# Text Image Super-Resolution using GAN for improved OCR accuracy

***A Report Submitted***
***in Partial Fulfillment of the Requirements for the Course***

**B.Tech. Project (CS399)**

*by*

**Qazi Sajid Azam**
(B16CS026)

**Anurag Shah**
(B16CS034)

*under the guidance of*

**Dr. Gaurav Harit**

**to the**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**INDIAN INSTITUTE OF TECHNOLOGY JODHPUR**

# CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**Text Image Super-Resolution using GAN for improved OCR accuracy**" is a bonafide work of **Qazi Sajid Azam** (**Roll No. B16CS026**) and **Anurag Shah** (**Roll No. B16CS034**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Jodhpur under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Dr. Gaurav Harit**

Assistant Professor,

April, 2019 — Department of Computer Science & Engineering,

Jodhpur. — Indian Institute of Technology Jodhpur, Rajasthan.

# Acknowledgements

We would like to thank our respected professor and mentor **Dr. Gaurav Harit** for guiding us during the course of this project.
We also acknowledge the institute for providing the necessary platform and resources for completing this project.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Recent advances in the field of deep learning have enabled us to create systems that can perform seemingly very complicated tasks. One such task is image super-resolution. Simply put, it involves recovering a high-resolution image from a low-resolution counterpart, while maintaining a high fidelity. This differs from traditional image upscaling algorithms such as bicubic, in that it actually does a somewhat intelligent analysis of the image to fill in the extra pixels, rather than calculating a pixel simply by its neighbouring pixels.
We are using a Generative Adversarial Network, also called GAN as our underlying neural network. It is trained on the ICDAR2015 Text Image dataset and performs 4x upscaling. It is inspired by [2] and the various parameters have been adjusted to give better results on text images.
The code for the project was written in Python 3.5 using libraries Numpy, TensorFlow, Tensorlayer, Scipy and Pillow. The project is available on our GitHub repository at following link:
`https://github.com/QaziSajid/Text-SRGAN`

# Chapter 2

# Literature Survey

## 2.1 Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network

In this paper [2], the authors present the concept of GAN applied for super-resolution. They emphasize that the conventional metrics to evaluate SR algorithms (such as MSE) are not indicative of actual visual fidelity and result in overly smooth textures. GAN can be used so that the network can generate highly detailed textures even on 4x upscaling factors, that looks more natural and photo realistic even if the metrics are not as high as contemporary state-of-the-art methods.

## 2.2 Deep Laplacian Pyramid Networks for Fast and Accurate Super-Resolution

In this paper [1], is proposed the Laplacian Pyramid Super-Resolution Network (LapSRN) to progressively reconstruct the sub-band residuals of high-resolution images. At each pyramid level, the model takes coarse-resolution feature maps as input, predicts the high-frequency residuals, and uses transposed convolutions for upsampling to the finer level. This method does not require the bicubic interpolation as the pre-processing step and thus dramatically reduces the computational complexity. Training of the proposed LapSRN with deep supervision is done using a robust Charbonnier loss function and achieve high-quality reconstruction. Furthermore, this network generates multi-scale predictions in one feed-forward pass through the progressive reconstruction, thereby facilitates resource-aware applications. Extensive quantitative and qualitative evaluations on benchmark datasets show that the proposed algorithm performs favorably against the state-of-the-art methods in terms of speed and accuracy.

## 2.3 Deep Laplacian Pyramid Network for Text Images Super-Resolution

In this paper [3], the network used in [1] which was proposed for natural images super-resolution, is adopted to use for single text image super-resolution. This paper has combined Gradient Difference Loss (GDL) with L1/L2 loss to enhance edges in the generated image. Quantitative and qualitative evaluations show that adding the GDL improves the super-resolution results for text image super-resolution.

