Project Report: Super Resolution

Using convolutional neural networks to enhance the quality of low-resolution images

Team: Khairul Islam, Eddie Steiner, Chetanye Maheshwari, Steven Subianto

Motivation

In today's digital world, the demand for high-quality images constantly increases across various fields such as photography, medical imaging, surveillance, and satellite imaging. However, capturing or storing high-resolution images often requires advanced hardware, higher storage capacities, and greater bandwidth, which may not always be feasible. Additionally, many older or degraded images lack the necessary quality for practical use, leading to a loss of valuable information. Traditional upscaling methods, such as bicubic interpolation, often fail to recover finer details, resulting in blurry and unappealing outputs. This creates a growing need for an efficient method to enhance low-resolution images while retaining important details.

This project is motivated by the potential of convolutional neural networks (CNNs) to address this problem. By training a model to understand and recreate the patterns and details present in high-resolution images, it becomes possible to enhance image quality effectively. The goal is to develop a scalable and reliable solution for improving image resolution, making it easier to restore lost details in images and enhance their usability across multiple applications. This not only saves resources but also bridges the gap between the limitations of hardware and the growing need for better-quality images.

Solution

Algorithm

This project focuses on using a Super-Resolution Convolutional Neural Network (SRCNN) to improve the quality of low-resolution images. The process begins by simulating low-resolution images from high-resolution ones. This is done by first reducing the image size using bicubic interpolation and then resizing it back to the original dimensions, which mimics the loss of

detail commonly seen in real-world low-resolution images. These images are then divided into small patches to make the training process more efficient and manageable.

The SRCNN model works by learning how to reconstruct high-resolution images from these low-resolution inputs. It starts with convolutional layers that use large filters to capture broad image patterns. As the layers go deeper, smaller filters are used to focus on fine details, ensuring that even the smallest features are enhanced. The model includes batch normalization layers to make training more stable and residual connections to help the network retain important features. These connections allow the model to combine new information with existing features, making the final output sharper and more accurate.

The model is trained using a loss function that compares the reconstructed images to the original high-resolution ones, ensuring the network learns to minimize their differences. An Adam optimizer with a very small learning rate is used to fine-tune the model's weights. Techniques like early stopping (to prevent overfitting), learning rate adjustments (to improve performance), and saving the best model weights during training are applied to make the training process reliable.

Once training is complete, the model is tested to ensure it produces high-quality images, and the results are saved. A notification system is also set up to inform the user when training is finished, along with details like the final validation loss. The final trained model is stored and can be used to transform low-resolution images into visually improved, high-quality images. This method has practical applications in fields like photography, medical imaging, and video enhancement.

This approach not only enhances the resolution of images but also ensures that the reconstructed images preserve important features such as edges, textures, and finer details, which are often lost in traditional upscaling methods. By leveraging the power of deep learning, the SRCNN can learn complex patterns and correlations between low-resolution and high-resolution images, making it highly effective in handling diverse image types and varying levels of degradation. Additionally, the model's ability to generalize allows it to work well on unseen images, making it suitable for real-world applications. Whether for restoring old photographs, improving video quality, or refining satellite images, this project demonstrates how convolutional neural networks can be a powerful tool in image processing and enhancement tasks.

Model Architecture Details

The architecture of the SRCNN model is designed to effectively enhance the resolution of low-resolution images by leveraging a series of convolutional layers, batch normalization, and residual connections. Below are the key components of the architecture along with their specific function:

1. Input Layer:
    a. Accepts images with three channels (RGB) and allows for flexible input dimensions using the shape (None, None, 3).
2. Initial Feature Extraction:
    a. Two convolutional layers are used with relatively large kernel sizes:
        i. Layer 1: 64 filters with a kernel size of (9, 9) and ReLU activation.
        ii. Layer 2: 64 filters with a kernel size of (5, 5) and ReLU activation.
    b. Purpose: Capture broad features and extract coarse-level information from the input image.
3. Deeper Layers for Detailed Feature Extraction:
    a. A series of convolutional layers with varied filter numbers and kernel sizes:
        i. Layer 3: 128 filters with a kernel size of (3, 3) and ReLU activation, followed by batch normalization.
        ii. Layer 4: 128 filters with a kernel size of (3, 3) and Sigmoid activation.
        iii. Layer 5: 256 filters with a kernel size of (3, 3) and ReLU activation, followed by batch normalization.
        iv. Layer 6: 256 filters with a kernel size of (3, 3) and Sigmoid activation.
    b. Purpose: Extract finer details and refine features through deeper layers.
4. Skip Connections and Residual Learning:
    a. A residual connection is added:
        i. Layer 7: 256 filters with (3, 3) kernel size and ReLU activation.
        ii. Layer 8: 256 filters with (3, 3) kernel size and Sigmoid activation.
        iii. A residual connection combines the output of Layer 8 with the input of Layer 7 using the Add() operation.
    b. Purpose: Improve gradient flow and stabilize the learning process.
5. Final Reconstruction Layers:
    a. Convolutional layers to reconstruct the high-resolution image:
        i. Layer 9: 128 filters with a kernel size of (3, 3) and ReLU activation.
        ii. Layer 10: 64 filters with a kernel size of (3, 3) and Sigmoid activation.
        iii. Output Layer: 3 filters with a kernel size of (5, 5) and Sigmoid activation to produce the RGB output.
    b. Purpose: Reconstruct the output image from the refined features.

