

Deep Learning and its applications to Signal and Image Processing and Analysis S2-2025

361-2-1120

**Final Project:
Super-Resolution Image Enhancement**

Tomer Abram 208931691

Laith Sadik 318679677

GitHub repository link: <https://github.com/laithsadik/Super-Resolution-Image-Enhancement-Final-Project.git>

Note: Because of the parallels arising from Google colab, we trained the models on 50 epochs.

3.2) Abstract

3.2.1) The challenge we chose is Single Image Super-Resolution (SISR), a computer vision task aimed at reconstructing a sharp and clear high-resolution image from a blurry or low-resolution one. This problem is crucial in real-world applications such as medical imaging, satellite photography, and video streaming, where images are often captured or transmitted at low quality. The core difficulty lies in recovering the fine details lost in the blurry image - a process that does not have a single correct answer, as multiple plausible interpretations may exist for the same low-quality input. The goal of this project is to develop a deep learning model capable of learning this mapping, and generating natural, realistic, and high-quality high-resolution images.

3.2.2) The first model we used was **SRResNet** - a classical architecture based on deep residual networks, providing a solid baseline for Super-Resolution tasks. This model served as our benchmark for comparison. The second model was **our own enhanced version of SRResNet** (it's like EDSR), which we built from scratch. It includes improvements such as normalization, a custom residual block design, and an improved upscaling strategy. These changes aimed to improve reconstruction quality while maintaining computational efficiency.

3.2.3) Our experiments showed that our custom model outperformed the baseline SRResNet in several quantitative metrics (including PSNR, SSIM, and FID) and produced sharper, more detailed visual results in most cases. Additionally, in our Ablation Study where we replaced the loss function from L1 to MSE, we observed that although the MSE-based model converged faster during training, it ultimately achieved slightly lower reconstruction quality. These results highlight the importance of the architectural enhancements and the choice of loss function in improving Super-Resolution performance.

3.3) Introduction, Objective, and Data Description:

3.3.1) The challenge of our project is Single Image Super-Resolution (SISR) — the task of reconstructing a high-resolution (HR) image from a corresponding low-resolution (LR) input. This problem is fundamental in computer vision and image processing, with applications in medical imaging, satellite imagery, surveillance, and more. The difficulty arises

from the fact that information is lost during downsampling; therefore, the process of upscaling is inherently ill-posed, a single low-resolution image may correspond to many possible high-resolution images. The goal is to train a deep learning model that can learn to recover fine details, textures, and sharpness that are missing in the LR input, in a way that is perceptually convincing and faithful to the original image.

3.3.2) Describe the Dataset:

The dataset used for this project is **DIV2K** (Diverse 2K resolution images), a widely-used benchmark dataset for **Single Image Super-Resolution** (SISR). These images are rich in textures and fine details, making them ideal for training and evaluating super-resolution models. It contains a total of **900 high-quality images** with a resolution of approximately **2040×1400 pixels**, designed to support training and evaluating models that enhance image resolution. The dataset is split as follows: **800 images for training, 100 images for validation**. Each image in the dataset is provided in a high-resolution (HR) version. For the purpose of super-resolution training, low-resolution (LR) versions are created by downsampling the HR images by a factor of $\times 4$ using bicubic interpolation. This simulates the degradation process typical in real-world scenarios.

In our project, we work with HR-LR image pairs, where:

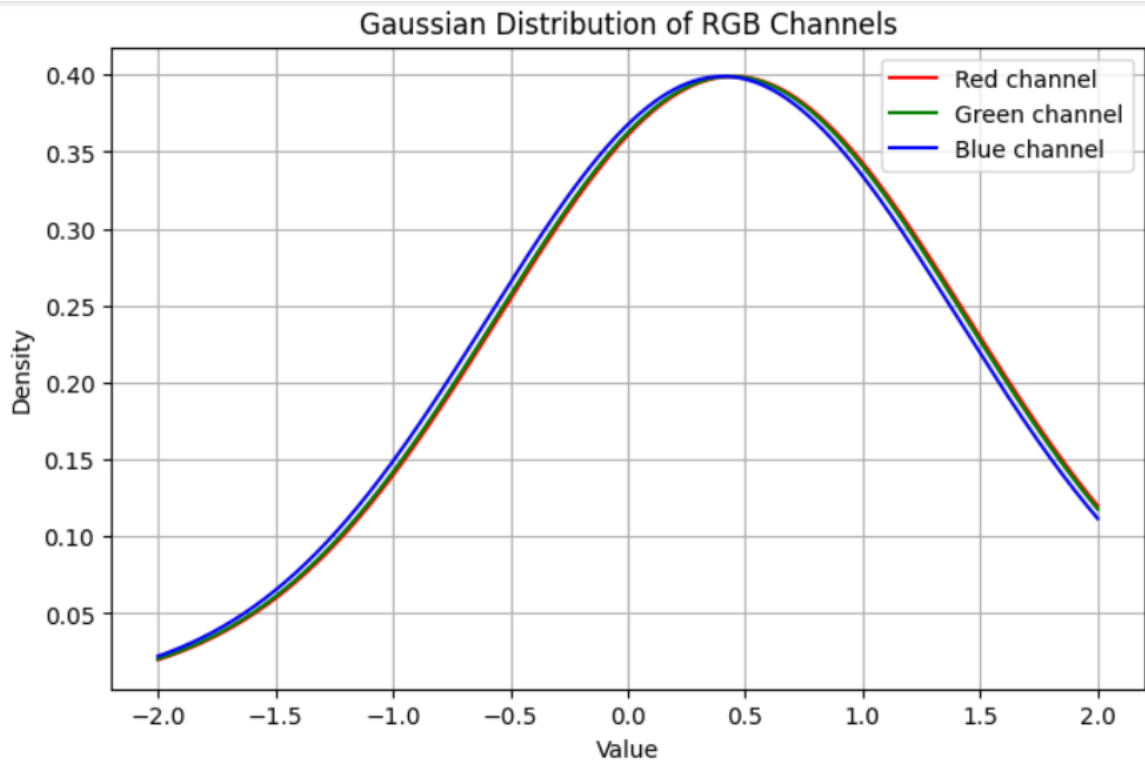
- The HR image is the target output the model tries to reconstruct.
- The LR image is the input image, which is a degraded (blurred and downsized) version of the HR.

During preprocessing, the images are:

- Cropped into fixed-size patches of 96×96 pixels (for HR).
- Corresponding LR patches are generated at 24×24 pixels.
- Pixel values are normalized to the range $[-1, 1]$ to facilitate stable training.

This dataset provides a good balance of diverse content and resolution, which helps the model learn to recover fine details and textures from significantly downsampled inputs.

RGB Channel Means in DIV2K Dataset (Normalized): The graph depicts the average pixel values in each color channel (red, green, blue) in the DIV2K dataset after normalization to the range $[0, 1]$. The values represent the average brightness level in each channel, and are used to normalize the images before feeding them to the model. Normalization helps stabilize training and improves network performance.



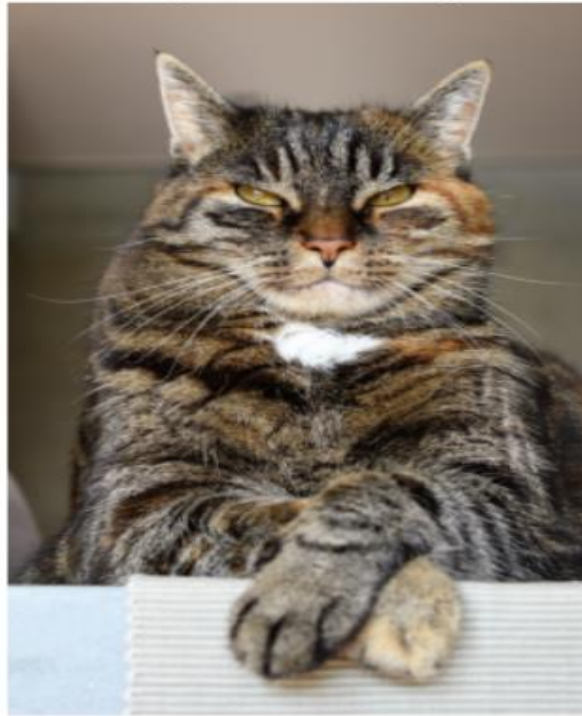
Exploratory Data Analysis (EDA) and Visualizations:

The DIV2K dataset contains high-quality images of diverse scenes including landscapes, buildings, people, animals, and everyday objects.

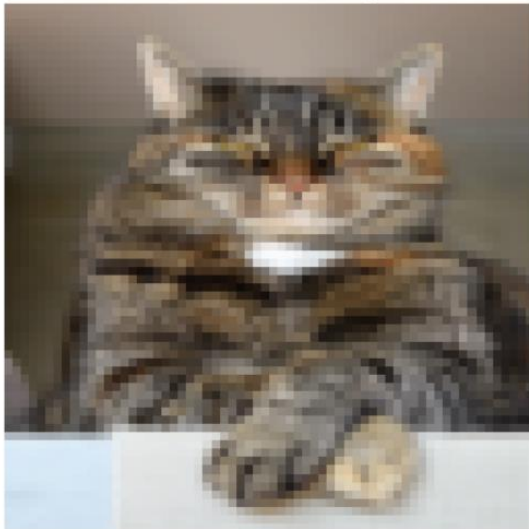


Displaying original image, Low Resolution image and Prediction image:

Original Full Image



Low Resolution Image

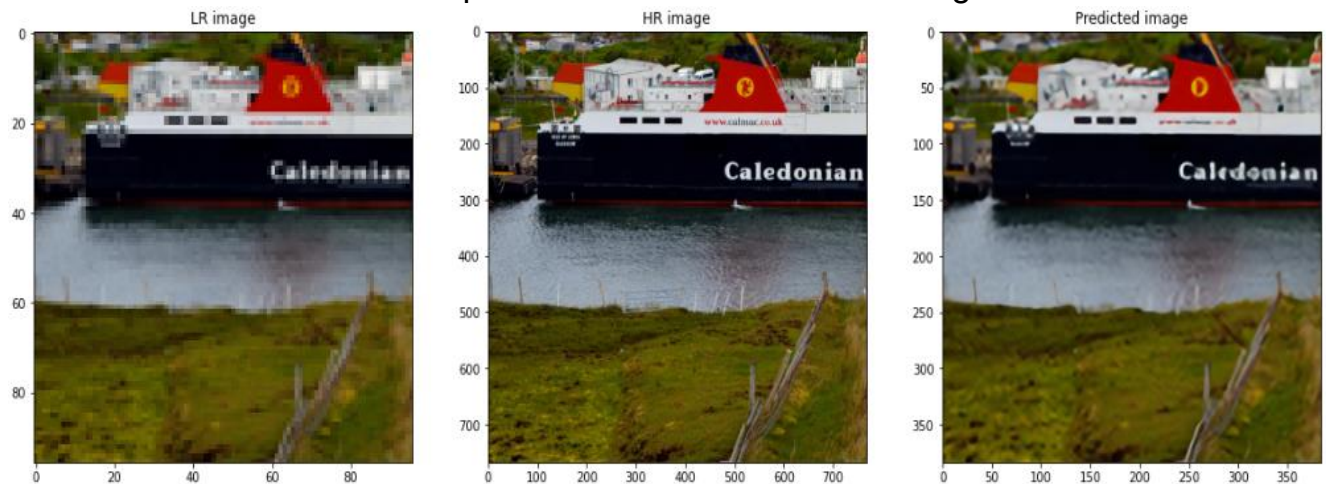


Generated SR Image

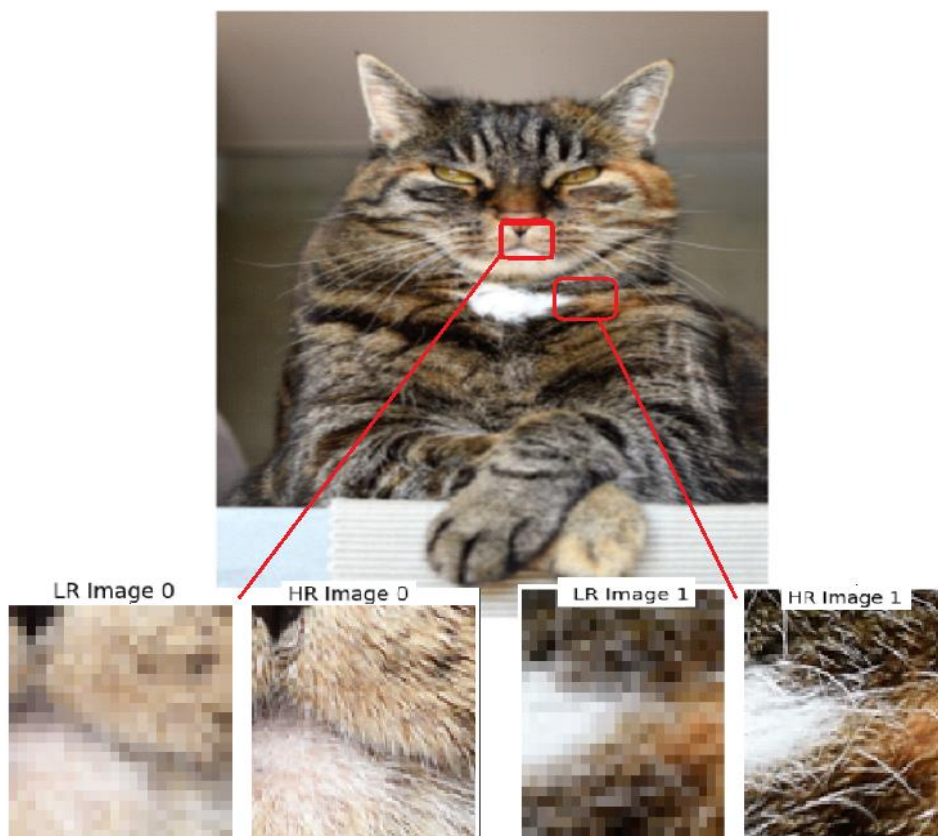


We can see that there is a difference between the two images (Low Resolution, Generated SR image) and it will take 50 iterations of training. The model converges after a thousand iterations in an amazing way.

