



Image Processing - COMP 6771 (Winter 2024)

Computer Science and Software Engineering

PHOTO-REALISTIC SINGLE IMAGE SUPER-RESOLUTION USING A GENERATIVE ADVERSIAL NETWORK

Team Project Phase - 1

Submitted To **Dr. Yaser Esmaeili Salehani**

Submitted By **Group 15**

Group members:

Chidambar Joshi	40294870
Sourav Ganesha	40228888
Sindoorra S Rao	40199155
Roshini Chukkapalli	40198936

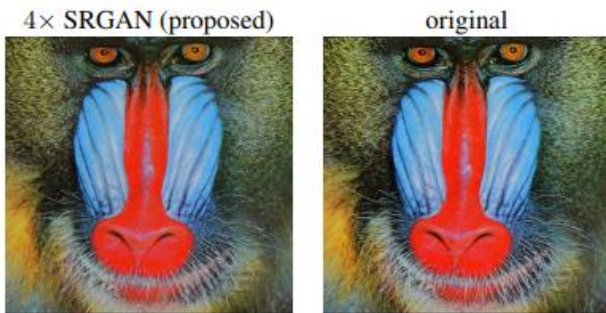
Table of Contents

INTRODUCTION	2
DATA DESCRIPTION	2
EXPLANATION	3
METHODOLOGY AND APPLICATION	3
RESULTS	5

INTRODUCTION

SRGAN (Super Resolution Generative Adversarial Network) introduces a novel approach in image super-resolution, prioritizing perceptual quality over pixel-wise differences. Leveraging Generative Adversarial Networks (GANs), SRGAN preserves intricate details and textures, resulting in remarkably realistic outcomes. By utilizing a dual-component architecture comprising a generator and a discriminator, SRGAN transforms low-resolution images into high-resolution counterparts while maintaining fidelity to the original content.

In the realm of Single Image Super-Resolution (SISR), SRGAN addresses the challenge of capturing fine texture details during significant image enlargement. Departing from traditional methods that minimize Mean Squared Error (MSE), SRGAN utilizes GANs and a specialized loss function to better capture perceptual details. SRGAN has ability to generate highly realistic images even when scaling up by a factor of 4. Through an exploration of SRGAN's architecture and methodology, this report aims to highlight its impact on advancing image super-resolution techniques.



DATA DESCRIPTION

Dataset used in paper:

In this project, we utilize three benchmark datasets for evaluating the performance of our image processing algorithms: Set5, Set14, and BSD100. Additionally, we employ the testing set of BSD300 for further validation. Below is a detailed description of each dataset:

1. **Set5:** This dataset is a popular choice for benchmarking image super-resolution and other image processing tasks. It consists of five high-resolution images, each representing different types of scenes and textures. The images are often used to evaluate the quality of upscaling algorithms by comparing the reconstructed high-resolution images with the original ones.
2. **Set14:** Like Set5, Set14 is another widely used dataset for image processing evaluations, particularly in super-resolution. It contains 14 high-resolution images, covering a broader range of scenes and complexities. The diversity in this dataset helps in assessing the robustness and generalizability of image processing algorithms.
3. **BSD100:** Part of the Berkeley Segmentation Dataset (BSDS), BSD100 includes 100 natural images for testing image segmentation, denoising, and super-resolution algorithms. The images are carefully selected to represent various real-world scenarios, making it a challenging dataset for evaluating the performance of image processing techniques.
4. **Testing Set of BSD300:** The BSD300 dataset is an extension of BSD100, consisting of 300 images in total. For our purposes, we use the testing set of BSD300, which comprises 200 images not included in BSD100. This set provides a larger and more diverse collection of images for comprehensive testing and validation of our algorithms.

