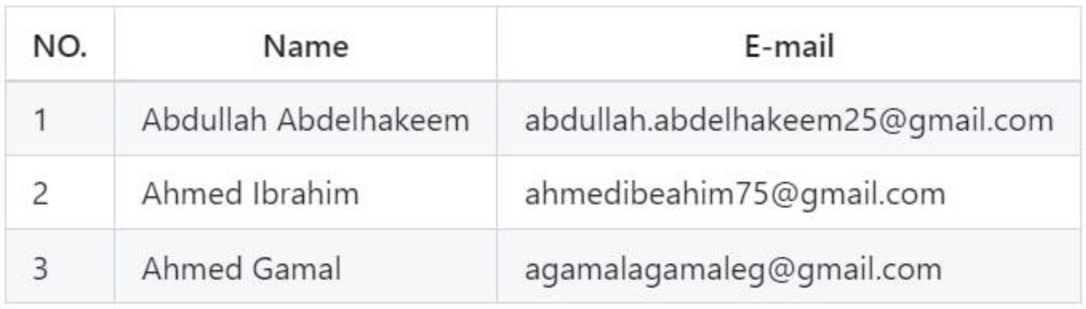
Document SRGAN

Single Image Super Resolution (SRGAN)



NO.	Name	E-mail
1	Abdullah Abdelhakeem	abdullah.abdelhakeem25@gmail.com
2	Ahmed Ibrahim	ahmedibeahim75@gmail.com
3	Ahmed Gamal	agamalagamaleg@gmail.com

Cluster Head of AI_Pro at ITI: ENG.George Iskander
Supervisor of BrightSkies : ENG.Ahmed Ayyad

Table of Contents

Table of Figures

1. Introduction

1.1.Motivation

Recently, learning-based models have enhanced the performance of single-image super-resolution (SISR). However, applying SISR successively to each video frame leads to a lack of temporal coherency. Convolutional neural networks (CNNs)outperform traditional approaches in terms of image quality metrics such as peak signal to noise ratio(PSNR) and structural similarity (SSIM). However, generative adversarial networks (GANs) offer a competitive advantage by being able to mitigate the issue of a lack of finer texture details, usually seen with CNNs when super-resolving at large upscaling factors. Furthermore, to improve the "naturality" of the super-resolved image while eliminating artifacts seen with traditional algorithms, we utilize the discriminator from super-resolution generative adversarial network (SRGAN). Although mean squared error (MSE) as a primary loss-minimization objective improves PSNR/SSIM

1.2.Problem Definition

This architecture used a pre-trained VGG-19 feature extractor and gave photorealistic results on large (4x) unsampled low resolution images. It has been applied to the DIV2K, CelebA and other natural image datasets and here we want to see how it performs on OCT data. This network will serve as a baseline for further experiments with upscaling, choice of feature extractor etcetera.

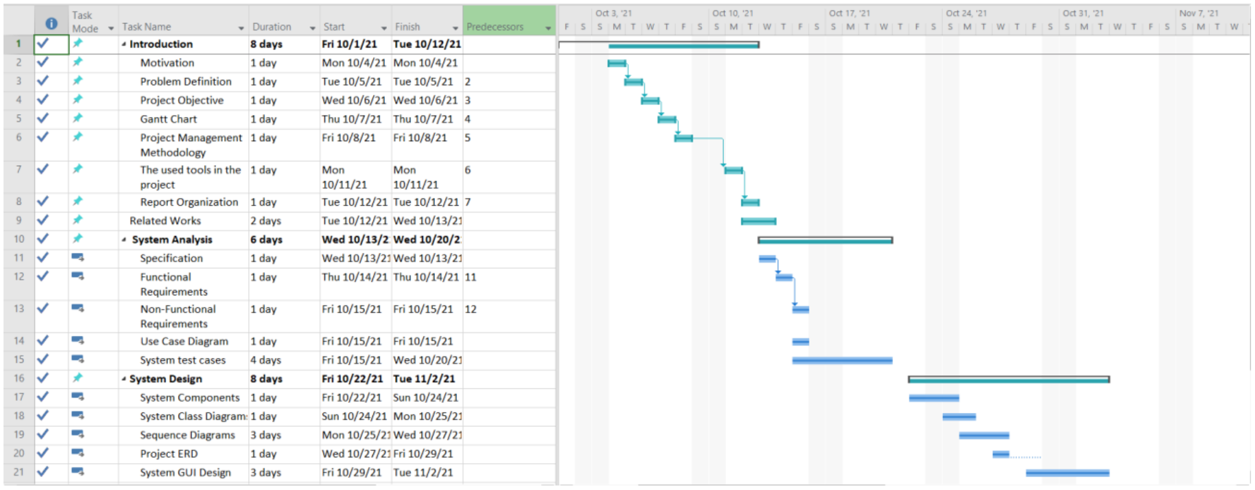
1.3.Project Objective

The computational enhancement of image resolution is known as super-resolution. Generative Adversarial Networks (GANs) are one of newer methods that have been applied to super resolution and in this notebook, we use a Super-Resolution GAN (SRGAN) to enhance subsampled OCT scans.

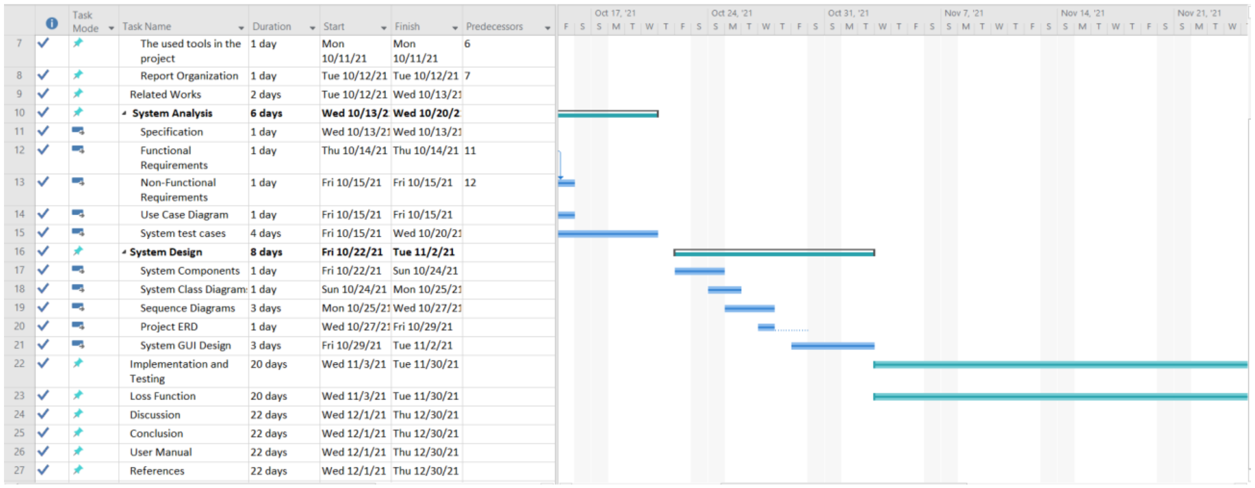
The SRGAN, introduced in 2016, addressed the issue of reconstructing high resolution (HR) images from low resolution (LR) images such that fine texture detail in the reconstructed super resolution (SR) images was not lost. Here the authors used a perceptual loss instead of a pixel-wise Mean Squared Error (MSE) loss. MSE loss approaches give a high Peak Signal-to-Noise (PSNR) value, but they also tend to produce overly-smooth images with insufficient high-frequency details. The perceptual loss by contrast has a content loss component that computes pixel-wise differences in feature space (not pixel space) and this results in an SR image that is closer to the subjective evaluation of human observers.

The SRGAN model uses a deep neural network (built with residual blocks) and optimized using perceptual loss in a GAN framework. A VGG-19 network is used for feature extraction; this allows us to compute the feature distance between the original and generated images sent through the feature extractor.

1.4.Gantt Chart



		Task Mode	Task Name	Duration	Start	Finish	Predecessors
1	✓	📌	**Introduction**	**8 days**	**Fri 10/1/21**	**Tue 10/12/21**	
2	✓	📌	Motivation	1 day	Mon 10/4/21	Mon 10/4/21	
3	✓	📌	Problem Definition	1 day	Tue 10/5/21	Tue 10/5/21	2
4	✓	📌	Project Objective	1 day	Wed 10/6/21	Wed 10/6/21	3
5	✓	📌	Gantt Chart	1 day	Thu 10/7/21	Thu 10/7/21	4
6	✓	📌	Project Management Methodology	1 day	Fri 10/8/21	Fri 10/8/21	5
7	✓	📌	The used tools in the project	1 day	Mon 10/11/21	Mon 10/11/21	6
8	✓	📌	Report Organization	1 day	Tue 10/12/21	Tue 10/12/21	7
9	✓	📌	Related Works	2 days	Tue 10/12/21	Wed 10/13/21	
10	✓	📌	**System Analysis**	**6 days**	**Wed 10/13/2**	**Wed 10/20/2**	
11	✓	➡	Specification	1 day	Wed 10/13/21	Wed 10/13/21	
12	✓	➡	Functional Requirements	1 day	Thu 10/14/21	Thu 10/14/21	11
13	✓	➡	Non-Functional Requirements	1 day	Fri 10/15/21	Fri 10/15/21	12
14	✓	➡	Use Case Diagram	1 day	Fri 10/15/21	Fri 10/15/21	
15	✓	➡	System test cases	4 days	Fri 10/15/21	Wed 10/20/21	
16	✓	📌	**System Design**	**8 days**	**Fri 10/22/21**	**Tue 11/2/21**	
17	✓	➡	System Components	1 day	Fri 10/22/21	Sun 10/24/21	
18	✓	➡	System Class Diagram:	1 day	Sun 10/24/21	Mon 10/25/21	
19	✓	➡	Sequence Diagrams	3 days	Mon 10/25/21	Wed 10/27/21	
20	✓	➡	Project ERD	1 day	Wed 10/27/21	Fri 10/29/21	
21	✓	➡	System GUI Design	3 days	Fri 10/29/21	Tue 11/2/21	



		Task Mode	Task Name	Duration	Start	Finish	Predecessors
7	✓	📌	The used tools in the project	1 day	Mon 10/11/21	Mon 10/11/21	6
8	✓	📌	Report Organization	1 day	Tue 10/12/21	Tue 10/12/21	7
9	✓	📌	Related Works	2 days	Tue 10/12/21	Wed 10/13/21	
10	✓	📌	**System Analysis**	**6 days**	**Wed 10/13/2**	**Wed 10/20/2**	
11	✓	➡	Specification	1 day	Wed 10/13/21	Wed 10/13/21	
12	✓	➡	Functional Requirements	1 day	Thu 10/14/21	Thu 10/14/21	11
13	✓	➡	Non-Functional Requirements	1 day	Fri 10/15/21	Fri 10/15/21	12
14	✓	➡	Use Case Diagram	1 day	Fri 10/15/21	Fri 10/15/21	
15	✓	➡	System test cases	4 days	Fri 10/15/21	Wed 10/20/21	
16	✓	📌	**System Design**	**8 days**	**Fri 10/22/21**	**Tue 11/2/21**	
17	✓	➡	System Components	1 day	Fri 10/22/21	Sun 10/24/21	
18	✓	➡	System Class Diagram:	1 day	Sun 10/24/21	Mon 10/25/21	
19	✓	➡	Sequence Diagrams	3 days	Mon 10/25/21	Wed 10/27/21	
20	✓	➡	Project ERD	1 day	Wed 10/27/21	Fri 10/29/21	
21	✓	➡	System GUI Design	3 days	Fri 10/29/21	Tue 11/2/21	
22	✓	📌	Implementation and Testing	20 days	Wed 11/3/21	Tue 11/30/21	
23	✓	📌	Loss Function	20 days	Wed 11/3/21	Tue 11/30/21	
24	✓	📌	Discussion	22 days	Wed 12/1/21	Thu 12/30/21	
25	✓	📌	Conclusion	22 days	Wed 12/1/21	Thu 12/30/21	
26	✓	📌	User Manual	22 days	Wed 12/1/21	Thu 12/30/21	
27	✓	📌	References	22 days	Wed 12/1/21	Thu 12/30/21	

1.5. Project Management Methodology

Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Instead of betting everything on a "big bang" launch, an agile team delivers work in small, but consumable, increments.

