BRAUDE
College of Engineering, karmiel

Capstone Project Phase A

Image Enhancement with Super-Resolution and Controlled Noise

24-1-R-12

Students:

Liza Shvachka– 206236176 Liza.shvachka@e.braude.ac.il

Avishai Hershkovitz– 209460443 Avishai.hershkovitz@e.braude.ac.il

Supervisor: Dr. Renata Avros

Advisor: Prof. Zeev Volkovich

Table of Contents

Abstract

This paper explores signal reconstruction using machine learning, with a specific focus on image enhancement tasks. The first approach uses a single model for various image reconstruction tasks, including denoising and denoising synthetic images, all based only on corrupted data. This demonstrates the effectiveness of learning recovery without explicit image or corruption models. The second approach addresses image super-resolution (SR), a challenging task due to its inherent ambiguity. Previous deep learning methods struggle with non-stationary degradations, limiting their effectiveness in some applications. To deal with this, we propose a basic U-net and a robust (RU-net) architecture that learns the relationship between degraded low-resolution images and high-resolution images. The main idea of the project is to test the most efficient approach for image enhancement tasks.

Our project files and source code will be available at: GitHub.

1. Introduction

Image restoration and super resolution, techniques for converting low quality images to high quality. Dealing with the challenge of restoring images based only on corrupted images. In the past, traditional statistical models such as linear regression and wavelet transformation were used. But these models require a deep understanding of the underlying processes and statistical properties. In the real world the images are varied including, different levels of exposure (short and long exposure shots), analysis of aerial photographs and when there is a latent/hidden object in the image. Furthermore, image super-resolution (SR) has made significant strides with the advent of deep learning techniques. Adverbial network (GAN) based methods have shown promise in generating realistic results. But they rely on a Bicubic down-sampling model that does not fully represent real-world scenarios, which limits their effectiveness, and they often face instability in training.

In this project we focus on the process of turning low quality images into high quality by learning corrupted images where the machine learns the way of corruption indirectly from the training data. The methodology involves training the U-net model several times using a data set that includes pairs of corrupted input. By analyzing the nature of the corruption, the model learns to efficiently restore the images, thereby improving their quality. To address the challenge of super-resolution, a robust U-net (RU-net) architecture is proposed. This approach includes connections between distant layers within the network and a degradation model that varies between different regions of the image.

The following articles were used: Noise2Noise: Learning Image Restoration without Clean Data [4]. RUNet: A Robust UNet Architecture for Image Super-Resolution [3].

2. Related Work

2.1 Red-Net30

Red-Net is a type of neural network that consists of 30 layers arranged in a sequence. It's designed with special connections between layers that help it work better. These connections, called residuals, make it easier for the network to learn and give better results.

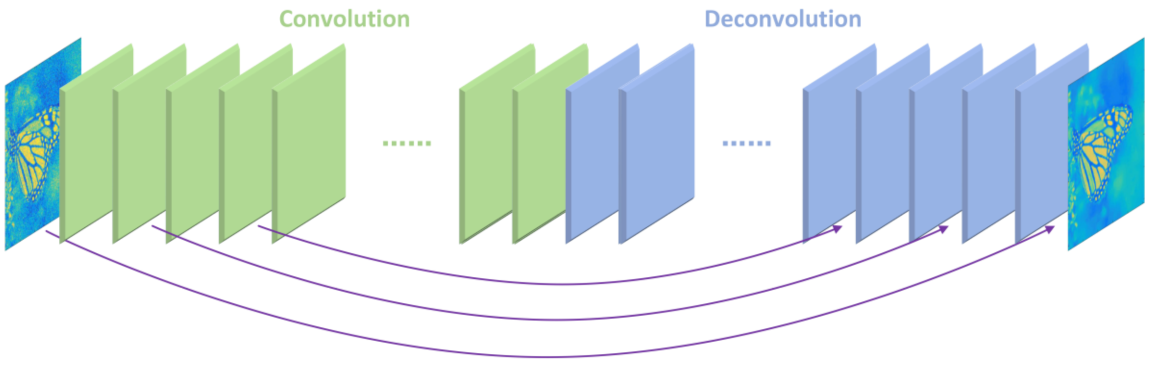


Figure 1. *Red-Net30 Architecture [5]*

The network comprises two layers: an encoder and a decoder, each with symmetric convolution. These layers map convolutional features. A skip connection exists between them, connecting the features from the encoder layer to their mirrored counterparts in the decoder layer.

2.2 BM3D

A denoise algorithm that operates on blocks of pixels in an image to identify similar patterns in the image and then applies a filtering approach to reduce noise while preserving the underlying structures in the image.

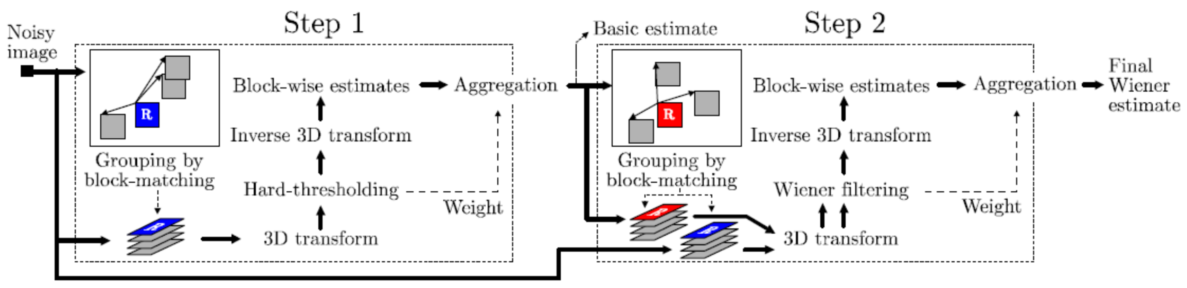
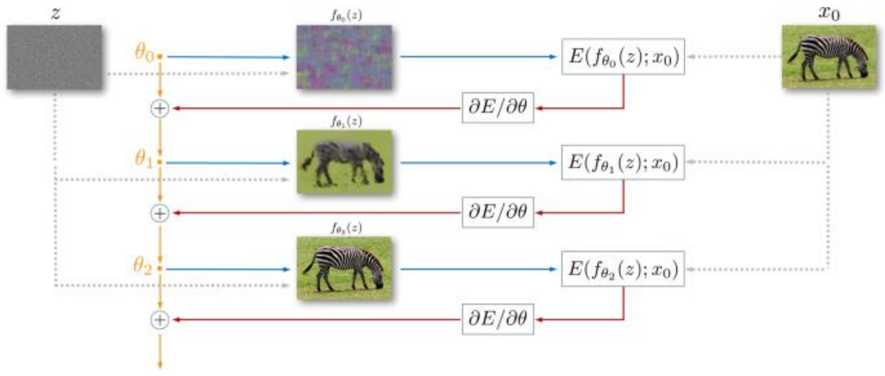


Figure 2. *Flowchart of image denoising BM3D algorithms [1]*

The algorithm divides images into overlapping blocks, identifies similar blocks and groups them into clusters. Collaborative filtering combines information from similar blocks to blur them and post-processing steps further improve image quality. Finally, aggregated blocks reconstruct the denoised image with proper weighting to prevent artifacts.

2.3 DIP- Deep Image Prior

Algorithm that reconstructs a noisy image without the need for previous training data. It works by inputting a noisy image into a neural network (CNN). However, unlike traditional methods learned from any dataset, the weights of the network in this algorithm are not learned from any specific dataset.



z
$f_{\theta_0}(z)$
x_0
θ_0
$E(f_{\theta_0}(z); x_0)$
$\partial E/\partial\theta$
$f_{\theta_1}(z)$
θ_1
$E(f_{\theta_1}(z); x_0)$
$\partial E/\partial\theta$
$f_{\theta_2}(z)$
θ_2
$E(f_{\theta_2}(z); x_0)$
$\partial E/\partial\theta$

Figure 3. *Flowchart iterative of image denoising DIP algorithm [9]*

This method operates by iteratively adjusting the input noisy image based on how much it "thinks" it differs from what it should be (using by loss functions optimize iteration) and ultimately produces a clearer output image.

2.4 Anscombe

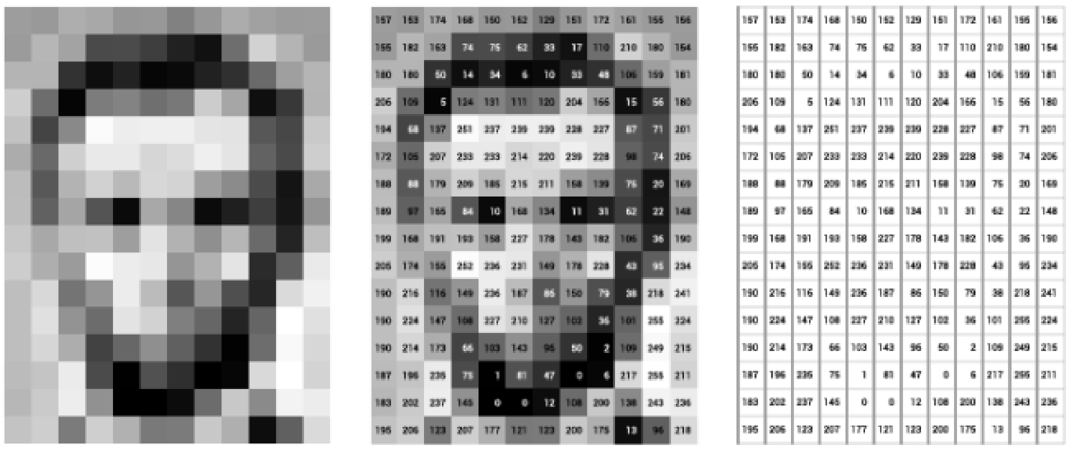
A mathematical transformation used in image processing to convert data that follows Poisson noise (data from images with low-light conditions) into data that is approximately Gaussian. This transformation meaning is direct dealing with Poisson noise can be challenging, so traditional noise reduction techniques may not work optimally. Therefore, this is a way to make the noise more suitable for reconstruction for traditional denoising techniques that can handle Gaussian noise.