6. Training Configuration:
    a. Optimizer: Adam optimizer with a learning rate of 1e-6.
    b. Loss Function: Mean Squared Error (MSE) to minimize pixel-level differences between predictions and ground truth.
    c. Metrics: MSE is used to evaluate training and validation performance.
7. Training Strategies:
    a. Batch Size: 128 patches per batch.
    b. Learning Rate Scheduler: Reduces the learning rate by a factor of 0.5 if the loss stagnates for 3 epochs.
    c. Checkpointing: Saves the model weights whenever there is an improvement in the training loss.

Changes Made/Iterations

Several iterations and changes were made to the algorithm during its development to improve performance and address challenges encountered in training. Initially, the SRCNN model was designed with fewer convolutional layers and larger filters to simplify computations. However, this approach led to suboptimal results, as it struggled to capture fine details in complex images. To address this, deeper layers with varied filter sizes were introduced, allowing the network to focus on both broader structures and intricate features.

Another key change involved adding batch normalization layers and residual connections. Batch normalization improved training stability and speed by normalizing intermediate layers' outputs. Residual connections allowed the network to retain critical information across layers, solving the issue of vanishing gradients and improving the reconstruction quality of fine details. These modifications significantly enhanced the model's ability to produce sharper and more accurate high-resolution images.

The choice of the loss function also underwent refinement. Initially, Mean Absolute Error (MAE) was tested but was later replaced with Mean Squared Error (MSE), which provided better results for minimizing pixel-wise differences. Additionally, early stopping and learning rate scheduling were incorporated into the training process after observing overfitting during initial runs. These changes ensured that the model trained effectively while maintaining generalization to unseen data.

Results

Overview

The outcomes of this project indicate that the Super Resolution Convolutional Neural Network (SRCNN) is capable of enhancing the quality of low-resolution images. Our evaluation includes both quantitative and qualitative analysis to measure our model's performance. Quantitative results are presented using Mean Squared Error (MSE) to measure how well the model performed during the training and validation phase. Qualitative results display the enhancement of the appearance or overall visual improvement in the super-resolution images as opposed to their low-resolution counterparts with better textures and less pixelated. Although the SRCNN provided noticeable improvements in reducing pixelation and enhancing texture smoothness, some issues include loss of sharpness in the details and a stretched-out appearance in the super-resolution images. Because of this, we require further research such as improving sharpness, addressing distortions, and exploring advanced architectures or loss functions to minimize the error and produce super-resolution images that closely resemble the original high-resolution images both in detail and in quality.

