Image Super-Resolution using Generative Adversarial Networks

EE 541: A Computational Introduction to Deep Learning

Prof. Brandon Franzke

Nikhil Manjunath (USC ID: 5091042121)

(**nikhilm6@usc.edu**)

Krishna Dheeraj Krovi (USC ID: 6220673324)

(**kkrovi@usc.edu**)

Abstract

Image super-resolution is the process of converting a low-resolution (LR) image into a high-resolution image (HR). Super-resolution (SR) is a very popular application in surveillance, media and medical fields. Use of deep learning systems in super-resolution has resulted in breakthroughs in generating highly accurate super-resolution images as well as faster computation speeds. In this project, we implement a Generative Adversarial Network (GAN) that performs super-resolution over a low-resolution image and generates a highly accurate super-resolved image which is as good as the target high-resolution image. The generator network is trained to generate super-resolved images from low-resolution images. The discriminator network is trained to distinguish accurately between 'fake' super-resolved images and 'real' high-resolution images. Hence, an efficient model is built that minimizes the error between the super-resolved image and the real high-resolution image.

Contents

1. Problem Statement

The goal of the project is to develop a deep learning model (GAN) that takes an input low-resolution image, performs a number of operations on it like upscaling, detail improvisation, defect removal, compression artifacts, and returns a high-resolution image. We train a Generator Network G that estimates a HR image given a LR image. We also train a Discriminator Network D that that can distinguish between real images and the super-resolved images. We want the generator to fool the discriminator by creating images that are highly similar to the real images. At the same time, we want the discriminator to be good at identifying the super-resolved images. This way, we would be building a highly efficient model, that minimizes the error between the super-resolved image and the HR real image.

2. Dataset

For the project, we will be using the DIV2K dataset. The dataset includes down-sampled images along with their corresponding high-resolution images. The dataset is divided into 800 training and 100 test images. Each image is of RGB format and is 804 X 1020 pixels.

In order to download the dataset, the following codes can be run:

Low-resolution train and test sets:

```
!wget "http://data.vision.ee.ethz.ch/cvl/DIV2K/DIV2K_train_LR_bicubic_X
2.zip"
!wget "http://data.vision.ee.ethz.ch/cvl/DIV2K/DIV2K_valid_LR_bicubic_X
2.zip"
```

Corresponding high-resolution train and test sets:

```
!wget "http://data.vision.ee.ethz.ch/cvl/DIV2K/DIV2K_train_HR.zip"
!wget "http://data.vision.ee.ethz.ch/cvl/DIV2K/DIV2K_valid_HR.zip"
```

A few images from the datasets:

Low-resolution images

High-resolution images

3. Literature Review

3.1 Previous Work

- Prediction based methods were the earliest super-resolution methods used. These methods used filters like linear filter and bicubic filter to recover HR images from LR images. However, this method was not very successful as it yielded overly smooth textures

- Compressed sensing to establish a complex LR-HR mapping has been used for super resolution as well. In Glasner et al. the authors exploit patch redundancy across scales to generate SR image.

- Yue et al. came up with the concept of generating SR images by correlating the LR image with similar content from the web and using a structure-aware matching criterion for alignment.

- Machine Learning algorithms have also been proposed as possible approaches to SR. A SR regression problem is solved with Gaussian process regression or Random Forests algorithm.

- Convolutional Neural Network (CNN) based SR algorithms are quite popular due to their superior performance. Dong et al. used a feed-forward network consisting of bicubic interpolation and a three layer deep fully convolutional network to generate SR images.

3.2 Generative Adversarial Networks (GANs)

GAN is a type of machine learning framework consisting of two neural networks – the generator G and the discriminator D, that compete with each other.

In our project:

- **Generator Network** – generates super-resolution images that are almost identical to high-resolution images.
- **Discriminator Network** – distinguishes between "real" high-resolution images and "fake" super-resolution images.

GANs are based on "indirect" learning. The generator network learns not by identifying how real is the super-resolved image, but by understanding how well it is able to fool the discriminator network.