Additionally, after applying the transformation, we need to perform an inverse transformation to convert the changed image back to its original form. [6]

3. Mathematical Background

3.1 Image representation

Image representation means turning a picture into a format that computers can understand. Pictures are made up of tiny dots called pixels, each with its own color or brightness. These pixels are arranged in a grid that shows the picture's size and colors (such as RGB for color images). In computer an image can be defined by a two-dimensional array specifically arranged in rows and columns, when x, y representing the pixel value ranging from 0 (black) to 255 (white). The aim is to pick out important details from the picture so the computer can do things like recognizing objects, sorting images, or dividing them into parts.



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Figure 4. *Image representation in computer [11]*

- **Grayscale\ Monochrome representation-** Images are represented using a single channel where each pixel contains a grayscale value ranging from 0 (black) to 255 (white). Each pixel represents the brightness or intensity of the image at that point.
- **Color representation** - Images are represented using the RGB (Red, Green, Blue) format. Each pixel is represented by three color channels (R, G, and B), with each channel containing an intensity value ranging from 0 to 255.

3.2 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a deep learning algorithm that focuses on analyzing visual data, such as images. CNN is a deep neural network which is like the structure of neurons in the brain. The network is particularly effective for tasks such as image segmentation and classification.

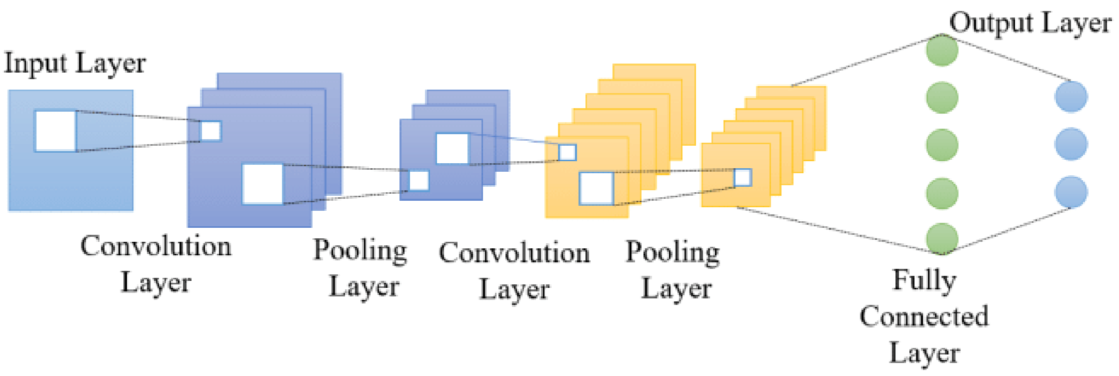


Figure 5. *Convolution Neural network [2]*

CNN architecture consists of five parts:

1. Input- the representation of the image according to its pixel values, usually divided into RGB (Red, Blue, and Green).

2. Convolution layer- the central layer whose function is to extract the features maps of the image. In each iteration, initialized filter of size $h \times w \times d$ (high, width, depth) which representing the weights. Convolve the filter on the image when it passes along the entire image according to its size and computing dot products, the connection of the pixels into one neuron. After this process, an activation function is activated on each neuron that enables the calculation to be represented visually and in a more understandable way for the user.
   The convolution layer uses hyper-parameters to perform the process:
   a. Stride – A parameter that determines the size of the step between each step that the filter passes over the image. In other words, the number of displacements of the filter on the image throughout its length and width.
   b. Padding- A parameter that defines the number of additional pixels, usually starting with zero, at the borders of the image. This method helps to maintain the spatial dimensions, adjusting the size of the filter on the image and preserves the features that exist at the edges of the image.
   c. Batch size- A parameter that defines the number of samples that will pass through the network in each forward/backward iteration. This affects the stability and speed of the training.
   d. Activation Function- Activation functions - a function that is applied to each neuron in the network and turns it into a non-linear processing unit. By introducing non-linearity, these functions empower the network to grasp and depict intricate relationships among data pixels, resulting in enhanced feature maps.
3. Pooling layer- its function is to finish extracting the feature maps by reducing the dimensions and allows to manage them in a better way.
4. Full connection layer- a layer that is designed to combine and normalize the features calculated from the previous layers. Contains neurons that connect to the entire input volume.
5. Output layer- a layer that includes neurons arranged in the form of a grid, where the number of neurons is determined according to the required conditions. Each neuron represents the probability of the input pixel belonging to a certain category.

3.3 U-net architecture

U-net is a convolutional network. Network work in an Encoder-Decoder way:

1. Encoder part- Shrinks the image, capturing high-level features like shapes and textures.
2. Decoder part- Expands the image back to original size, incorporating precise location details from the encoder.

Another component of the model consists of Skip Connections. Concatenation, where feature maps from the encoder layer are combined with those from the decoder layer. This is crucial for accurate segmentation.

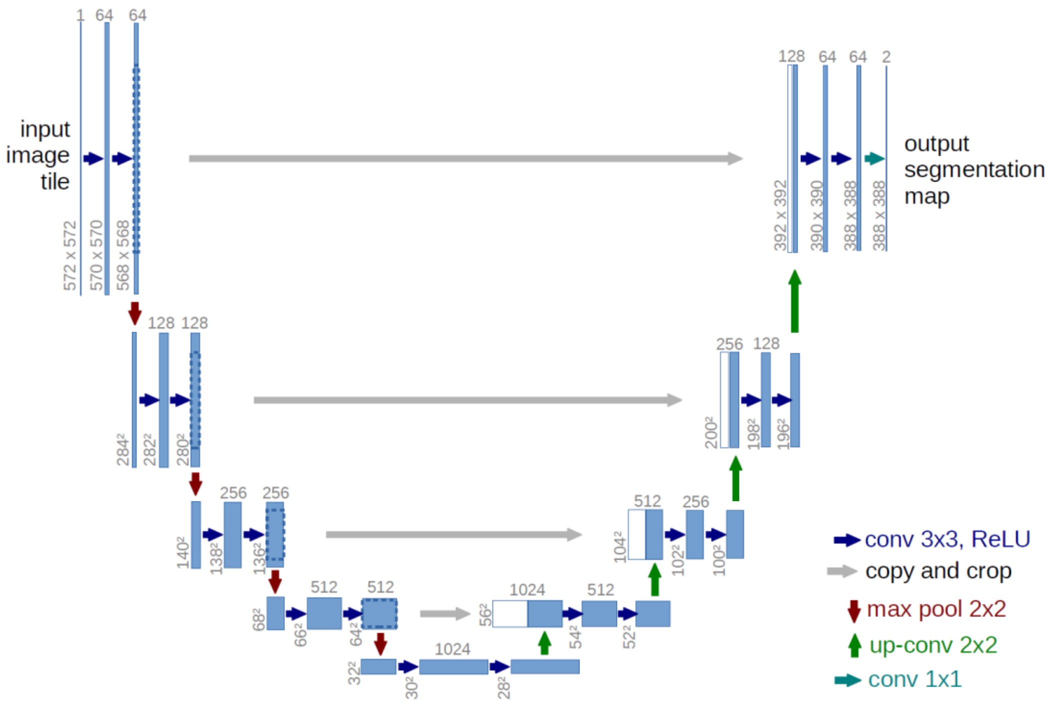


Figure 6. Architecture of U-net model [7]

The model's structure, resembling a U shape, and each layer acts like the typical contracting path in a CNN, the U-net block (in section 3.3.1).

Let's us break Down each part of the model:

1. Encoder part consists of multiple convolutional layer block with max pooling between each of them. These layers are responsible for capturing high-level features and reducing the spatial dimensions of the input image through down-sampling (in section 3.3.2) operations.
2. Decoder part consists of multiple convolutional layer block with up-sampling (in section 3.3.2) layer between each of them. Up-sampling layers are used to increase the spatial dimensions of the feature maps, effectively expanding them.
3. Skip connections (connecting the previous layers) allow the network to recover spatial details lost during down-sampling by directly concatenating feature maps from the contraction path to the corresponding layers in the expanding path.
4. The final layer usually consists of a convolutional layer 1x1 to reduce the number of channels that generated over the network. So, in the end we have one picture representation with the same dimension as the input image. This is followed by an activation function for semantic segmentation tasks.

This model is trained using loss function, MSE (in section 3.4).

3.3.1 U-net block

The U-net block use convolutions like a standard CNN block. Unlike just applying a filter, the U-net block performs multiple convolutions (often 2-3) followed by activation functions to learn more complex features. The output of the U-net convolutional block is a feature map, which captures specific aspects of the image based on the filters used.

This block consisting of two identical component, convolutional and activation function.

a. Activate on the image a filter 3x3xC_{out} when in encoding layer we increase the number of features maps and in decoding we decrease them. The filter weights are trained during the model.

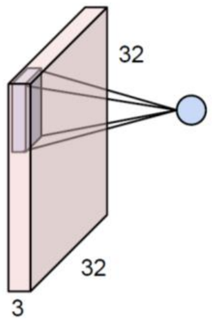


Figure 7. *Convolve filter [12].*

b. Activation function ReLU- It gives back 0 for negative inputs and keeps positive inputs unchanged, adding non-linearity without requiring much computation.