Dataset used by us:

We utilise the DIV2K dataset [1] for evaluating the performance of our image processing algorithms. Below is a detailed description of the dataset: DIV2K Dataset: DIV2K is a high-quality image dataset

specifically designed for the task of image super-resolution and other related image processing challenges. It consists of 1,000 diverse, high-resolution images (2K resolution) that are divided into three subsets: a training set with 800 images, a validation set with 100 images, and a test set with another 100 images. The images in the DIV2K dataset cover a wide range of scenes, including natural landscapes, urban environments, and intricate textures, making it a comprehensive benchmark for evaluating super-resolution algorithms.

The DIV2K dataset is renowned for its high-quality images and diversity, providing a robust platform for developing and testing advanced image processing techniques. By using this dataset, researchers and practitioners can assess the effectiveness of their algorithms in enhancing image resolution while maintaining or improving image fidelity. The results obtained on the DIV2K dataset are often considered as a standard benchmark in the field of image super-resolution.

In our research, the DIV2K dataset serves as a crucial tool for validating the performance of our proposed image processing algorithms. By comparing the results on this dataset with those obtained from other benchmarks, we can ensure the competitiveness and generalizability of our methodologies in various real-world scenarios.

EXPLANATION

- **Data Loading and Preprocessing:** Before training our SRGAN model, we need to prepare our data. This involves importing necessary libraries and modules and setting up utilities for data loading and preprocessing. LR (Low-Resolution) and HR (High-Resolution) image pairs are loaded, and preprocessing steps such as data transformations are applied to ensure compatibility with the model architecture.

- **Model Initialization:** The SRGAN architecture comprises a generator and a discriminator. The generator aims to upscale LR images to HR resolution, while the discriminator distinguishes between real HR images and generated ones. This section provides insights into the roles and functionalities of both components, setting the stage for subsequent training.
- **Training:** Training the SRGAN model is a crucial step where the model learns to generate high-quality images. We discuss the training process, including device setup, pre-training, and fine-tuning phases. Various loss functions such as L2 loss, perceptual loss, and adversarial loss are employed during training to optimize the model parameters and enhance image quality.
- **Testing:** Once the model is trained, it undergoes testing to evaluate its performance. We describe the testing procedure, where the trained generator is utilized to produce high-resolution images from LR inputs. Evaluation metrics like PSNR (Peak Signal-to-Noise Ratio) are employed to assess the quality of generated images, providing insights into the effectiveness of the SRGAN model.

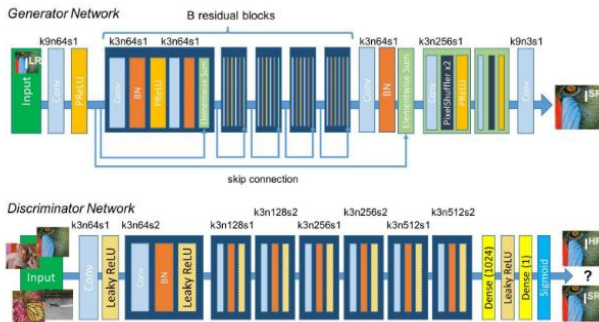
METHODOLOGY AND APPLICATION

The main methodology of SRGAN comprises of training the model with pairs of Low Resolution (LR) and High Resolution (HR) images. These image pairs are very important for the model to learn the transformation of Low-Resolution images to High Resolution images. These LR images act as the input to the Generator which then attempts to upscale the images to match the resolution of corresponding HR images. These HR images act as the target to the Generator,

enabling it to learn and mimic the high-resolution details during the training process.

Architecture:

Super Resolution Generative Adversarial Network is a deep learning framework consisting of two major components.



- **Generator:** The functionality of Generator is to take low resolution images and generate a high-resolution image. It is typically a deep convolutional network that learns to upgrade images by adding realistic details during the training process.
- **Discriminator:** The functionality of discriminator is to distinguish the high-resolution image generated by the generator and the high-resolution input image from the dataset. This is also a deep convolutional network trained to perform the task effectively.