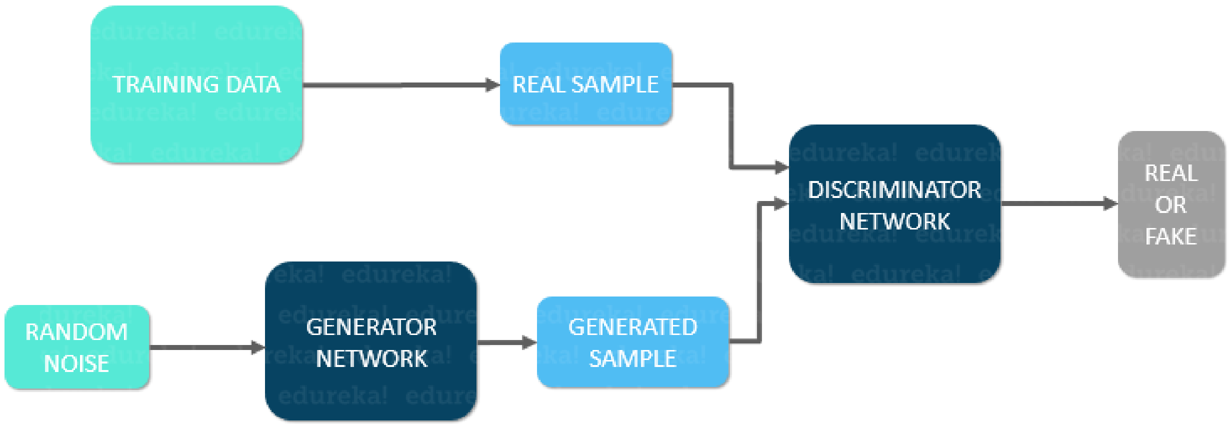


Figure 7: GAN Flowchart

3 Architecture

3.2 Generator and Discriminator

The below figures represent the architectures of the Generator and the Discriminator similar to the one we have constructed:

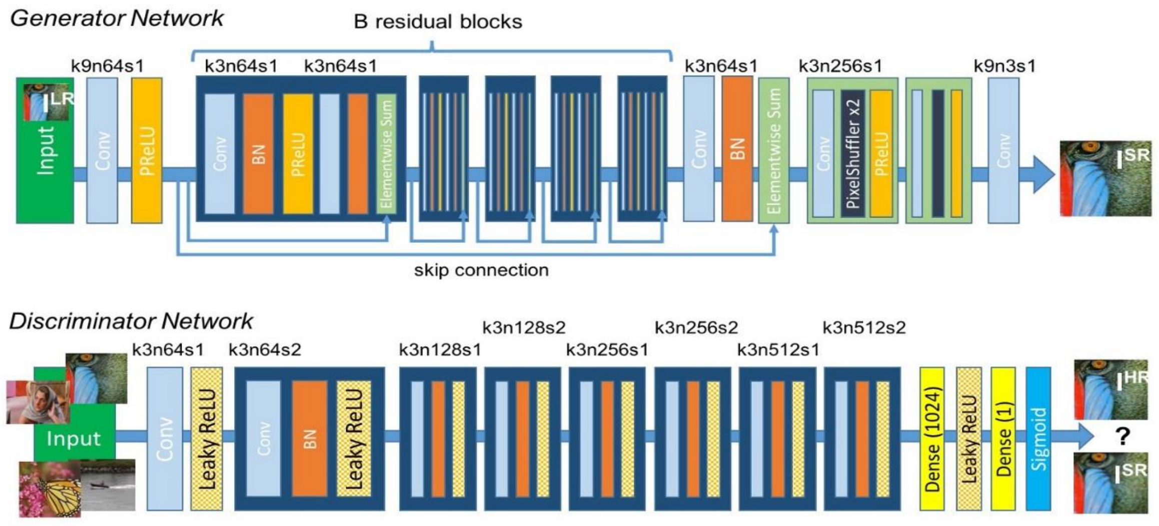


Figure 8: Generator and Discriminator Networks

The networks mostly consist of convolution blocks, residual blocks, batch normalization, ReLU blocks (Leaky and Parameterized) and pixel shufflers. Skip connections are implemented as well for the Generator network.

- Convolution blocks: consists of the convolution layer and pooling layer. These blocks are used for spatial feature extraction.
- Residual blocks: enables easy and faster implementation of deep neural networks resulting in improved performance.
- Batch Normalization: standardize network inputs and accelerate training process.
- Parameterized ReLU: generalizes the original ReLU activation with a slope for negative values learnt from backpropagation. This accounts for the different types of non-linearity associated with different layers.
- Pixel Shufflers: It is an up-sampler that rearranges elements in a tensor of shape (*, C x r^2, H, W) to a tensor of shape (*,C, H*r, W*r). It aids efficient sub-pixel convolutions.

We adopt an architecture similar to the one implemented by Christian Ledig et al.