The image below a representative output from our model after 1000 training epochs. We selected this example from the Kaggle DIV2K SRResNet Code to qualitatively showcase the model's potential. Due to the computational limitations of Google Colab, we were unable to run all experiments for 1000 epochs, and therefore included this specific result to illustrate the model's full capabilities under extended training.



Showing low resolution and high resolution for a random crop of a random image in the validation set:



During training, the model learns to reconstruct small areas (random

crops), and when it encounters an entire image during testing, it manages to apply what it has learned and reconstruct it in high resolution and high quality.

3.3.3) To comprehensively evaluate the performance of super-resolution models, three widely used metrics are applied, each capturing different aspects of image quality:

- **PSNR (Peak Signal-to-Noise Ratio):** PSNR is a pixel-wise quantitative metric that measures the ratio between the maximum possible value of a pixel (usually 255 for 8-bit images) and the power of the error introduced during reconstruction. It is derived from the Mean Squared Error (MSE) between the super-resolved image and the original high-resolution image. A higher PSNR value indicates that the reconstructed image is closer to the ground truth and has less distortion or noise. However, PSNR is sensitive to small pixel differences and may not always reflect perceptual visual quality.
- **SSIM (Structural Similarity Index):** SSIM is a perceptual metric that evaluates the structural similarity between two images, considering local patterns of pixel intensities that have been normalized for luminance and contrast. It compares features such as texture, edges, and structural content, which are crucial for human visual perception. The value ranges from -1 to 1, where 1 means perfect structural similarity. SSIM is particularly useful in assessing whether the model preserves meaningful visual details and structures.
- **FID (Fréchet Inception Distance):** FID is a deep-learning-based metric that compares the feature distributions of real and generated images using a pretrained Inception network. Rather than comparing pixels, FID evaluates how realistic and natural the generated images look in terms of their semantic content. A lower FID score indicates that the generated images are closer in distribution to real high-resolution images, and thus look more convincing to the human eye.

3.4.1) we chose Option 1 (Baseline Model from Kaggle).

3.4.1.1) The baseline model we chose to use is SRResNet, one of the classic architectures for Super-Resolution.

Architecture structure:

- Input: Low-Resolution image.
- Output: High-Resolution image, 4 times the size of the input.

1. conv_block1:

- Convolution layer with kernel size 9×9 , 3 input channels (RGB) and $n_channels$ output (64).
- Activation: PReLU.
- Goal: Allow the model to learn small, subtle changes that need to be added to the image, while preserving important information from the original image. The residuals allow for smoother gradient flow in training.

2. Residual Blocks (x16 Blocks):

- Each block includes two Conv layers with kernel size 3×3 , with BatchNorm and PReLU between them.
- skip connection: $Output = F(x) + x$, which help in learning identities.
- Goal: Learn small corrections and deepen the network without damaging the gradient flow.

3. conv_block2 (Post-Residual Conv Layer):

- An additional Conv layer with kernel size 3×3 + BatchNorm that is added back to the original features before the Residuals (Connects to the skip connection that came from conv_block1).
- Goal: Reinforces the original information that passed directly without changes.

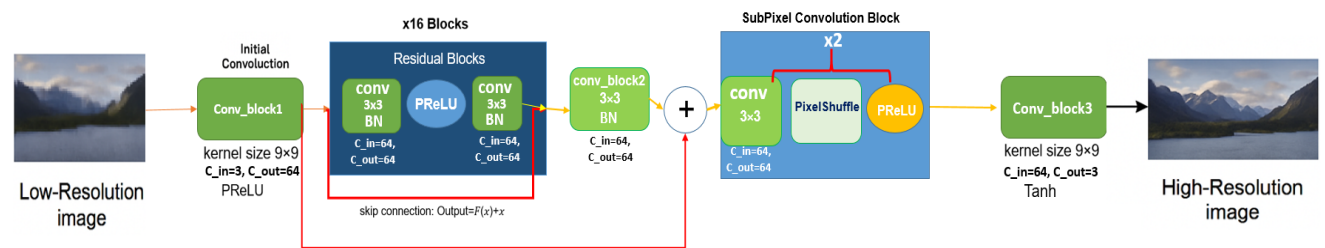
4. SubPixel Convolution Blocks (Upsampling Layers):

- Two layers of PixelShuffle (Upsampling $\times 2$ each, together $\times 4$).
- Before each PixelShuffle there is a Conv layer to prepare the features.
- Activation after each PixelShuffle: PReLU.
- Goal: Instead of doing simple UpSampling, it learns how to properly increase.

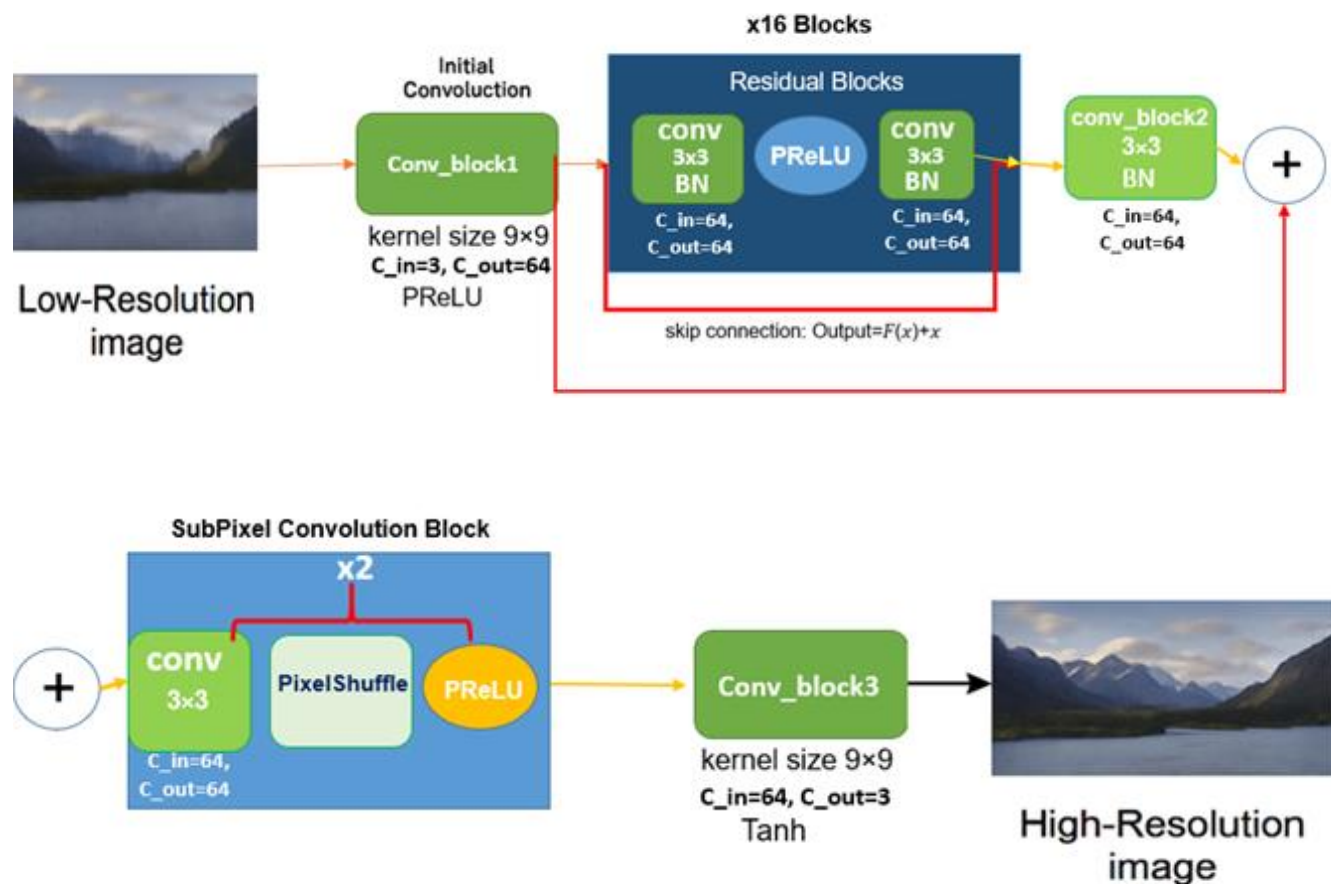
5. conv_block3 (Final Output Layer):

- Conv layer with 3 filters (RGB), kernel size 9×9 that returns back to a normal size image (3 channels, RGB).
- Activation: Tanh (for image in range [-1, 1]).
- Goal: Converts the features back to a super-resolution image.
- Output the high-resolution image.

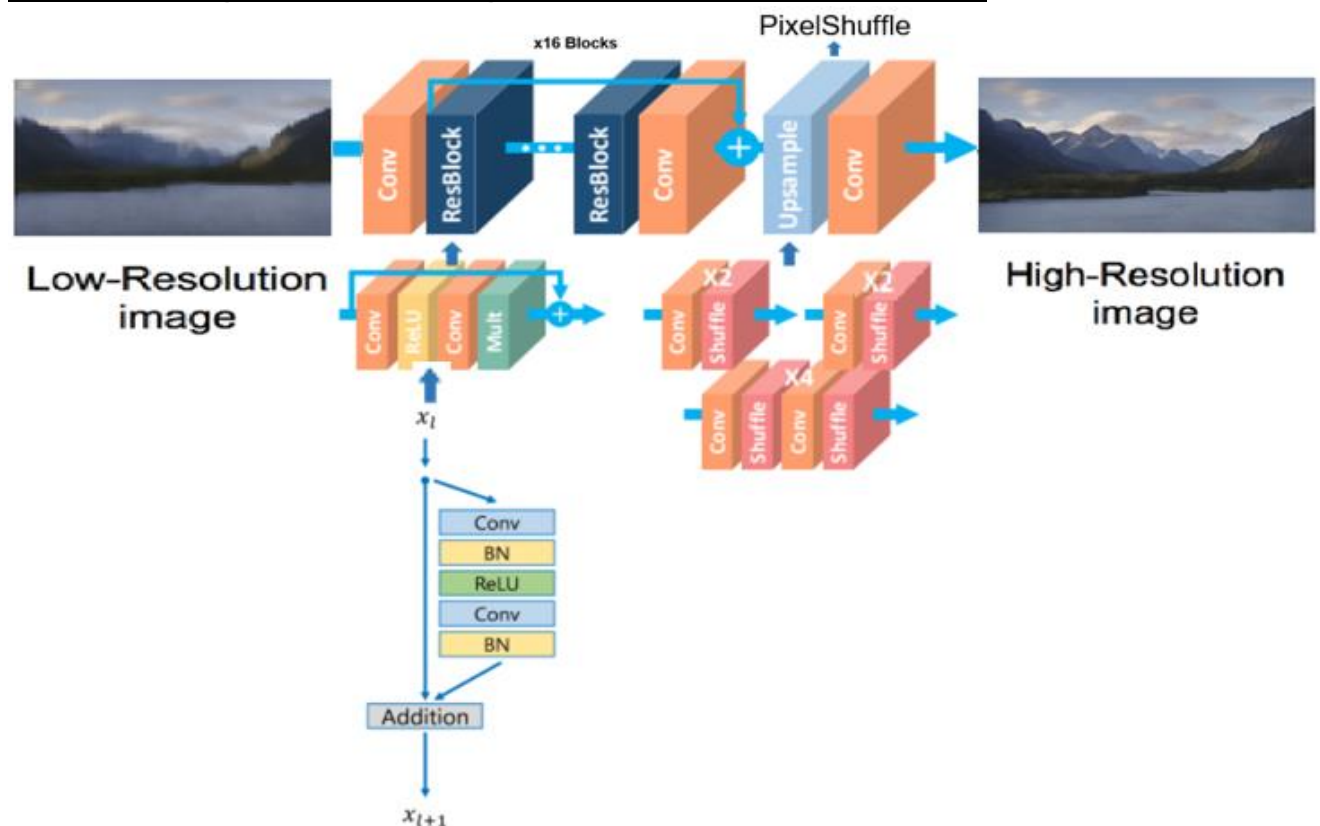
3.4.1.2.a) diagram illustrating the model structure:



3.4.1.2.b) Divided model structure into two parts (ZOOM-IN):



Another diagram explaining the structure of the model:



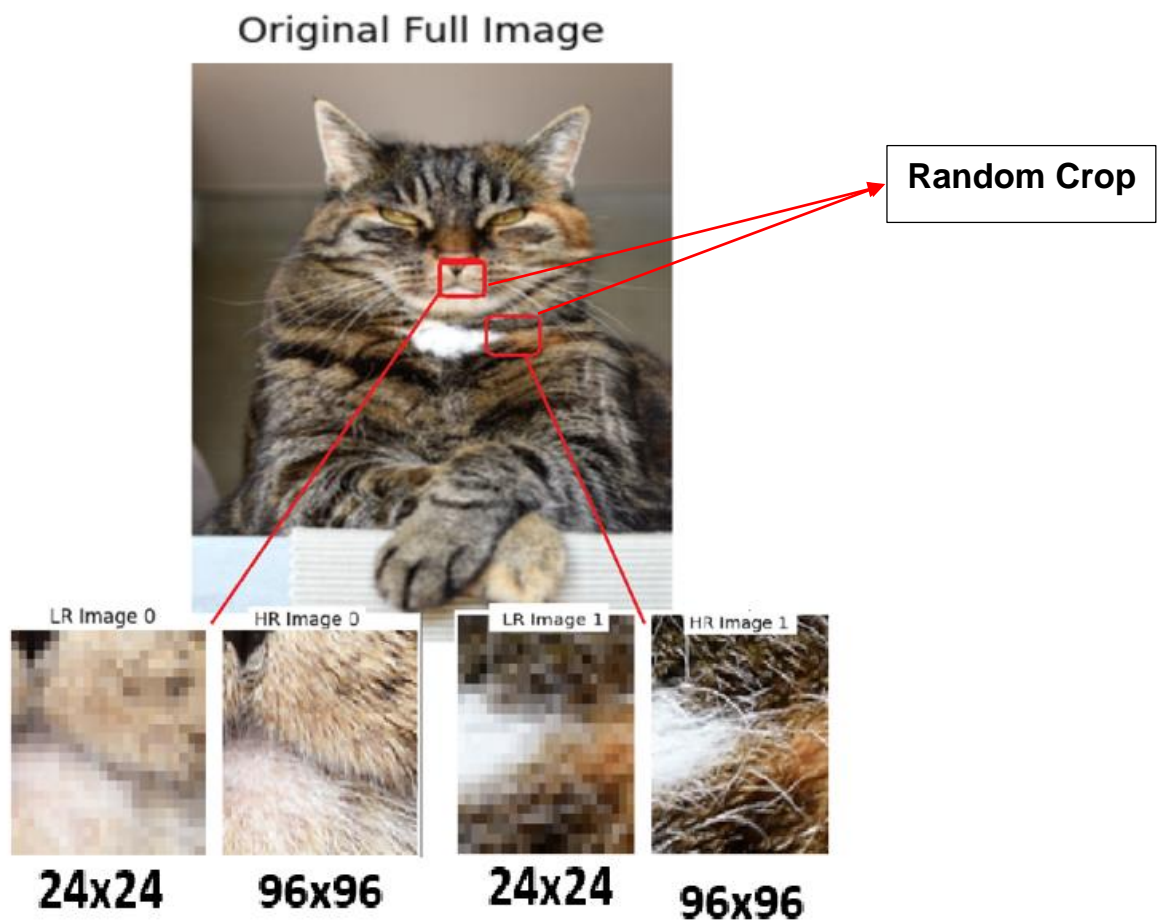
3.4.1.3) data preprocessing steps and visualizations:

Data Preprocessing Steps:

Before training, the images undergo several preprocessing steps

- **Cropping:** High-resolution images are randomly cropped into 96×96 patches to increase data variety and allow the model to focus on local details.
- **Downsampling:** Each cropped high-resolution image is downscaled by a factor of 4 using bicubic interpolation to generate its corresponding low-resolution version.
- **Normalization:** Pixel values are scaled from [0, 255] to [-1, 1], which helps accelerate training and stabilize convergence.
- **Conversion to Tensors:** Images are converted to PyTorch tensors to be compatible with the neural network.
- **Batching and Shuffling:** The data is divided into batches and shuffled each epoch to improve generalization and avoid overfitting.

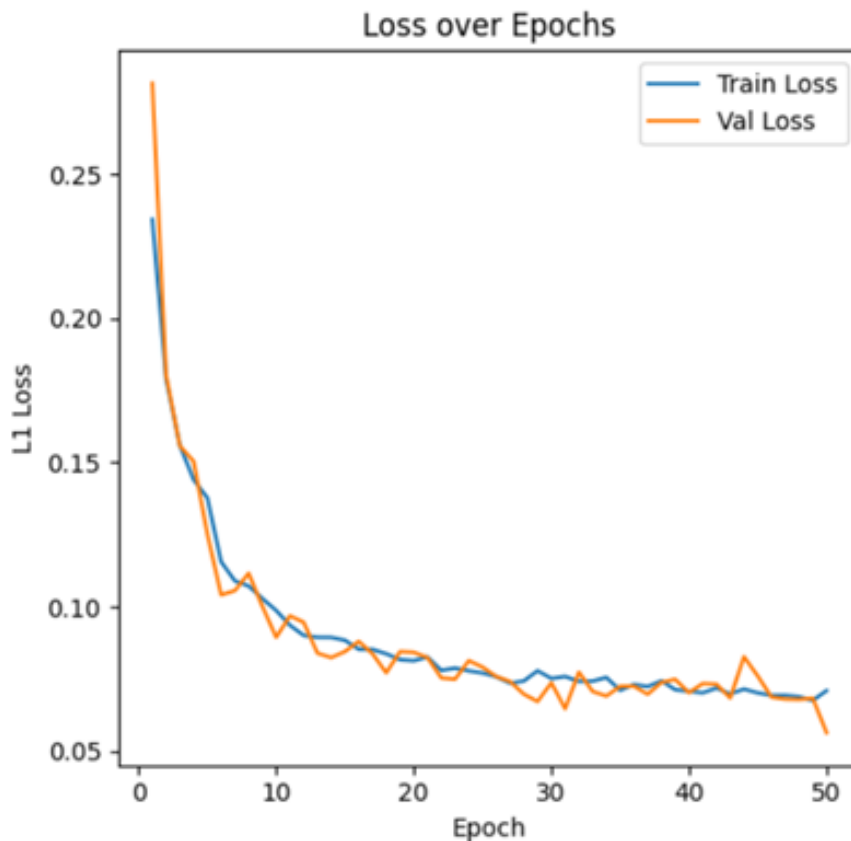
Visualizations: Original image, HR and LR



3.4.1.4) Presentation of quantitative metrics and qualitative examples showing both successful and failed predictions:

1. Quantitative metrics (after 50 epoch of training)

Training and Validation Loss Over Epochs – Model Convergence: The graph shows the training and validation loss values over the epochs. It can be seen that the loss is getting smaller, which indicates that the model is successfully converging and learning to improve its performance. The small gap between the Train Loss and Valid Loss also shows that the model is able to generalize well to data it has not seen.

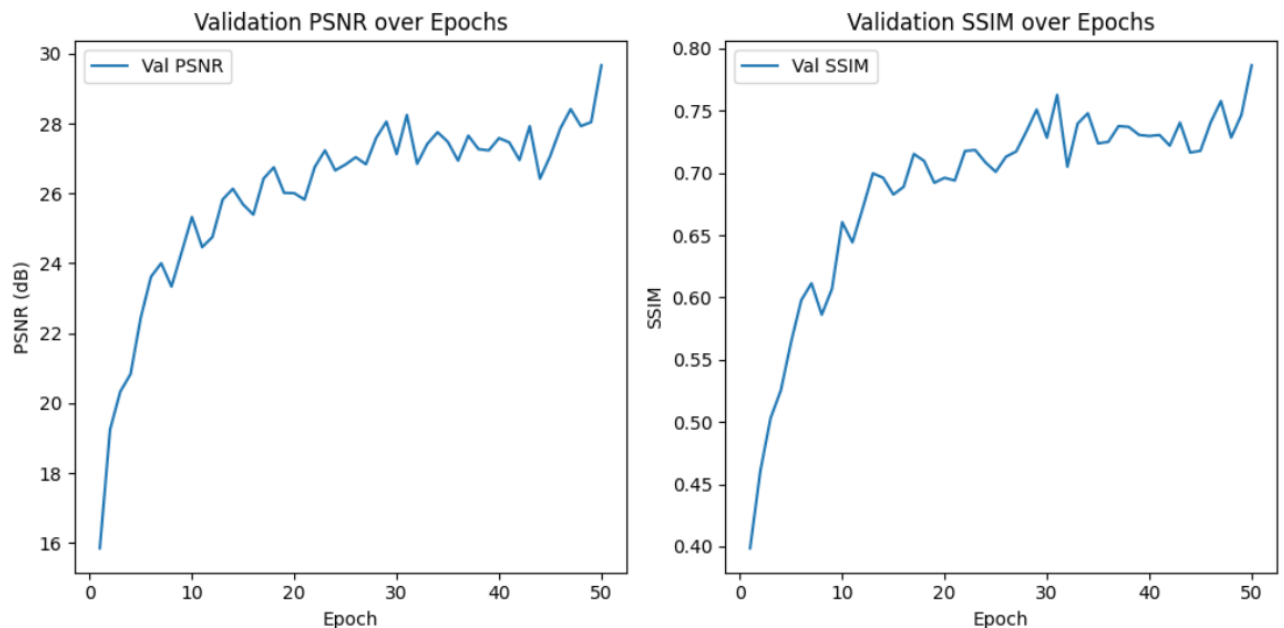


We printed the values of the three criteria every 10 epoch:

	Epoch	PSNR	SSIM	FID
0	10	23.868413	0.617302	307.365980
1	20	26.767133	0.706533	279.923050
2	30	28.463374	0.751977	254.213479
3	40	26.537657	0.727512	255.996600
4	49	27.560224	0.734378	223.127739

- A clear improvement trend is observed from epoch 10 to epoch 30, both in PSNR and SSIM. The PSNR increases from 23.87 to 28.46 – indicating that the average pixel-wise error between the super-resolved (SR) image and the high-resolution (HR) ground truth has significantly decreased.
- Similarly, SSIM improves from 0.617 to 0.752, reflecting better preservation of structural and perceptual details such as textures and edges. Both metrics suggest that the model is learning to reconstruct more accurate and visually faithful images over time.
- A consistent decrease in FID shows that the model's output images are becoming more natural and realistic.

We can see from the graphs that there is a significant improvement as the number of epoch increases.

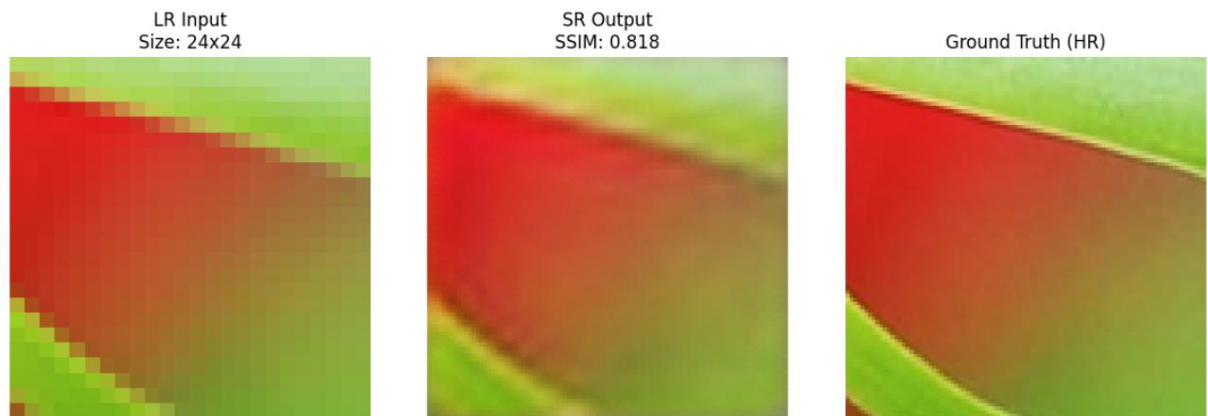


2. Qualitative Examples (after 50 epoch of training):

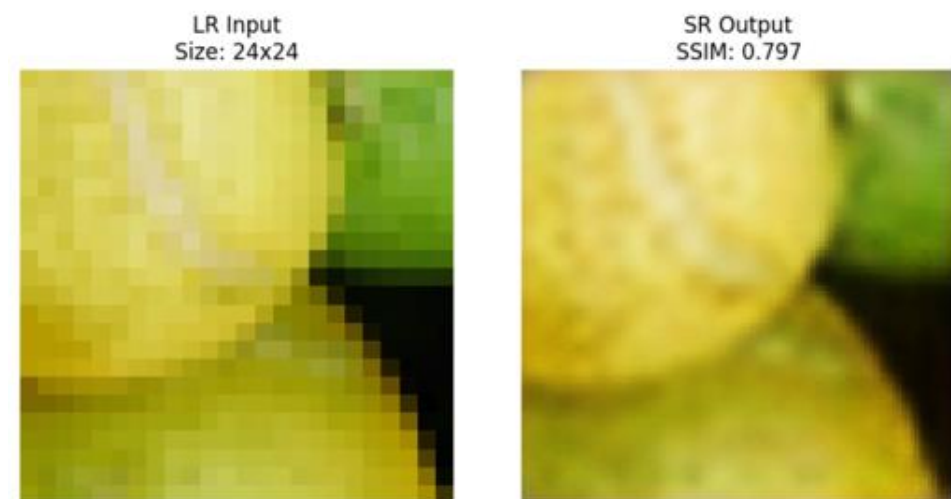
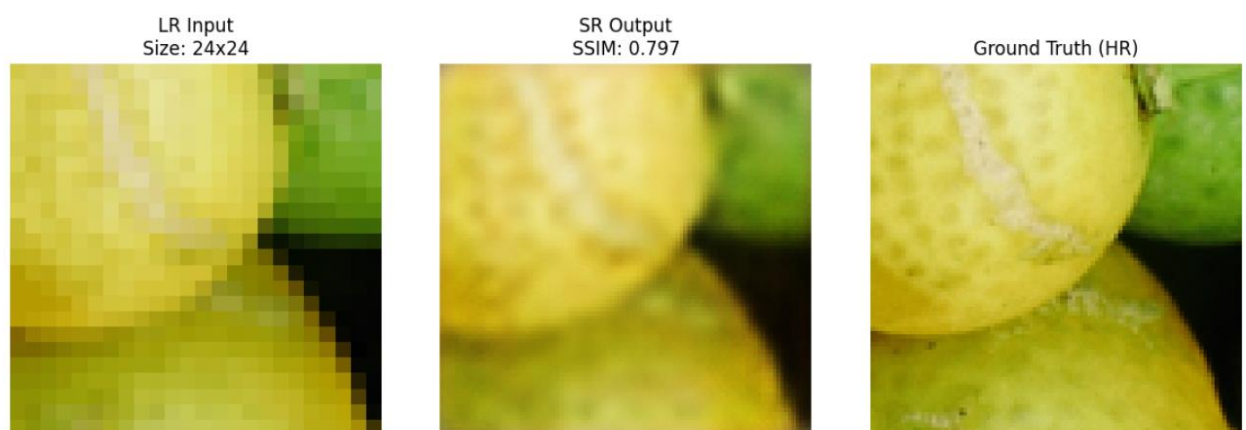
SSIM was chosen as the primary criterion for comparison because it measures structural similarity between the reconstructed image and the original—that is, how well important details such as contrast, structure, and texture are preserved, in a way that is close to how humans perceive image quality. Unlike simple metrics such as **PSNR** that only measure pixel-by-pixel error, **SSIM** provides a more qualitative metric that is suitable for subjective image quality assessment.