Quantitative Results

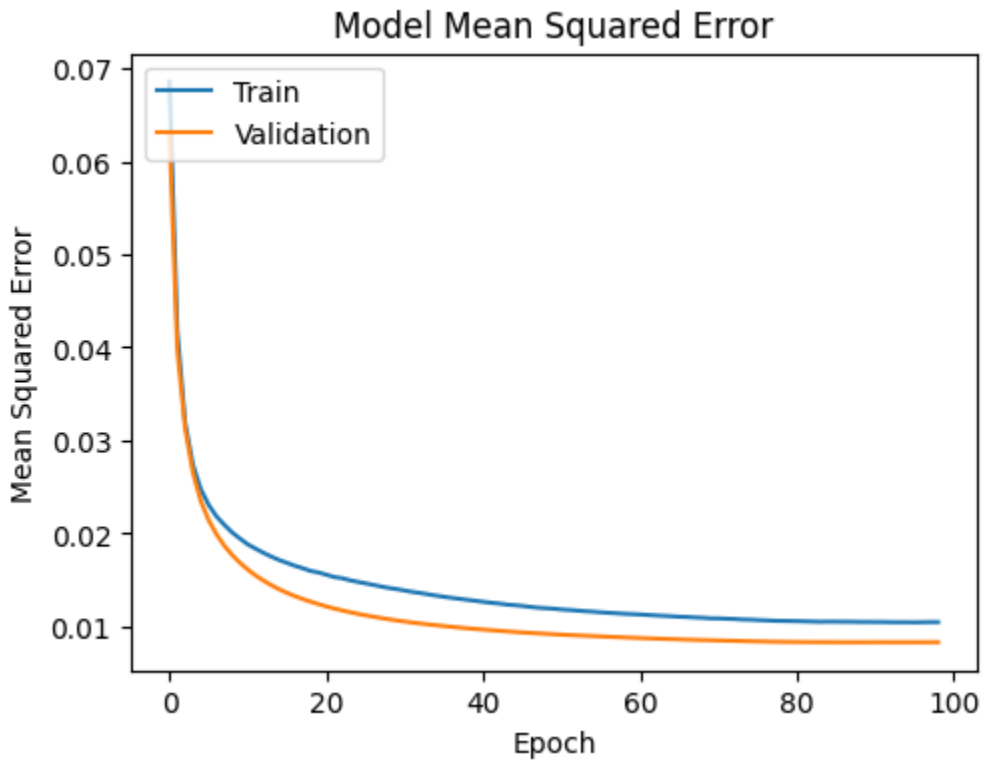

Figure 1: Training and validation MSE trends showing the model's stable learning process over 99 epochs

This SRCNN model's performance was evaluated using the Mean Squared Error (MSE) as shown above (Figure 1), a metric that we used to compute the pixel-to-pixel difference between the predicted super-resolution images and the original high-resolution images. During the 99 training iterations, the MSE just fell for both the training as well as the validation data. This consistent decrease in error during each iteration indicates that the model was learning and there were improvements in the ability of the model to generate high-resolution details from low-resolution inputs.

The training was conducted on the DIV2k dataset which had high (2k) resolution images. An epoch took about 43 seconds which gave the total training time of about 71 minutes (1 hour and 11 minutes). Some of the measures that were taken include implementing early stopping to avoid over-fitting and using a learning rate scheduler to vary the learning rate during the training phase. These methods played a role in maintaining the stability of the training and also enabled the model to learn well from the data that was not used during the training. The continuous decrease in MSE proves that the SRCNN has the potential to learn from the given dataset and produce high-resolution images.

Qualitative Results

The qualitative results of the SRCNN model were assessed by visually comparing the low-resolution (LR), high-resolution (HR) and super-resolution (SR) images. The comparison shows a dramatic enhancement of the super-resolution outputs from the low-resolution inputs in aspects such as texture and reduction of pixelation. The outcomes also show that there is still a challenge in reproducing some of the finer details as the SR images were not as sharp and accurate as the original high-resolution images.

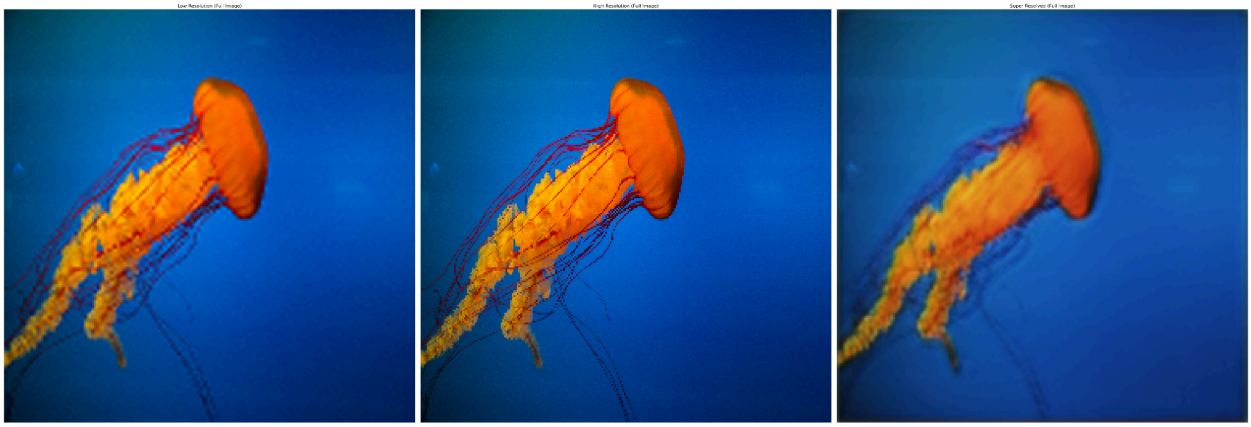

Figure 2: Comparison of low-resolution, high-resolution, and super-resolution (left to right) jellyfish images, highlighting smoother textures in the SR output

In the jellyfish example (Figure 2), the low-resolution image can be seen as very noisy and pixelated. The output of the SRCNN is much more clear than the initial image and has smoother and less pixelated surfaces. Nevertheless, when compared to the high-resolution image, the SR image is still not very defined and does not capture all the details that are depicted in the initial image.