Generator Architecture:

Layer/Block	In Channels	Out Channels	Kernel Size	Stride	Loss Function	Optimizer
Convolutional	3	64	9	1	Cross Entropy	Adam
PReLU	-	-	-	-	-	-
Residual	64	64	3	1	Cross Entropy	Adam
Convolutional	64	64	3	1	Cross Entropy	Adam
Batch Normalization	64	64	-	-	-	-
Pixel Shuffler	64	64	-	-	-	-
Convolutional	64	3	9	1	Cross Entropy	Adam

Residual Block:

Layer	In Channels	Out Channels	Kernel Size
Convolutional	64	64	3
Batch Norm	64	64	-
PReLU	-	-	-
Convolutional	64	64	3
Batch Normalization	64	64	-

Pixel Shuffler:

Layer	In Channels	Out Channels	Kernel Size
Convolutional	64	64	3
PReLU	-	-	-

Discriminator Architecture:

Layer/Block	In Channels	Out Channels	Kernel Size	Stride	Loss Function	Optimizer
Convolutional	3	64	3	1	Cross Entropy	Adam
Leaky ReLU	-	-	-	-	-	-
Convolutional	64	512	3	1	Cross Entropy	Adam
Convolutional	512	1024	1	1	Cross Entropy	Adam
Leaky ReLU	-	-	-	-	-	-
Convolutional	1024	1	1	1	Cross Entropy	Adam

4.1 Loss Functions

4.1.1 Perceptual Loss

Although the Generator network may produce a super-resolved image with good image quality, we also need to make sure that the image produced is perceptually satisfying and does not lack the high frequency details. Hence, perceptual loss is defined to capture the perceptual or per-pixel loss between the output SR image and the ground-truth HR image.

Perceptual loss is equal to the weighted sum of content loss and adversarial loss:

$$l^{SR} = l^{SR}_{X} + 10^{-3}\, l^{SR}_{Gen}$$

Where l^{SR} is the perceptual loss, l^{SR}_{X} is the content loss and $10^{-3}\, l^{SR}_{Gen}$ is the adversarial loss.

4.1.2 Content Loss

Content loss uses MSE loss. It is the most widely used loss function for recovering realistic features from heavily down sampled images. However, the MSE loss sometimes does not consider high frequency details, resulting in images appearing perceptually unsatisfying.

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

Pixel-wise MSE loss

Hence, another loss function called the VGG loss function is defined for Content Loss. It is based on the ReLU activation layers of the pretrained 19-layer VGG network. VGG loss is equal to the Euclidean distance between the Super-resolved image features and the target HR 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

VGG 19 Architecture

VGG19 is a variant of the VGG model consisting of 19 layers – 16 convolution layers, 3 fully connected layers, 5 MaxPool layers and 1 Softmax layer. The architecture is as shown below:

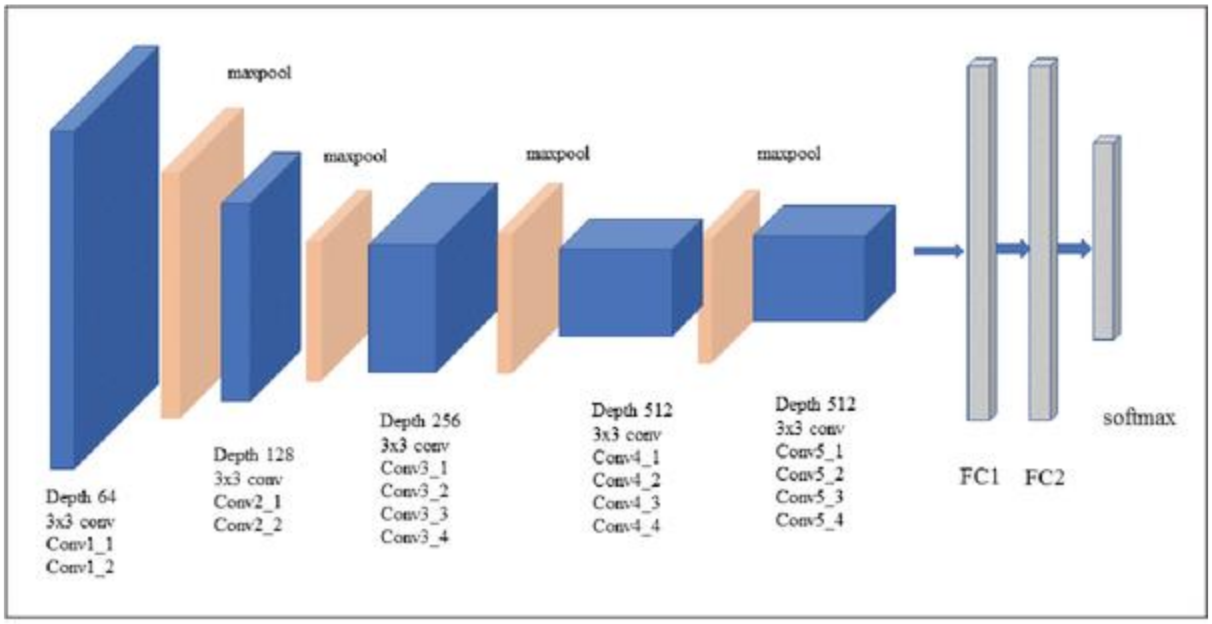


Fig. 3. VGG-19 network architecture

4.1.3 Adversarial Loss

Adversarial Loss is used by the Discriminator network to differentiate between the super-resolved images and the original high-resolution images.

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

5 Training

5.1 Process

1. Downloaded the DIV2K dataset and split the images into train and test sets.
2. Defined parameter details like upscale factor, batch size, number of epochs, learning rate and loss coefficients.
3. Built the Generator network using Convolutional blocks, Residual blocks and Pixel Shufflers. TanH is used as the activation function in the output layer.
4. Discriminator network built using Convolutional blocks and Batch Normalization.
5. Perceptual Loss function built. Calculates perceptual, adversarial and VGG losses.
6. Save Checkpoints function built so that the training can continue from checkpoints if it is stopped mistakenly or when we want to change hyperparameters. Helps reduce time spent on training.
7. Training block is developed and run.
8. Model performance is evaluated.
9. Changes made in the Generator and Discriminator networks by exploring different combinations of layers, number of layers in each block, and changing parameters.
10. Optimized model tested on the test set.

5.2 Parameters

Parameter	**Values**
Optimizer	Adam
Generator Learning Rate	10^{-4}
Discriminator Learning Rate	10^{-4}
Batch Size	16
Beta Values for Adam Optimizer	(0.9,0.9999)
Number of GPU’s	1
Discriminator steps per Generator	1
Weight Initialization	Xavier
Adversarial Loss	0.0010
Perceptual Loss	0.0041
vgg Loss	0.0010
Input Dimensions	804 X 1020
Output Dimensions	804 X 1020

5.3 Training Details

We started the training by keeping the number of epochs small (5-10). The resulting outputs were not of good resolution. Hence, we added warmups and also increased the epochs to 40. There were a few challenges as the model built was extremely complex and each epoch was taking approximately 5-7 minutes to finish. However, the outputs generated were of a much better quality and the loss values were also extremely low.

The statistics of the training are described below:

Parameter	**Description**
GPU	Google Colab Pro
GPU Memory	16 GB
Number of CPU Cores	4
System RAM	32 GB
Time per epoch	5-7 minutes
Cost	$10
Time to train	4 hours