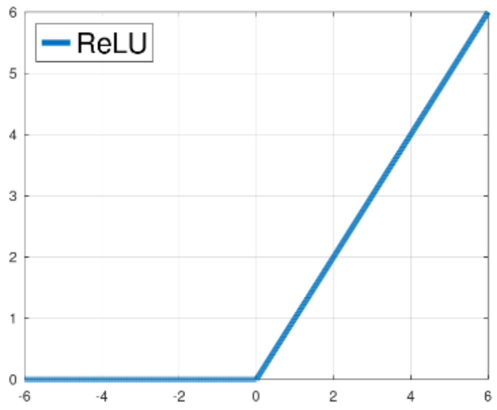


Figure 8. *ReLU activation function,* $R(x) = max\ (0, x)$ *[14].*

3.3.2 Up and Down sampling

Up-sampling and Down-sampling are two methods to either increase or decrease the size of an image. Up-sampling, known as pooling, halves the size of the image and down-sampling doubles the size of the image.

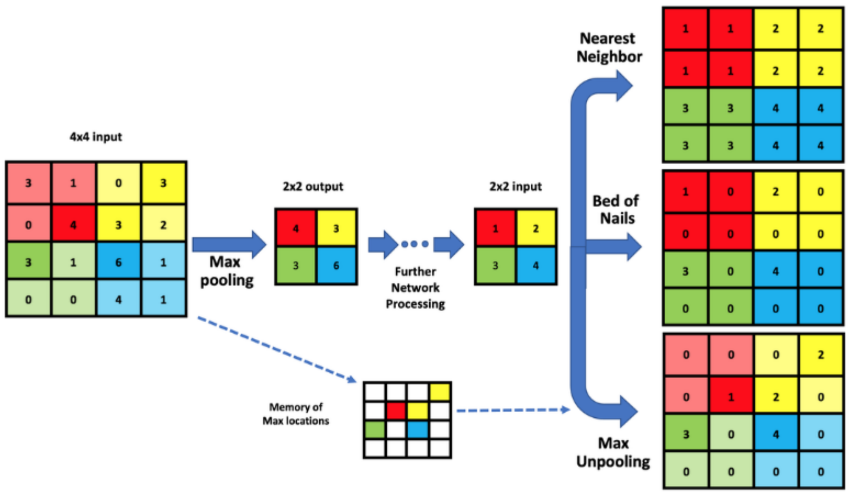


Figure 9. *Up and Down sampling [13]*

a. Down-Sampling- In generally the method used is max-pooling. When you take a filter of a certain size and calculate the maximum value from the image according to the size of the filter.
b. Up-Sampling - There are several methods to increase the image.
    1) Nearest Neighbors - when the value is duplicated to the neighbors according to the size of the filter.
    2) Bed of Nails - when placed in a certain index according to the size of the filter, consistently for the entire image.
    3) Max Up-pooling - depending on the index chosen to be taken to the maximum, we will place the value in the same index in the increase step.
    4) Pixel Shuffle- rearranges the pixels within each feature map. Each set of r x r elements in the channel (where r is the scaling factor) is reshaped into a single element in a new, enlarged height and width dimension.

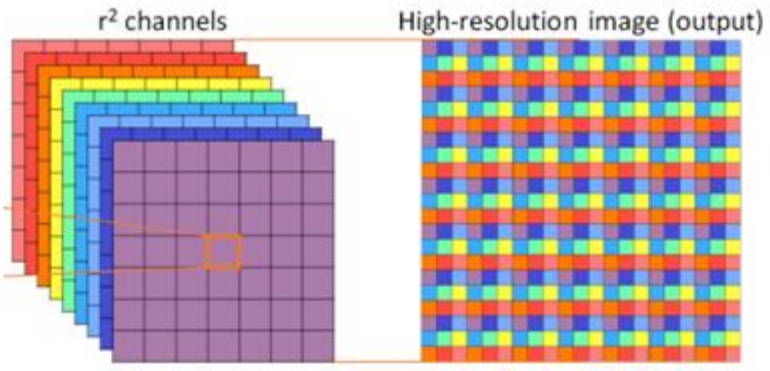


Figure 10. *Pixel Shuffle [8]*

3.4 Loss Function

Measures the difference between the model's predictions and the actual target values. It calculates the discrepancy between the model's predicted values (x) and the actual target values (y) in the training dataset. The formula defined by the following equation:

$$L\ (x, y)$$

The loss function varies depending on the type of task (for example, regression or classification) and the nature of the data.

- <u>L0 Function</u> – loss function, where $\varepsilon = 10^{-8}$, where γ is annealed linearly from 2 to 0 during training.

$$L_0(x, y) = (|x - y| + \varepsilon)^\gamma$$

- <u>L1 Function</u>- loss calculates the sum of the absolute differences between the predicted and actual values.

$$L_1(x, y) = |x - y|$$

- <u>L2 Function</u>- standard MSE loss, Mean Squared Error that measures the squared error between the predicted and actual values.

$$L_2\ (x, y) = (x - y)^2$$

- <u>L-HDR Function</u> - The L_{HDR} loss function is defined as the squared difference between the denoiser's output x and the clean target y, divided by the square of the denoiser's output plus a small constant 0.01 squared.

$$\mathrm{L}_{HDR}(\mathrm{x}, \mathrm{y}) = \frac{(\mathrm{x} - \mathrm{y})^2}{(\mathrm{x} + 0.01)^2}$$

- <u>L- Perceptual functions</u>- Compare high-level features between the target and output images, rather than pixel-level differences. The loss is calculated as the Euclidean distance between the feature maps in one or more layers.
Where $\phi_j(\hat{I})$ denoting the feature map generated at the j^{th} layer for a given input image I, I_{HR} a target high resolution image and $\mathrm{C}_j\mathrm{H}_j\mathrm{W}_j$ feature map size.

$$\mathrm{L}_j = \frac{1}{\mathrm{C}_j\mathrm{H}_j\mathrm{W}_j} \left\| \phi_j(\hat{I}) - \phi_j(I_{HR}) \right\|_2^2$$

3.5 Image quality assessment

Comparison in image processing evaluates the similarity or quality of images based on various criteria. The indices help in evaluating the quality of the image and guiding optimization processes. We'll explore two common benchmarks:

1. <u>Peak Signal-to-Noise Ratio, PSNR</u>- measures the quality of a reconstructed image by comparing it to a reference image, focusing on the signal-to-noise ratio. As the signal index increases, the quality of the processed image also increases, expressed in decibels (dB).

2. Structural Similarity Index Measure, SSIM - assesses how two images are structurally similar by considering factors such as brightness, contrast, and pixel arrangement, providing a score between 0 and 1.

3.6 Noise

Noise is essentially disturbances that obscure or interfere with the intended signal for certain data. Noise is created by environmental sources or electronic interference. In the field of image processing, noise manifests as unwanted pixel values, leading to loss of brightness and details in the image. In our model, we train by deal with various types of noise and to simulate reality so that it can handle a noisy image resulting from environmental features (brightness/light/shadows) and be able to cope with and address these noise disturbances.

3.6.1 Gaussian

Gaussian (normal) distributed random noise is applied to the gray values of original images. Gaussian noise is commonly used in modeling scenarios due to its mathematical properties and prevalence in real-world noise sources. The mathematical properties of this distribution allow averaging over a large number of pixels to help detect and cancel out the noise.

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

3.6.2 Poisson

Poisson noise is like the randomness that happens in photos, modeled by a mathematical function. This function helps understand the chance of different noise levels in pictures, especially when things like photons (particles of light) randomly show up. Dealing with this noise is a real challenge in fixing up images because it depends on the signal.

In the training of the model, we up by adjusting the noise level (λ) from 0 to 50. This lets the model handle various amounts of Poisson noise, making it super adaptable to fix and enhance photos, even in low-light situations. We're guiding our model to learn and deal with the randomness of Poisson noise during the process of making images look better.

$$P(x = k) = \frac{e^{-\lambda} * \lambda^k}{k!}$$

3.6.3 Multiplicative Bernoulli

Multiplicative Bernoulli noise, also known as binomial noise, generates a random mask. It constructs a random mask of parameter m that is "flag" (0/1). This mask is 1 for valid pixels and 0 for zeroed/missing pixels. The purpose is to address the issue of backpropagating gradients (basic algorithm used for training neural networks that enables optimization of weights based on error and loss) from missing pixels.

$$argmin_{\theta} \sum_{i} \left(\mathrm{m} \odot (\mathrm{f}_{\theta}(\hat{\mathrm{x}}_{i}) - \hat{\mathrm{y}}_{i})\right)^{2}$$

3.6.4 Monte Carlo

Rendering algorithms that simulate how light interacts with objects in the scene. The technique involves tracing random paths of light while interacting with surfaces and materials in the virtual environment [10]. Calculates the brightness or luminance of each pixel in the final rendered image based on the contributions of different light paths in the scene. The Monte Carlo used in path tracking is designed to ensure that the intensity of each pixel is an accurate representation of the expected radiation, despite the randomness inherent in the path sampling process. This means that the sampling noise, or variance in the sampled paths, averages to zero over time.

This technique has a significant characteristic because it ensures that the overall brightness or intensity of the processed image is not systematically biased by the random sampling process. Instead, the sampling noise tends to cancel out or average to zero over time, allowing the processed image to accurately represent the expected irradiance values for each pixel in the scene. Monte Carlo noise refers to random variations or artifacts that can appear in images processed with the Monte Carlo path tracking technique due to the inherent randomness of the method.