A similar pattern can be observed in the staircase example (Figure 3). The low-resolution picture is blurry and there is no clear definition of edges and small features. The super-resolution image which is produced by the SRCNN enhances details such as edges and decreases the pixelation that is present in the LR input. Nevertheless, it can be noted that the SR image has not reached the level of the high-resolution image in terms of the sharpness and detail reconstruction.

Figure 3: Comparison of low-resolution, high-resolution, and super-resolution staircase images (left to right), demonstrating enhanced sharpness in the SR version

In order to understand the limitations of SRCNN, a zoomed-in part of the staircase was considered. As seen below (Figure 4), the super-resolution image contains more information than the low-resolution input but still does not have the level of detail found in the high-resolution image. This is because there is a need to enhance the model's capability to predict convoluted textures and other high-precision features.

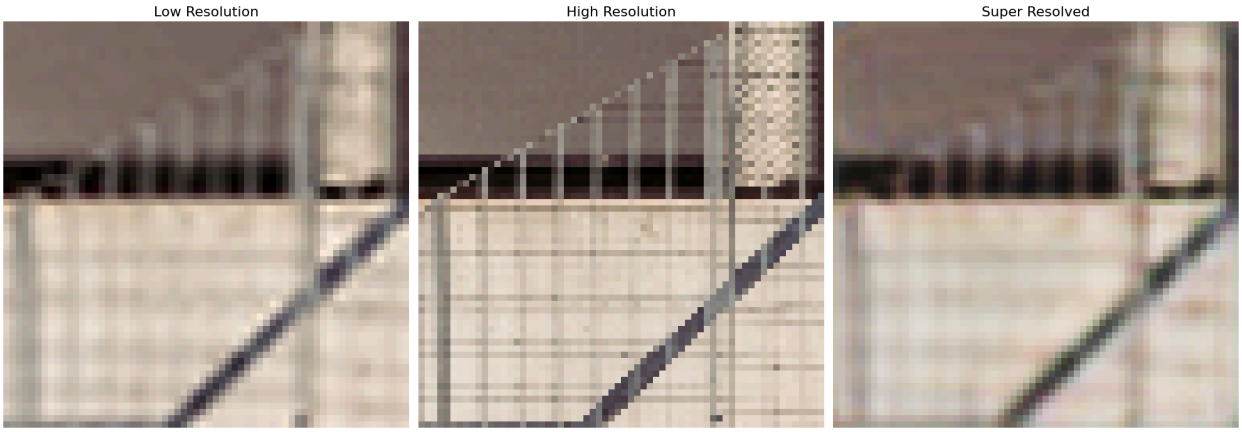

Figure 4: Zoomed-in view of the staircase image, showing some improvement in details but highlighting limitations in sharpness

A possible explanation behind the blurriness of the SR images is due to the loss function we've used, the mean squared error. Since the MSE, as described by its name, takes the average of the squared error, this could be the reason why the super-resolved images look blurry and smoothened as the model converges faster instead of taking its time to refine its definitions. While it is known that other SRCNNs have used the MSE to great effect, our low training time may be another reason for this apparent limitation.