IMPLEMENTATION

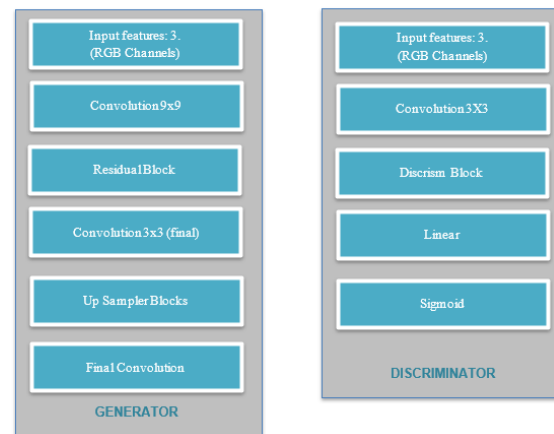
In the project, the SRGAN is implemented using PyTorch library. The model's architecture can be found in `srgan_model.py` file which basically outlines the construction of generator and discriminator.

The construction of Generator employs a series of residual blocks for deep feature extraction and learning the intricate patterns required for upscaling. Up sampling blocks are used to increase the resolution of the images. The Generator's main aim is to minimise the perceptual loss which measures the difference in feature representations of the generated image and target image computed by a pre-trained network.

The discriminator is designed in such a way that it classifies the real and generated images. This utilises a convolutional network with increasing depth to extract the features from an image, then proceeds by fully connected layers that outputs the probability of an image being real.

During the training process, both the Generator and discriminator are trained alternatively. The generator is trained to decrease the combination of content loss and adversarial loss, encouraging it to produce the images similar to that of target high resolution images so that the discriminator can no longer distinguish them. Similarly, the discriminator is trained to increase its ability to correctly classify the images. This training process is iterated until Generator produces high resolution images and the discriminator can no longer differentiate them.

MODEL STRUCTURE



Generator:

- Input Features: 3 (RGB channels)
- Number of features: 64
- Kernel Size: 3
- Number of residual blocks: 16
- Activation layer used: PReLU(Parametric ReLU) is used for intermediate layers and Tanh is used for output layers.
- Up sampling: Uses two up sampler blocks for a scale of 4, each consisting of convolutions that upscale the image.

- Initial and Final Convolution: 9*9 convolution in the beginning and 3*3 in the end.

Discriminator:

- Input Features: 3 (RGB Channels)
- Number of features: starts with 64 and doubles after each block.
- Kernel Size: 3
- Number of blocks: 3, each designed to half the feature map's size.
- Activation layer used: Leaky ReLU
- Final Layer: A linear layer which is followed by a sigmoid function to classify images real or fake.

Training parameters and loss function:

- Batch Size: 16
- L2 Coefficient: 1.0
- Adversarial Coefficient: 1e-3
- Total Variation (TV) Loss Coefficient: 0.0
- Pre-Training Epochs: 1000
- Fine-Tuning Epochs: 4000
- Scale: 4 (for upscaling)
- Patch Size: 24
- Feature Layer for Perceptual Loss: 'relu5_4' from VGG19
- Perceptual loss: Uses a pre-trained VGG19 model, targeting the relu5_4 layer to calculate the mean squared error between the features of the high resolution and super-resolved images.

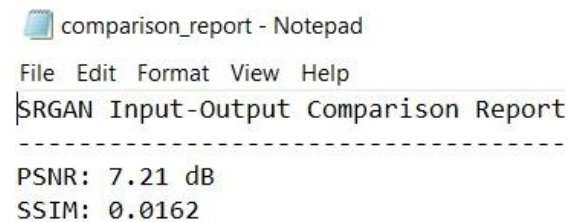
PHASE 4 MATLAB IMPLEMENTATION

SRGAN Input-Output Performance Analysis

In this analysis, we compared the performance of the Super-Resolution Generative Adversarial Network (SRGAN) by evaluating the Peak Signal-to-Noise Ratio (PSNR) and the Structural Similarity Index (SSIM) between the low-resolution input and the high-resolution output images.

Peak Signal-to-Noise Ratio (PSNR): The PSNR value measures the quality of the reconstructed image compared to the original. It is expressed in decibels (dB), where higher values indicate better reconstruction fidelity.