# Chapter 3

# Overview Of Methodology

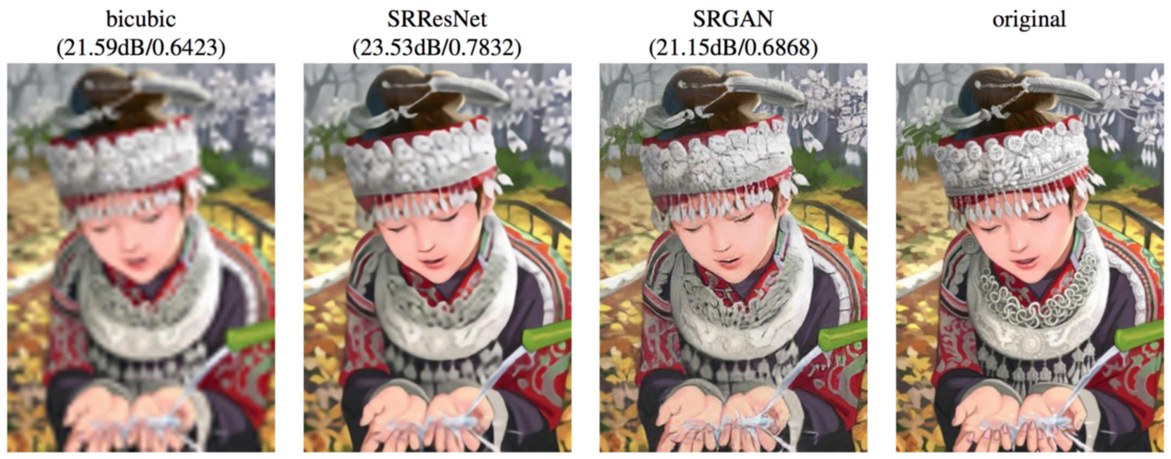

**Fig. 3.1** Various methods compared, with (PSNR, SSIM) as shown in [2]

Super-resolution GAN applies a deep network in combination with an adversary network to produce higher resolution images. As shown above, SRGAN is more appealing to a human with more details compared with the similar design without GAN (SRResNet). During the training, a high-resolution image (HR) is downsampled to a low-resolution image (LR). A GAN generator upsamples LR images to super-resolved images (SR). We use a discriminator to distinguish the HR images and backpropagate the Generator and Discriminator loss to train the generator and discriminator.