So what is Agile methodology in project management? It’s a process for managing a project that involves constant collaboration and working in iterations. Today, the word Agile can refer to these values and the frameworks for implementing them, including Scrum, Kanban, Extreme Programming (XP), and Adaptive Project Framework (APF).

Agile encourages a high degree of input and collaboration between the client and development team. This leads to happier clients because there is transparency throughout the process and developers are better informed on client needs and wants.

By breaking down the development process into iterative sprints, project managers can more accurately estimate costs and set clear, predictable timelines. This makes stakeholders happier because they know what to expect and can plan budgets and marketing strategies more precisely. It also makes the development process easier for teams because they can focus on delivering quickly and reliably and test software regularly for quality and efficacy.

Agile project management is all about being nimble so teams can adapt to changes quickly while reducing sunk costs. Agile allows teams to pivot due to changing client needs, shifts in market demands, or in response to evolving product requirements. This gives teams the flexibility to refine and reprioritize the product backlog so that they are always delivering high-quality, relevant products on time and on budget.

Agile product development integrates regular testing into the development process. This makes it easier for the product owner to identify any issues early on and make changes as needed. The result is higher quality products that are relevant and thoroughly vetted.

1.6.The used tools in the project

1.6.1. Software

We used python programming language to implement our solution as it’s so popular with machine learning and deep learning techniques with the help of libraries like:

- Numpy: for fast matrix arithmetic operations
- OpenCv: for loading the images and performing resize operations on input images
- Matplotlib: for plotting the images
- Pickle: for dumping the model into a “pkl” format file in order to load it in the flask application
- Flask: flask framework used to develop website make it easier for user to deal with it
- Keras and Tensorflow

1.6.2. Hardware

We used:

- Colab
- Kaggle
- Azure

1.7.Report Organization

The material presented at the project is organized into five chapters. After this introductory chapter, **chapter 2** describes the related work which presents The closest examples of the project and the main differences between them and our project. **Chapter 3** summarizes the system analysis which demonstrates the functional requirements, non-functional requirements, use case diagrams and system test cases. **Chapter 4** provides information about System component diagrams, system class diagrams, sequence diagrams, project ERD. **Chapter 5** demonstrates the implementation phases and samples of the applied test cases.

2. Related Works

The SRGAN is implemented as follows:

Training

We downsample HR OCT images by 4x to synthetically create LR training data. This gives us pairs of HR and LR images for the training data set. The Generator upsamples LR images by 4x and will be trained to generate SR images. The discriminator will be trained to distinguish between HR/SR images; the GAN loss is backpropagated to the discriminator and the generator. Evaluation The visual quality of generated images will be observed. In addition standard quantitative metrics, Peak Signal-to-Noise Ratio and Structural Similarity Index (PSNR, SSIM), will be used to assess the results.

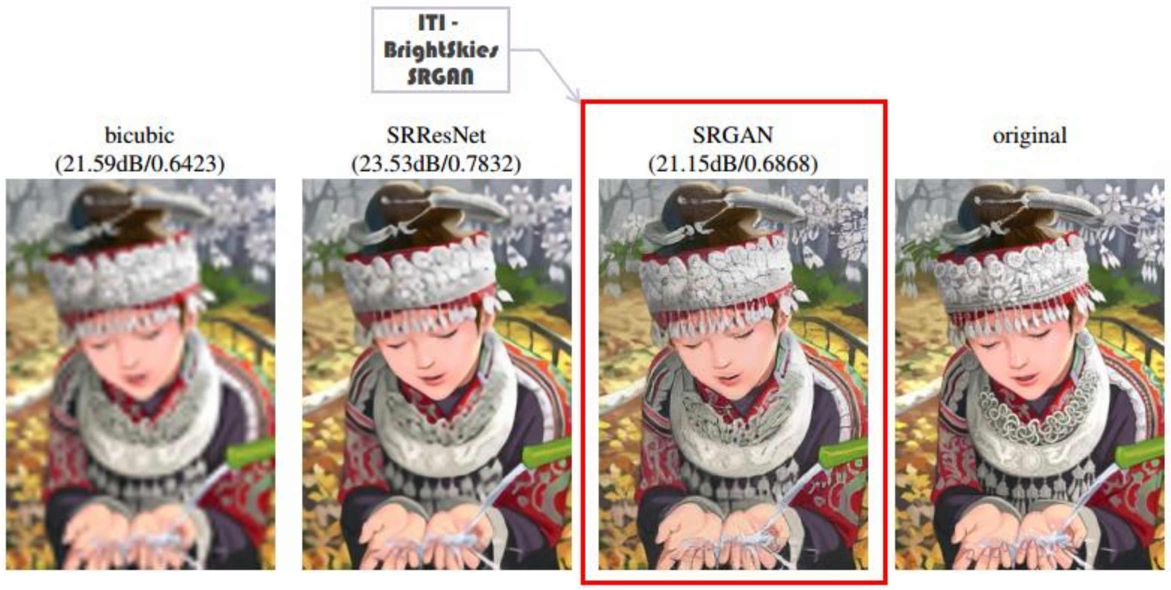


Image (01) – SR-GAN Output

was proposed by researchers at Twitter. The motive of this architecture is to recover finer textures from the image when we upscale it so that it's quality cannot be compromised. There are other methods such as Bilinear Interpolation that can be used to perform this task but they suffer from image information loss and smoothing. In this paper, the authors proposed two architectures the one without GAN (SRResNet) and one with GAN (SRGAN). It is concluded that SRGAN has better accuracy and generate image more pleasing to eyes as compared to SRGAN.

Architecture:

Similar to GAN architectures, the Super Resolution GAN also contains two parts Generator and Discriminator where generator produces some data based on the probability distribution and discriminator tries to guess weather data coming from input dataset or generator. Generator than tries to optimize the generated data so that it can fool the discriminator. Below are the generator and discriminator architectural details:

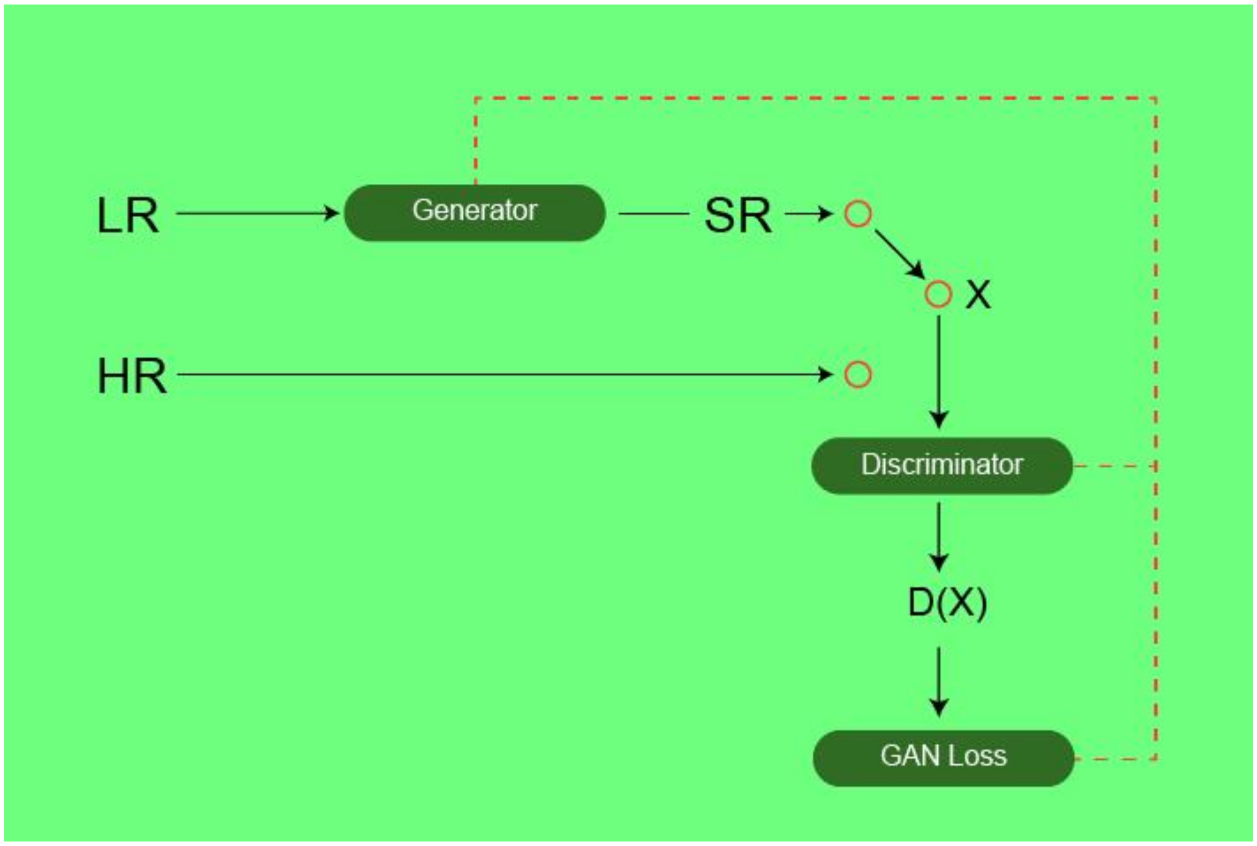


Image (02) – SR-GAN Pipeline

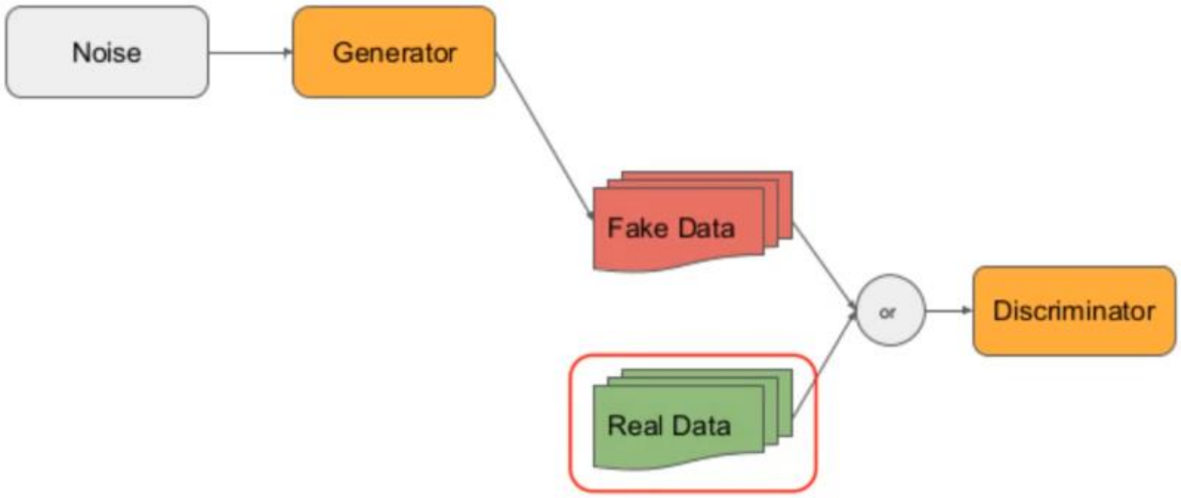
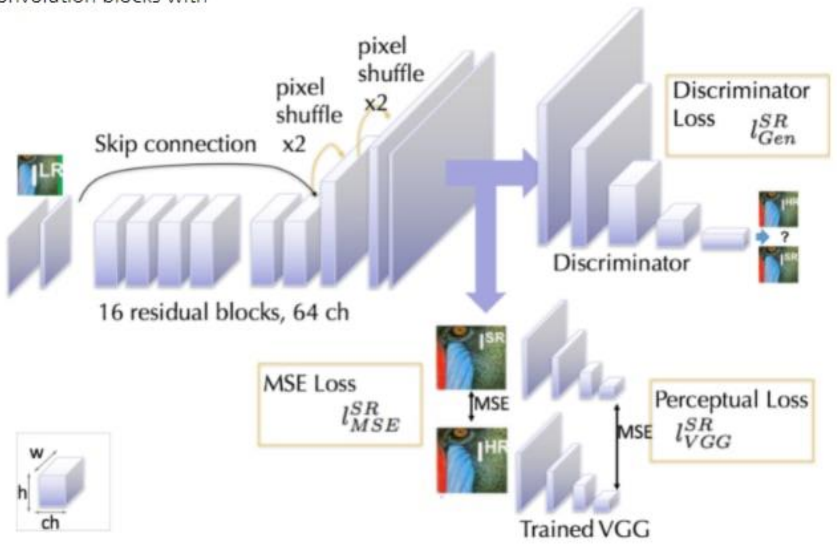


Image (03) - SR-GAN

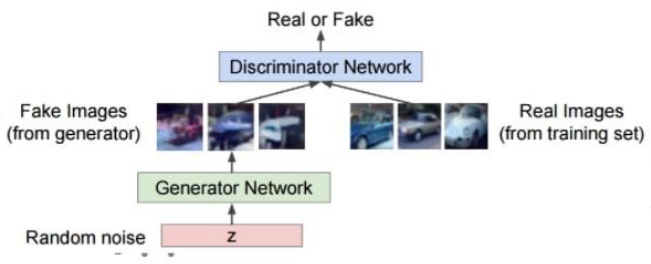
SRResNet

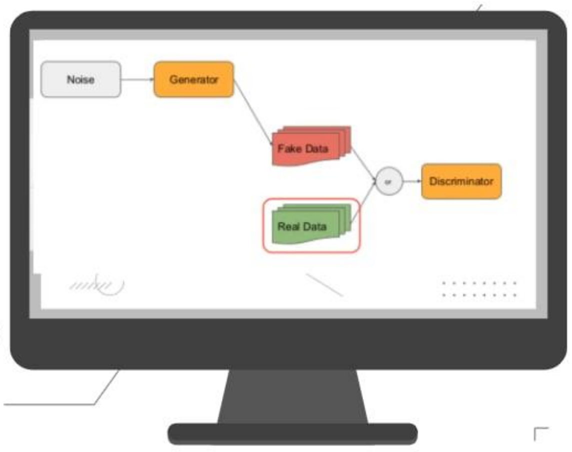
CNN with skip connections
SRResNets replaced simple convolution blocks with residual blocks.



SRGAN

SRGAN has three neural networks, a generator, a discriminator, and a pre-trained VGG19 network on the Imagenet dataset.





Generator Architecture:

The generator architecture contains residual network instead of deep convolution networks because residual networks are easy to train and allows them to be substantially deeper in order to generate better results. This is because the residual network used a type of connections called skip connections.

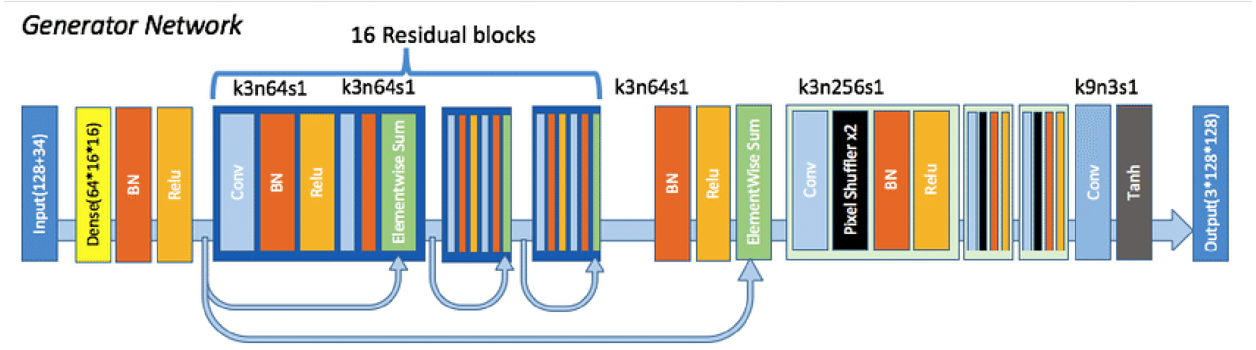


Image (04) – Generator a.

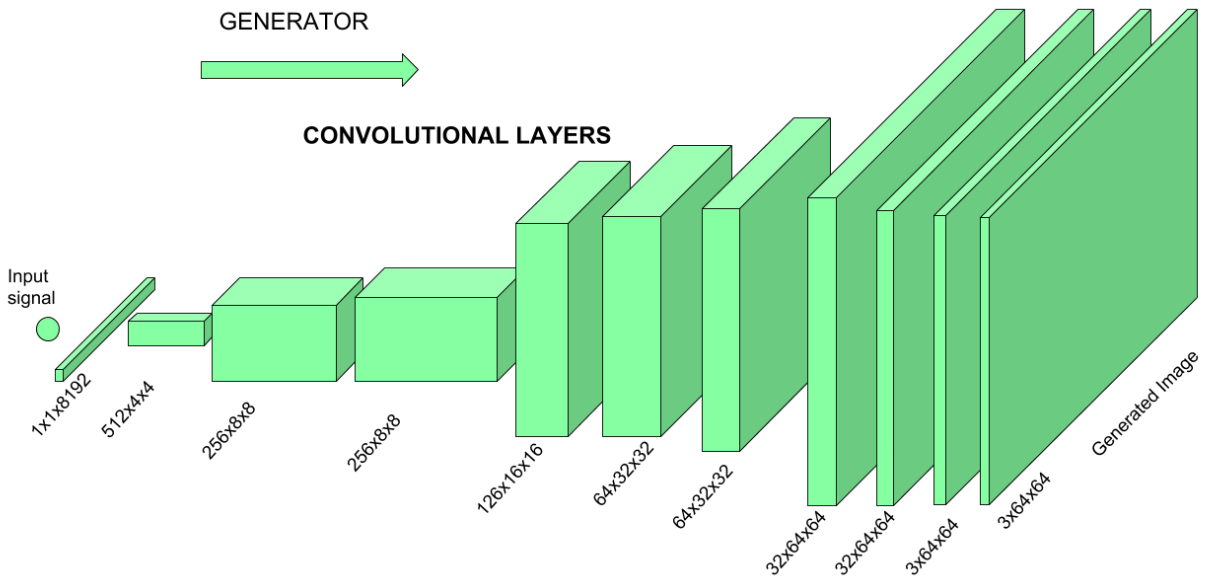


Image (05) – Generator b.

There are B residual blocks (16), originated by ResNet. Within the residual block, two convolutional layers are used, with small 3×3 kernels and 64 feature maps followed by batch-normalization layers and ParametricReLU as the activation function.

The resolution of the input image is increased with two trained sub-pixel convolution layers.

This generator architecture also uses parametric ReLU as an activation function which instead of using a fixed value for a parameter of the rectifier (alpha) like LeakyReLU. It adaptively learns the parameters of rectifier and improves the accuracy at negligible extra computational cost

During the training, A high-resolution image (HR) is downsampled to a low-resolution image (LR). The generator architecture than tries to upsample the image from low resolution to super-resolution. After then the image is passed into the discriminator, the discriminator and tries to distinguish between a super-resolution and High-Resolution image and generate the adversarial loss which then backpropagated into the generator architecture.

Discriminator Architecture:

The task of the discriminator is to discriminate between real HR images and generated SR images. The discriminator architecture used in this paper is similar to DC- GAN architecture with LeakyReLU as activation. The network contains eight convolutional layers with of 3×3 filter kernels, increasing by a factor of 2 from 64 to 512 kernels. Strided convolutions are used to reduce the image resolution each time the number of features is doubled. The resulting 512 feature maps are followed by two dense layers and a leakyReLU applied between and a final sigmoid activation function to obtain a probability for sample classification.

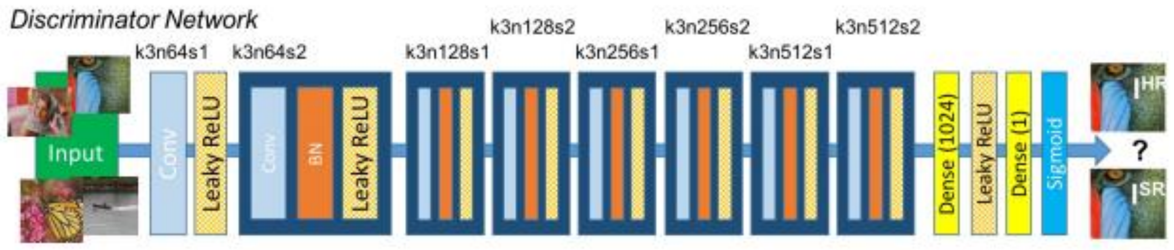


Image (06) – Discriminator a.

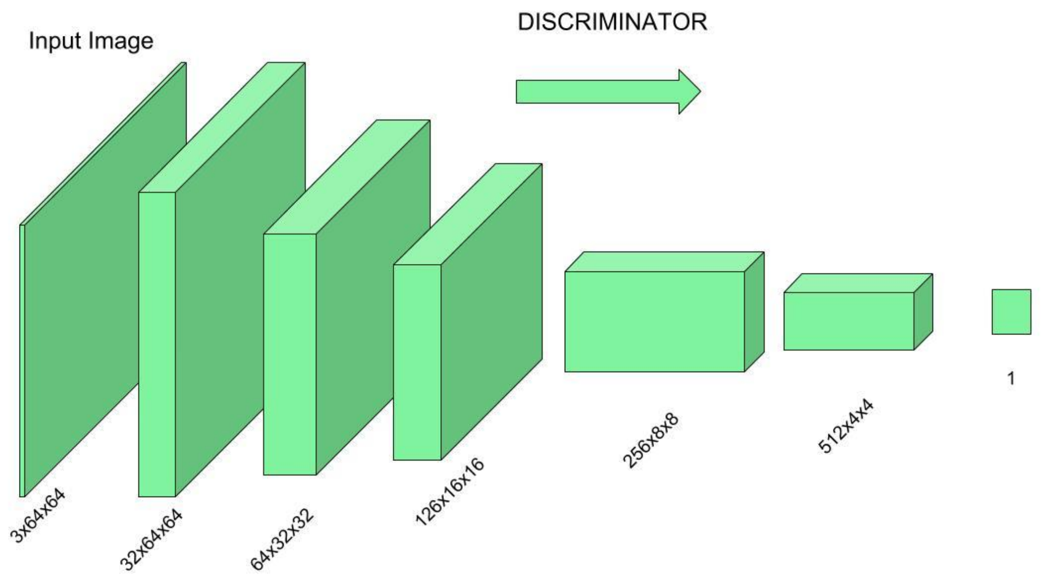


Image (07) – Discriminator b.