Showing two random crop of two images, one with a poor **SSIM** criterion and the other with a better **SSIM** criterion:

We see **below** that there is an improvement in the crop image of the first image, we got a value of **0.818**. You can see how the model performs gradual improvement, until it produces images that are very close to the original ones.

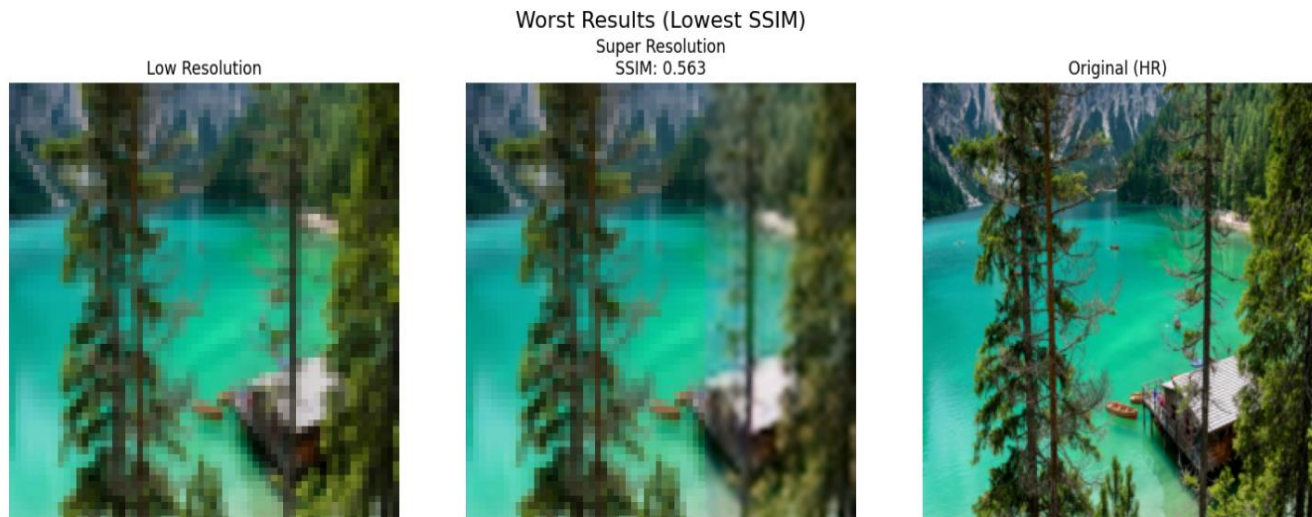


In the second image, the value of the **SSIM (0.7)** is less good than the first, but nevertheless we see a small improvement. We see that the blur started to disappear.

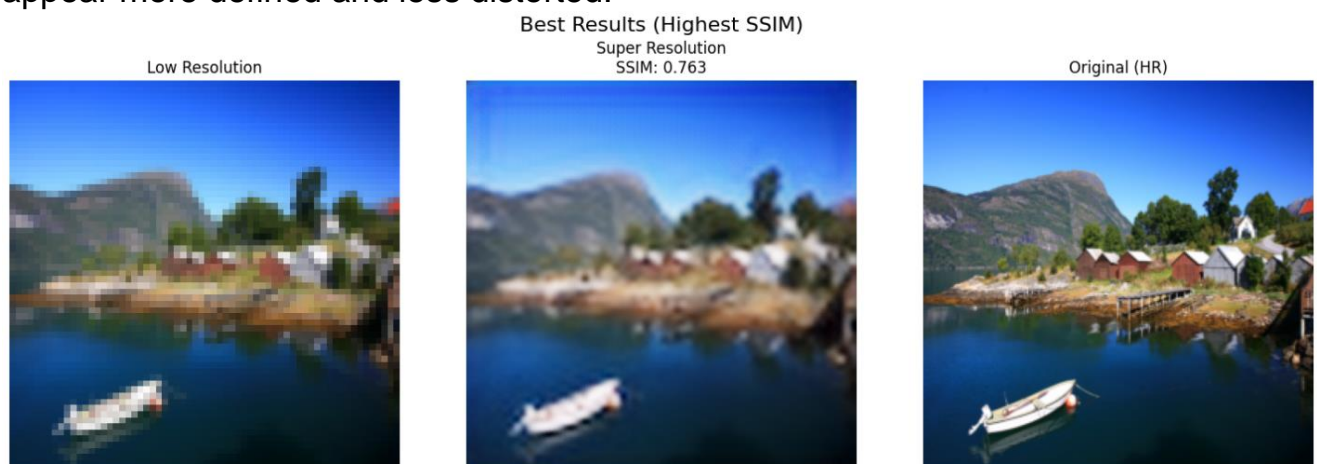


Showing two images, one with a poor **SSIM criterion** and the other with a better **SSIM criterion**:

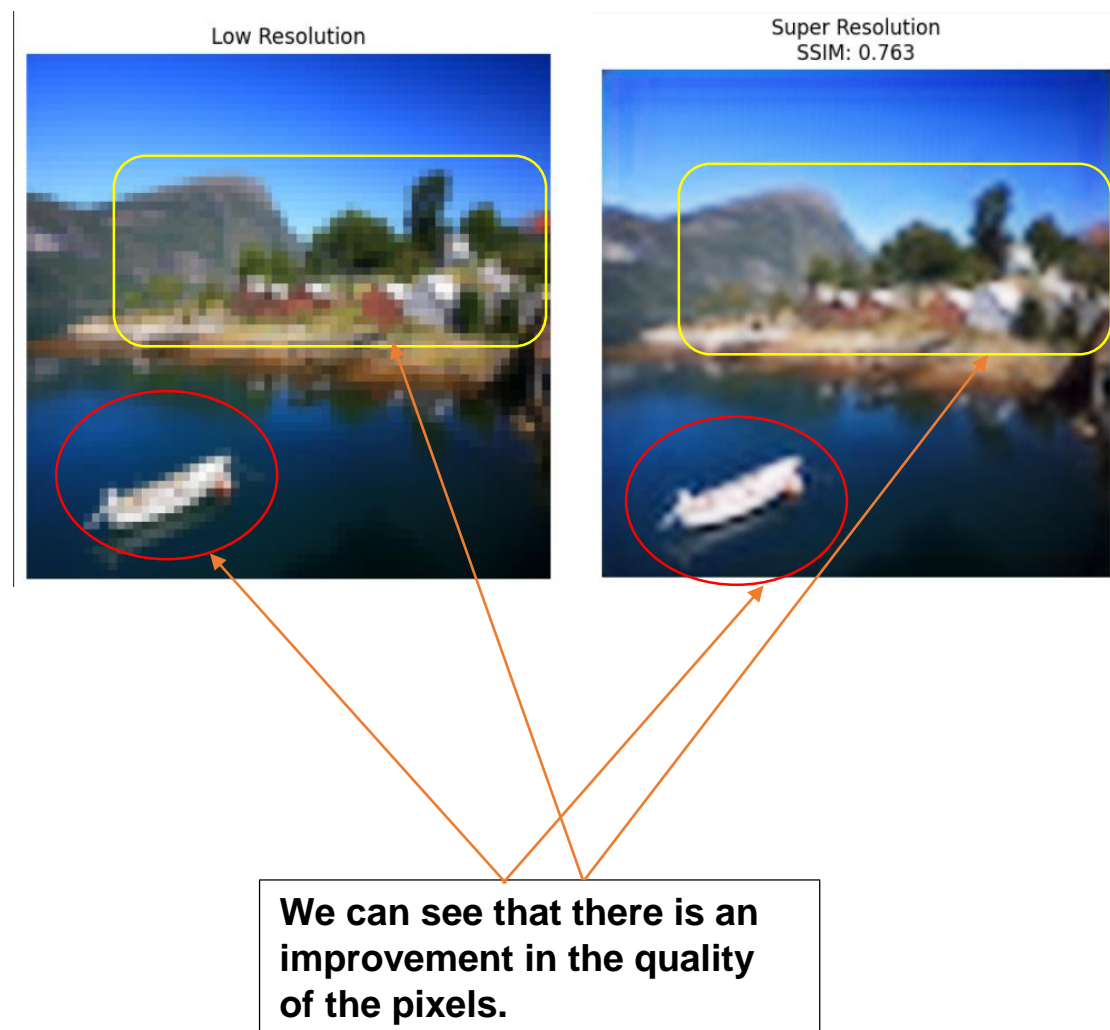
The first image we did not get a very good **SSIM (0.5)** value, so as you can see, the image in the middle is the image created by the model, it doesn't look that good, almost the blur remains.



The second image achieved a relatively high SSIM score of 0.763, indicating a significant improvement in perceptual quality. As illustrated, the model-generated image (center) exhibits noticeable enhancements over the low-resolution input (left). Specifically, the blurring has been substantially reduced. This improvement is clearly visible in finer details such as the boat, where the edges appear sharper, as well as in the background structures like the houses and mountains, which now appear more defined and less distorted.



ZOOM-IN:



3.5.1) We built the model from scratch, but it is based on the principles of the EDSR model – which is an improved version of SRResNet. That is, our model was built manually, but it is also an improvement on SRResNet using deeper residual blocks than SRRESNET, removing BatchNorm, and using PixelShuffle to learn the augmentation in a smart way. Plus, using adaptive Learning Rate.

Architecture structure:

- **Input:** Low-Resolution image (24x24).
- **Output:** High-Resolution image, 4 times the spatial size of the input (96x96).

0. NormAndDeNorm (Normalization / Denormalization Layer):

- Conv layer with kernel size 1×1 , 3 input channels (RGB), and 3 output channels.
- Weights and bias are fixed (not learnable) – based on the RGB mean and std values.
- Used twice in the model:
 - sign = -1: Before entering the network – normalization of input.
 - sign = +1: After the output – denormalization back to image space.
- Goal:
 - Helps stabilize training by ensuring the pixel values are centered and scaled properly.
 - Returns the output image to the correct dynamic range for viewing and evaluation.

1. conv_block1 (Initial Feature Extraction):

- Conv Layer: kernel size 3×3 , input channels = 3 (RGB), output n_channels (256).
- Activation: None (as in original EDSR).
- Goal: Extract low-level features from the input LR image.

2. Residual Blocks (x32 Blocks):

- Each block consists of: Conv \rightarrow ReLU \rightarrow Conv (all with kernel size 3×3 , padding=1).
- Skip connection: Each block adds its input to its output:
Output = $F(x) + x$.
- BatchNorm: Removed, to avoid artifacts and preserve dynamic range. BatchNorm compresses the dynamic range of values (light-dark range). As a result, this creates a loss of small details, and sometimes also unwanted artifacts (such as noise, unnatural blur).
- Goal: Learn small but deep residual corrections without vanishing gradients, the residual connection (skip connection) helps prevent vanishing gradients.

3. conv_block2 (Post-Residual Conv Layer):

- An additional Conv layer with kernel size 3×3 that is added back to the original features before the Residuals (Connects to the skip connection that came from conv_block1).
- Input/output = n_channels(256)
- Goal: Reinforces the original information that passed directly without changes.

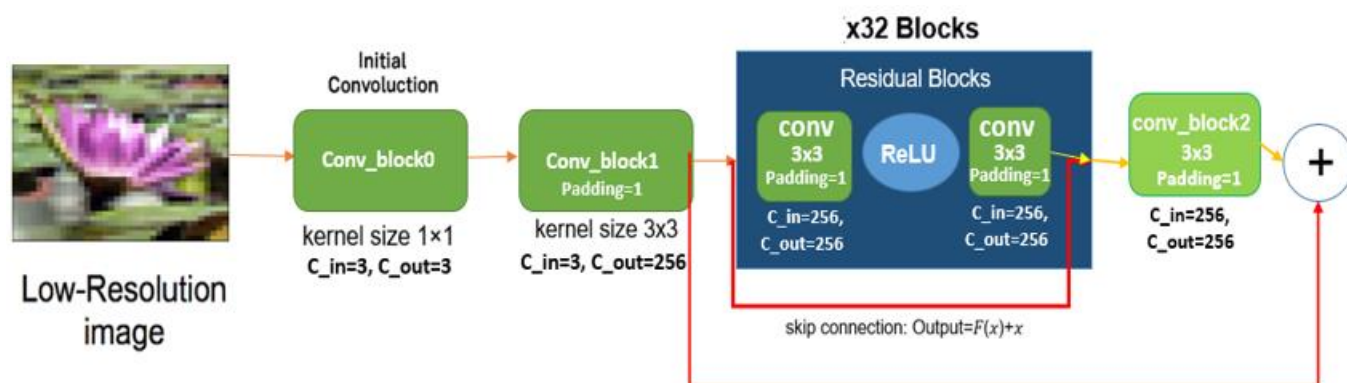
4. SubPixel Convolution Blocks – Upsampling Layers

- Two subpixel (PixelShuffle) blocks, each upscales by ×2 (total×4).
- Before each PixelShuffle: Conv layer that expands the number of channels appropriately.
- Activation: None (follows EDSR which omits PReLU in these layers).
- Goal: Learn how to upsample efficiently and accurately using learnable filters.

5. conv_block3 – Final RGB Reconstruction

- Conv layer with 3 filters (RGB), kernel size 3×3 that returns back to a normal size image (3 channels, RGB).
- Goal: Converts the features back to a super-resolution image.
- Output the high-resolution image.