Dealing with the difficulty of Monte Carlo denoising is by using auxiliary information during the denoising process. The denoiser is provided not only with per-pixel luminance values (brightness), but also with additional information such as the average albedo (surface reflectivity or texture color) and the normal vector (surface orientation) of the visible surfaces at each pixel. Therefore, the denoiser can better understand the basic characteristics of the scene and make more informed decisions when removing noises.

3.6.5 Text Removal

Noise composed of several random strings in random places on the image, as well as random font size and color, but the font and direction remain constant. Removing random text overlays corresponds to seeking the median pixel color.

3.6.6 Random-Valued Impulse

Replaces some pixels with noise and retains the colors. This noise retains the color of each pixel in the image with probability p and otherwise replaces pixels with random pixel values between $[0-1]^3$.

4. Expected Achievements

In this project, the goal is to achieve several outcomes that will contribute to improving super-resolution techniques and transforming corrupted images into clean ones without relying on explicit images. This can be achieved by increasing image resolution and/or handling image corruption. To increase resolution, we upscale low-resolution images by duplicating and expanding them, followed by model training to generate high-resolution images. For the case of image corruption, we will use different models to create noise in the image. This process emphasizes the random nature of the corruption, which tends to average out over many pixels, resulting in cleaner images. The aim is to evaluate and determine which model yields the most effective image reconstruction results.

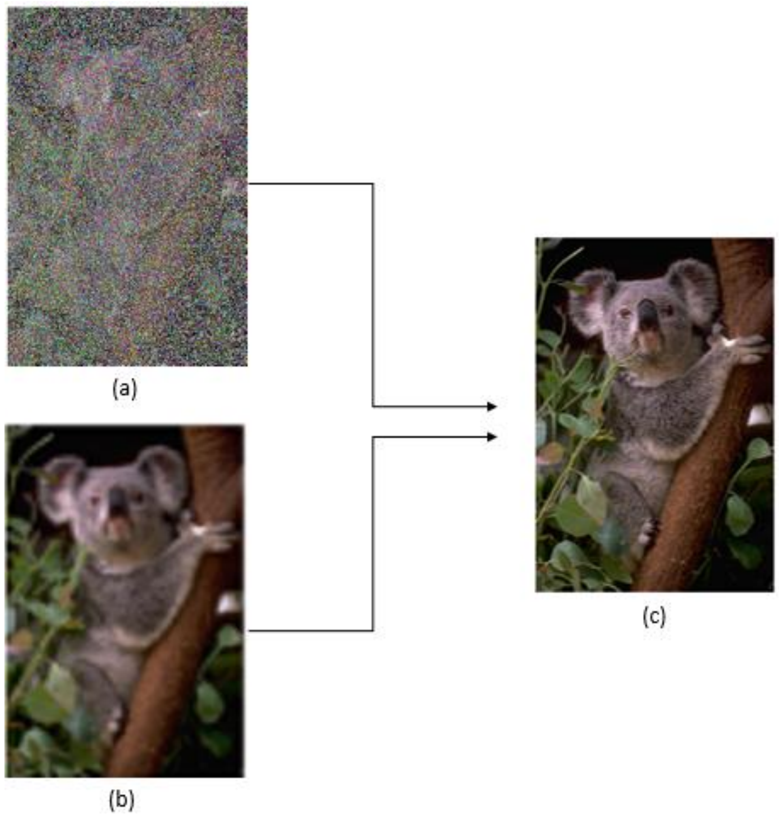


Figure 11. *Image reconstruction, (a) Image corrupted by random noise. (b) Low resoluton image. (c) Output- High resolution and clean image.*

Success Criteria

The success of the project will be measured based on the accuracy and efficiency of the algorithm developed in image reconstruction.

The success criteria include:

1. Creating a higher resolution image.
2. Image corruption by synthetic noise models.
3. Restoring an image that will be sharper and less grainy.
4. Achieve the most effective image reconstruction model.
5. Reduced time and effort required to obtain clean data.

5. Research Process

5.1 Process

The main goal of the project is to transform an image from low resolution to high resolution. We do this by studying how noise affects the image and what types of noise can be used to restore the image to the cleanest version.

In part A of the project, the research process began by studying new fields in the vast field of image reconstruction. Then, a study was conducted on the articles, algorithms, models and architecture of U-net during this study, we divided our work into two main parts. First, we classified noisy images into different types of noise such as Gaussian, Bernoulli, and Monte Carlo, which helps simulate real-world conditions. We delved into the characteristics and challenges of each type of noise and tested methods on how to handle them and restore the images. Second, a comprehensive study of the U-net architecture and implementation process. This involved analyzing how the architecture works, what input it takes and what output it produces. We also tested techniques to improve image resolution to obtain cleaner images. Therefore, further research was carried out on the use of the existing model to compare the two models and see which one is more effective in performing the task of image reconstruction.

In part B of the project, the presented model will be developed including the 2 parts. The goal is to estimate the average accuracy and overall performance of the model.

5.2 Product

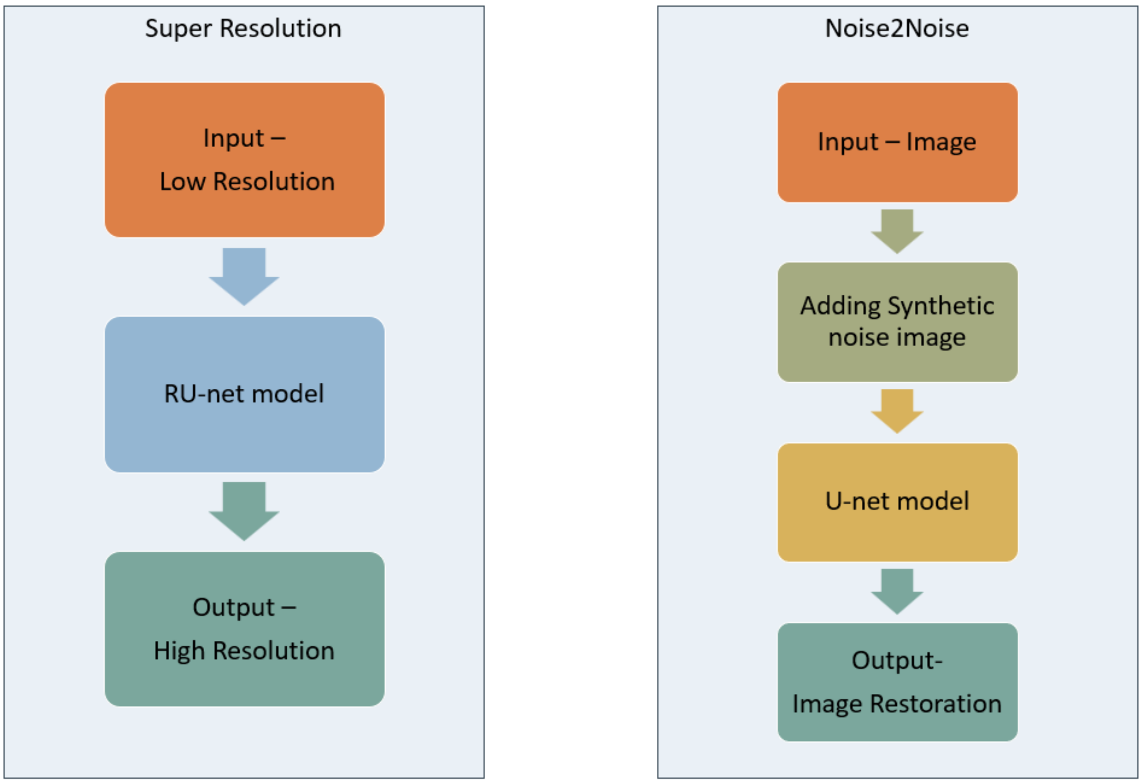


Figure 12. *Workflow of the product.*

Our product focuses on improving image quality through two models: Super-Resolution and Noise2Noise, image restoration without clean data. In the process of super- resolution, we use a Robust U-Net model to upscale low-resolution images to high resolution, reproducing fine details and improving clarity. Additionally, at noise2noise, we add synthetic noise to images to simulate real-world scenario. We use a U-net model to indirectly learn the nature of the noise and restore the image to a clean image.

5.2.1 Super Resolution

Input – An Image

The input is collection of high-resolution images. During the training phase, shown in Figure 13 in purple block, the high-resolution image first down-sampled by a factor of two in both direction, then blurred using filter. Next, the image up-sampled by a factor of two. It's a blurry picture where no specific features can be discerned. All the steps of the algorithm will be performed on these images. The high-resolution images will be used as a comparison source for the segmentation task. In the test phase, shown in Figure 13 in green block, it will receive as input the image with a low resolution, up-sampling by factor of two and then enter to the trained model.

RU-net Model

Following the input phase, we start training our model. Utilizing both blurred images and their original high-resolution counterparts, the model learns to distinguish and enhance image features.