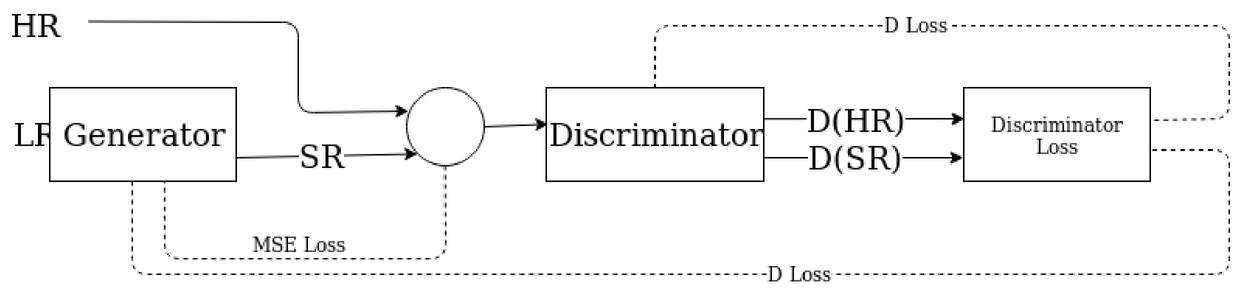

**Fig. 3.2** GAN Flow Chart

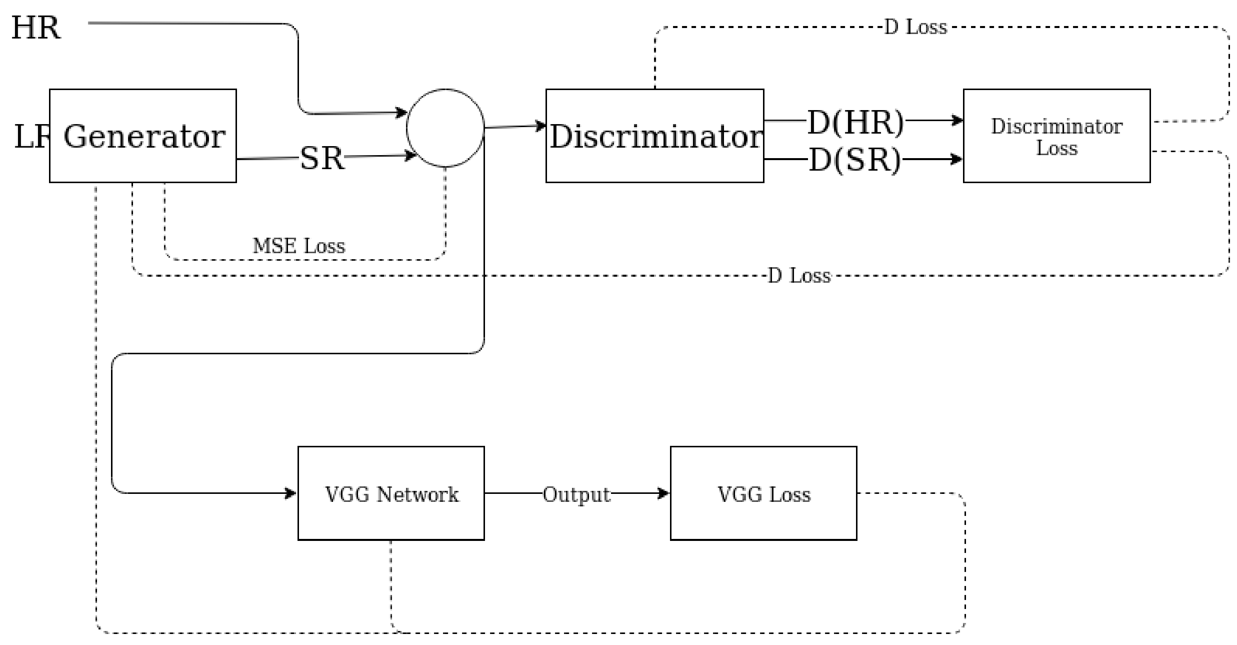

**Fig. 3.3** SRGAN Flow Chart

# Chapter 4

# Work Done

To achieve our results, we have used the procedure which is described below.

## 4.1 Preparing the Dataset

As is the case with any deep learning approach, we need a dataset to train the network. In our model, we are using the ICDAR2015 text images dataset, which contains 567 images of printed text with proper annotation and in 1x, 2x and 4x resolutions. We will use the 1x and 4x images to train the network for 4x upscaling.

## 4.2 Network Architecture

We are using the network architecture inspired by [2], which is a GAN as shown in Figure 4.1 below. It consists of two parts, the Generator and Discriminator. The Discriminator tries to distinguish between a super-resolved and an original image, while the Generator tries to produce an image so that the Discriminator is fooled into classifying it as an original image. Working together, both networks eventually improve at their tasks. The following equation 4.1 describes the minimax problem which is solved by these networks. The discriminator outputs a value *D(x)* indicating the probability that $x$ is a real image whereas

*G(x)* denotes the generated image obtained by the generator network. Our objective is to maximize the chance to recognize real images as real and generated images as fake. i.e. the maximum likelihood of the observed data. To measure the loss, we use cross-entropy as in most Deep Learning: *p log(q)*. For real image, *p* (the true label for real images) equals to 1. For generated images, we reverse the label (i.e. one minus label). So the objective becomes:

$$min_{\theta_G}\ max_{\theta_D}\ \mathbb{E}_{I^{HR}\sim p_{train}(I^{HR})}\left[logD_{\theta_D}\left(I^{HR}\right)\right]+\mathbb{E}_{I^{LR}\sim p_G(I^{LR})}\left[log\left(1-D_{\theta_D}\left(G_{\theta_G}\left(I^{LR}\right)\right)\right)\right] \tag{4.1}$$

In information theory, Entropy (*H(x)*) is used to measure the amount of information. We define entropy as

$$H(x) = E_{x\sim p}[I(x)] \tag{4.2}$$

where $x$ is an event and *I(x)* represents information of that event

$$I(x) = -log_2(P(x)) \tag{4.3}$$

where *P(x)* is probability that event $x$ occurs.

