CV Super Resolution using SRGAN

Sanskruti Santosh Raut
University of Southern California
sanskrut@usc.edu

Aditya Suresh Nayak
University of Southern California
nayaka@usc.edu

***Abstract*—Image super-resolution refers to the process of enhancing the resolution and quality of a given image to produce a higher-resolution version. The goal of super-resolution is to generate a more detailed and clearer image from a lower-resolution input. This process is particularly valuable in various applications where high-quality images are crucial, such as in medical imaging, surveillance, satellite imaging, and content creation etc.**

In this project, we have implemented Super Resolution Generative Adversarial Networks to increase the resolution of the images.

***Index Terms*—Generative Adversarial Networks, Deep Learning**

I. Introduction

Generative Adversarial Networks are a deep-learning-based generative model.The GAN architecture was first described in the 2014 paper by Ian Goodfellow, et al. titled "Generative Adversarial Networks."[1]

Generative Adversarial Networks (GANs) play a crucial role in image super-resolution by introducing a novel way to generate realistic and high-quality high-resolution images from low-resolution inputs. GANs consist of two main components: a generator and a discriminator, which are trained in an adversarial manner.

Generator: The generator learns to create fake data by taking feedback from the discriminator. Refer Fig [3].

Discriminator: The discriminator is a classifier that tries to classify fake data and real data from the generator. The real data is classified as positive instances and the fake data as negative instances in training. The discriminator loss penalizes for incorrect classification and updates weights from back-propagation from discriminator loss to discriminator network. Refer Fig [4]

Adversarial training: A GAN is called adversarial because it involves orchestrating a competition between two distinct networks. This adversarial training dynamic makes the generator to continuously refine its data generation approach, aiming to outsmart the discriminator. Conversely, the discriminator refines its ability to differentiate between authentic and generated data.

II. Literature Survey

In Paper [2], the author describes the introduction of deep residual networks called SRResNet, which achieves a new state-of-the-art performance on public benchmark datasets when assessed using the widely used PSNR (Peak Signal-to-Noise Ratio) measure. However, there are certain limitations to the PSNR approach to image super-resolution. The authors then introduced SRGAN which addresses these limitations by enhancing the content loss function by incorporating Generative Adversarial training. In summary, the introduction to Generative Adversarial Networks enhances the image quality, and the super-resolution is perceived as more pleasing to the eye.

The authors of this paper [3] provide a concise overview of deep learning algorithms, applied to Single Image Super-Resolution (SISR). The paper is categorized in two groups: Deep Architectures for SISR and Optimization Objectives for SISR. The optimization technique highlights loss functions or metrics that guide the learning process to generate high-quality images. In this paper, the authors describe various optimization strategies to enhance the efficiency and effectiveness of the SISR process.

Insufficient feature extraction and blurred images are the two problems addressed in this paper [4]. Therefore, the authors propose to reconstruct the SR model using GAN. The quality of the generated image is optimized by recovering the missing details in the image. Extensive experiments are conducted on various datasets, including DIV2K, Set5, Set14, and BSD100. The results show that the proposed model excels in recovering high-frequency details when dealing with prominent upscaling factors (e.g., 4×) and outperforms other state-of-the-art methods in terms of commonly used metrics such as PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity Index).

III. Methodology

A. Data Preprocessing

In this project, we used the DIV2K dataset. The dataset includes low-resolution and high-resolution images. The dataset contains a total of 900 images split into 800 training images and 100 for testing. Each image is in RGB format and has 2k resolution.

- Downloading the dataset

First, we downloaded and extracted the zip files using the 'wget' command. The Zip files contain high-resolution and low-resolution images for training and validation datasets in bicubic X2 downsampled versions. Then, we used the 'unzip' command to extract the contents inside the files. We have defined directory paths where the extracted high and low-resolution images are stored for both training and testing. In summary, the data-processing code snippets perform the

process of downloading, unzipping, and organizing the DIV2K dataset in the specified directories.

Fig. 1. High-resolution (right) and Low-resolution images (left)

- Data Augmentation

Two sets of transformations are used: one for high-resolution images and the other for low-resolution images. The transformations include center cropping, random rotation up to 5 degrees, and conversion to a PyTorch tensor.