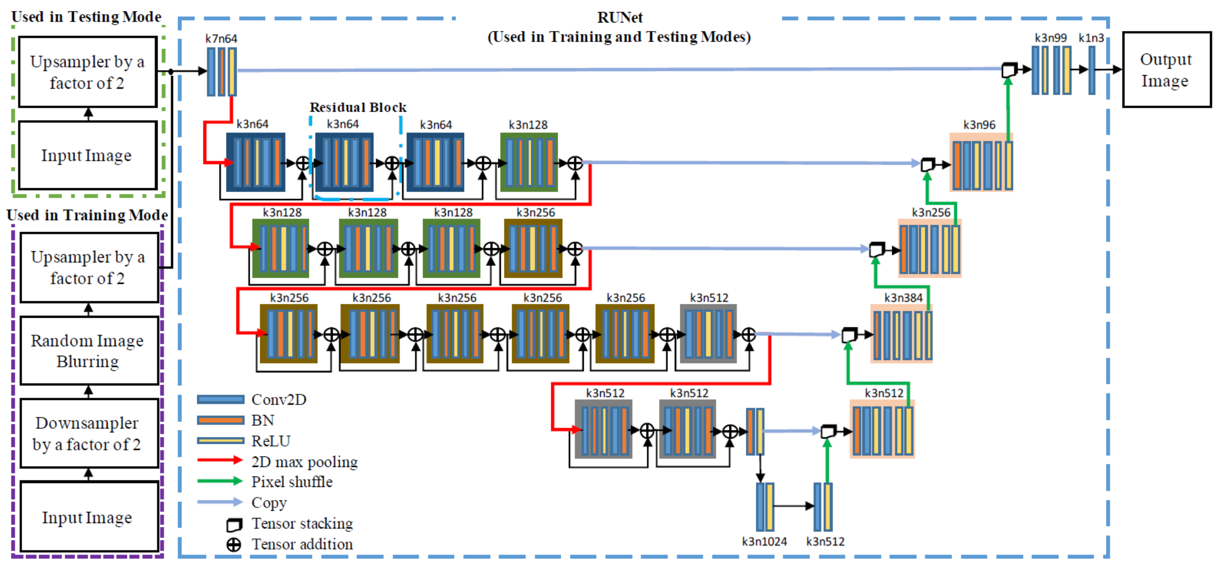


Figure 13. *RU-net model [3].*

The proposed RU-net resolution enhancement program introduces a new model that includes two key components: a degradation module and a new U-net architecture. The degradation module takes practice images and makes them corrupted in different ways. The U-net architecture focuses on enhancing their resolution. Together, these components form a comprehensive solution that aims to improve image quality, as shown in Figure 11.

The RU-net architecture consists of convolutional layer, ReLU activation function and max pooling in the encoder and decoder phase like in the basic U-net model. In addition to this, the model contains additional components, including concatenate blocks and pixel shuffle. Concatenation of blocks not only between the encoder and the decoder but in each layer between each convolution operation, the concatenation is performed by a connection operation. This action enables a more global exchange of information and thus it is possible to capture more complex patterns and relationships in the image. Instead of basic up-sampling on the decoder phase, the model uses pixel shuffle (in section 3.3.2). This method preserves spatial information more efficiently and prevents the introduction of additional parameters for learning, making it computationally efficient. Likewise, we will also use the perceptual loss function (in section 3.4) during the training.

Output – High Resolution Image

The output of the RU-net model is a high-resolution image that has undergone denoising and restoration processes. This image typically exhibits reduced noise and enhanced visual quality compared to the input image.

5.2.2 Noise2Noise

Input – An Image

The input to the model is an image that can be from various sources, such as cameras or sensors, imaging devices, landscape images, etc. For training the model we use a data set containing N clean images, and M representing the number of noise realizations for each clean image. In our model we will use N = 100 and M = 20, so the number of points will be 100 * 20 = 2000. Using noisy images provides benefits of reduced generation time, allowing the generation of a larger set of training data. The network therefore has the potential for substantial improvement with a larger training set.

Adding Synthetic Noise

In this phase, synthetic noise is artificially introduced to the dataset containing images. Various types of synthetic noise, such as Gaussian noise or Poisson noise, are added to the images to simulate real-world noisy conditions. This exposure to different types of corruption can help the model learn to better generalize and adapt to different manifestations of the noise, leading to improved denoising performance.

U-net Model

After preparing the image with synthetic noise, we will train the U-Net model.

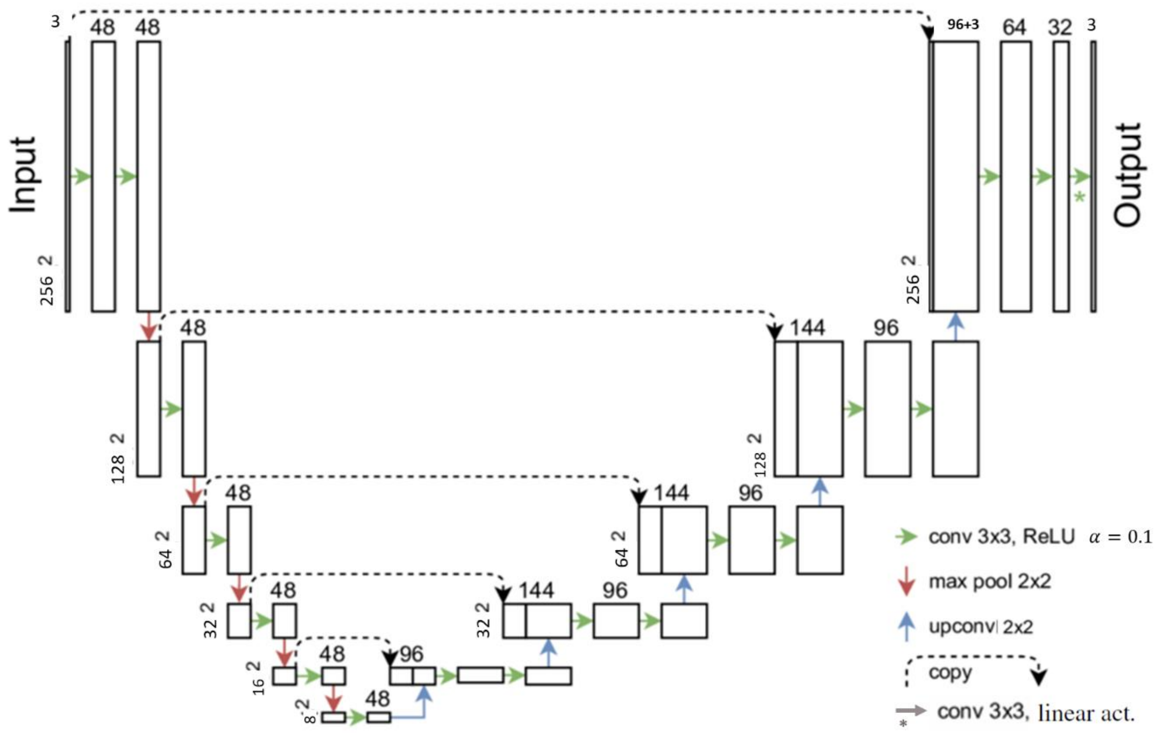


Figure 14. *U-net Noise2Noise model [4]. An image is provided as input to the trained model. Then process the image through its layers, leveraging its learned denoising capabilities to remove noise while preserving important image details.*

During training, the model iteratively learns to map noisy input images to clean output images by adjusting its parameters. Each training consists of a pair of images, a single noise realization and the corresponding clean image, obtained by averaging M-1 noisy images. (M- number of noise realizations)

The input includes an image size of 256 x 256 pixels, with pixel values were represented in range [-0.5, 0.5]. The Minibatch size is set to 4, and a padding of 1 is applied. The size of input/output features channels varies based on the type of images being processed. For general images n=m=3, for Monte Carlo simulations n=9, m=3, and the up-sampling type utilized is nearest neighbor. Training was done using ADAM with parameter values $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 10^{-8}$. The learning rate is 0.001 except Monte Carlo denoising' where 0.0003.

The noise gradually tends to cancel out when averaging over a sufficiently large number of pixels. The weight gradients are averaged over a large number of pixels which helps to smooth the noise in the gradients, making them more stable and less affected by the built-in noise during training. The presence of clean weight gradients allows the optimization algorithm to efficiently update the network parameters and converge towards a solution, despite the noisy nature of the task.

Output - Image Restoration

After sufficient training, the U-Net model becomes able to rule out new images. The result is an image with significantly reduced noise and improved visual quality. The model achieves high-resolution reconstruction even in the absence of clean data during the corruption process.