Hence entropy can be written as

$$H(x) = -E_{x\sim p}[log_2(P(x))] \tag{4.4}$$

$$H(x) = -\sum_x P(x)log_2(P(x)) \tag{4.5}$$

Coming to our network we need to optimize the parameters $\theta_D$ and $\theta_G$ that maximizes the probability of the observed data (a model that fits the data the best).

We use Maximum Likelihood Estimation to obtain this.

**Maximum Likelihood Estimation**:

We want to build a model with $\hat{\theta}$ that maximizes the probability of the observed data (a model that fits the data the best: Maximum Likelihood Estimation MLE):

$$\hat{\theta} = \arg\max_{\theta} \prod_{i=1}^{N} p(x_i|\theta) \tag{4.6}$$

However, multiplications overflow or underflow easily. Since *log(x)* is monotonic, optimize *log(f(x))* is the same as optimize *f(x).* So instead of the MLE, we take the log and minimize the negative log likelihood (NLL).

$$\hat{\theta} = \arg\min_{\theta} -\sum_{i=1}^{N} \log p(x_i|\theta) \tag{4.7}$$

$$\hat{\theta} = \arg\min_{\theta} -\sum_{i=1}^{N} \log q(x_i|\theta) = \arg\min_{\theta} -\sum_{x\in X} p(x)\log q(x|\theta) = \arg\min_{\theta} H(p,q) \tag{4.8}$$

where *H(p,q)* = Cross entropy = minimum of bits to encode x with distribution P (distribution of the true labels) using the wrong optimized encoding scheme from Q(Q is the probability distribution of the predictions from the deep network.).

From the above knowledge we can deduce the following 4.9 and 4.10 in our network.

$$max_D\, V(D) = \underbrace{\mathbb{E}_{x\sim p_{data(x)}}\left[logD\left(x\right)\right]}_{\text{recognize real images}} + \underbrace{\mathbb{E}_{z\sim p_z(z)}\left[log\left(1-D\left(G\left(z\right)\right)\right)\right]}_{\text{recognize generated images}} \tag{4.9}$$

$$min_G\, V(G) = \underbrace{\mathbb{E}_{z\sim p_z(z)}\left[log\left(1-D\left(G\left(z\right)\right)\right)\right]}_{\text{Optimize G to fool discriminator}} \tag{4.10}$$

Once both objective functions are defined, they are learned jointly by the alternating gradient descent. We fix the generator models parameters and perform a single iteration of gradient descent on the discriminator using the real and the generated images. Then we switch sides. Fix the discriminator and train the generator for another single iteration. We train both networks in alternating steps until the generator produces good quality images.

### 4.2.1 The Generator

The generator which is illustrated in 4.1 has 16 residual blocks with identical layout. Specifically, we use two convolutional layers with small $3 \times 3$ kernels and 64 feature maps followed by batch-normalization layers and ParametricReLU as the activation function. We increase the resolution of the input image with two trained sub-pixel convolution layers.

### 4.2.2 The Discriminator

The architecture of discriminator is shown in 4.1. We use LeakyReLU activation ($\alpha = 0.2$) and avoid max-pooling throughout the network. The discriminator network is trained to solve the maximization problem in equation. It contains 8 convolutional layers with an increasing number of $3 \times 3$ filter kernels, increasing by a factor of 2 from 64 to 512 kernels as in the VGG network. Strided convolutions are used to reduce the resolution each time the number of features is doubled. The resulting 512 feature maps are followed by 2 dense layers and a sigmoid activation to obtain a probability for sample classification.

## 4.3 Loss Function

We use the perceptual loss function proposed in [2] with some adjustments as our loss function. This loss function is defined in 4.11.

$$l^{SR} = \underbrace{\underbrace{l_{\mathrm{x}}^{SR}}_{\text{content loss}} + \underbrace{10^{-3} l_{Gen}^{SR}}_{\text{adversarial loss}}}_{\text{perceptual loss}} \tag{4.11}$$