3. System Analysis

3.1.Specification

3.1.1. Functional Requirements

- Visualize the image and the result of it as well as feedback for the user about the result.
- Visualize all images uploaded by the user and their corresponding classification
- Build Generator and Discriminator.
- Build GVV-19 for feature extraction.
- Upload image and get SR Image and show it in website.
- Visualize PSNR and SSIM.

3.1.2. Non-Functional Requirements

- Response time: 20 second per image.
- Availability: The tool will be available for everyone, in any time
- Capacity: Only 1 image at a time for all users
- Reliability: The website is using flask
- Usability: The tool will be very easy for all users to use

3.2. Use Case Diagram

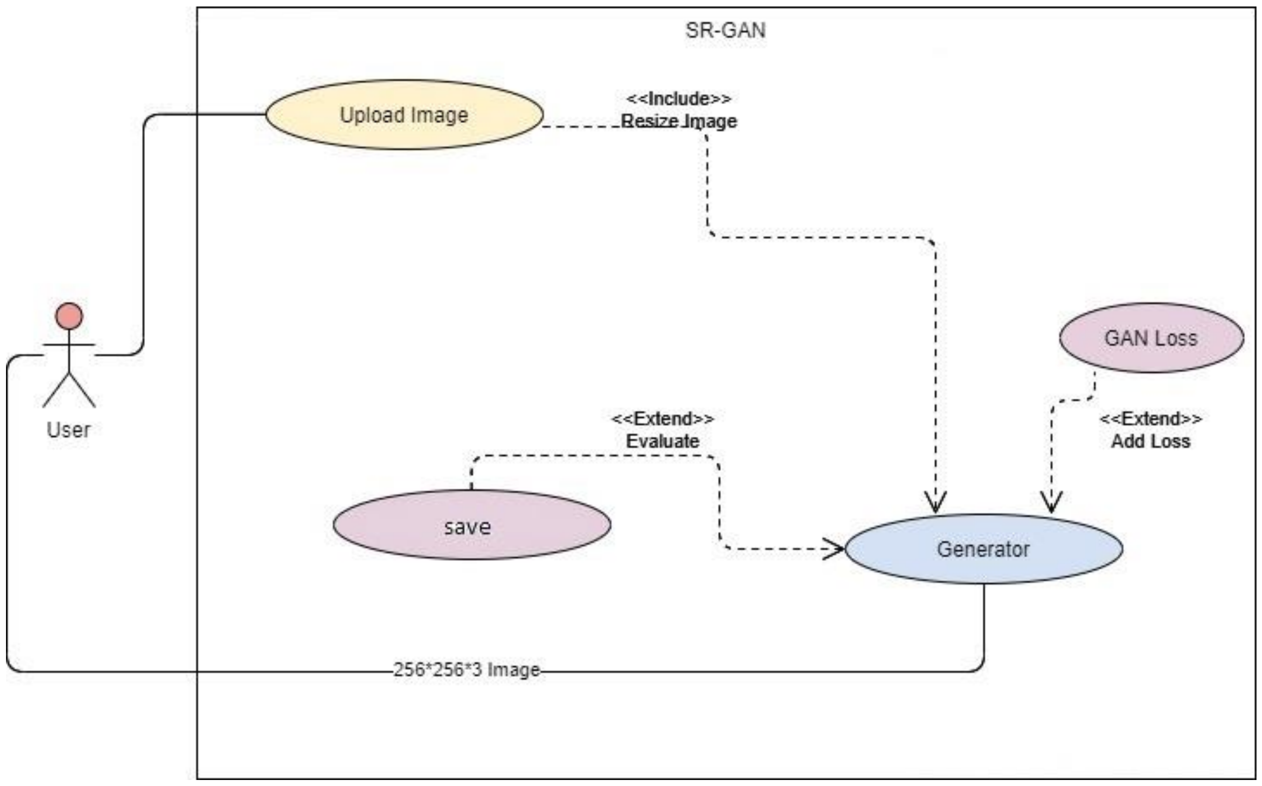
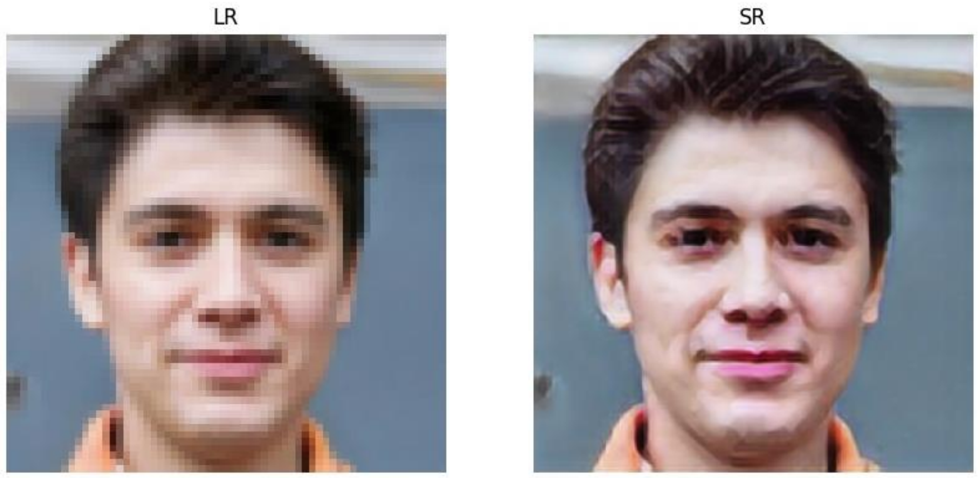


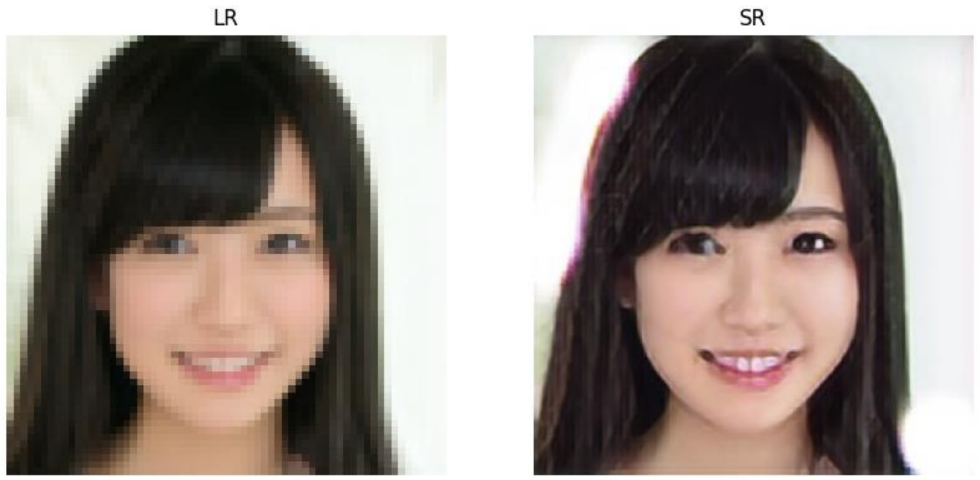
Image (08) -Use Case Diagram

3.3.System test cases

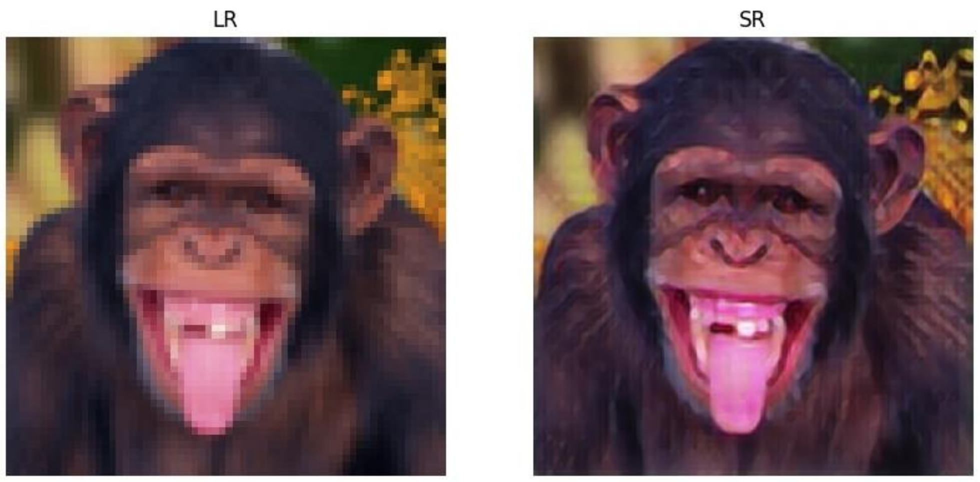
Test case ID	Test case Scenario	Test case Steps	Expected	Result
1	Human face	Using a human photo with 64*64 pixel	High resolution image with 256 * 256 pixel	Pass
2	Human face with white background	Using a human photo with 64*64 pixel	High resolution image with 256 * 256 pixel and get noise in white position	Pass
3	Monkey picture with background	Using 64*64 image	High resolution image with 256 * 256 pixel without noise from human eye perspective.	Pass
4	Monkey picture with white background	Using same image 64*64 without background	High resolution image with 256 * 256 pixel and get noise in white position	Pass
5	Cat image	64 * 64 pixel	Perfect high resolution image with 256 * 256 pixel	Pass



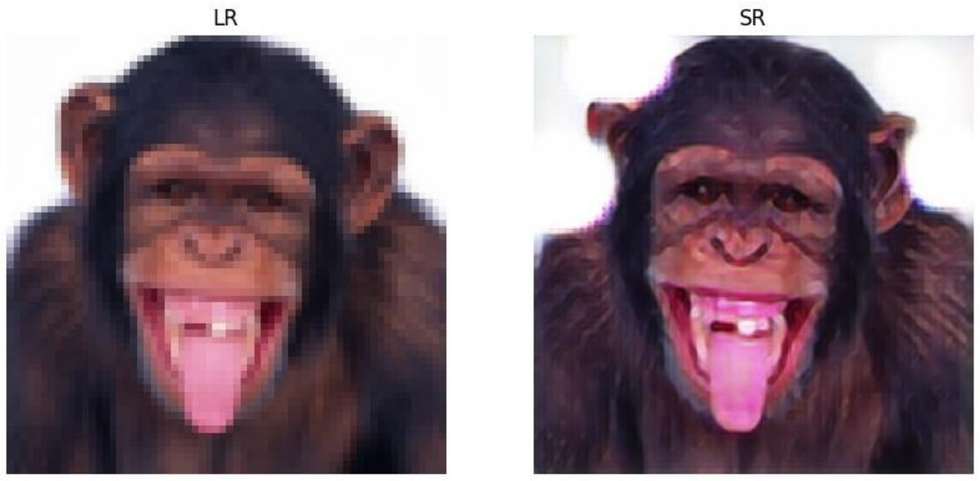
High resolution image with 256 * 256 pixel



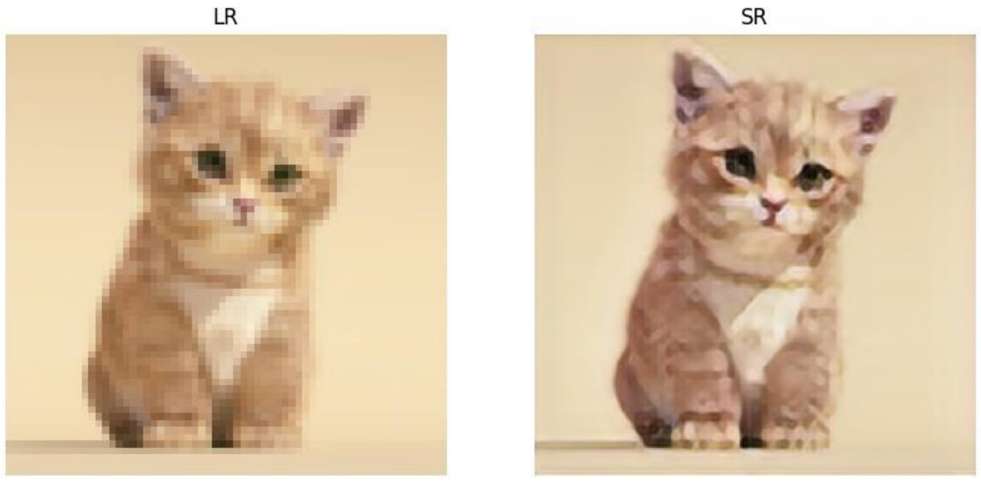
High resolution image with 256 * 256 pixel and get noise in white position



High resolution image with 256 * 256 pixel without noise from human eye perspective.