5.3 Related Diagram

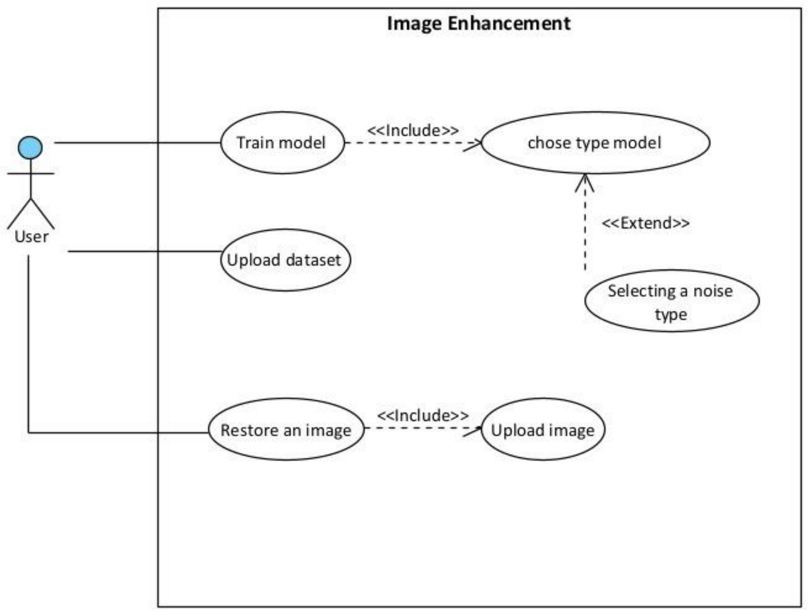


Figure 15 . *Use-Case diagram.*

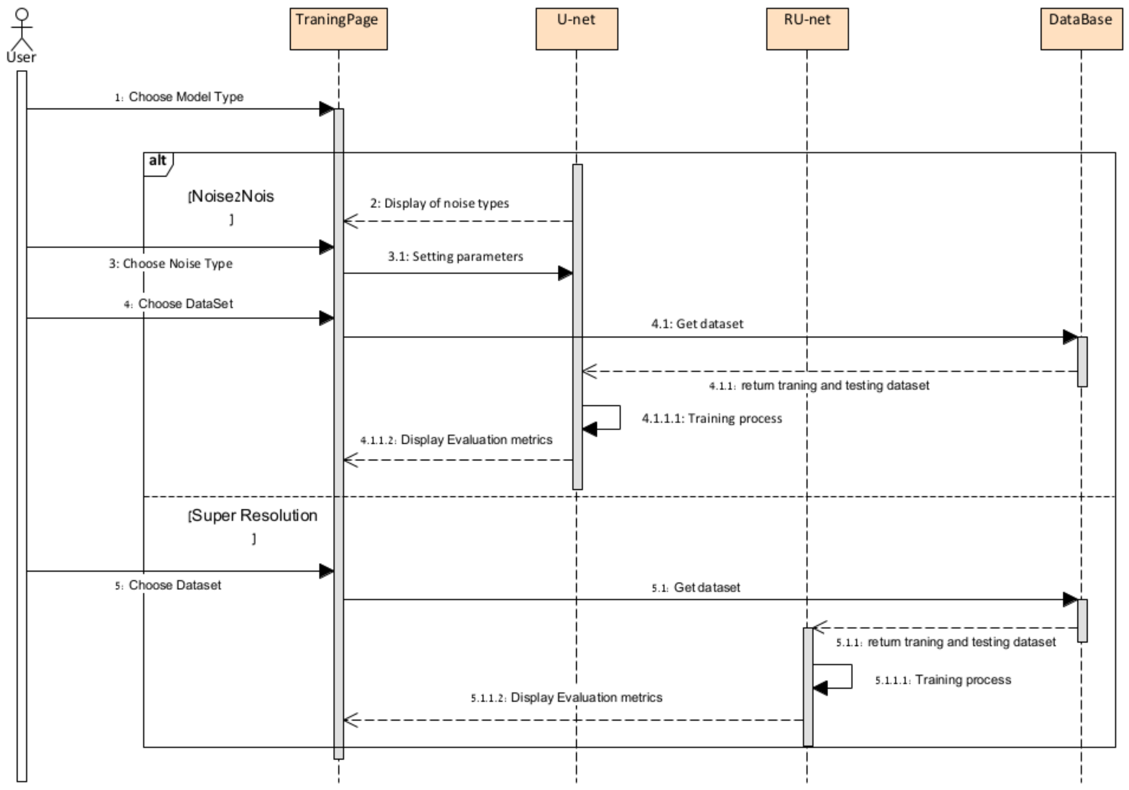
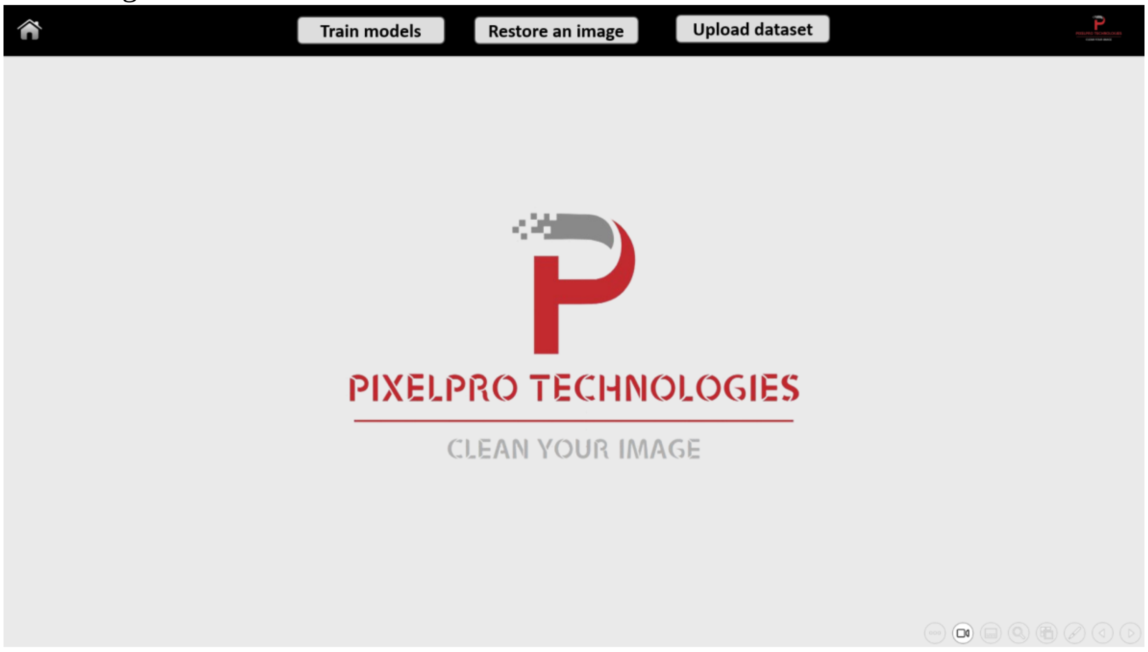

User
TraningPage
U-net
RU-net
DataBase
1: Choose Model Type
alt
[Noise2Nois]
2: Display of noise types
3: Choose Noise Type
3.1: Setting parameters
4: Choose DataSet
4.1: Get dataset
4.1.1: return traning and testing dataset
4.1.1.1: Training process
4.1.1.2: Display Evaluation metrics
[Super Resolution]
5: Choose Dataset
5.1: Get dataset
5.1.1: return traning and testing dataset
5.1.1.1: Training process
5.1.1.2: Display Evaluation metrics