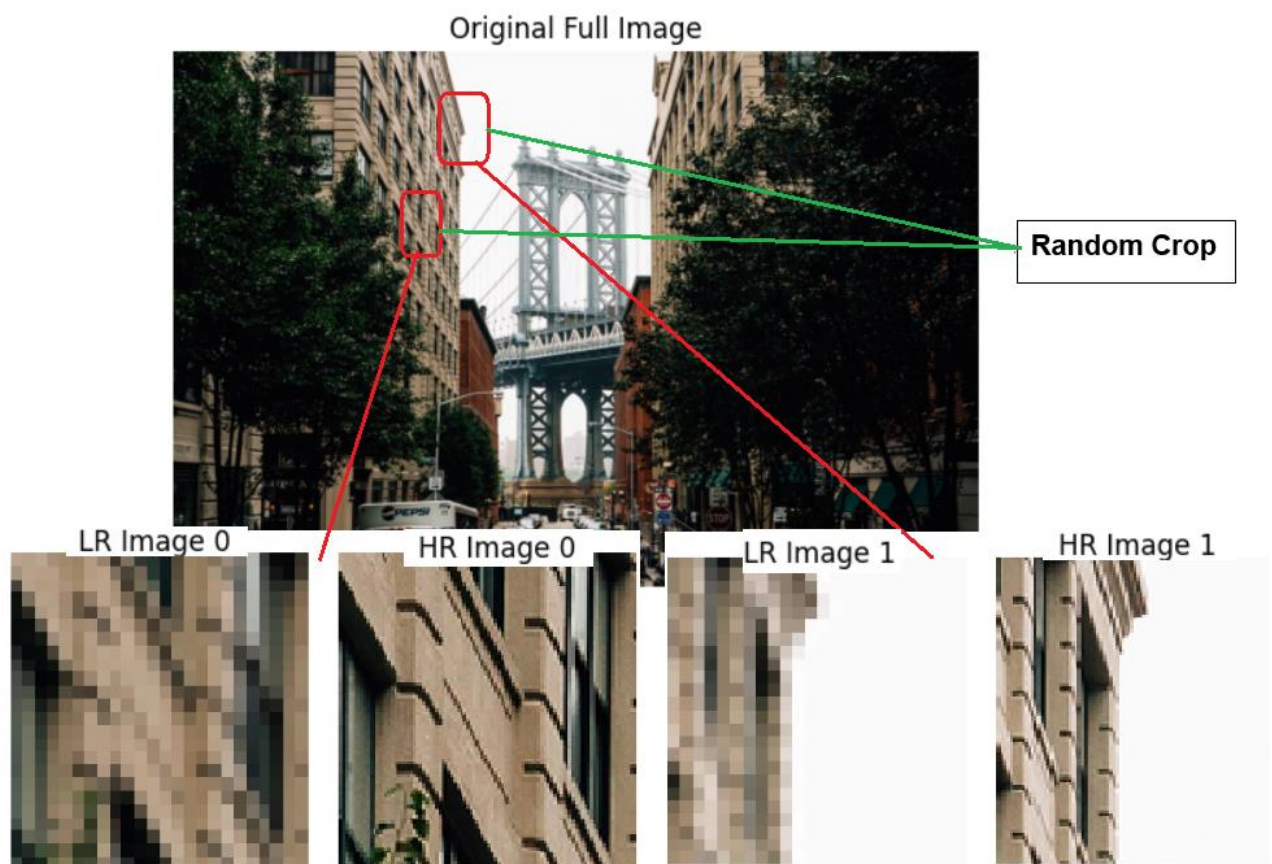
3.5.2) diagram illustrating the model structure:



increases variability and focuses the model on local textures and patterns.

- Downsampling: Each HR crop is resized by a factor of 4 using bicubic interpolation to generate the corresponding low-resolution (LR) image (24×24). This simulates the degradation process.
- Normalization to [0, 1]: Both HR and LR images are converted to float32 and divided by 255 to scale pixel values to the range [0,1].
- Per-Channel Normalization: A 1×1 convolution layer (NormAndDeNorm) further normalizes the input using the mean RGB values of the DIV2K dataset: (0.4488, 0.4371, 0.4040), centering the distribution and helping model convergence.
- Conversion to Tensor: Images are transposed from HWC (height, width, channel) format to CHW and converted to PyTorch tensors.
- Batching & Shuffling: During training, data is loaded in shuffled batches using a PyTorch DataLoader to improve generalization.

Visualizations: Original image, HR and LR



3.5.4) Training Regime, Hyperparameters and Loss Function:

3.5.4.1) Training Regime:

- **Loss Function:** L1 Loss (Mean Absolute Error).
- **Optimizer:** Adam optimizer with initial learning rate of $1e-4$ was chosen for its fast convergence properties.
- **Learning Rate Scheduler (Adaptive LR):** A step scheduler was applied: the learning rate is halved every 10 epochs to help with stability and convergence.
- **Training Duration:** The model was trained for 50 epochs using batches of 32 images.
- **Validation Metrics:** After each epoch, the model was evaluated on:
 - **PSNR** (Peak Signal-to-Noise Ratio) – assesses reconstruction quality.
 - **SSIM** (Structural Similarity Index) – measures structural similarity to ground truth.
 - **FID** (Fréchet Inception Distance) – evaluates perceptual quality.
- **Checkpointing:** The best model (based on PSNR and Loss together) was saved for later inference.

3.5.4.2) Hyperparameters:

1. Model Hyperparameters:

- **Scale = 4**, Super-resolution magnification factor (x4).
- **crop_size = 96**, Crop size from high-resolution images.
- **n_resblocks = 32**, Number of residual blocks in our model.
- **n_channels = 256**, Number of features in each convolution.

2. Training Hyperparameters:

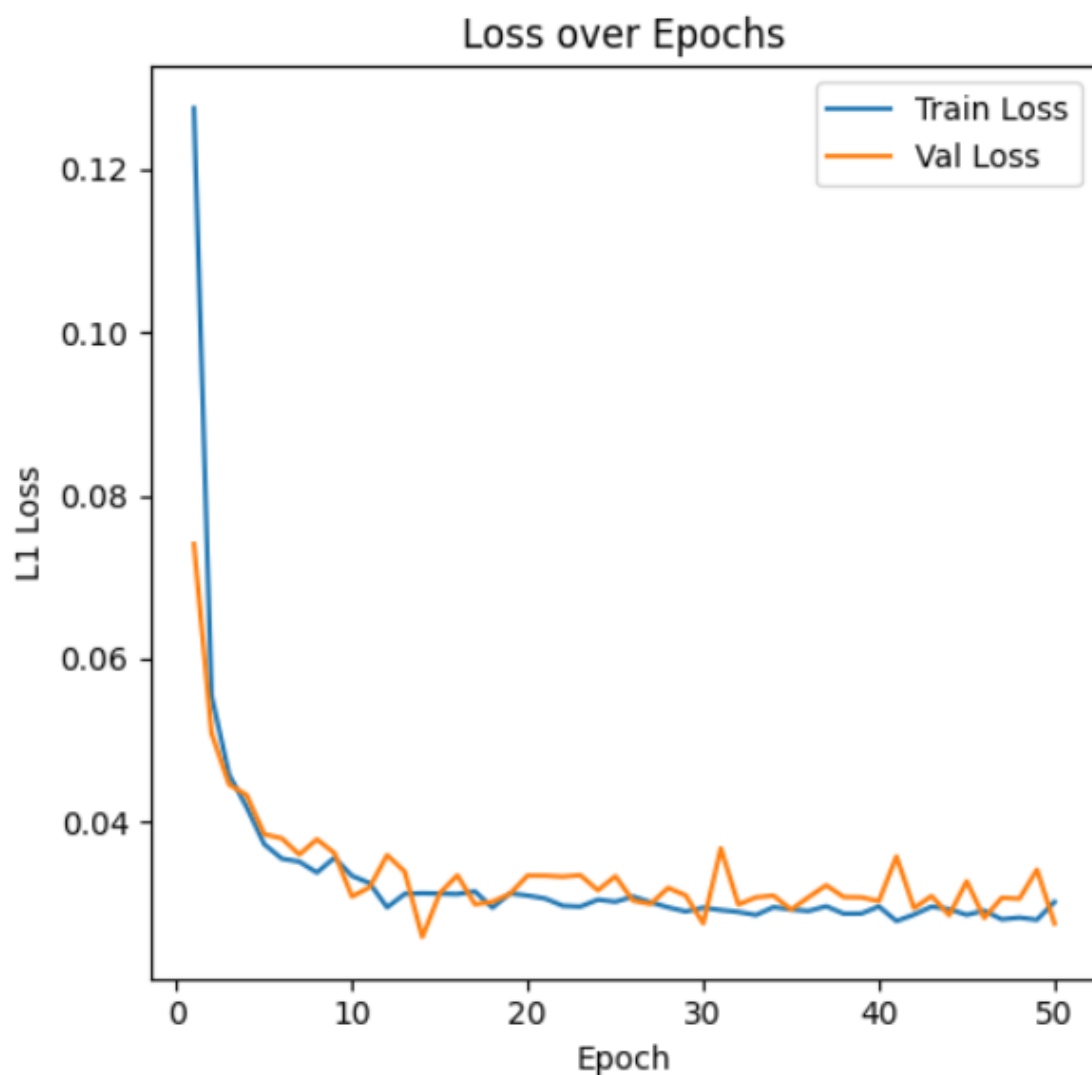
- epochs = 50, number of epochs in training.
- batch_size = 32, Batch size for the dataloader.
- learning rate = $1e-4$, Learning rate of the optimizer
- optimizer: Adam Common optimizer for training neural networks
- criterion L1Loss(). Loss function mean Absolute Error.
- scheduler StepLR (step_size=10, gamma=0.5) Every 10 epochs reduces the learning rate by half.

3.5.4.3) Loss Function:

L1 Loss (Mean Absolute Error) was used as the main loss function due to its ability to preserve fine image details better than MSE. L1 Loss is a loss function that measures the average of the absolute values of the differences between the reconstructed image (SR) and the original high-resolution image (HR).

3.5.5) Training Loss Over Epochs:

The graph shows a steady decrease in training loss over time, indicating that the model is learning effectively and converging as training progresses. The small gap between the Train Loss and Valid Loss also shows that the model is able to generalize well to data it has not seen.



3.5.6) Presentation of quantitative metrics and qualitative examples showing both successful and failed predictions:

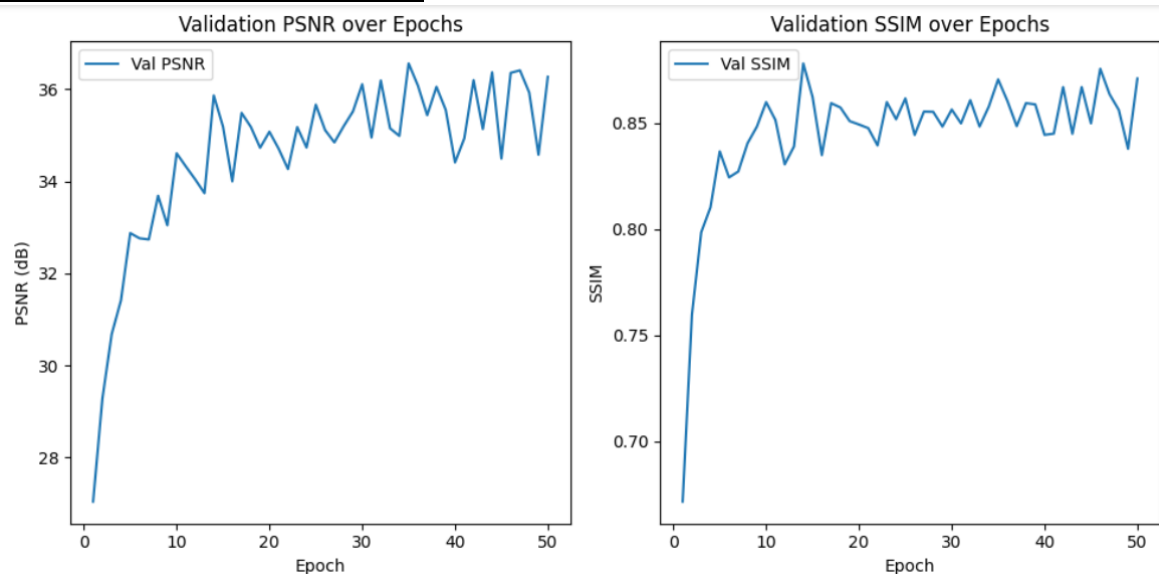
1. Quantitative Evaluation Metrics (with 50 epoch of training)

We printed the values of the three criteria every 10 epoch:

	Epoch	PSNR	SSIM	FID
0	0	27.042374	0.671461	314.184155
1	10	34.317708	0.851470	191.138865
2	20	34.703767	0.847639	166.156181
3	30	34.950142	0.849879	174.559980
4	40	34.929528	0.845058	174.072374
5	49	36.266410	0.871029	173.079152

- The table shows evaluation metrics on the validation set across different epochs. We observe that as training progresses, the model achieves:
- Higher PSNR (Peak Signal-to-Noise Ratio), indicating better reconstruction fidelity.
- Higher SSIM (Structural Similarity Index), indicating better structural similarity to the ground truth.
- Lower FID (Fréchet Inception Distance), suggesting the generated images are more perceptually realistic.
- We see that by epoch 49, the model achieves its best performance with: PSNR: 36.26 dB, SSIM: 0.871, FID: 173.07

As we see from the graphs, there is a significant improvement as the number of epoch increases:



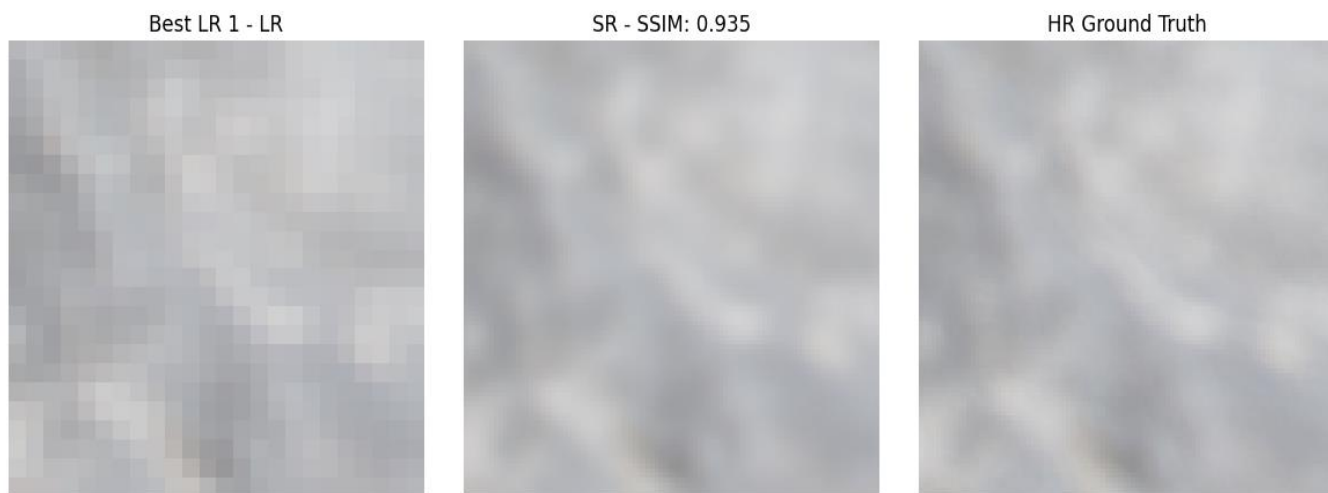
2. Qualitative Examples (after 50 epoch of training):

Showing two random crop of two images, one with a poor **SSIM** criterion and the other with a better **SSIM** criterion:

In this crop image example, we see three versions of the same image:

- **Left: The Low Resolution (LR) input Crop** - a heavily downsampled image ($\times 4$ smaller), where much of the detail is lost.
- **Middle: The Super-Resolution (SR) output Crop** - generated by our model.
- **Right: The High Resolution (HR) ground truth Crop** - the original, high-quality image.