High resolution image with 256 * 256 pixel and get noise in white position



Perfect high resolution image with 256 * 256 pixel

4. System Design

4.1.System Components

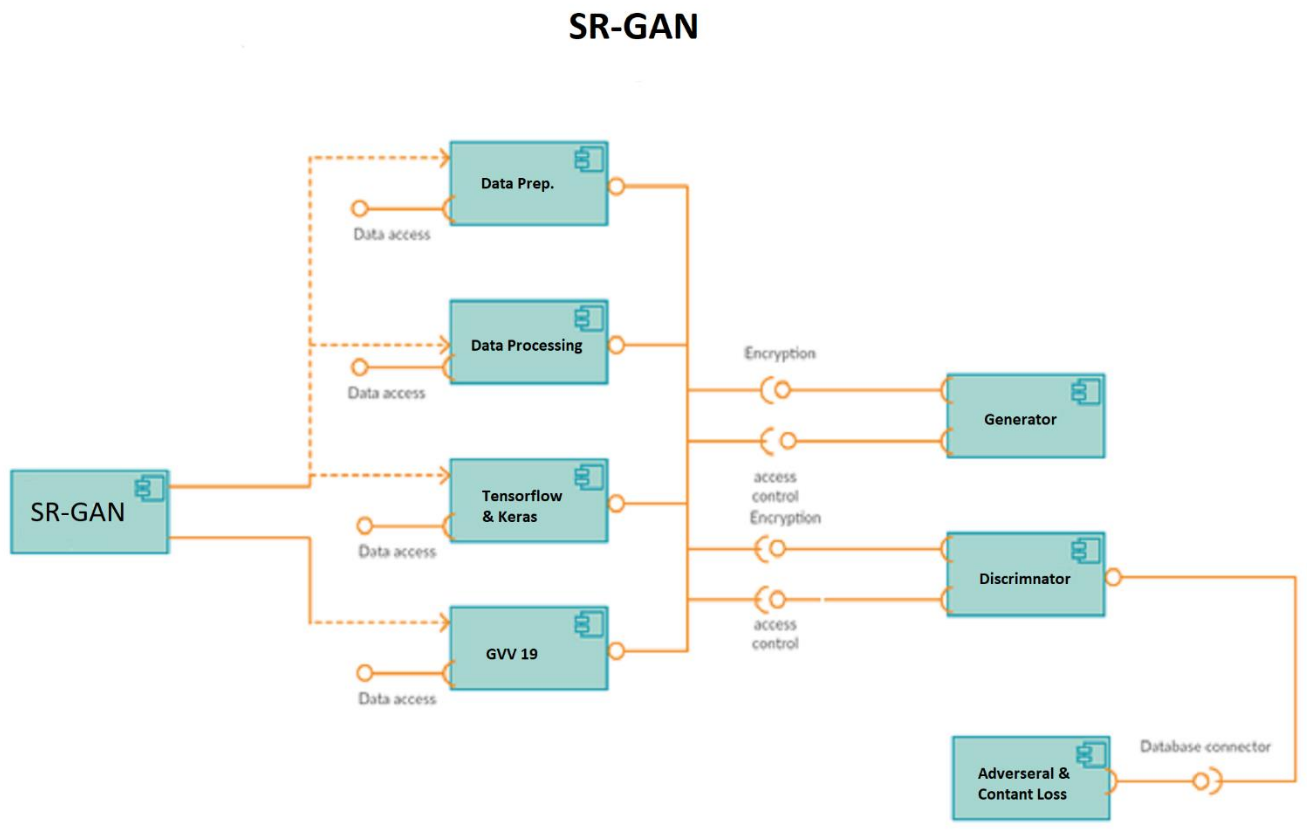


Image (09) - System Components

4.2.Sequence Diagrams

Inference Sequence Diagram

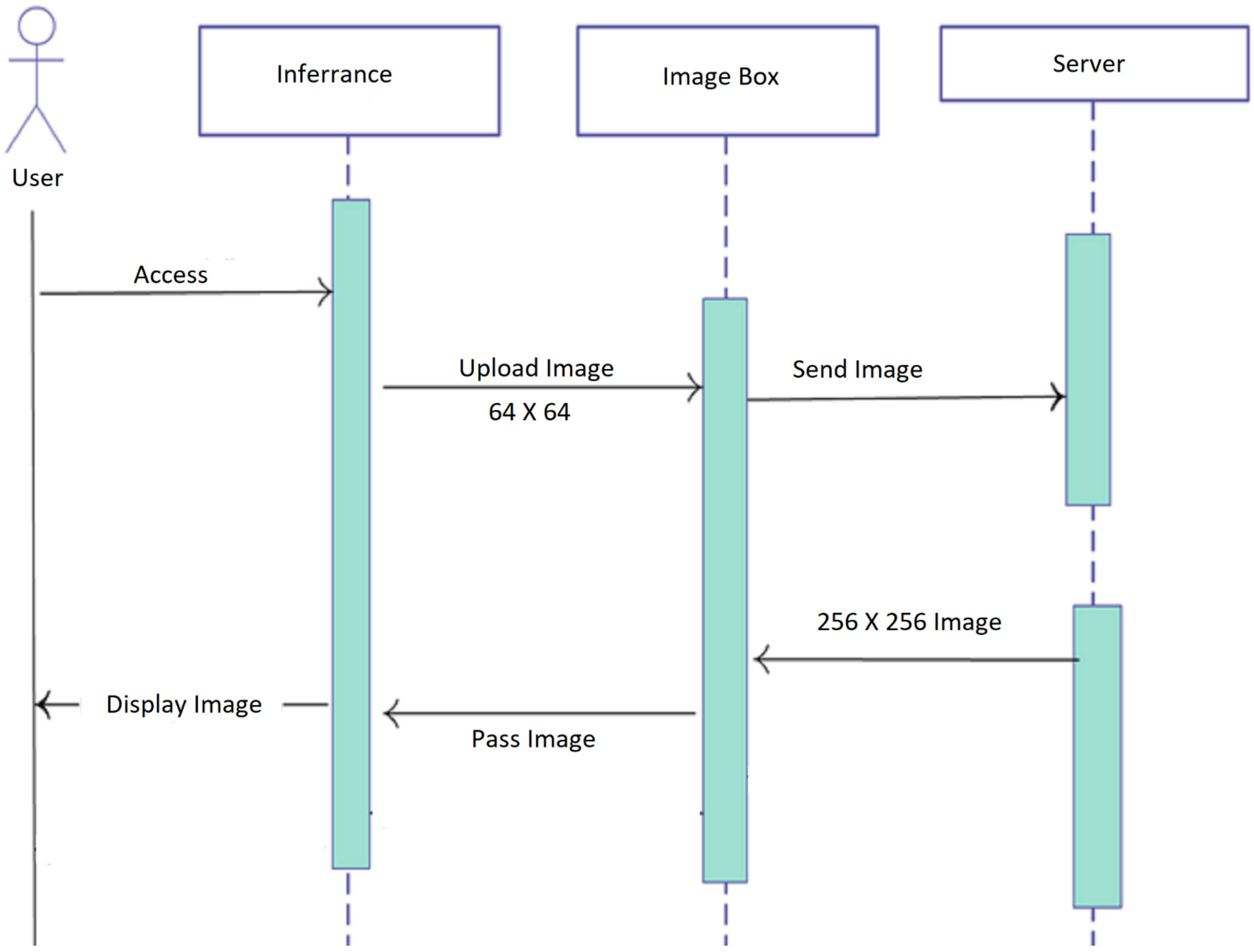


Image (10) - Sequence Diagram

5. Implementation and Testing

Code consists of python modules that work together to run the application in web browser with simple UI to make it easy for non-technical user want to use our service.

6. Loss Function

The SRGAN uses perpectual loss function (LSR) which is the weighted sum of two loss components : content loss and adversarial loss. This loss is very important for the performance of the generator architecture:

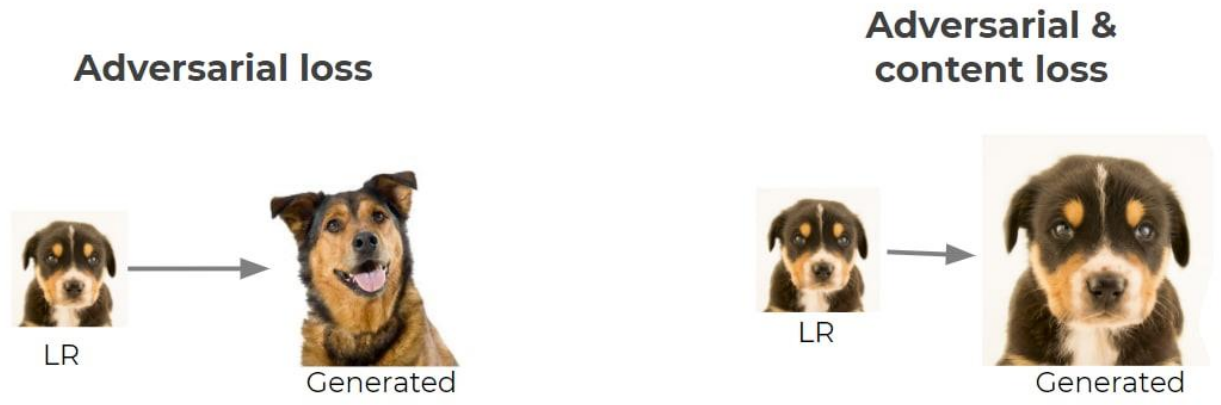
- **Content Loss:** We use two types of content loss in this paper : pixelwise MSE loss for the SRResnet architecture, which is most common MSE loss for image Super Resolution. However MSE loss does not able to deal with high frequency content in the image that resulted in producing overly smooth images. Therefore the authors of the paper decided to use loss of different VGG layers. This VGG loss is based on the ReLU activation layers of the pre-trained 19 layer VGG network. This loss is defined as follows:

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

- **Adversarial Loss:** The Adversarial loss is the loss function that forces the generator to image more similar to high resolution image by using a discriminator that is trained to differentiate between high resolution and super resolution images.