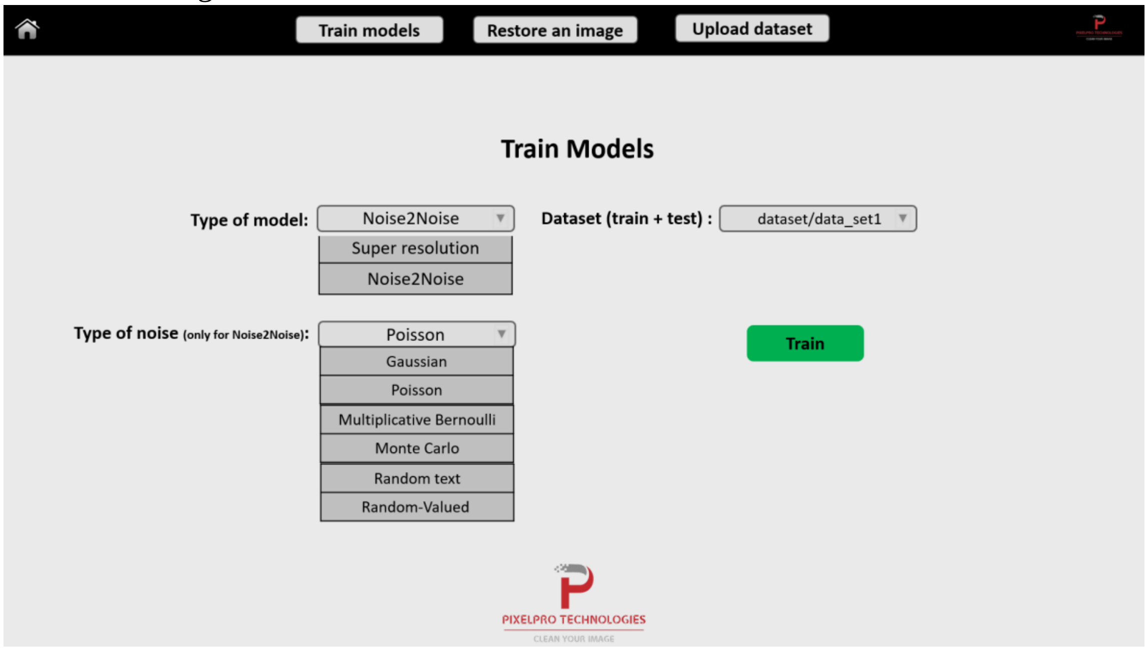
Figure 16. *Sequence diagram.*

5.4 GUI

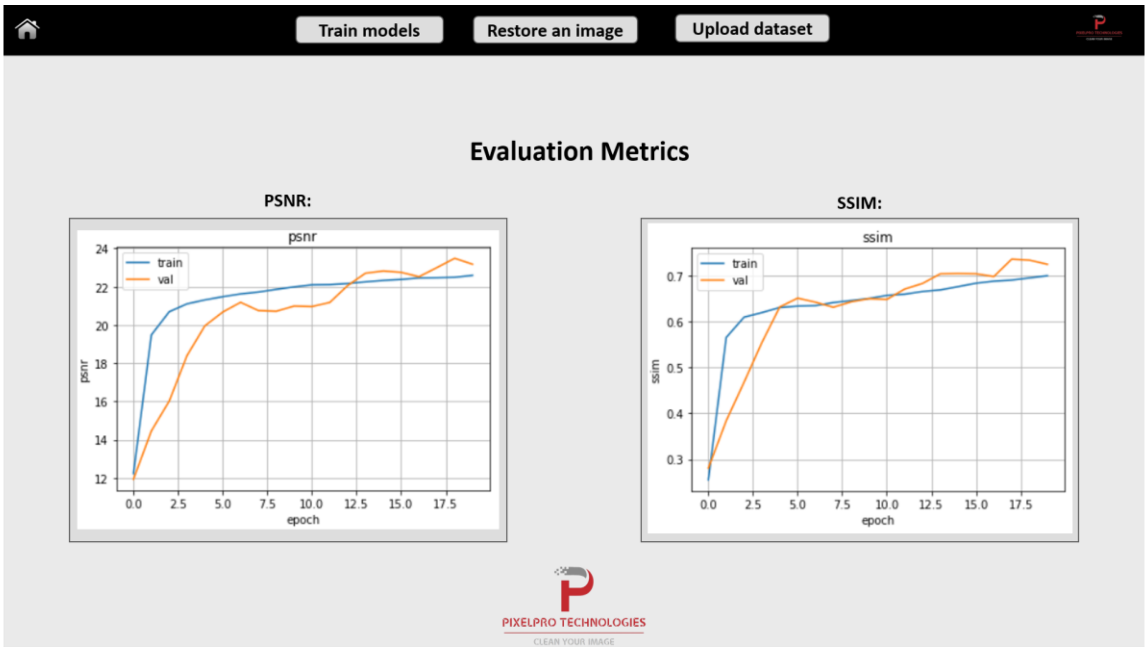
Home Page



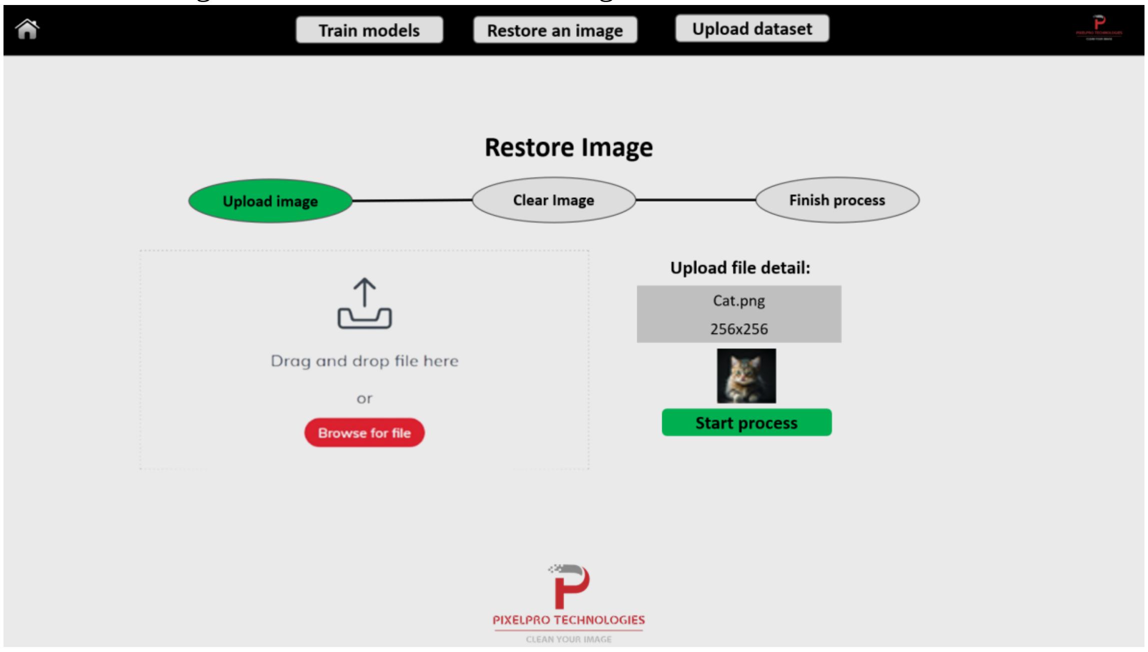
Choosing type of model you want to train. When choosing a Noise2Noise model, it is necessary to choose which type of noise to add to the image. Also, selecting a dataset from an existing list.



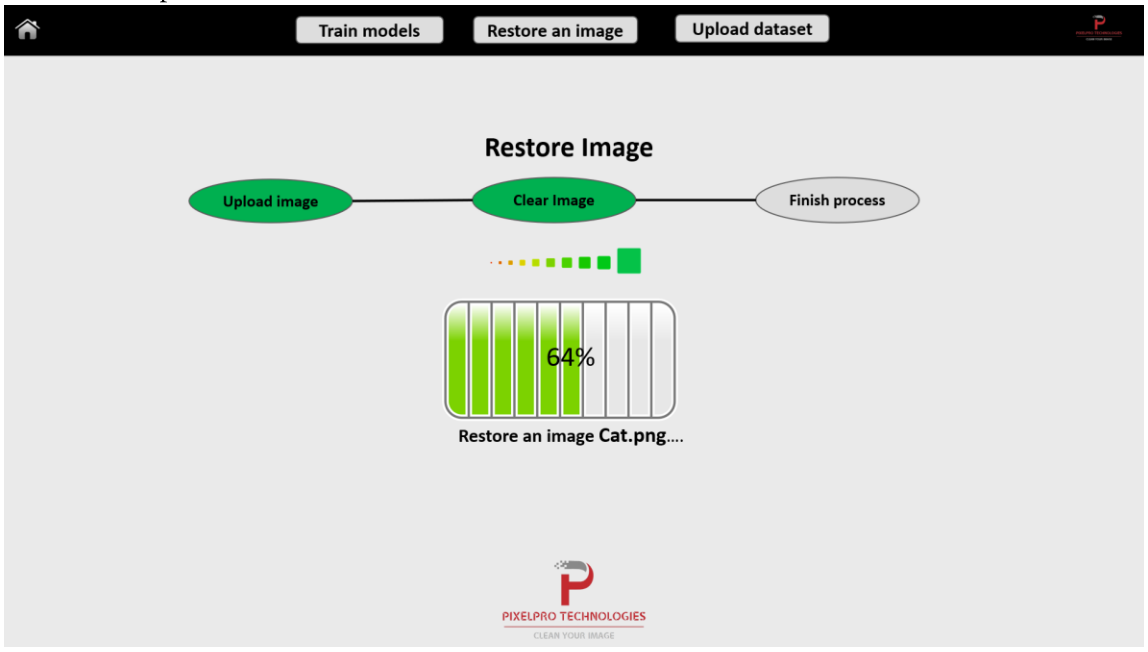
Evaluation metrics of the training process.



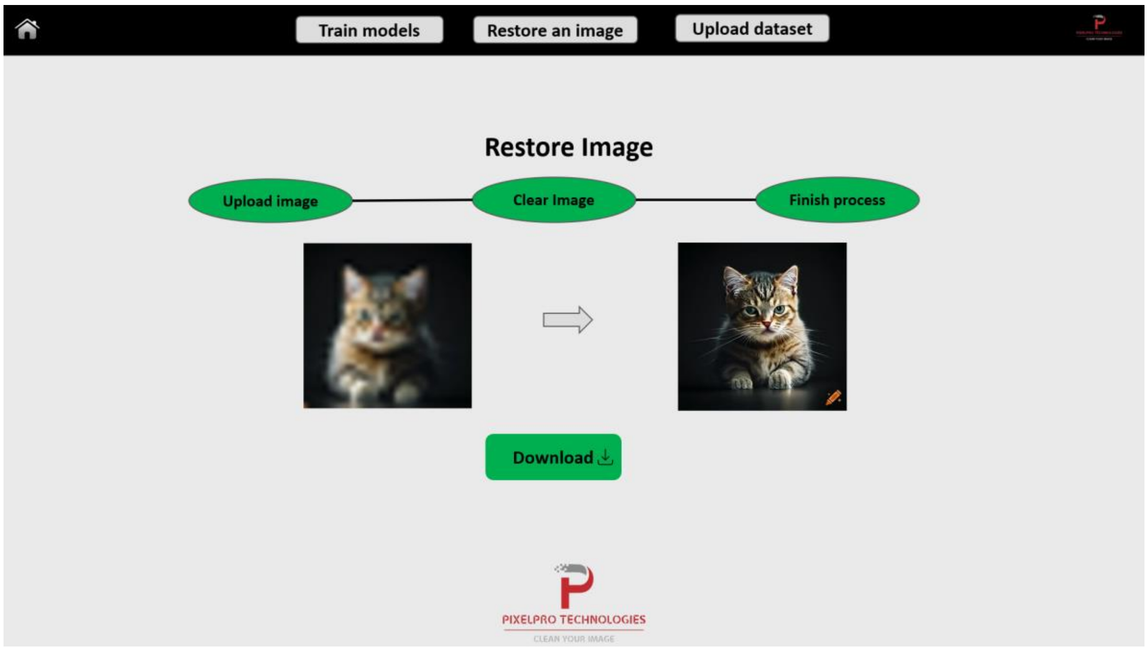
Restore an image - select a low-resolution image for enhancement.