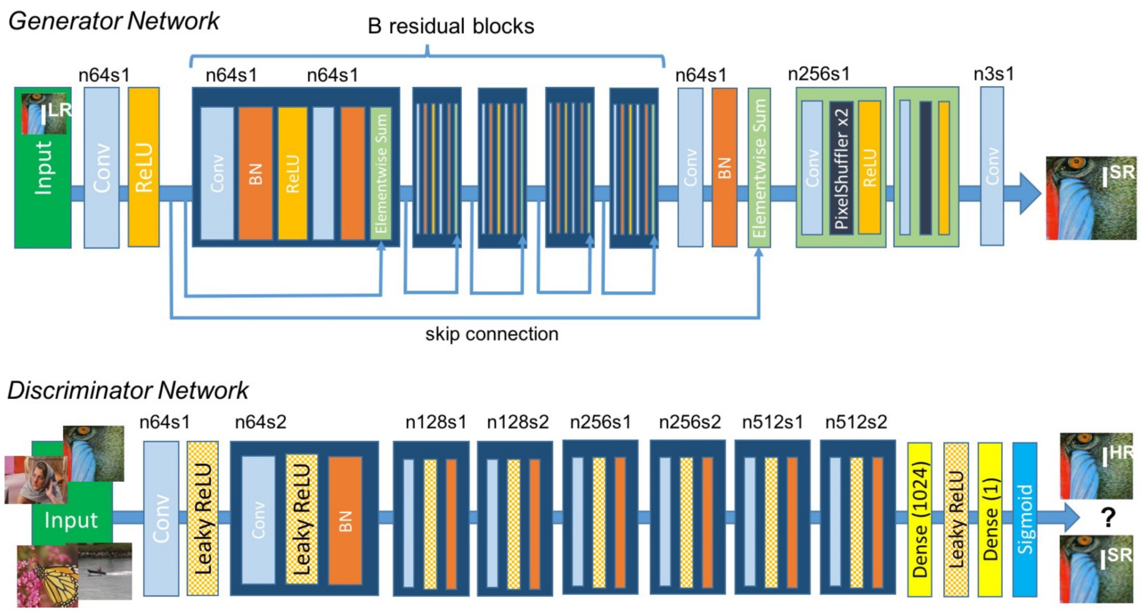

**Fig. 4.1** Network Architecture

### 4.3.1 Mean Square Error(MSE) Content loss

This is the most widely used optimization target for image SR on which many state-of-the-art approaches rely. This loss function calculates the mean of square of pixel value difference between the target image and predicted image. However, while achieving high PSNR, such solutions often lack high frequency content which results in perceptually unsatisfying solutions with overly smooth textures. The MSE loss is defined in 4.12.

$$l_{MSE}^{SR} = \frac{1}{r^2\,WH} \sum_{x=1}^{rW} \sum_{y=1}^{rH} \left( I_{x,y}^{HR} - G_{\theta_G}\left(I^{LR}\right)_{x,y} \right)^2 \tag{4.12}$$

Instead of relying only on pixel-wise losses we use VGG loss also for feature maps. Thus the content loss becomes 4.13

$$l_{\mathrm{x}}^{SR} = l_{MSE}^{SR} + 2 \times 10^{-3}\ l_{VGG/i,j}^{SR} \tag{4.13}$$

The value $2 \times 10^{-3}$ is set empirically.

### 4.3.2 VGG Content loss

We define the VGG loss based on the ReLU activation layers of the pre-trained 19 layer VGG network. With $\phi_{i,j}$ we indicate the feature map obtained by the j-th convolution (after activation) before the i-th maxpooling layer within the VGG19 network, which we consider given. We then define the VGG loss as the euclidean distance between the feature representations of a reconstructed image $G_{\theta_G}(I_{LR})$ and the reference image $\hat{I}_{HR}$.

$$l^{SR}_{VGG/i,j} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} \left( \phi_{i,j}\left(I^{HR}\right)_{x,y} - \phi_{i,j}\left(G_{\theta_G}\left(I^{LR}\right)\right)_{x,y} \right)^2 \tag{4.14}$$

### 4.3.3 Adversarial loss

In addition to the content losses described so far, we also add the adversarial component of our GAN to the perceptual loss. This encourages our network to favor solutions that reside on the manifold of natural images, by trying to fool the discriminator network. The adversarial loss $l^{SR}_{GEN}$ is defined based on the probabilities of the discriminator $D_\theta(G_\theta(I^{LR}))$ over all training samples.