Architecture

The architecture of our Super-Resolution Generative Adversarial Network (SRGAN) consists of 2 networks: a generator and a discriminator. SRGAN is a type of Generative Adversarial Network (GAN) designed for super-resolution challenges. The key elements include specific layer configurations, skip connections, and loss functions within the networks.

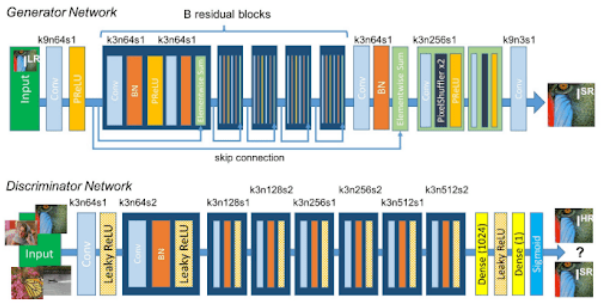

Fig. 2. SRGAN Architecture

Generator

The generator's primary task is to take low-resolution images and generate corresponding high-resolution images. The architecture combines features of the super-resolution convolutional neural network (SRCNN) and residual network (ResNet).

1) The initial layer (`conv1`) of the generator has 64 filters, a kernel size of 9, and a padding of 4 to extract high-level features. It is followed by a Parametric Rectified Linear Unit (PReLU) activation function (`prelu1`) for non-linearity.
2) The output from `prelu1` passes through a stack of residual blocks (`res_blocks`) designed to capture and refine features. The number of residual blocks is a hyper-parameter. Each block has two convolutional layers with 3x3 kernels, batch normalization, and PReLU activation. The skip connection preserves low-level details.

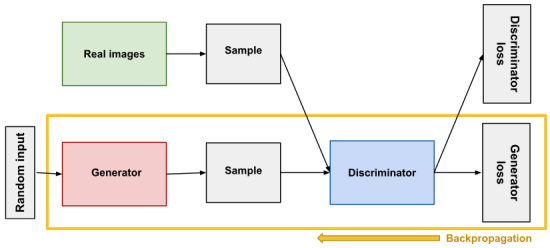

Fig. 3. Generator

3) An intermediate Convolutional Layer (`conv2`) follows the residual blocks with 64 filters, a kernel size of 3, and batch normalization.
4) The pixel shuffler upscales the output using a convolutional layer with 256 channels, a kernel size of 3, and padding of 1. PixelShuffle increases spatial resolution by a factor of 2 with PReLU activation.
5) The final convolutional layer (`conv3`) has 3 filters for RGB channels, a kernel size of 9, and padding of 4. Tanh activation is applied to limit values between [-1,1], suitable for image data.

Discriminator

The discriminator aims to classify images as real or generated. It uses several convolutional blocks.

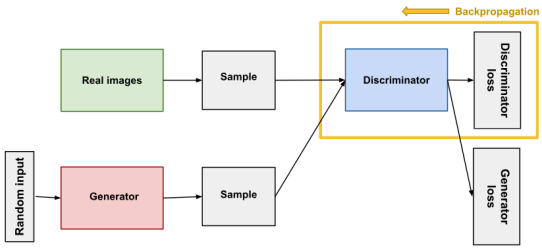

Fig. 4. Discriminator

1) The initial layer (`conv1`) is a convolutional layer with 64 filters, a 3x3 kernel size, and padding of 1. It is followed by LeakyReLU activation with a negative slope of 0.2 for non-linearity.
2) Several `ConvBlocks` follow, each with increasing channels, batch normalization, and LeakyReLU activation.
3) Feature maps from the convolutional blocks pass through a global average pooling layer (`global_pooling`), reducing the spatial dimension to 1x1.
4) The output from global pooling feature maps is flattened and passed through fully connected layers (`fc1` and `fc2`). The final output is a single neuron for discriminator decision, passed through a sigmoid activation to obtain a probability of the input being real.

B. Training

1) The training loop runs for the number of epochs provided as an hyperparameter.
2) In each epoch, first the discriminator gets calculated the discriminator loss using error from real image and error from generated image during classification.
3) After the discriminator update, generator loss which consists of vgg loss, adversarial loss and pixel loss are used to backprop and update generator parameters.
4) We also tried to put a n:1 ratio for generator and discriminator update, where generator gets updated n times while discriminator gets updated only once in each epoch. This helps generator to keep up with discriminator and avoid scenarios where discriminator always wins. Due to high GPU ram requirements, we could not run this procedure and had to adjust to single updates resulting in poor performance by the generator.
5) The discriminator is trained to distinguish between real HR images and fake HR images generated by the generator.
6) The generator is trained to minimize the adversarial loss and generate realistic HR images.