We can clearly see that the SR Crop image in the middle is visually very close to the HR Crop image on the right. Fine details, edges, and textures are well preserved. Compared to the blurry LR Crop image on the left, there's a significant improvement in sharpness and structure. This is also reflected in the high SSIM value of 0.935, indicating strong structural similarity to the original



Looking at these three images below:

- **Left: The Low Resolution (LR) input Crop** - blurred and lacking detail.
- **Middle: The Super-Resolution (SR) output Crop** - generated by the model.
- **Right: The High Resolution (HR) Crop** - ground truth image.

We can immediately notice that the **SR image Crop in the middle is still blurry**, and **does not show a meaningful improvement** over the LR input Crop. The fine details and textures from the HR image Crop on the right are **still missing**, and **edges remain soft or undefined**. There is **no clear visual enhancement**, and in some areas, the SR image Crop looks almost the same as the LR image Crop.

This poor reconstruction is reflected in the **low SSIM value of 0.430**, indicating that the model **failed to accurately recover the high-frequency information** from the original image.



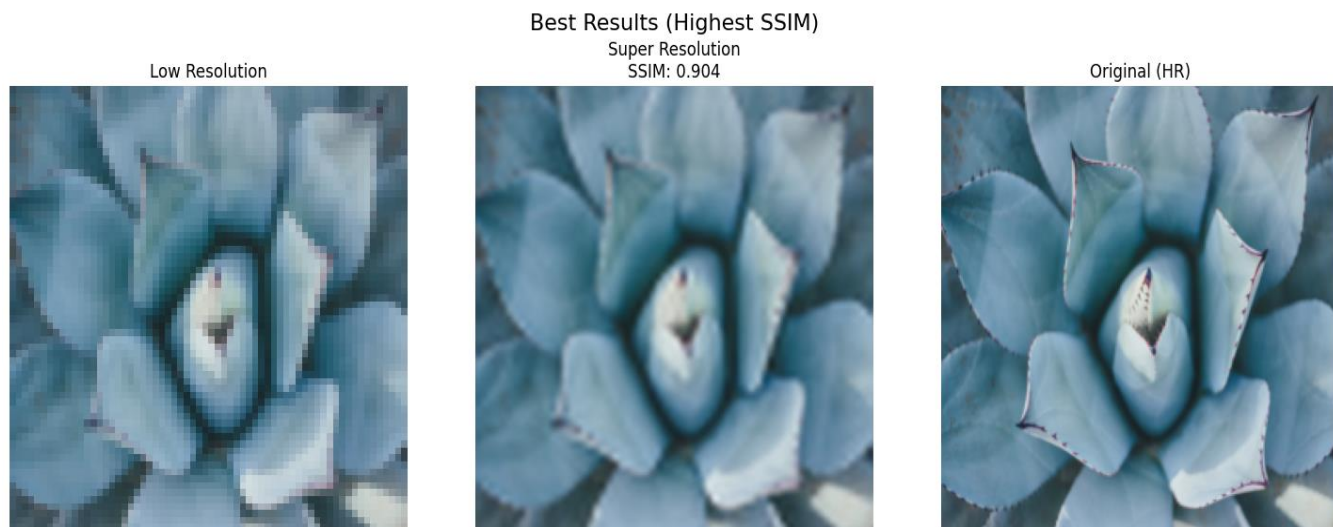
Showing two images, one with a poor **SSIM criterion** and the other with a better **SSIM criterion**:

In this image example, we see three versions of the same image:

- Left: The Low Resolution (LR) input: a heavily downsampled image ($\times 4$ smaller), where much of the detail is lost.
- Middle: The Super-Resolution (SR) output generated by our model.
- Right: The High Resolution (HR) ground truth: the original, high-quality image.

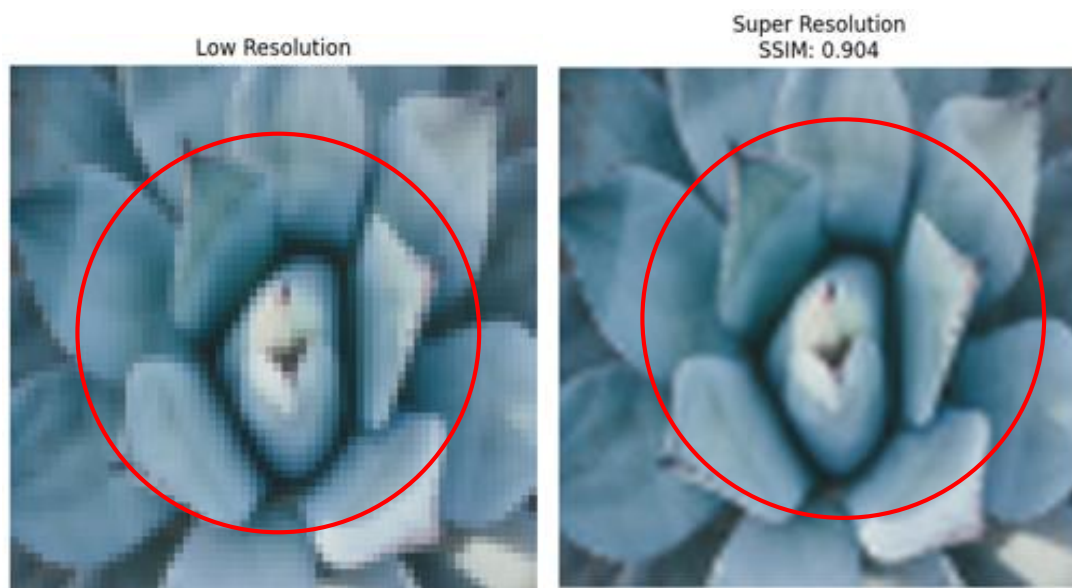
We can clearly see that the SR image in the middle is visually very close to the HR image on the right. Fine details, edges, and textures are well preserved. Compared to the blurry LR image on the left, there's a significant improvement in sharpness and structure. This is also

reflected in the high SSIM value of 0.904, indicating strong structural similarity to the original



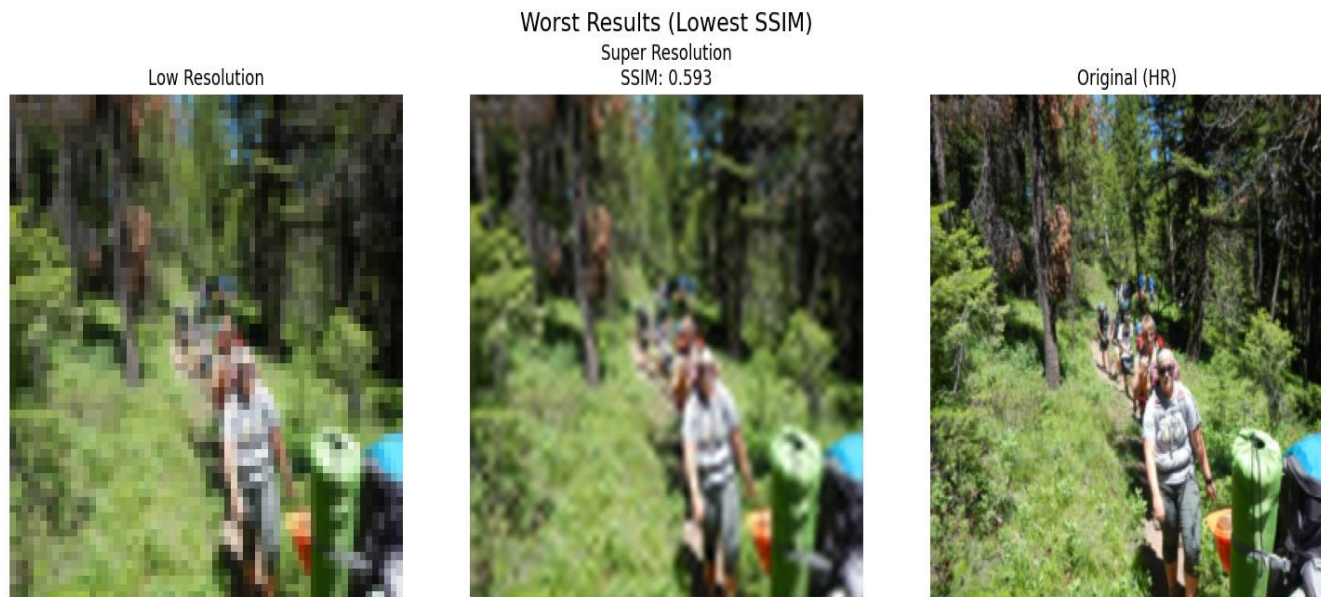
In the SR image (middle), we can observe that the blurriness in the central region — which is highlighted by the red circle — starts to fade and the details become noticeably sharper. This area begins to resemble the original high-resolution image on the right, rather than the blurry low-resolution input on the left.

It shows that the model was able to partially recover lost information, bringing the SR output closer to the ground truth



Although the SR image shows some noticeable improvement over the LR input, the result is still not close enough to the original HR image.

Fine details and textures remain partially missing, and edges are only moderately enhanced. With an SSIM value of 0.59, this reconstruction is considered moderate or suboptimal, indicating that the model made progress but still struggled to recover full image fidelity.



3.6.1) Results:

3.6.1.1) One examples of images where both models performed well:

in the visual comparison below, we can see that both models performed well, as we see our model produced an image with clearer textures and sharper edges compared to SRResNet. This is also reflected in the SSIM score: our model achieved a higher SSIM (0.907), indicating better structural similarity to the original HR image.

As we can clearly see, the blurriness that remains in the SRResNet output (left) is significantly reduced in our model's output (middle). Edges are sharper, textures are better restored, and overall the result looks more natural and closer to the HR image (right).

This visual improvement is reflected in the SSIM score, where our model outperformed SRResNet.

These results were achieved thanks to architectural modifications such as removing BatchNorm, using a 1×1 normalization layer at the input, and training with an adaptive learning rate.



3.6.1.2) One examples of images where both models performed poorly:

In the two images below, it can be clearly seen that the overall quality is still not good — in both there are noticeable blurs and the details are not sharp enough. That is, visually, both results suffer from a loss of sharpness and appear to be of lower quality.

If we examine the value of the SSIM (Structural Similarity Index) index, our model achieves a significantly higher result — about 0.3 points more than the competing model. This means that although the visual difference is not yet optimal, in terms of preserving the structure and features of the original image, our model manages to reconstruct an image with a better resemblance to the original according to this quality criterion.

In conclusion, although the visual improvement is still not impressive, the SSIM quality index indicates a clear advantage for our model in reconstructing structural information in the image.



3.6.1.3) One examples of images where Model 1 performed well and Model 2 did not: None.

During our evaluation, we did not find any test images where the baseline model (SRResNet) significantly outperformed our model in terms of SSIM by a large margin. This indicates that our model consistently matches or surpasses the baseline in reconstructing fine image details and structural similarity. as we said the improvement is mainly due to the architectural changes we made.

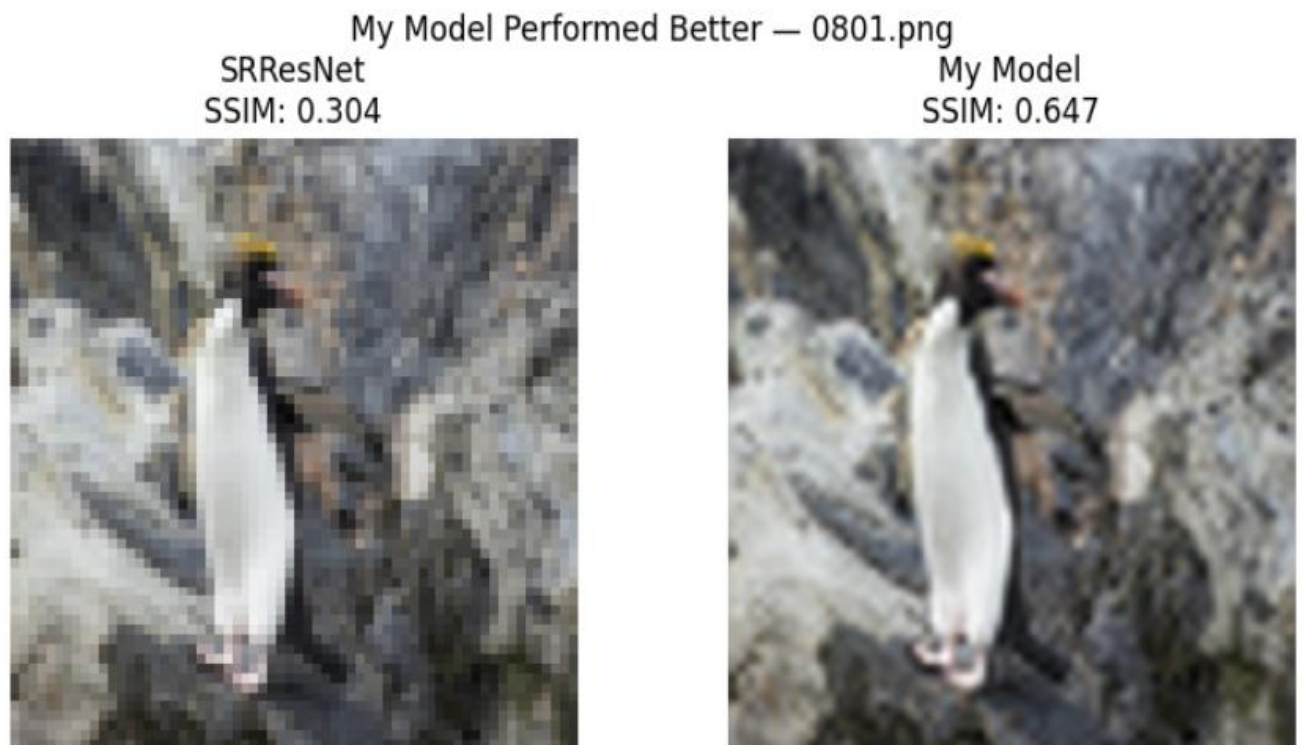
3.6.1.4) One examples of images where Model 2 performed well and Model 1 did not:

Our model consistently produces sharper and more detailed super-resolution images compared to SRResNet. In these cases, the model is better at preserving edges, textures, and fine structures, which translates into higher SSIM scores and better visual quality. As we said before the improvement is mainly due to the architectural changes we made.