$$l^{SR}_{Gen} = \sum_{n=1}^{N} -log D_{\theta_D}\left(G_{\theta_G}\left(I^{LR}\right)\right) \tag{4.15}$$

Here, $D_\theta(G_\theta(I^{LR}))$ is the probability that the reconstructed image $l^{SR}_{GEN}$ is a natural HR image. For better gradient behavior, we minimize $log(D_\theta(G_\theta(I^{LR})))$ instead of $log[1 - D_\theta(G_\theta(I^{LR}))]$

## 4.4 Calculation of Accuracy

We are using metrics such as PSNR, MSE and SSIM for evaluation of the results, in addition to the performance in intended usecase, i.e. OCR. We use the Tesseract OCR library to find the accuracy of our results in OCR, and analyse the improvement over traditional techniques.

# Chapter 5

# Results

We are using the MSE, PSNR and SSIM metrics, alongwith the Tesseract OCR library to evaluate our results, as mentioned earlier.

## 5.1 Performance

The dataset we are using contains 567 samples, and we use those with a batch size of 21. The training happens in two phases, the initial training of generator and the adversarial learning. We use 60 epochs for the first and 300 epochs for the second.
For our final training, we used the system provided by our institiute, which has eight NVIDIA GeForce GTX 1080Ti GPUs, each with 11GB of VRAM, and the full system has 256GB of system RAM. On this machine, each epoch in the first phase takes around 4 seconds, and each epoch in the second phase takes around 300 seconds. Thus the full training takes nearly 25 hours.
For evaluation, we use 141 images different from the training data. We can specify whether we want to use OCR or not with an argument when running the code. Without OCR the evaluation of 141 images completes in nearly 30 seconds, while with OCR it takes nearly 5 minutes.

The figures 5.1 and 5.2 show the gradual decrease of loss as epochs proceed.