TABLE I
HYPERPARAMETER TUNING

Parameter	Values
Beta Values for Optimizer	beta 1= 0.9., beta 2: 0.9999
Learning Rate	0.001
Weight Initialization	Kaiming He
Batch Size	8
VGG Loss Coefficient	0.006
Adversarial Loss Coefficient	0.001
Number of Residual Blocks	25

Below are the mathematical equations for VGG Loss, Pixel Loss, and Adversarial Loss respectively.

To address this limitation, an alternative loss function known as VGG loss is introduced for Content Loss. Derived from the ReLU activation layers of a pretrained 19-layer VGG network, it calculates the Euclidean distance between the features of the super-resolved image and the target high-resolution image.

$$l^{SR}_{VGG/i.j} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2$$

Vgg Loss

Content loss typically employs Mean Squared Error (MSE) loss, widely used for reconstructing realistic features from significantly downsampled images. However, MSE loss may overlook high-frequency details, leading to perceptually unsatisfying images.

$$l^{SR}_{MSE} = \frac{1}{r^2WH} \sum_{x=1}^{rW} \sum_{y=1}^{rH} (I^{HR}_{x,y} - G_{\theta_G}(I^{LR})_{x,y})^2$$

Pixel-wise MSE loss

The Discriminator network employs Adversarial Loss to distinguish between the super-resolved images and the original high-resolution images.

$$l^{SR}_{Gen} = \sum_{n=1}^{N} -\log D_{\theta_D}(G_{\theta_G}(I^{LR}))$$

C. Testing

Testing is designed to evaluate a Super-Resolution Generative Adversarial Network (SRGAN) model on a set of low-resolution (LR) images. By loading the pre-trained generator model, we get generated super-resolution images from a specified directory of LR images. The generated SR images are saved in a separate directory. It is a practical demonstration of the model's performance on a test set, offering insights into its ability to enhance image resolution. Adjustments to parameters and hyperparameters is need for high quality output from test set.

IV. RESULTS AND CONCLUSION

Fig. 5. Trained Image 1

As observed from the generated HR image from the LR input, it is evident that SRGAN models play a crucial role in producing high-quality images. Due to time constraints for hyperparameter tuning and testing, as well as limitations in compute and hardware, there is still room for improvement in terms of performance.

Github Repository Link: https://github.com/Adi-glitch/Super-Resolution-Generative-Adversarial-Networks.

Fig. 6. Trained Image 2

V. Challenges

- Hardware Limitations

Due to the substantial size of the DIV2K dataset with about 900 images, training our model on the laptop's CPU proved impractical since it took 43 minutes for one epoch. GAN models require a large number of epochs to train effectively. Hence, due to computational demands, it required the utilization of a GPU. Even after using Google Colab Pro as an external GPU to increase the computing power, the model still consumed considerable GPU power. Consequently, we had to train the model for fewer epochs to manage the computational resources effectively.

- Hyperparameter tuning time restrictions

We encountered challenges in tuning hyperparameters properly since it took high computational power and training time for the model.

VI. Future Work

- We can apply transfer learning techniques to improve SRGAN's results.
- For the generator to perform well over discriminator, we can update generator n:1 ratio to that of discriminator for each epoch.

References

[1] I. J. Goodfellow et al., "Generative Adversarial Networks," arXiv preprint arXiv:1406.2661, Jun. 2014.

[2] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, and Z. Wang, "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 8, pp. 1717-1730, Aug. 2018. DOI: 10.1109/TPAMI.2017.2778105.

[3] W. Yang, X. Zhang, Y. Tian, W. Wang, J.-H. Xue, Q. Liao, M. S. Durkee, R. Abraham, J. Ai, J. D. Fuhrman, M. R. Clark, M. L. Giger, "Deep Learning for Single Image Super-Resolution: A Brief Review."

[4] X. Zhu, L. Zhang, L. Zhang, X. Liu, Y. Shen, S. Zhao, "GAN-Based Image Super-Resolution with a Novel Quality Loss," Mathematical Problems in Engineering, vol. 2020, Article ID 5217429, Feb. 2020. doi: 10.1155/2020/5217429.