When looking at the two images below, it is clear that the image on the right (which was captured by our model) has less blur, sharper lines,

and better details. In contrast, the image on the left (from the first model) appears more blurry and less accurate in details.

This difference demonstrates the ability of our model to reconstruct higher-quality, clearer images, with better preservation of textures and fine details.



3.6.2) Our variation, based on an enhanced SRResNet architecture (it's like EDSR), outperformed the baseline SRResNet model in several aspects — including higher SSIM, PSNR scores and visually sharper reconstructions with more realistic textures and cleaner edges. Also, we get a low FID score, which indicates that the distribution of generated images is closer to the distribution of real high-resolution images. This suggests that our model not only preserves structural similarity (as shown by SSIM), but also captures more realistic texture and perceptual quality, as reflected by the FID metric.

Key improvements in our model make our variation outperformed the baseline model:

- The first layer in our model is a lightweight 1x1 convolutional layer used for input normalization, helping the network adapt to the scale and statistics of the data in a simple and efficient way.
- Removal of all normalization layers (BatchNorm) which prevents distortion in the feature distribution and helps retain important low-frequency and high-frequency details.
- No activation functions (such as ReLU) inside the residual blocks - this promotes smoother information flow and minimizes undesired nonlinear effects.
- Deeper architecture with more residual blocks and wider channels, enhancing the model's capacity to learn fine structures.
- Use of residual scaling to stabilize training despite the increased model depth and complexity.
- Adaptive learning rate scheduling — our training strategy includes dynamic adjustment of the learning rate, which allows the model to converge more efficiently by lowering the learning rate when progress stalls.
- While SRResNet performed well in simpler cases, our model demonstrated superior performance in more challenging and detail-rich regions.

Quantitative Performance comparison between our model and the first model (SRResNet):

- As can be seen in the two tables below, our model achieves better performance in all the criteria tested.
- In the PSNR metric, in every 10 epochs our model receives significantly higher values than the first model, indicating an improvement in the reconstruction quality.
- In the SSIM metric, every and throughout all 10 epochs, our values are closer to 1, reflecting a better structural similarity to the original image compared to the first model.
- In the last metric (probably FID or a similar metric), our model receives lower values, indicating better quality results with less deviations or noise.
- Overall, these data demonstrate that our model consistently improves the quality of reconstructions and leads to better results than the baseline product.

Baseline model table:

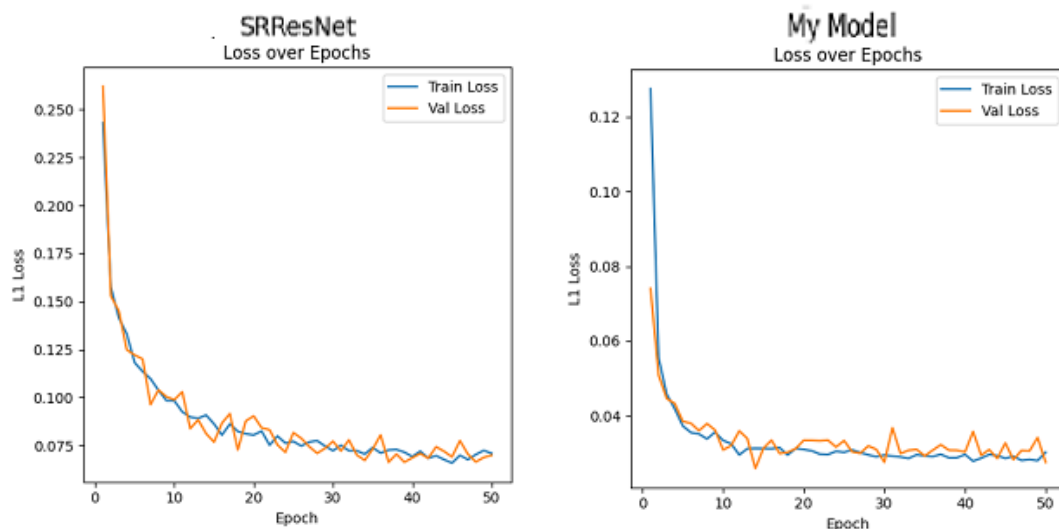
	Epoch	PSNR	SSIM	FID
0	10	23.868413	0.617302	307.365980
1	20	26.767133	0.706533	279.923050
2	30	28.463374	0.751977	254.213479
3	40	26.537657	0.727512	255.996600
4	49	27.560224	0.734378	223.127739

Our model table:

	Epoch	PSNR	SSIM	FID
0	0	27.042374	0.671461	314.184155
1	10	34.317708	0.851470	191.138865
2	20	34.703767	0.847639	166.156181
3	30	34.950142	0.849879	174.559980
4	40	34.929528	0.845058	174.072374
5	49	36.266410	0.871029	173.079152

Comparison of convergence graphs: our model versus the baseline model:

As we see in the graphs, our model converges faster and reaches lower loss values in both training (train value < 0.04) and validation (validation value < 0.04) in contrast to baseline model which reaches loss values in both training and validation Approximately 0.075. In addition, in the two models the validation graph shows higher stability, the Training Loss and Validation Loss graphs converge in a similar way, meaning they advance together and decline at the same rate without a large gap between them, which is a positive indication that the model is not overfitting and that it is learning in a general way (generalization).



Qualitative Performance comparison between our model and the first model (SRResNet):

- In the two Crop images below, you can clearly see that our model reproduces images with better quality.
- The result of our model is less blurry, with sharper and clearer lines and shapes.
- The reproductions look more natural and accurate, which significantly improves the viewing experience.
- Compared to the first model, our model manages to preserve fine details and create a sharper image.

SRResNet



My Model



SRResNet



My Model

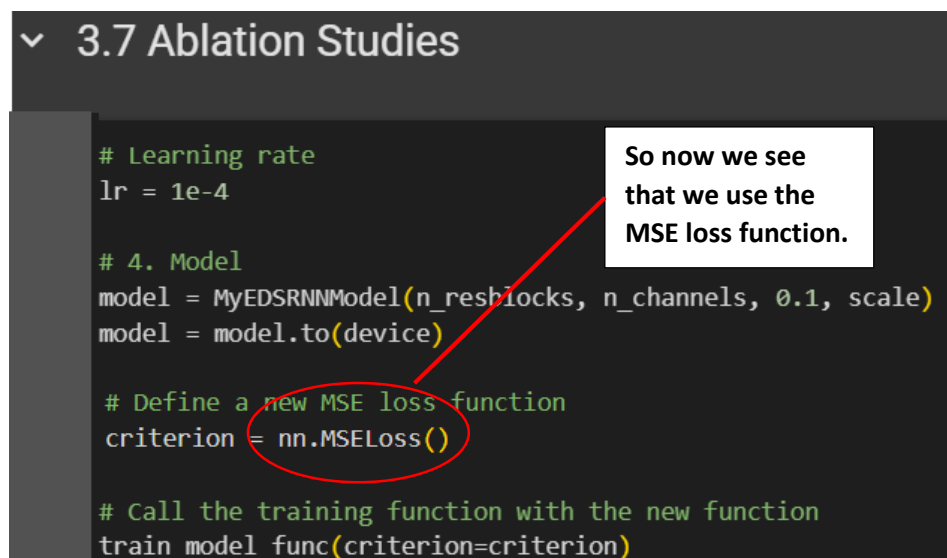


3.7) Ablation Studies:

3.7.1) Ablation Setup:

- In the ablation experiment, we chose to examine the effect of the loss function on the model's performance. Our original model used L1Loss, while in the "ablated" version we switched to MSE-Loss.
- The goal is to test how this change affects the model's performance, in terms of reconstruction quality, PSNR, SSIM, and FID metrics, as well as the convergence of the loss during training.
- All other model components and parameters remained the same, so the impact of the change is only due to the replacement of the loss function.
- The motivation for the change was to test whether a loss function that is more sensitive to large changes in pixels (such as MSE) would affect the model's ability to reproduce fine details, compared to L1, which is characterized by low sensitivity to noise and is considered more "forgiving".

Visualizations: The code snippet We replaced



```
✓ 3.7 Ablation Studies

# Learning rate
lr = 1e-4

# 4. Model
model = MyEDSRNNModel(n_resblocks, n_channels, 0.1, scale)
model = model.to(device)

# Define a new MSE loss function
criterion = nn.MSELoss()

# Call the training function with the new function
train_model_func(criterion=criterion)
```

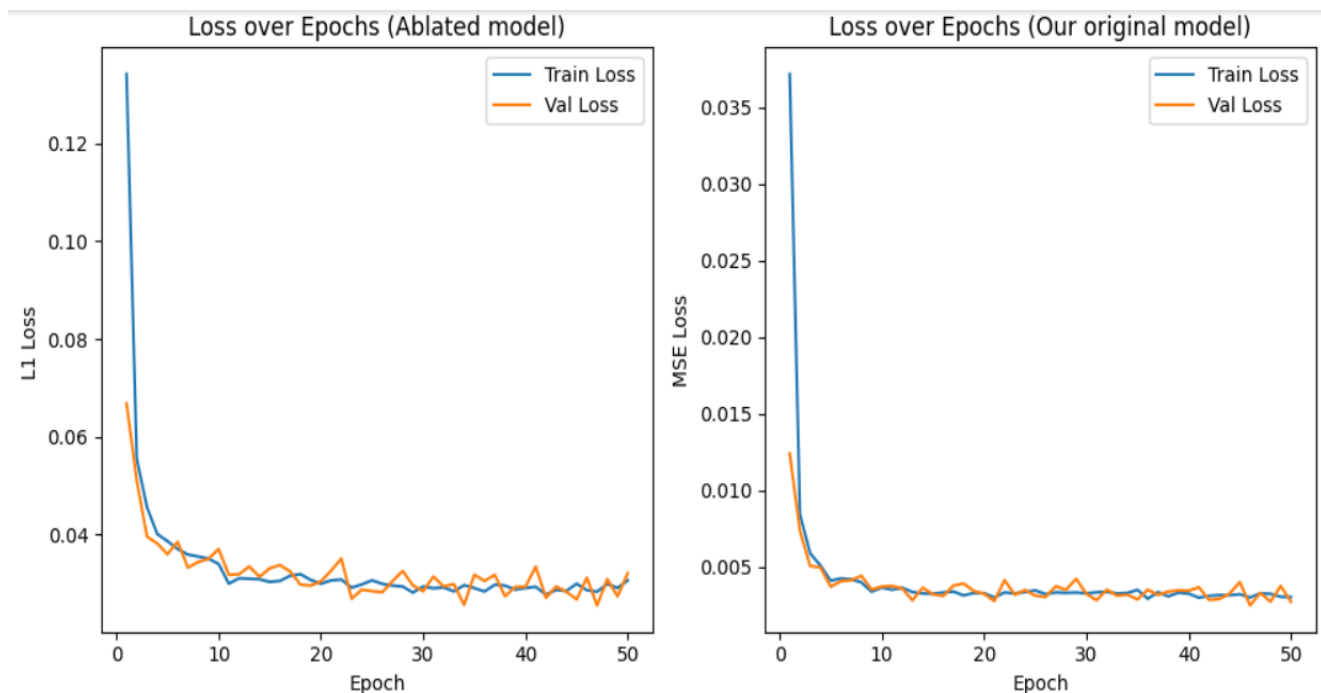
So now we see that we use the MSE loss function.

3.7.2) compare the training loss curves for the original and ablated versions of the model:

- When comparing the Training Loss graphs between the two models - our original model (with an L1 Loss function) and the Ablated model (with MSE Loss) - it can be seen that the graph of the model with MSE Loss converges faster and displays lower

loss values throughout training. That is, in terms of the decrease in the Loss value during the epochs, the model with MSE seems to “learn faster” or adapt to the data more sharply.

- However, it is important to note that MSE tends to penalize larger errors more strongly (because it increases the error by the square), so the rapid convergence may be due to the way it calculates the loss — and not necessarily to a better general ability of the model.
- In contrast, L1 Loss: converges more slowly at times, because its slopes are smaller.
- But it better preserves fine details in the image (such as edges, textures), because it penalizes errors at a constant rate and does not highlight large errors like MSE.



3.7.3) Quantitative performance metrics for each version:

When looking at the performance tables of the **two models below** – the original model (with L1 Loss) and the enhanced model (with MSE Loss) – it can be seen that the values in PSNR, SSIM and FID are very close throughout all the epochs, but nevertheless:

- **PSNR** (Peak Signal-to-Noise Ratio): Although the differences are not large, our model (with L1 Loss) manages to reach higher PSNR values at the end of training, which indicates a more

accurate reconstruction of the pixels compared to the original image.