6 Results

Successfully Super-Resolved Images

Super-resolved Image

Original HR Image

Super-resolved Image

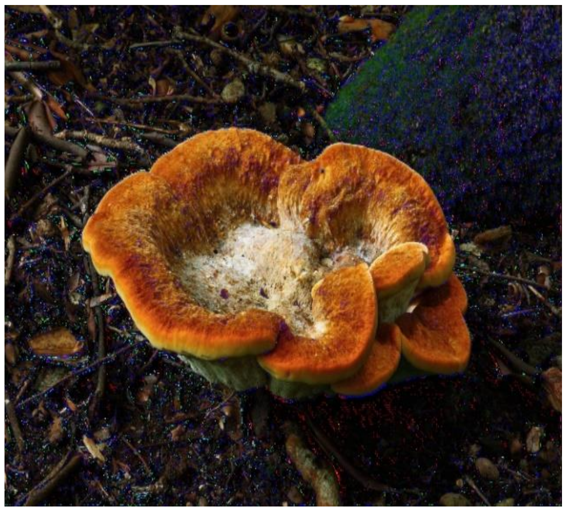
Original HR Image

Super-resolved Image

Original HR Image

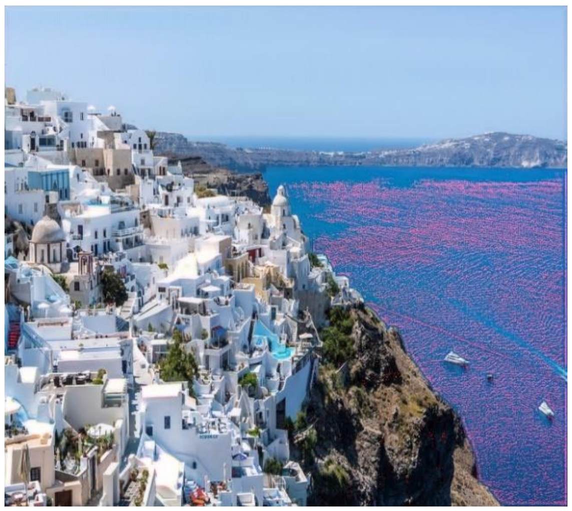
Unsuccessful Super-resolution samples

Super-resolved Image



Original HR Image

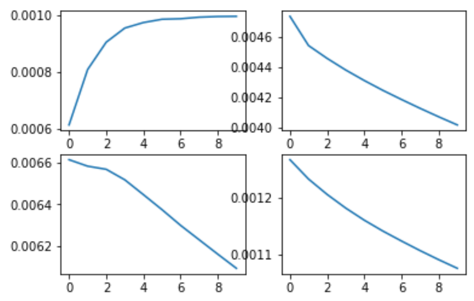
Super-resolved Image



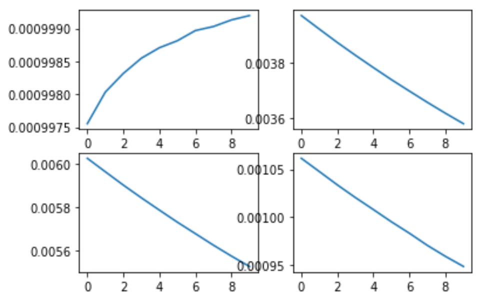
Original HR Image

Super-Resolution Comparison based on number of epochs

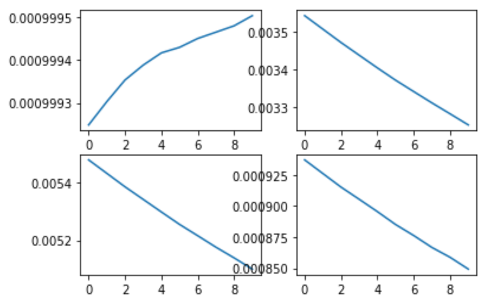
10 epochs



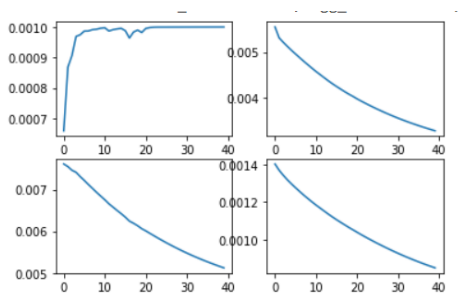
20 epochs



30 epochs



40 epochs



In each of the above plots:

- 1st Graph: Adversial Loss vs Number of Epochs
- 2nd Graph: Pixel Loss vs Number of Epochs
- 3rd Graph: Generator Error vs Number of Epochs
- 4th graph: vgg Loss vs Number of Epochs

7 Challenges

- **Hardware Limitations –** The DIV2K dataset is quite large with about 1600 (800 Low resolution + 800 High Resolution) images in total. Also, the GAN that we built was extremely complicated involving 5,215,425 trainable parameters running over 40 epochs. Hence, the model ending up using a lot of GPU. We had to train the models with reduced batch size and on Google Colab pro.

- **Performance Evaluation –** As mentioned before, GAN is based on "indirect learning", by fooling the discriminator. Hence, there is no loss function available to optimize, which makes evaluating how well the GAN will perform is a real challenge.

8 Future Work

- The project can be expanded to incorporate super-resolution for video.

- Could try developing other models for super-resolution like Encoder-Decoder networks, and deep CNNs.

9 Applications

Image super-resolution have applications in various fields. Some of them are listed below:

- Surveillance: For facial recognition on low-resolution images from CCTV cameras.

- Medical: Generating high-resolution of MRI images which usually have low-resolution.

- Media: Super-resolution can be used by streaming websites to convert a low-resolution video to a higher resolution.

10 Conclusion

In this project, we developed a GAN model on the DIV2K dataset to generate high-resolution images from input low-resolution images. The model developed performs really well on the datasets used. However, it does have some limitations in performance when there is a change in color gradient. This could be eliminated by performing more epochs.

11 References

[1] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, Wenzhe Shi Twitter. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network

[2] H. Yue, X. Sun, J. Yang, and F. Wu. Landmark image superresolution by retrieving web images. IEEE Transactions on Image Processing

[3] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In IEEE International Conference on Computer Vision (ICCV)

[4] W. Dong, L. Zhang, G. Shi, and X. Wu. Image deblurring and superresolution by adaptive sparse domain selection and adaptive regularization. IEEE Transactions on Image Processing

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Advances in Neural Information Processing Systems (NIPS)