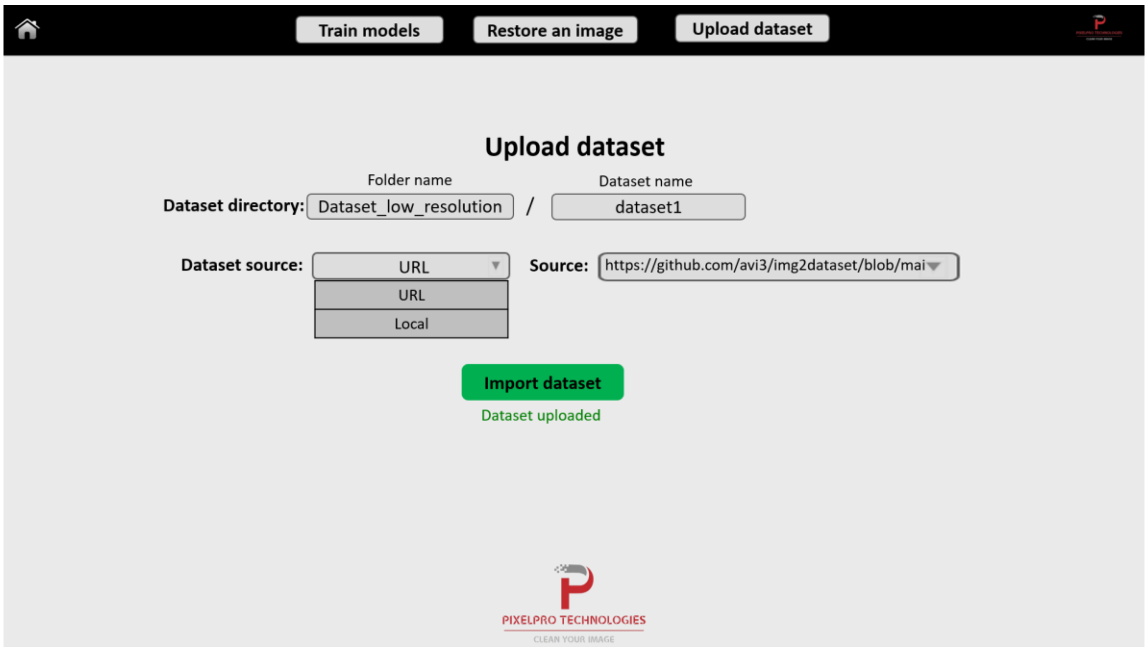
Restoration process.



Output image in high-resolution.



Upload Dataset- uploading a data set from the local computer or through a link from a website.



Upload dataset

Folder name

Dataset name

Dataset directory: Dataset_low_resolution / dataset1

Dataset source: URL

URL

Local

Source: https://github.com/avi3/img2dataset/blob/mai

Dataset uploaded

6. Evaluation Plan

The evaluation plan for an image reconstruction model involves several stages to ensure a thorough assessment of its performance. The first stage is the selection of a dataset of images that includes a various of images. Afterward, data preprocessing involves several steps, including reducing resolution, duplicating the image, expanding dimensions, and adding various types of noise to the image. The next step is to train the models on the pre-processed dataset, using a validation set to monitor the model's performance during the training process. After training is complete, the model is evaluated on a test set of images, using metrics such as PSNR and SSIM (3.5) to measure the quality of the images. Finally, we will compare the performance of the different models trained on each type of dataset.

6.1 Testing process

Test	Module	Test description	Excepted result
1	Home page	Click 'Home' button in navbar menu	Open 'Home' window
2	Home page	Click 'Train Models' in navbar menu	Open 'Train Models' window
3	Home page	Click 'Restore an image' button in navbar menu	Open 'Restore an image' window
4	Home page	Click 'Upload dataset' button in navbar menu	Open 'Upload dataset' window
5	Train Models window>Type of Model	Click on combo box 'Type of model' and choose 'Super resolution'	Option ‘Super resolution' should be chosen.
6	Train Models window ->Type of model	Click on combo box 'Type of model' and choose 'Noise2Noise'	Option ‘Noise2Noise' should be chosen.
7	Train Models window ->Type of model (Noise2Noise)->Type of noise	Click on combo box 'Type of noise' and choose 'Gaussian'	Option ‘Gaussian' should be chosen
8	Train Models window ->Type of model (Noise2Noise)->Type of noise	Click on combo box 'Type of noise' and choose 'Poisson'	Option ‘Poisson' should be chosen
9	Train Models window ->Type of model (Noise2Noise)->Type of noise	Click on combo box 'Type of noise' and choose 'Multiplicative Bernoulli.' '	Option ‘Multiplicative Bernoulli' should be chosen
10	Train Models window ->Type of model (Noise2Noise)->Type of noise	Click on combo box 'Type of noise' and choose 'Monte Carlo'	Option ‘Monte Carlo' should be chosen
11	Train Models window ->Type of model (Noise2Noise)->Type of noise	Click on combo box 'Type of noise' and choose 'Random text.'	Option ‘Random text' should be chosen
12	Train Models window ->Type of model (Noise2Noise)->Type of noise	Click on combo box 'Type of noise' and choose 'Random-Valued '	Option ‘Random-Valued should be chosen
13	Train Models->Dataset	Click on combo box ' Dataset (train + test) ' and choose ' dataset/data_set1.'	Option ' dataset/data_set1' should be chosen
14	Train Models window ->Train Process window	Click on 'Train' button	Open 'Train process' window with active training timer and current PSNR present window.
15	Train Models window	Click on 'Train' button (failure case: One or more combo box not chosen)	an alert in red will pop up with message 'one or more fields are not chosen'
16	Train Models ->->Type of model (Super resolution)->Type of noise	Click on 'Train' button (failure case: hosen Model type: 'Super resolution' and chose also type of noise)	an alert in red will pop up with message 'please remove the Type of noise in model Super resolution'.
17	Restore an image window->Upload image	Click on ‘Browse for file'	Open the file explorer to select the files

18	Restore an image window->Upload image	Drag and drop file into the box	File dragged should be uploaded.
19	Restore an image window->Clear image	Click on button 'Start process'	Advance to step of Clear image processing with active precent of the clear
20	Restore an image window->Finish process	Click on button 'download'	Download the clean image to your computer
21	Upload dataset-> Dataset directory	Fill a folder name	Set folder name
22	Upload dataset window-> Dataset direction	Fill a dataset name	Set dataset name
23	Upload dataset window->Dataset source	Click on combo box 'Dataset source' and choose 'URL.'	Option ‘URL' should be chosen
24	Upload dataset window->Dataset source	Click on combo box 'Dataset source' and choose 'Local.'	Option ‘Local' should be chosen
25	Upload dataset window->Dataset source ('URL')->source	Fill URL of the dataset	Set the URL dataset
26	Upload dataset window ->Dataset source ('Local')->source	Click on Drop down 'source' and choose file to upload.	The file should be chosen and present in 'source' field
27	Upload dataset window	Click on 'Import dataset' button	an alert in green will pop up with message 'Dataset uploaded'
28	Upload dataset window	Click on 'Import dataset' button (failure case: One or more field empty)	an alert in red will pop up with message ' One or more field empty
29	Upload dataset window	Click on 'Import dataset' button (failure case: the dataset name already exists)	an alert in red will pop up with message 'this name of dataset already exist'
30	Upload dataset window	Click on 'Import dataset' button (failure case: invalid URL directory of dataset)	an alert in red will pop up with message 'the URL is

7. AI Tools

We used ChatGPT and Gemini for checking spelling and grammar of sentences and sometimes also referring to articles about complex topics. Also, we used ResearchRabbit, which provided us with easier access to the existing references in the articles, to explore all the existing subtopics.

Prompts:
' Can you check the spelling and grammar in the following sentence: [text].
' ' How to get high resolution image via noise2noise. '

8. References

[1]. Dabov, K., Foi, A., Katkovnik, V., & Egiazarian, K. (2007). Image denoising by sparse 3-D transform-domain collaborative filtering. IEEE Transactions on image processing, 16(8), 2080-2095.

[2]. Gu, H., Wang, Y., Hong, S., & Gui, G. (Year of publication). Blind Channel Identification Aided Generalized Automatic Modulation Recognition based on Deep Learning. Digital Object Identifier 10.1109/ACCESS.2017

[3]. Hu, X., Naiel, M. A., Wong, A., Lamm, M., & Fieguth, P. (2019). RUNet: A robust UNet architecture for image super-resolution. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (pp. 0-0).

[4]. Lehtinen, J., Munkberg, J., Hasselgren, J., Laine, S., Karras, T., Aittala, M., & Aila, T. (2018). Noise2Noise: Learning image restoration without clean data. arXiv preprint arXiv:1803.04189.

[5]. Mao, X. J., Shen, C., & Yang, Y. B. (2016). Image restoration using convolutional auto-encoders with symmetric skip connections. arXiv preprint arXiv:1606.08921.

[6]. Makitalo, M., & Foi, A. (2010). Optimal inversion of the Anscombe transformation in low-count Poisson image denoising. IEEE transactions on Image Processing, 20(1), 99-109.

[7]. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18 (pp. 234-241). Springer International Publishing.

[8]. Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., ... & Wang, Z. (2016). Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1874-1883).

[9]. Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2018). Deep image prior. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 9446-9454).

[10]. Veach, E., & Guibas, L. J. (1995, September). Optimally combining sampling techniques for Monte Carlo rendering. In Proceedings of the 22nd annual conference on Computer graphics and interactive techniques (pp. 419-428).

[11]. Baheti, P. (2022, October 5). Image Recognition: Definition, Algorithms & Uses. Retrieved from https://www.v7labs.com/blog/image-recognition-guide.

[12] Fei-Fei Li, Jiajun Wu, Ruohan Gao, Lecture 5- CS231n.

[13] Up and Down sampling - https://mriquestions.com/upsampling.html.

[14] Shamir, G., & Lin, D. (2022, April 5). Reproducibility in Deep Learning and Smooth Activations. Posted by Gil Shamir and Dong Lin, Research Software Engineers, Google Research.https://research.google/blog/reproducibility-in-deep-learning-and-smooth-activations/