- **SSIM** (Structural Similarity Index): Also in the SSIM index, the gaps between the two models are limited, but our model (with L1 Loss) shows a slight but consistent advantage, which indicates better preservation of structures and visual content in the image.
- **FID** (Fréchet Inception Distance): In FID values, the gap is more noticeable – our model (with L1 Loss) obtains a lower FID at the end of training, indicating more realistic reconstructed images that are more similar to the original data distribution.

My Original metric model				
	Epoch	PSNR	SSIM	FID
0	0	27.042374	0.671461	301.776321
1	10	34.317708	0.851470	182.514691
2	20	34.703767	0.847639	169.544478
3	30	34.950142	0.849879	177.073004
4	40	34.929528	0.845058	173.787831
5	49	36.266410	0.871029	163.069173

My Ablated metric model				
	Epoch	PSNR	SSIM	FID
0	0	26.587760	0.660607	309.096982
1	10	34.506134	0.838523	195.497272
2	20	35.525945	0.856320	177.879520
3	30	36.230540	0.859952	165.601224
4	40	36.004721	0.858586	173.087264
5	49	35.867668	0.863453	170.268673

3.7.4) Qualitative Comparison and Analysis:

- During the qualitative comparison between the two models – our original model using L1Loss, and the obliterated model using MSE-Loss - a consistency in performance was observed across most examples. In almost all images tested, the two models produced very similar results, with close values in the three criteria: PSNR, SSIM, and FID. Although a small advantage can be seen for our model (L1) in most examples, the differences were mostly negligible and not visually significant.
- In both examples where both models were able to reconstruct the image well, and in examples where both struggled - the reconstruction quality was maintained at more or less the same level. No clear example was found where the obliterated model (MSE) presented a better result than our original model (L1), either in terms of sharpness, accuracy in small details, or overall quality.
- This result reinforces the hypothesis that the L1Loss function is better able to preserve fine details in images, while MSELoss tends to favor "average" and smoother solutions - which may damage sharp details.

**Was there an image where the ablated model performed better?
Can you explain why?**

- No example was found where the ablated model (using MSE-Loss) performed clearly better than our original model (using L1-Loss). In no image was the MSE model superior in terms of sharpness, detail preservation, or quantitative metrics (PSNR, SSIM, FID).
- Generally, both models produced similar results, but our model consistently delivered slightly better results. This supports the hypothesis that L1Loss is better suited for Super-Resolution tasks, as it is less sensitive to outliers and better preserves fine details.

3.7.5) Summary about the insights gained:

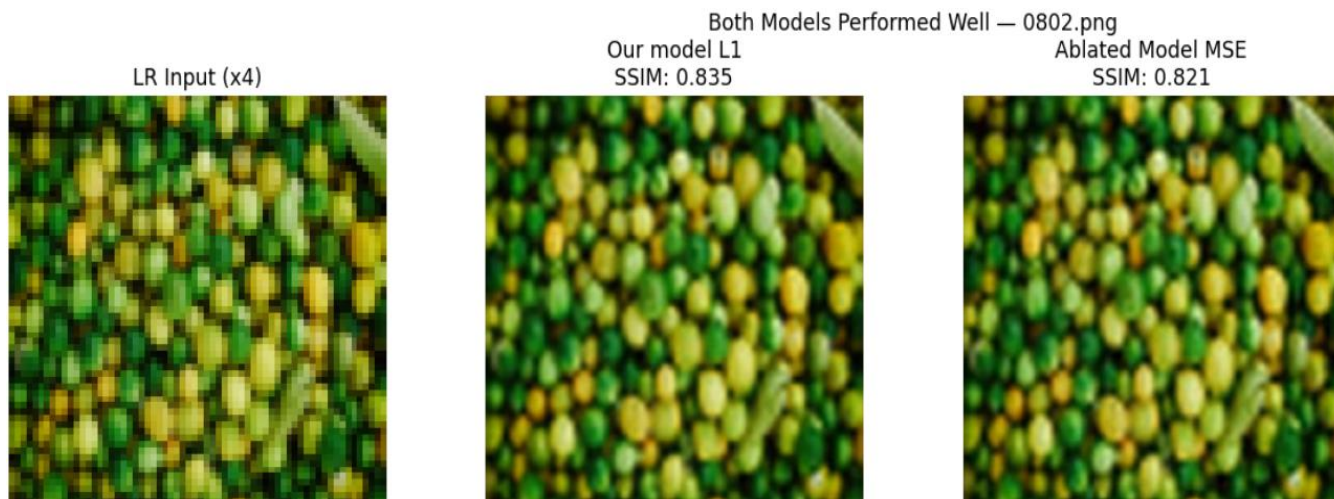
- The experiment we conducted demonstrated the impact of the loss function on the model's performance. When we changed the loss function from L1Loss to MSELoss, we saw that the ablative model (with MSE) converged faster during training and presented lower loss values, but when examining the quantitative performance (PSNR, SSIM, FID) and in the visual examples – the

model with L1Loss provided consistent and better results, even if by a small margin.

- The impact of the loss function is reflected in the level of fine detail preservation: while MSELoss tends to penalize larger errors more strongly and consequently “flatten” the reconstruction, L1Loss preserves edges and details better – which is critical in improving resolution.
- Therefore, although the differences in the numbers were small (sometimes at the level of $\pm 0.00X$), it can be seen that L1Loss has a real contribution in maintaining the quality of the reconstruction, especially in the subtleties of the image structure. However, both models produced quite similar results, so the version with MSELoss can also be considered a reasonable option, depending on the application requirements.

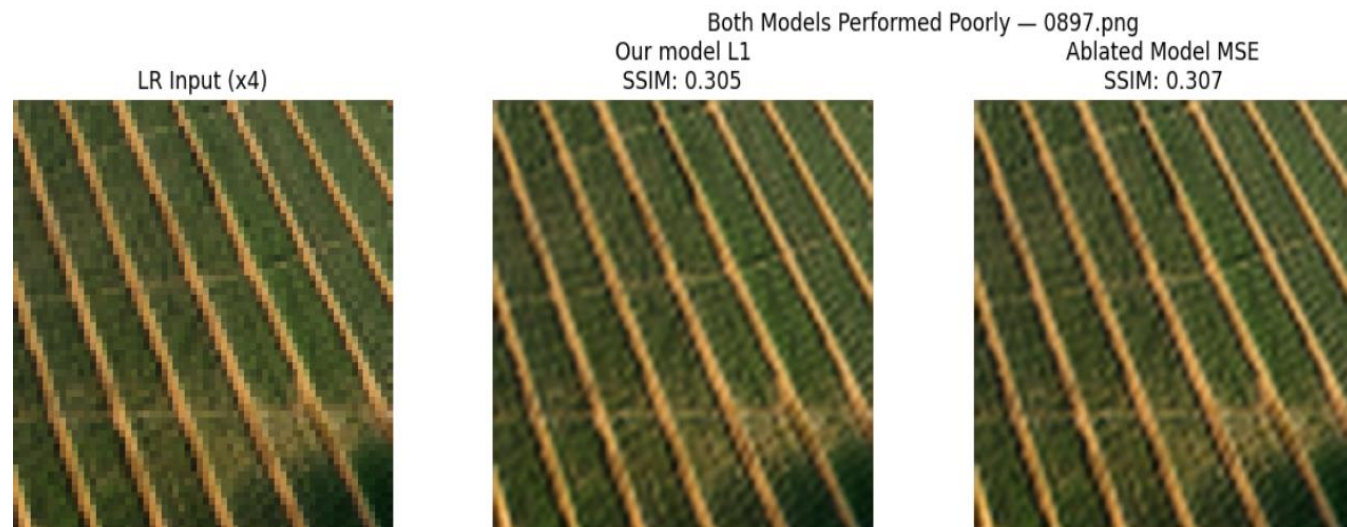
One where both models performed well:

As We can see below, both models were able to reproduce the details in a similar way. The blur in the original image was effectively removed by both models, and they produced results of almost the same quality - both in terms of sharpness and fine details.



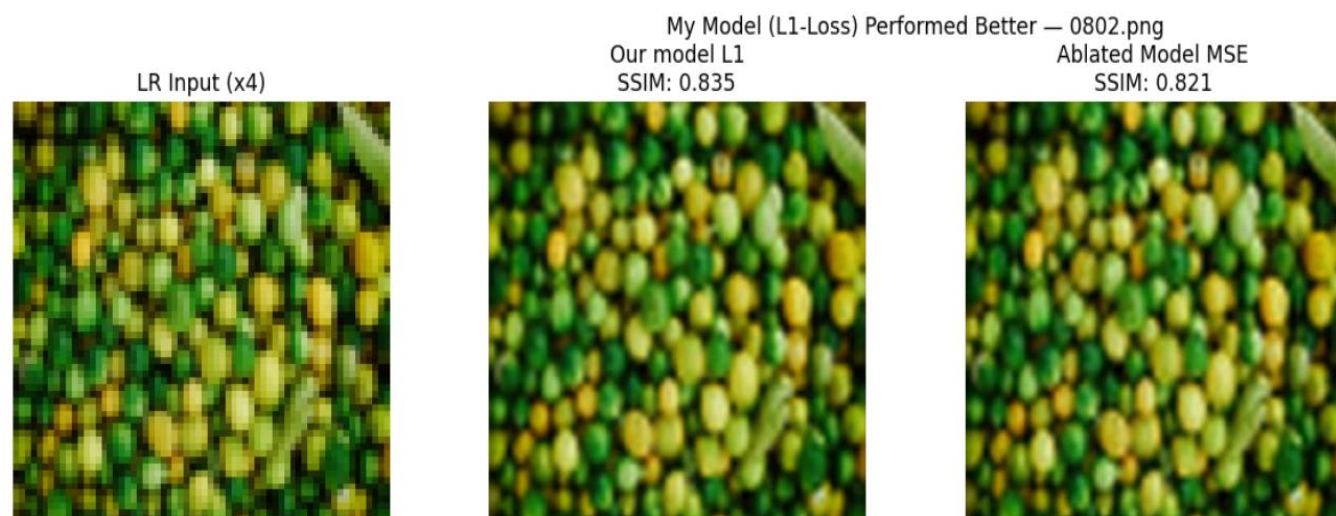
One where both models performed poorly:

In this images below, both models struggled with high-quality reconstruction. The output from both remained blurry, lacking sharp details. The poor performance in this image was similar in both cases, indicating general limitations of both models in certain types of images.



One where Model 1 (L1) performed well and Model 2(MSE) did not

Here we can see a slight but clear difference: our model (with L1Loss) was able to recover sharper details, especially in areas where edges or fine textures need to be preserved. In contrast, the output of the model with MSELoss is slightly smoother and more blurry. Although the difference is not large, our model shows a clear advantage.



One where Model 2 (MSE) performed well and Model 1 (L1) did not:

- During the qualitative testing, we did not find a single example where the Ablative model, trained with the MSELoss loss function, provided a better result than our model (trained with L1Loss).
- In most images, the two models produced very similar results, but when there was a difference – it tended to favor our model, both in terms of sharpness and in preserving fine details.
- This highlights the contribution of the L1Loss loss function to improving visual quality, especially in areas with sharp edges and complex textures.

3.8) 1. Conclusions and Summary:

- In this project, we dealt with the Super-Resolution task – improving the resolution of high-quality images. The process involved working in three main stages:
- Base model (Model 1): We started with the SRResNet model, a classic and well-established architecture in the field, which served as a starting point for comparison.
- Improved model (Model 2): We then developed our own advanced model – a substantial improvement of SRResNet, built from scratch while adding changes to the architecture (such as updated residual layers, use of PixelShuffle, BatchNorm, and more), and using L1 Loss.
- Ablation Study: Finally, in order to analyze the impact of a significant component in the model, we performed an experiment in which we replaced the loss function from L1Loss to MSELoss, while keeping the other settings and architecture identical. This allowed us to isolate and examine the contribution of the loss function to the model's performance.

2. Key findings:

- The model with MSELoss (the ablated version) converged faster and exhibited lower training loss values throughout training.
- However, when examining the performance of the models on the validation set and the full images, our model with L1Loss provided consistent and better performance according to the three criteria:

- Higher PSNR – better sharpness.
- Higher SSIM – preservation of visual structure.
- Lower FID – natural appearance and closer to the original image.
- Visually, in most examples – both models (with L1 and MSE) produced similar results, but our model tended to produce slightly sharper images with more accurate details.
- No image was found where the model with MSELoss provided a better result than the model with L1Loss.

3. General conclusions:

- The loss function is a critical component that affects not only the speed of convergence, but also the quality of the actual output.
- Although MSELoss allows for a faster decrease in the value of the function, it tends to smooth out important details – which harms the quality of image reconstruction.
- L1Loss, on the other hand, preserves edges and fine details better, and is therefore preferable for Super-Resolution tasks, even if the convergence is slightly slower.
- The improvement we made to the SRResNet model proved itself both quantitatively and visually – and showed that a carefully built model, with an understanding of the architecture and loss, is able to outperform existing solutions.
- The entire experiment emphasizes the importance of proper design of the model components, and in particular the large impact of the loss function – even when it comes to small differences in structure or set of settings.

3.9) References:

[1] Lim, B., Son, S., Kim, H., Nah, S., & Mu Lee, K. (2017). **Enhanced Deep Residual Networks for Single Image Super-Resolution**. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 136–144.

<https://arxiv.org/abs/1707.02921>

[2] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., ... & Shi, W. (2017). **Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network**. CVPR, 4681–4690.

<https://doi.org/10.1109/CVPR.2017.19>

- An important background in the field of image resolution improvement, the SRGAN model is an inspiration for model improvements.

[3] Zhang, Y., Tian, Y., Kong, Y., Zhong, B., & Fu, Y. (2018). **Residual Dense Network for Image Super-Resolution**. CVPR, 2472–2481.

<https://doi.org/10.1109/CVPR.2018.00263>

- An explanation of deep networks with residual blocks, similar to the architectures we used.

[4] **DIV2K Dataset** for Image Super-Resolution. Available at: <https://www.kaggle.com/datasets/joe1995/div2k-dataset/data>

- The main dataset we used to train and test the models.