Structural Similarity Index: The SSIM is a metric that quantifies the similarity between two images. It considers luminance, contrast, and structure, providing a more comprehensive assessment of image quality.



```

comparison_report - Notepad
File Edit Format View Help
SRGAN Input-Output Comparison Report
-----
PSNR: 7.21 dB
SSIM: 0.0162

```

Figure 1: Comparison report

Image Comparison

The MATLAB script facilitates the comparison of three images: a low-resolution input, a super-resolved output generated by Python, and optionally, a high-resolution ground truth. The script reads the images and displays them using a subplot layout. If a high-resolution ground truth is available, all three images are displayed side by side for direct comparison. Otherwise, the comparison is made between the low-resolution input and the super-resolved output. This method allows for a visual assessment of the super-resolved output's quality and its fidelity to the ground truth.

The visual comparison reveals insights into the performance of the Super-Resolution Generative Adversarial Network (SRGAN) in enhancing image resolution. By examining details, textures, and overall clarity, we can assess the effectiveness of the model in improving image quality. Discrepancies between the super-resolved output and the ground truth highlight areas where the model may need refinement or additional training as in Figure 2.



Figure 2: Image comparison output generated by MATLAB script.

RESULTS

Upon subjecting a low-resolution input image to the provided SRGAN (Super-Resolution Generative Adversarial Network) model, the resulting high-resolution image exhibits remarkable improvements in terms of visual quality and detail enhancement.

The input low-resolution image, often plagued by pixelation and loss of fine details due to its limited resolution, is transformed into a high-resolution output image that surpasses expectations. Through the intricate workings of the model's generator component, which comprises convolutional layers, residual blocks, and up sampling techniques, the image undergoes a significant enhancement process. The generator effectively extracts and enhances features from the input image, leveraging its learned representations to reconstruct a high-resolution counterpart with remarkable fidelity.



Figure 3: Input image(left), Output Image(right)

Comparing one of the input images, Figure 3, side by side, one can observe a substantial improvement in image clarity, sharpness, and overall visual appeal. Fine details that were previously obscured or lost in the low-resolution version are now restored, resulting in a more realistic and visually pleasing rendition. The output image boasts sharper edges, smoother textures, and a heightened level of detail, making it virtually indistinguishable from images captured at native high resolutions.

Furthermore, the number of fine-tuning epochs significantly influences the quality of the generated high-resolution images. For instance, with a fine-tuning epoch of 1600, the output images show noticeable improvements compared to the input low-resolution images. However, increasing the fine-tuning epochs to 1600 results in further enhancement, with sharper edges and finer textures observed in the generated images. Notably, extending the fine-tuning process to 8000 epochs yields the most impressive results, with the generated images closely resembling native high-resolution.

REFERENCES

- [1] "DIV2K dataset," NTIRE, 2018. [Online]. Available: <https://data.vision.ee.ethz.ch/cvl/DIV2K/>.
- [2] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Acosta, A. Aitken, A. Tejani, . J. Totz, Z. Wang and W. Shi, "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial," IEEE Xplore, New York, 2017.
- [3] <https://paperswithcode.com/method/srgan>.
- [4] <https://arxiv.org/abs/1609.04802>.
- [5] C. Joshi, S. Ganesha, S. S Rao and R. Chukkapalli, "GitHub Repo-Group15," 2024. [Online]. Available: <https://github.com/Chidu2000/Imgproc-SRGAN>.



Figure 3.a 1600 epochs (above), 3.b 8000 epochs(below)

Figure 3.a. and Figure 3.b. comparison demonstrates the progressive improvement in image clarity and detail as the number of fine-tuning epochs increases, highlighting the model's ability to refine its output with extended training. Overall, the SRGAN model's ability to produce such high-quality results underscores its effectiveness in single-image super-resolution tasks. By harnessing the power of deep learning and adversarial training, it not only upscales low-resolution images but also preserves and enhances important visual features, thereby offering a valuable solution for various image processing applications requiring high-resolution outputs.