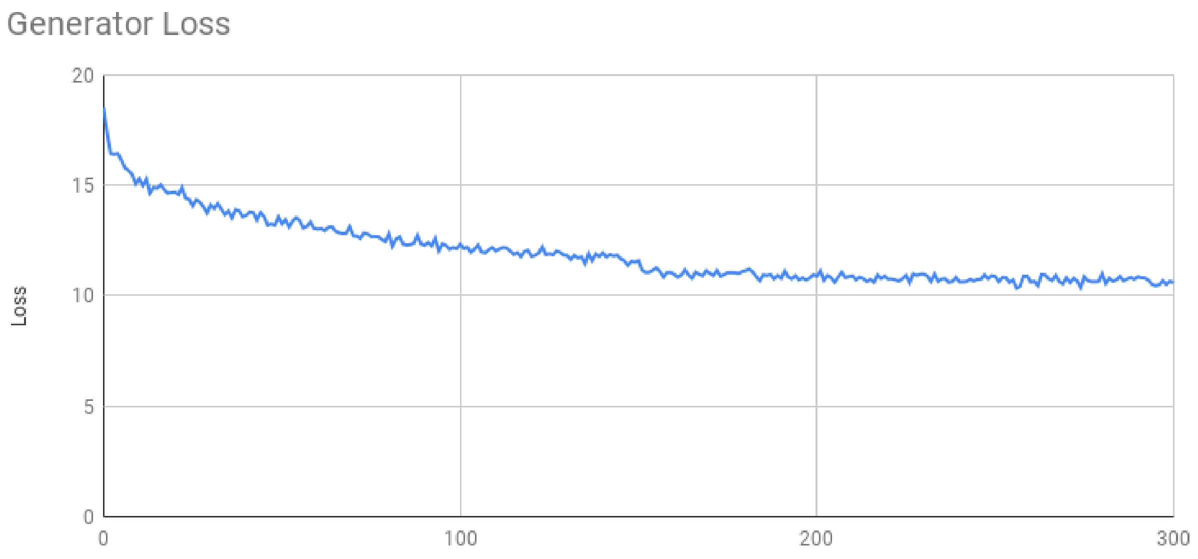

**Fig. 5.1** Generator loss

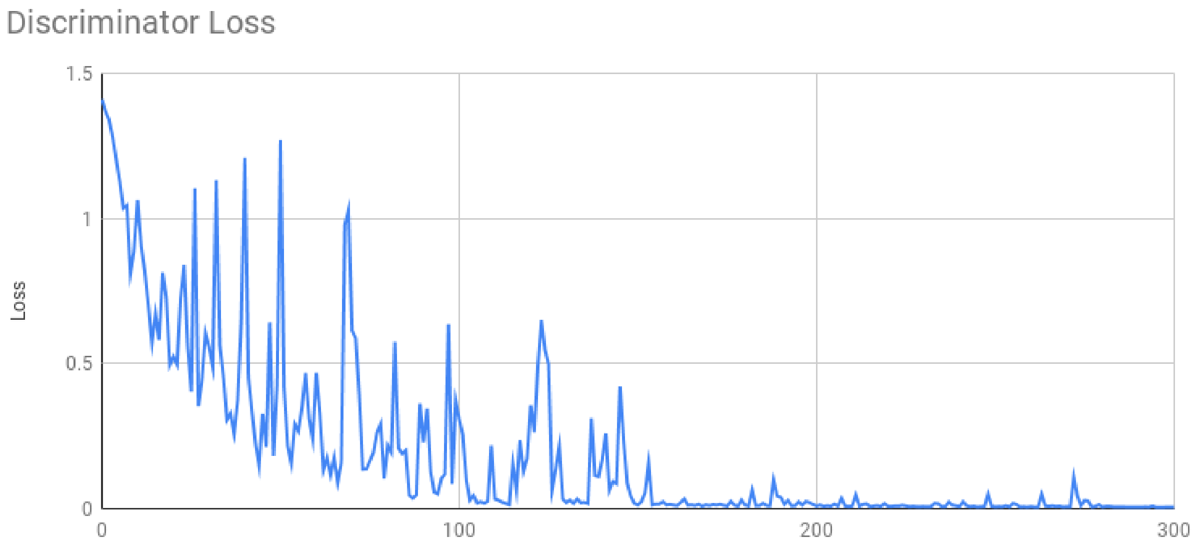

**Fig. 5.2** Discriminator loss

## 5.2 Result Metrics

Although our model is not deeply optimized for metrics such as PSNR and is rather focused on the perceptual loss, it still performs decently with these metrics. Our final model achieves an average PSNR value of 21.86047, MSE value of 475.5829 and SSIM value of 0.921915.

This is not as much as other state-of-the-art methods for reasons we discuss in Chapter 6, as well as the fact that it does not focus on these metrics, unlike the other methods.

The table 5.1 shows the results with the different methods.

| | PSNR | MSE | SSIM |
|---|---|---|---|
| Bicubic | 22.11959 | 618.1672 | 0.885283 |
| SRGAN trained on DIV2K | 21.45826 | 643.0672 | 0.864593 |
| SRGAN trained on ICDAR2015 | 19.33736 | 833.2610 | 0.876426 |
| Text SRGAN | 21.86047 | 475.5829 | 0.921915 |

**Table 5.1** Quantitative comparison of the results

## 5.3 Accuracy with OCR

We apply OCR on the image and then match the text with the ground truth annotation. For text matching, we use the SequenceMatcher function from the Difflib library in Python. This gives us the result as a number between 0 and 1, where 1 is a perfect match and 0 is a failure. The original images get an average score of 0.875629, or nearly 87.56%. Our network can generate 4x upscaled images from the LR test data, which get an average score of 0.836612 or nearly 83.66%. This is a significant improvement over LR (score=0.21) and bicubic (score=0.68), and is much closer to accuracy of HR results. The accuracy of generated images relative to HR is nearly 95.54%.

## 5.4 Visual Quality

The figure 5.3 shows some of the output samples obtained from the model.