Pixel Intensity Analysis

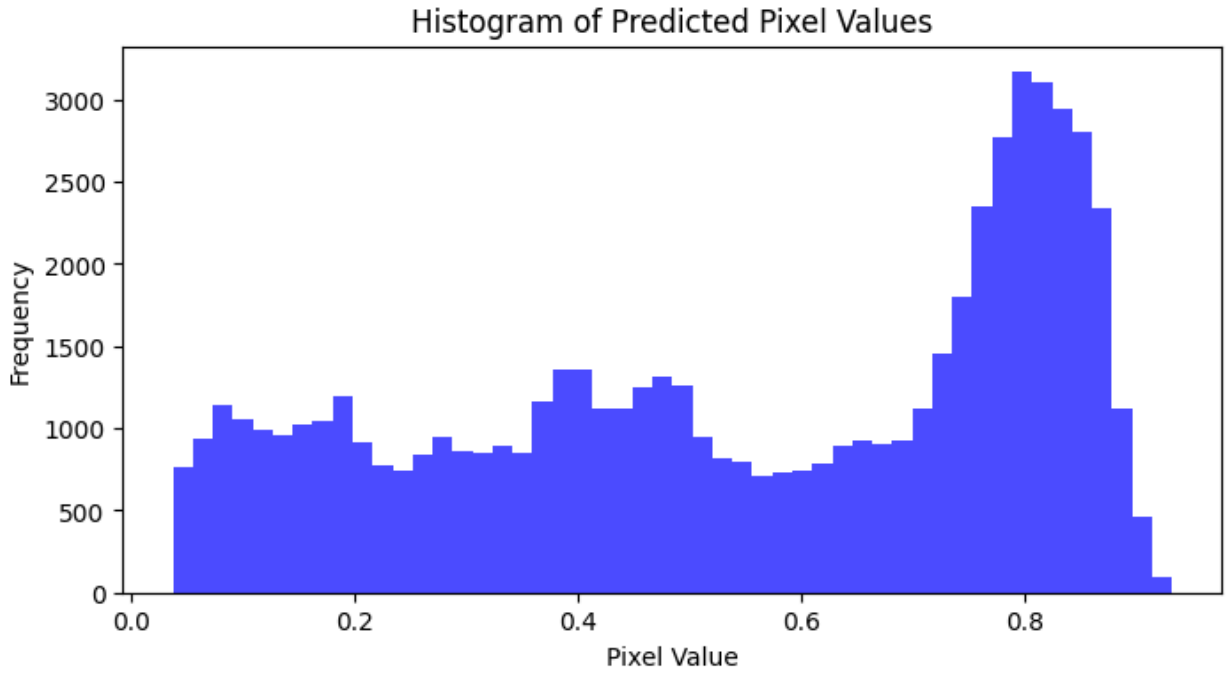

Figure 5: Showing pixel intensity values in super-resolution images, confirming appropriate brightness and color balance

In addition to the visual comparisons above, a histogram analysis of pixel intensity values in the super-resolution images was conducted. The histogram reveals a well-balanced range of pixel intensities, suggesting that the SRCNN preserved appropriate brightness and color distributions in the output images. This further validates the model's ability to produce visually realistic images.

Future Work

Further improvements include making use of a larger dataset such as the following Flickr dataset: https://www.kaggle.com/datasets/hsankesara/flickr-image-dataset. This dataset contains a lot more images at smaller sizes, thus making it possible for our model to learn at an increased rate. This is especially relevant when we compare to the DIV2K dataset, where each image file is measured in megabytes (MB) whereas the Flickr dataset contains at least 31k images, with each file measured in kilobytes (kB). However, there is the possibility that training our model on these low-definition images could be detrimental instead. Thus, experimentation is required.

Another possible exploration is to experiment with different loss functions, namely perceptual loss. A few others include gradient-based loss functions and structural similarity index. With a different loss function, we could mitigate the blurriness of our current model and achieve the sharpness and definition of high-quality images. Following this avenue, we could combine the different loss functions to create a sort of 'ensemble' of these loss functions. By conducting hyper-parameter tuning, we could find an appropriate weight for each loss which will allow the functions to work together effectively without overwhelming one another. This can also allow us to make use of the discarded Mean Absolute Error.

Finally, we could decide to have the model remain as is; instead, we could let it train for far longer than the 71 minutes it has taken. Another super-resolution neural network project done by user Affine[4] was reported to have taken up to at least 12+ hours to train. Other models were reported to have done with far more than an hour of training time as well. Thus, extending the training time is a perfectly viable experiment, given ample time and resources.

References

[1] J. Schaefferkoetter et al., “Convolutional neural networks for improving image quality with noisy PET data,” EJNMMI Research, vol. 10, no. 1, Sep. 2020, doi: https://doi.org/10.1186/s13550-020-00695-1.

[2] Yussi Eikelman, “Low resolution image classification challenge - Analytics Vidhya - Medium,” Medium, Oct. 03, 2019. https://medium.com/analytics-vidhya/low-resolution-image-classification-e725a4c31fd2 (accessed Dec. 05, 2024).

[3] B. Garber, A. Grossman, and S. Johnson-Yu, “Image Super-Resolution Via a Convolutional Neural Network.” Available: https://cs229.stanford.edu/proj2020spr/report/Garber_Grossman_Johnson-Yu.pdf

[4] Affine, “Super-Resolution With Deep Learning For Image Enhancement,” Medium, Aug 16, 2022. https://affine.medium.com/super-resolution-with-deep-learning-for-image-enhancement-e3c2c892fa52 (accessed Dec 5, 2024).