Adversarial loss for original GAN

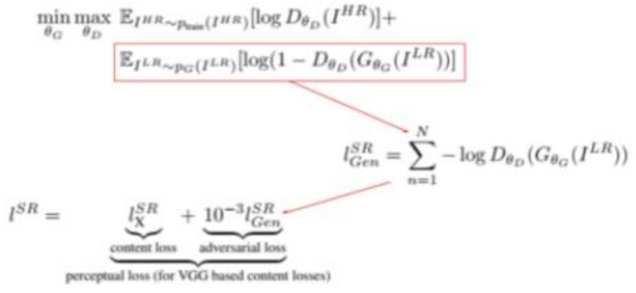
$$l^{SR} = \underbrace{\underbrace{l^{SR}_{X}}_{\text{content loss}} + \underbrace{\boxed{10^{-3} l^{SR}_{Gen}}}_{\text{adversarial loss}}}_{\text{perceptual loss (for VGG based content losses)}}$$



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

- Therefore total content loss of this architecture will be :

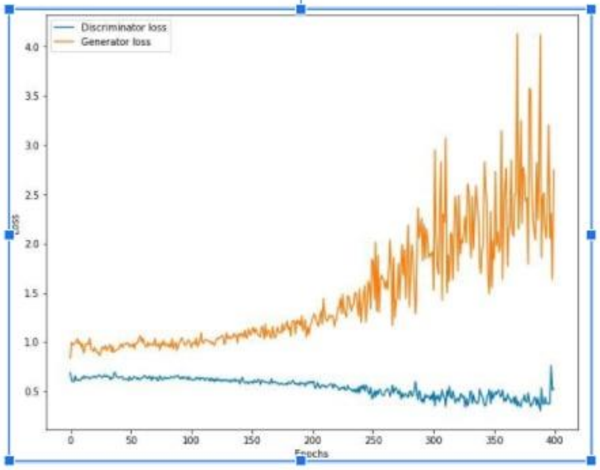
$$l^{SR} = \underbrace{\underbrace{l^{SR}_{X}}_{\text{content loss}} + \underbrace{10^{-3} l^{SR}_{Gen}}_{\text{adversarial loss}}}_{\text{perceptual loss (for VGG based content losses)}} \tag{3}$$

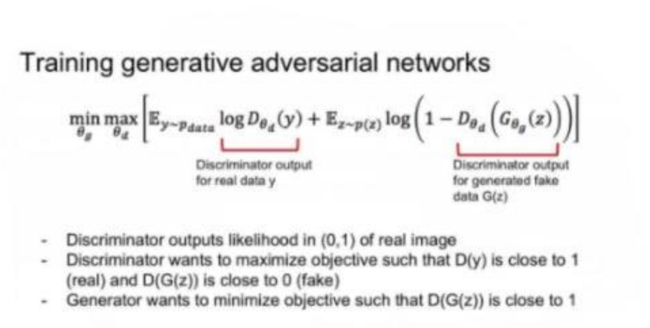


$$\min_{\theta_G}\max_{\theta_D}\ \mathbb{E}_{I^{HR}\sim p_{train}(I^{HR})}[\log D_{\theta_D}(I^{HR})]+$$
$$\boxed{\mathbb{E}_{I^{LR}\sim p_G(I^{LR})}[\log(1-D_{\theta_D}(G_{\theta_G}(I^{LR}))]}$$

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

$$l^{SR}=\underbrace{\underbrace{l^{SR}_{X}}_{\text{content loss}}+\underbrace{10^{-3}l^{SR}_{Gen}}_{\text{adversarial loss}}}_{\text{perceptual loss (for VGG based content losses)}}$$





Training generative adversarial networks

$$\min_{\theta_g}\max_{\theta_d}\left[\mathbb{E}_{y\sim p_{data}}\underbrace{\log D_{\theta_d}(y)}_{\text{Discriminator output for real data y}}+\mathbb{E}_{z\sim p(z)}\log\left(1-\underbrace{D_{\theta_d}\left(G_{\theta_g}(z)\right)}_{\text{Discriminator output for generated fake data G(z)}}\right)\right]$$

- Discriminator outputs likelihood in (0,1) of real image
- Discriminator wants to maximize objective such that D(y) is close to 1 (real) and D(G(z)) is close to 0 (fake)
- Generator wants to minimize objective such that D(G(z)) is close to 1

7. Discussion

The main contributions of this paper is:

- This paper generates state-of-the-art results on upsampling (4x) as measured by PNSR (Peak Signal-to-Noise Ratio) and SSIM(Structural Similarity) with 16 block deep SRResNet network optimize for MSE.
- The authors propose a new Super Resolution GAN in which the authors replace the MSE based content loss with the loss calculated on VGG layer.
- SRGAN was able to generate state-of-the-art results which the author validated with extensive Mean Opinion Score (MOS) test on three public benchmark datasets.

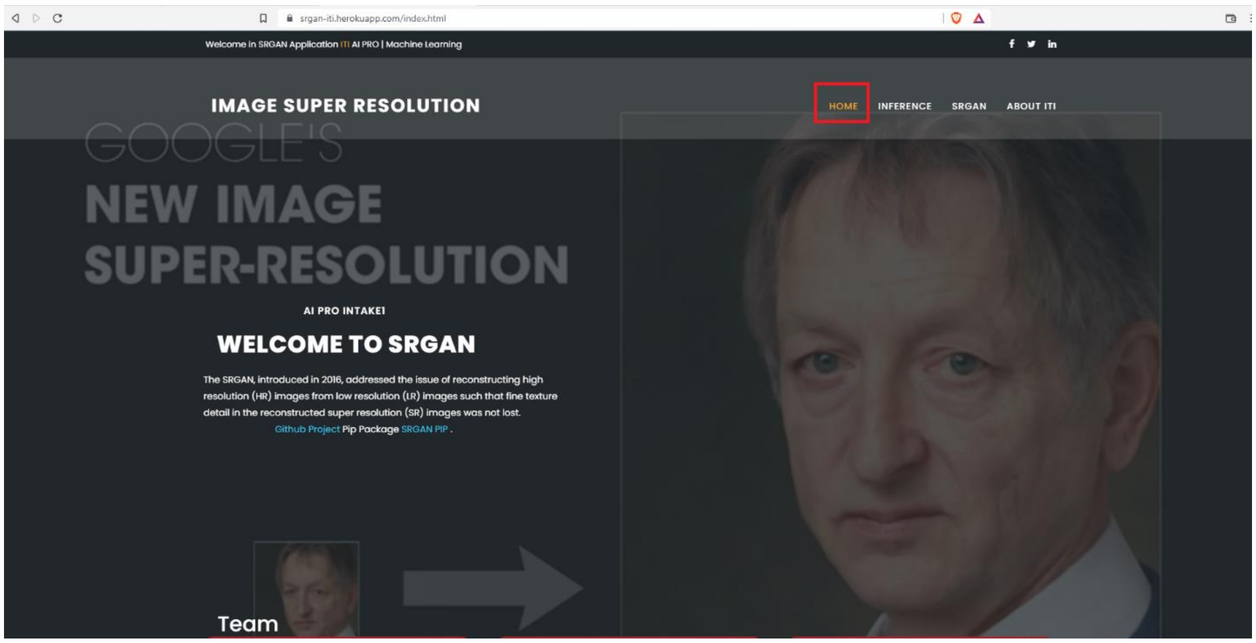
8.Conclusion

In the above layer MSE means we take simple mean squared pixelwise error as content loss, VGG22 indicate the feature map obtained by the 2nd convolution (after activation) before the 2nd maxpooling layer within the VGG19 network and wwe calculate the VGG loss using formula described above. This loss is thus loss on the low-level features. Similarly VGG 54 uses loss calculated on the the feature map obtained by the 4th convolution (after activation) before the 5th maxpooling layer within the VGG19 network. This represents loss on higher level features from deeper network layers with more potential to focus on the content of the images

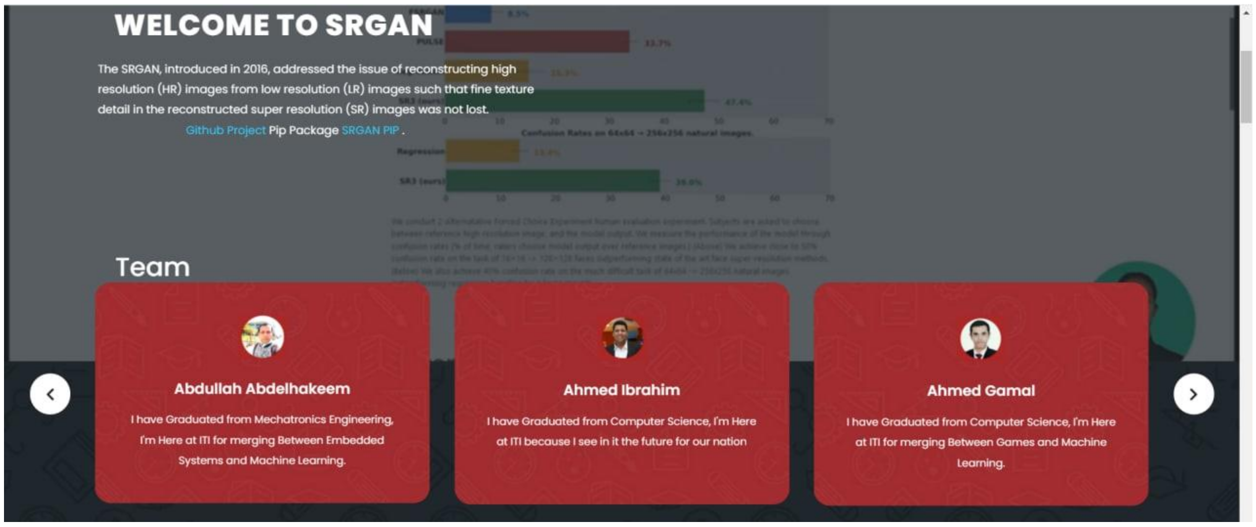
Set5	SRResNet	SRGAN
PSNR	32.05	29.40
SSIM	0.9019	0.8472
Set14		
PSNR	28.49	26.02
SSIM	0.8184	0.7397
DIV2K		
PSNR	34.22	30.03
SSIM	0.8431	0.8676
FLICKR2K		
PSNR	29,30	27.24
SSIM	0.7814	0.6992
BSD100		
PSNR	27.58	25.16
SSIM	0.7620	0.6688
ANIMAL FACES		
PSNR	33.21	30.83
SSIM	0.36	0.39

9.User Manual

You can browse the website through the home page



The SRGAN, introduced in 2016, addressed the issue of reconstructing high resolution (HR) images from low resolution (LR) images such that fine texture detail in the reconstructed super resolution (SR) images was not lost.
Github Project Pip Package SRGAN PIP .