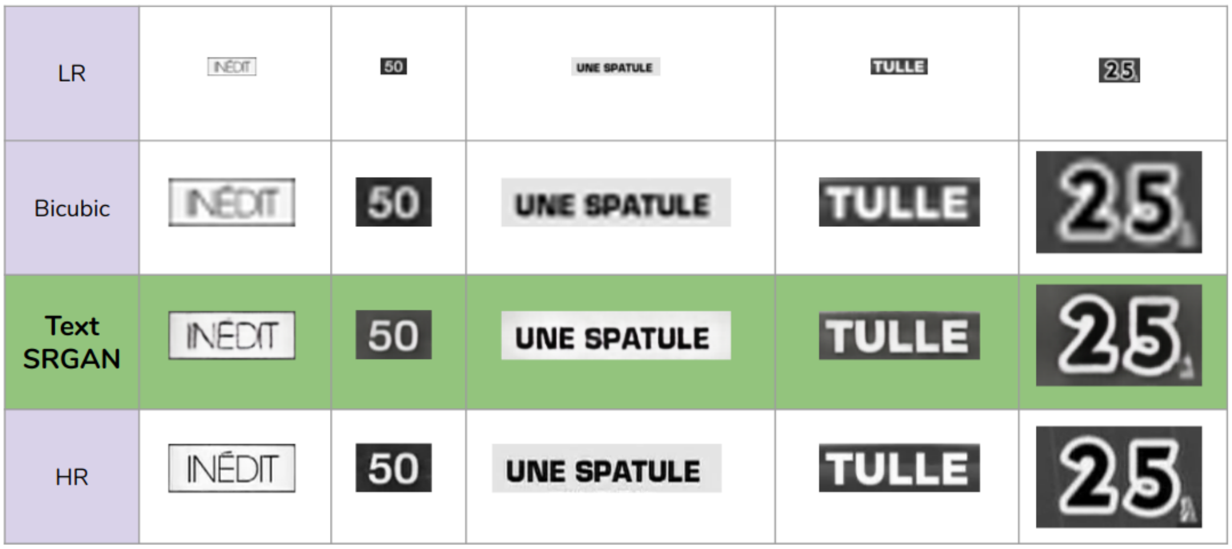
LR
Bicubic
Text SRGAN
HR
INÉDIT
50
UNE SPATULE
TULLE
25

**Fig. 5.3** Few of the samples

# Chapter 6

# Future Work

Even though our system performs impressively in improving OCR accuracy, it is still not as good as some other state-of-the-art methods in terms of super-resolution, because those methods have been optimized for natural images. The main reason for this is that much more research has been done on the super-resolution problem involving natural images than text images, and much more training data is available for natural images. Some other network architectures like Deep Laplacian Pyramid Network [1] also perform very well. Now we will discuss some of the future work that can be done in this project by anyone who decides to improve upon it.

## 6.1 Optimizations to the Loss Function

Due to limited time and resources, we have not been able to extensively explore different loss functions. We suggest using loss functions that incorporate the OCR accuracy for better results. Also, the loss function in our case has a component of the VGG loss which is originally optimised for natural images. We recommend using a similar network trained on text images to get better results.

## 6.2 Training on a larger dataset

The dataset we are using, the ICDAR2015 text images dataset, is a fairly small one. Also, it only contains images of printed text so our system is not suited for handwritten text. One of the major improvements to accuracy will come from using a larger dataset with more variations, and also train it on handwritten images to increase its potential applications.

## 6.3 Improvement of Network Architecture

An enhanced version of SRGAN is proposed in [4] which may improve the results significantly. However the network is more complex and hence more resource-intensive. So it was not feasible for us due to the limited time and resources. Although with enough resources, it is fairly easy to train and obtain results, which may be done in the future.

# References

[1] Wei-Sheng Lai et al. "Fast and Accurate Image Super-Resolution with Deep Laplacian Pyramid Networks". In: *CoRR* abs/1710.01992 (2017). arXiv: 1710.01992. URL: http://arxiv.org/abs/1710.01992.

[2] Christian Ledig et al. "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network". In: *CoRR* abs/1609.04802 (2016). arXiv: 1609.04802. URL: http://arxiv.org/abs/1609.04802.

[3] Hanh T. M. Tran and Tien Ho-Phuoc. "Deep Laplacian Pyramid Network for Text Images Super-Resolution". In: *CoRR* abs/1811.10449 (2018). arXiv: 1811.10449. URL: http://arxiv.org/abs/1811.10449.

[4] Xintao Wang et al. "ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks". In: *CoRR* abs/1809.00219 (2018). arXiv: 1809.00219. URL: http://arxiv.org/abs/1809.00219.