really does perform better. By comparing the results of U-Net and ResNet50 under the same conditions, we hope to show if ResNet50 provides any improvement in performance.

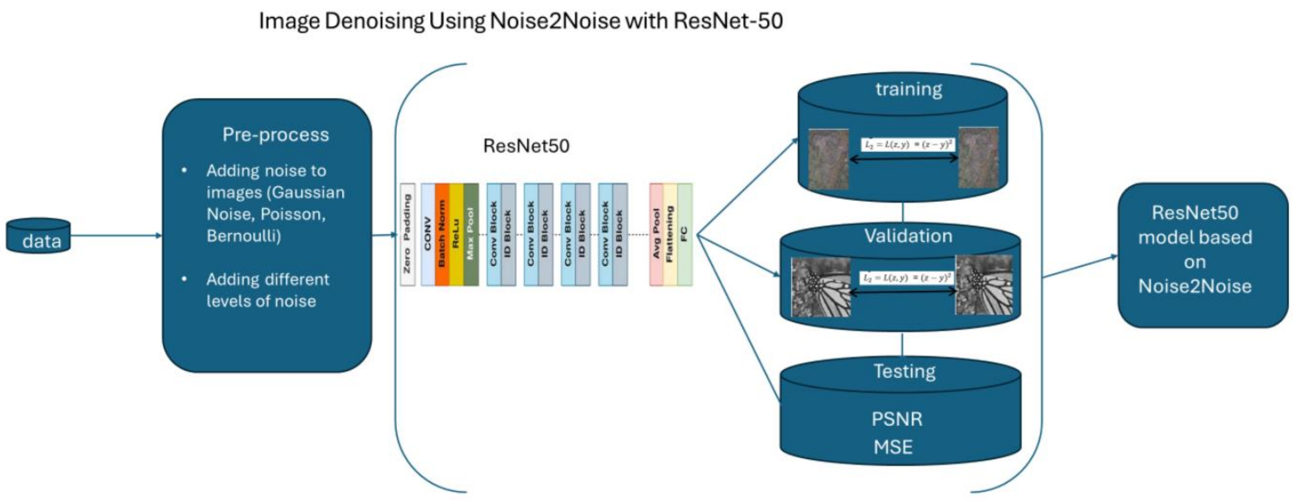
## 4. Process

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 ResNet50 instead of U-NET. By using Noise2Noise model based on ResNet50, 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.

### 4.1. Pre-processing

Since the aim is to train ResNet50 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 ResNet50 model in the Noise2Noise framework. This approach ultimately aims to enable effective noise removal from images through the model's learning process.

## 5. Model Diagram



This diagram shows the steps in using ResNet50 for the Noise2Noise algorithm. The approach begins with adding various types of noise to the data, then using the modified ResNet50 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 ResNet50's residual learning to effectively handle the image denoising.

### 5.1. Training, Validation and Testing processes

**Training:**

Process noisy image pairs, update model parameters based on L2 loss function.

**Validation:**

Assess model performance on unseen data without updating weights.

**Testing:**

Evaluate denoising quality using PSNR and MSE on independent dataset.

## 6. Testing Plan

There are many methods available to test and evaluate models, including metrics used to examine model efficiency: Peak Signal-to-Noise Ratio (PSNR) and Mean Squared Error (MSE). For example, PSNR measures the peak error between the original and the denoised image [8]. It is used to quantify the difference between the original and denoised image, with higher values indicating better performance.

In this project, we will compare the performance of the Noise2Noise model using ResNet50 architecture against the traditional U-Net architecture to determine which model performs better under various conditions.
The images from the testing dataset are compared to the denoised images generated by the model.

| Test number | Test subject | Expected Results |
|---|---|---|
| 1 | Load Noisy Image | The noisy image will load onto the screen. |
| 2 | Press Denoise | The algorithm will denoise the image, displaying the clean version on the screen. |
| 3 | Upload Invalid Image Format (Invalid input) | A message will appear asking the user to upload a valid image file. |
| 4 | No Image Loaded and Denoise button has pressed | An error message will appear on the user' screen saying that no image is loaded into the system. |
| 5 | Upload Image while denoising is in progress | An error message will appear saying that denoising is in progress and a new image should be uploaded only when finished. |
| 6 | Exiting Program Mid-Denoise | The program will ask the user if they want to exit, with a warning about losing the current progress. |
| 7 | Compare denoised Image with original image | The system will calculate and display PSNR and SSIM metrics to compare the denoised image with the original image. |

| | | |
|---|---|---|
| 8 | Handle Low-Resolution Images | The algorithm will denoise low-resolution images. |
| 9 | Handle High-Resolution Images | The algorithm will successfully denoise high-resolution images. |

## 7. 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