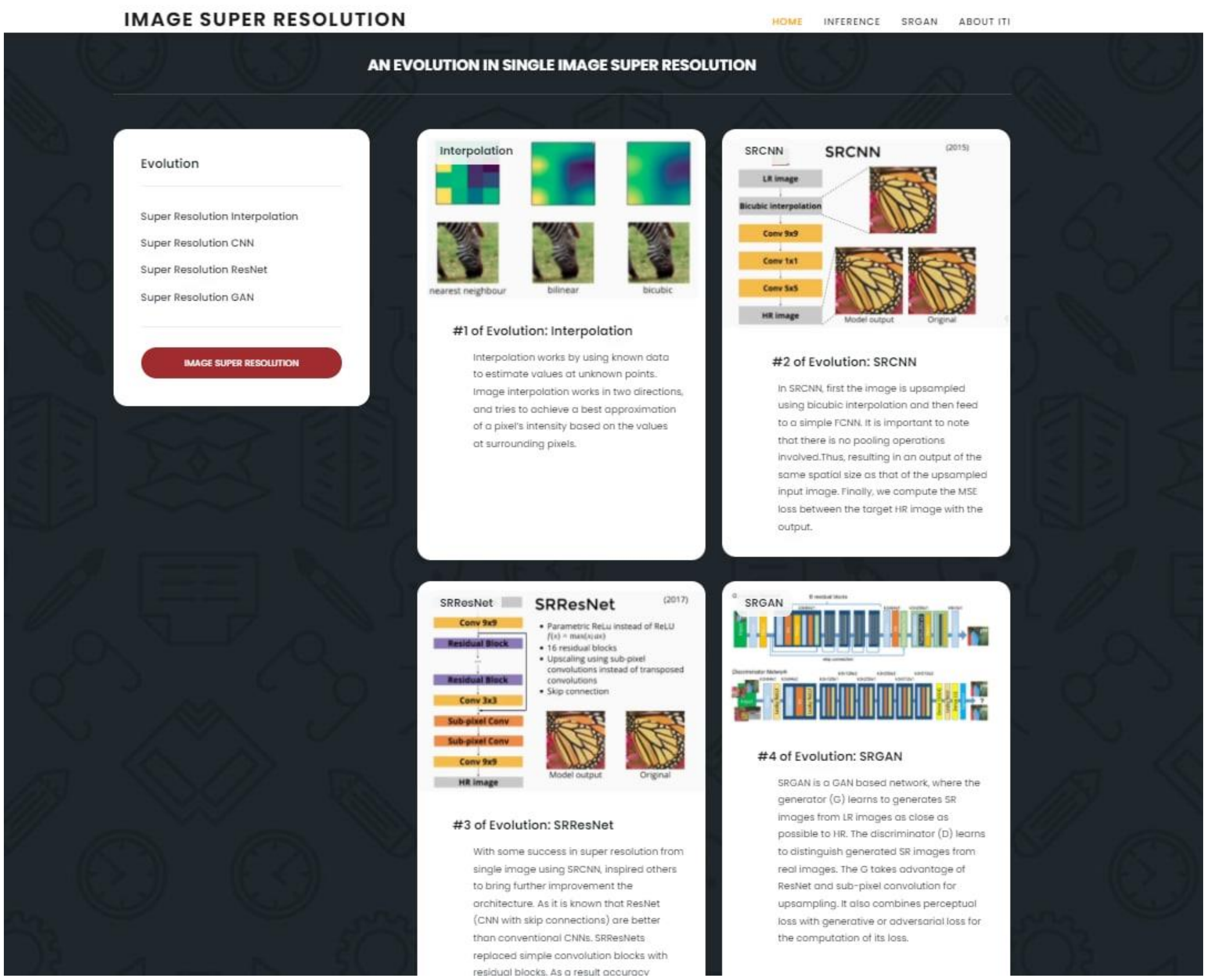
The SRGAN, introduced in 2016, addressed the issue of reconstructing high resolution (HR) images from low resolution (LR) images such that fine texture detail in the reconstructed super resolution (SR) images was not lost.
Github Project Pip Package SRGAN PIP .

Abdullah Abdelhakeem
I have Graduated from Mechatronics Engineering, I'm Here at ITI for merging Between Embedded Systems and Machine Learning.

Ahmed Ibrahim
I have Graduated from Computer Science, I'm Here at ITI because I see in it the future for our nation

Ahmed Gamal
I have Graduated from Computer Science, I'm Here at ITI for merging Between Games and Machine Learning.

in “AN EVOLUTION IN SINGLE IMAGE SUPER RESOLUTION” part you can know more about image super resolution history



AN EVOLUTION IN SINGLE IMAGE SUPER RESOLUTION

#1 of Evolution: Interpolation

Interpolation works by using known data to estimate values at unknown points. Image interpolation works in two directions, and tries to achieve a best approximation of a pixel's intensity based on the values at surrounding pixels.

#2 of Evolution: SRCNN

In SRCNN, first the image is upsampled using bicubic interpolation and then feed to a simple FCNN. It is important to note that there is no pooling operations involved.Thus, resulting in an output of the same spatial size as that of the upsampled input image. Finally, we compute the MSE loss between the target HR image with the output.

- Parametric ReLu instead of ReLU $f(x) = \max(x, ax)$
- 16 residual blocks
- Upscaling using sub-pixel convolutions instead of transposed convolutions
- Skip connection

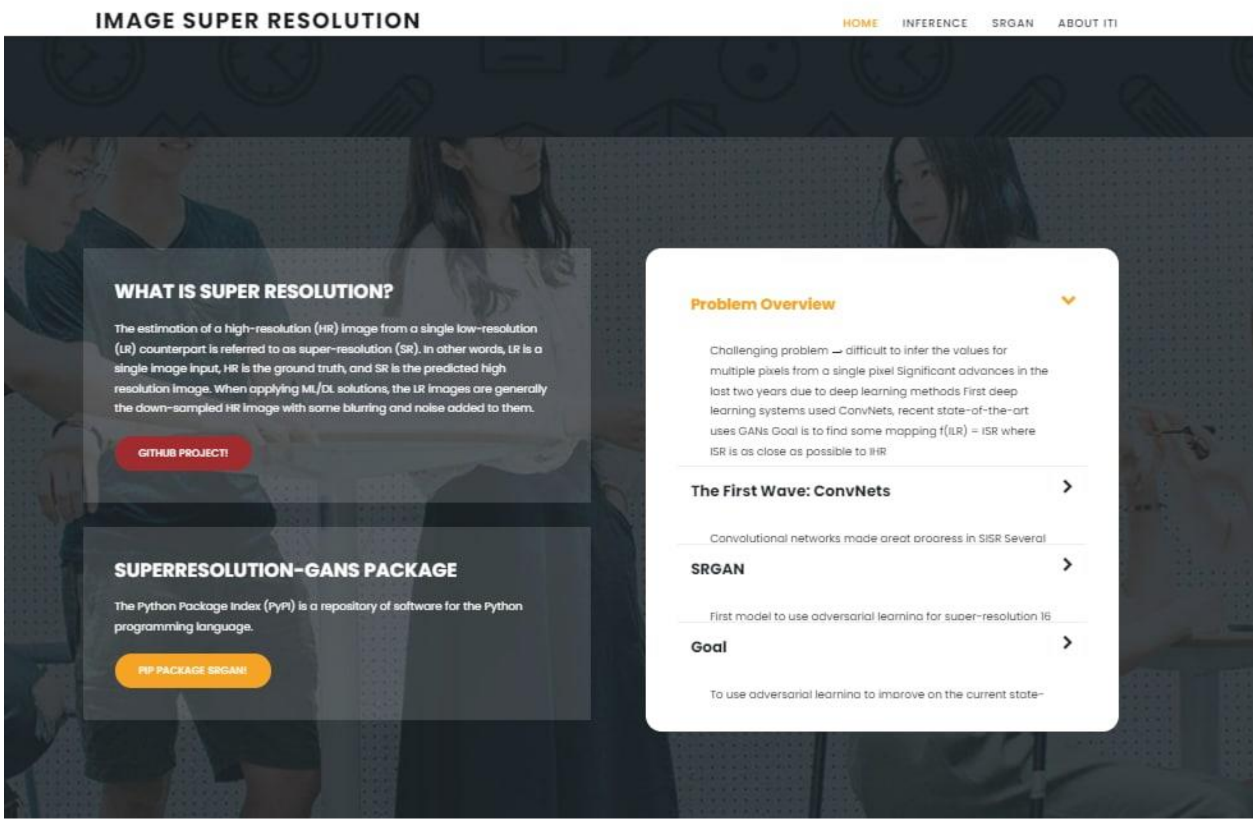
#3 of Evolution: SRResNet

With some success in super resolution from single image using SRCNN, inspired others to bring further improvement the architecture. As it is known that ResNet (CNN with skip connections) are better than conventional CNNs. SRResNets replaced simple convolution blocks with residual blocks. As a result accuracy

#4 of Evolution: SRGAN

SRGAN is a GAN based network, where the generator (G) learns to generates SR images from LR images as close as possible to HR. The discriminator (D) learns to distinguish generated SR images from real images. The G takes advantage of ResNet and sub-pixel convolution for upsampling. It also combines perceptual loss with generative or adversarial loss for the computation of its loss.

in the “Home section also you can know more about the problem



WHAT IS SUPER RESOLUTION?

The estimation of a high-resolution (HR) image from a single low-resolution (LR) counterpart is referred to as super-resolution (SR). In other words, LR is a single image input, HR is the ground truth, and SR is the predicted high resolution image. When applying ML/DL solutions, the LR images are generally the down-sampled HR image with some blurring and noise added to them.

SUPERRESOLUTION-GANS PACKAGE

The Python Package Index (PyPI) is a repository of software for the Python programming language.

Problem Overview

Challenging problem → difficult to infer the values for multiple pixels from a single pixel Significant advances in the last two years due to deep learning methods First deep learning systems used ConvNets, recent state-of-the-art uses GANs Goal is to find some mapping f(ILR) = ISR where ISR is as close as possible to IHR

The First Wave: ConvNets

Convolutional networks made great progress in SISR Several

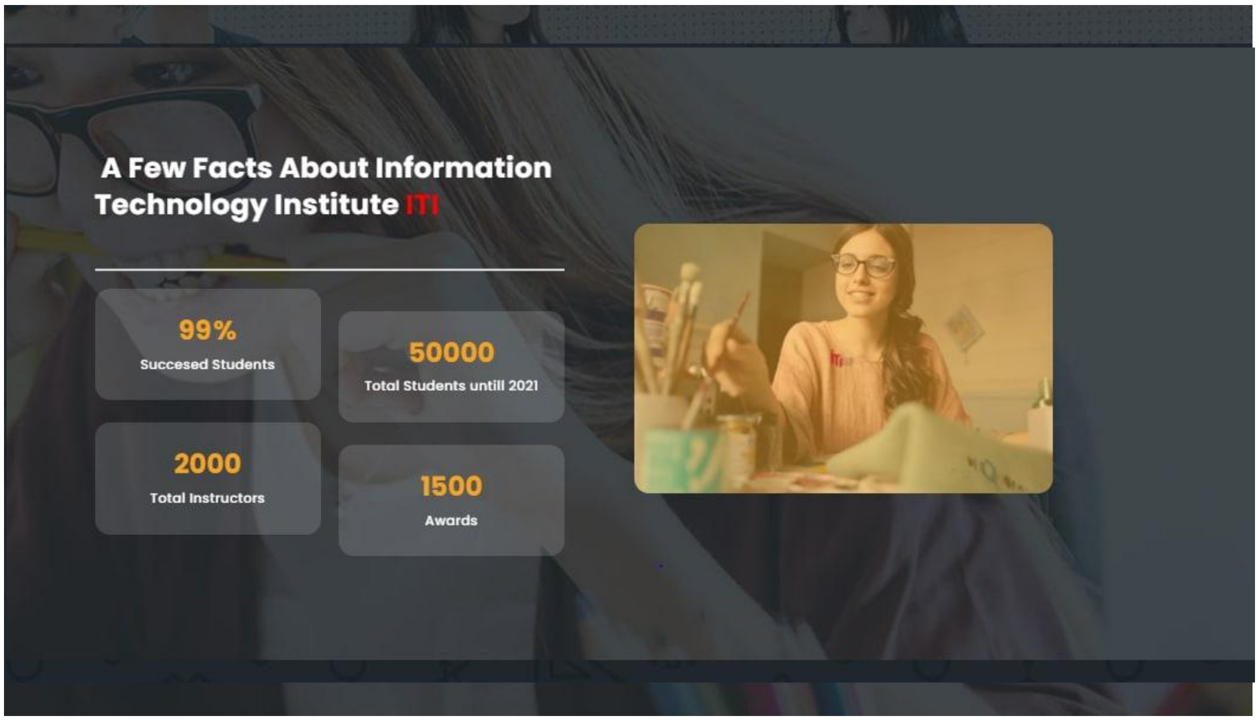
SRGAN

First model to use adversarial learning for super-resolution 16

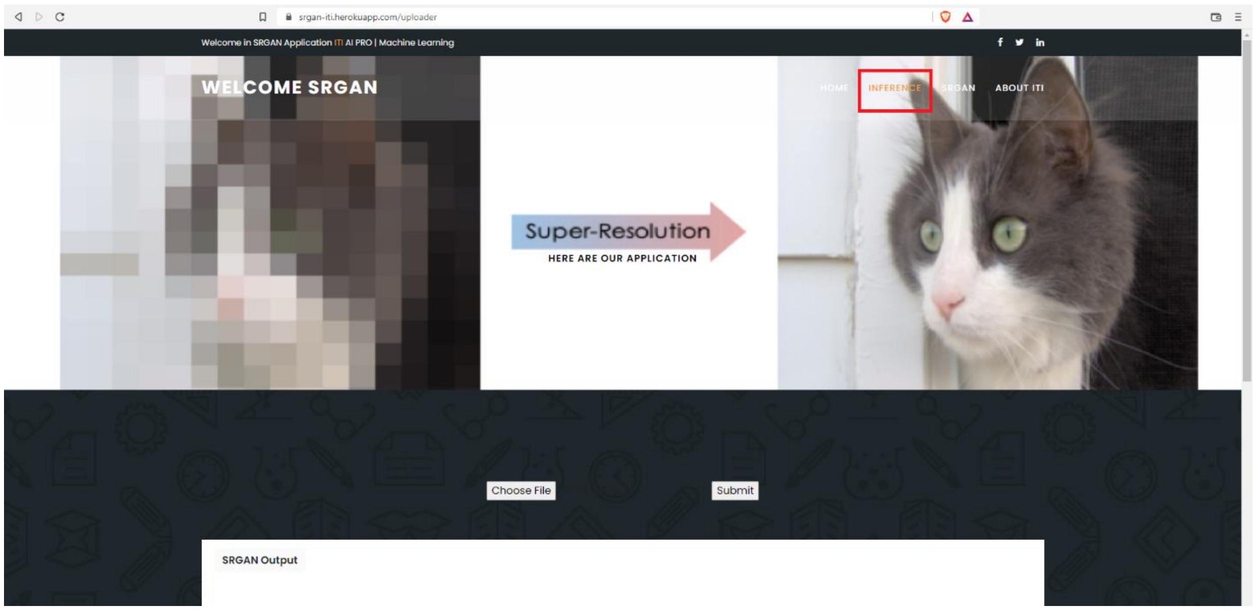
Goal

To use adversarial learning to improve on the current state-

Also, you can know more about the courses that we needed to learn to do this project:

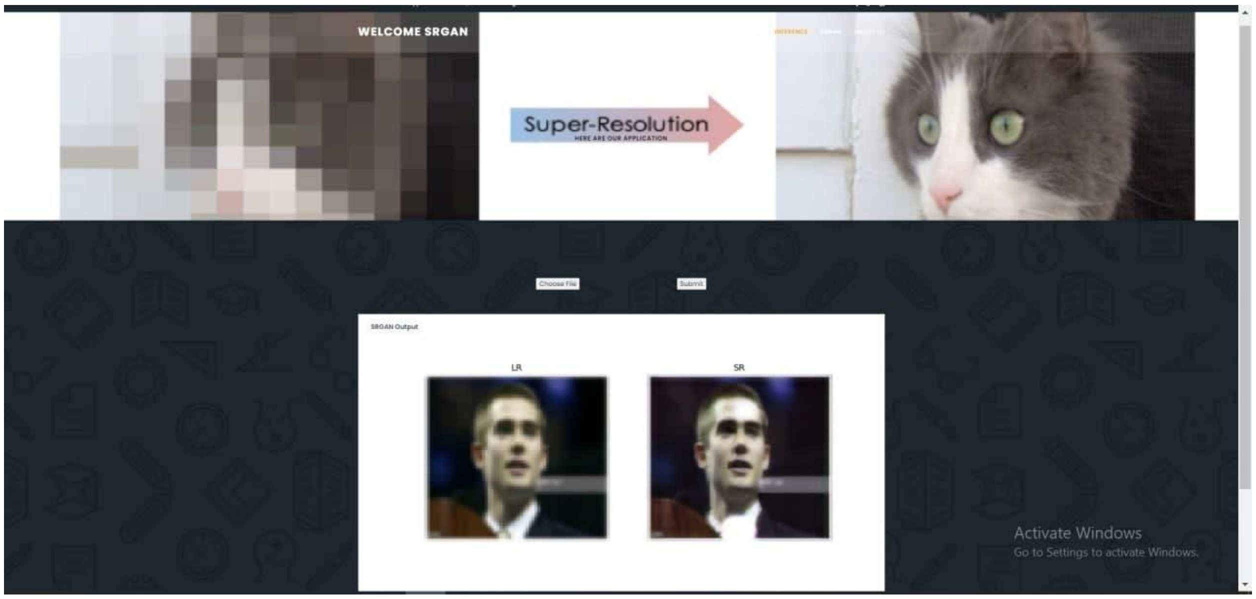

A Few Facts About Information Technology Institute ITI
99%
Succesed Students
50000
Total Students untill 2021
2000
Total Instructors
1500
Awards

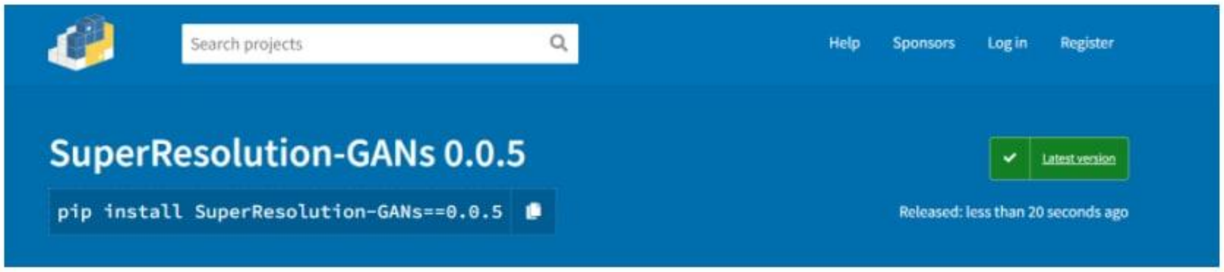
And now you can enjoy trying our model from "inference" Section



Just click on the "Choose File" button and select your image you want to try it.

After that, click on the "Done" button and wait for your image to be processed and it will appear to you as in this figure

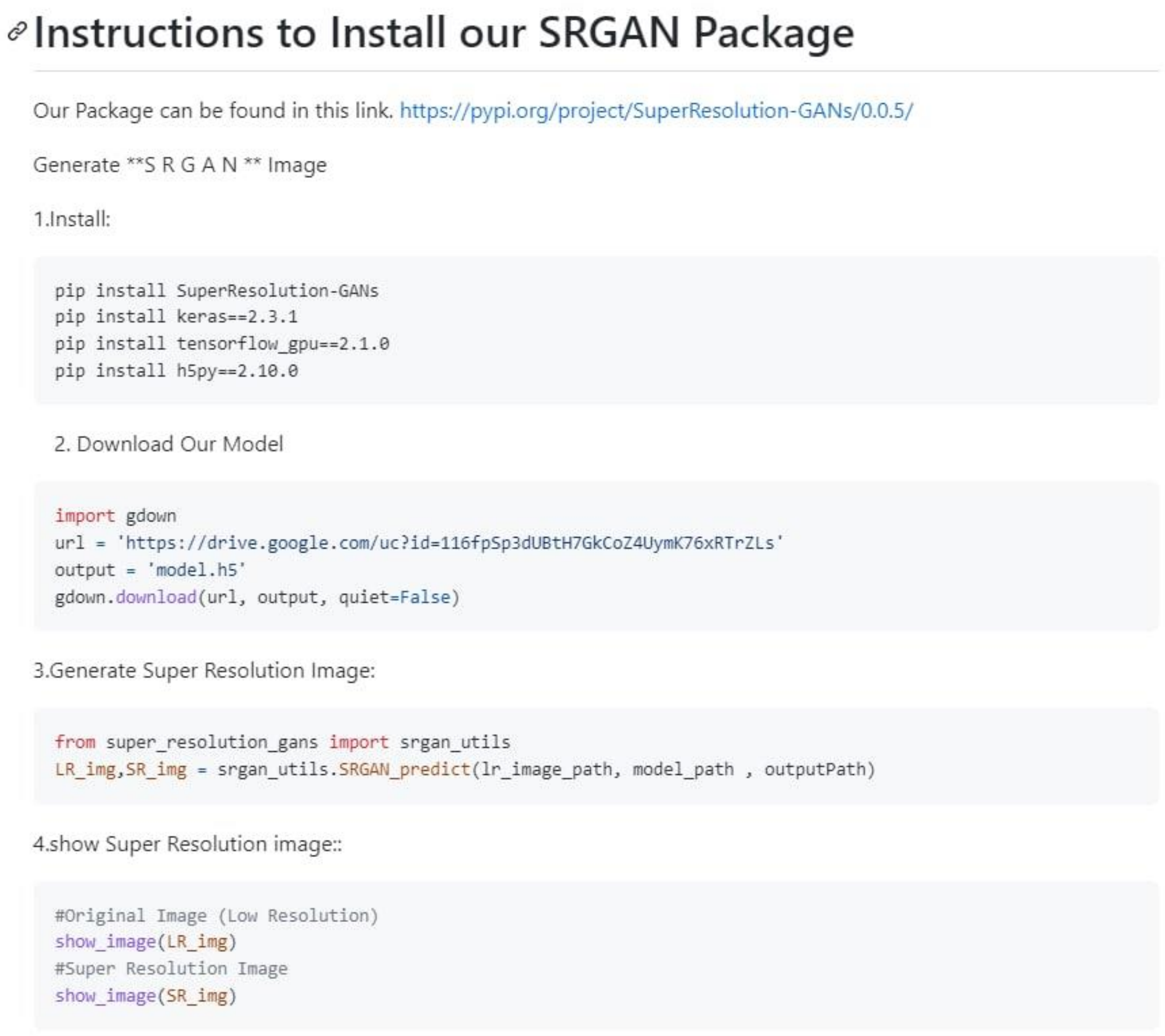




SuperResolution-GANs 0.0.5

```
pip install SuperResolution-GANs==0.0.5
```

Released: less than 20 seconds ago



Instructions to Install our SRGAN Package

Our Package can be found in this link. https://pypi.org/project/SuperResolution-GANs/0.0.5/

Generate **S R G A N ** Image

1.Install:

```
pip install SuperResolution-GANs
pip install keras==2.3.1
pip install tensorflow_gpu==2.1.0
pip install h5py==2.10.0
```

2. Download Our Model

```
import gdown
url = 'https://drive.google.com/uc?id=116fpSp3dUBtH7GkCoZ4UymK76xRTrZLs'
output = 'model.h5'
gdown.download(url, output, quiet=False)
```

3.Generate Super Resolution Image:

```
from super_resolution_gans import srgan_utils
LR_img,SR_img = srgan_utils.SRGAN_predict(lr_image_path, model_path , outputPath)
```

4.show Super Resolution image::

```
#Original Image (Low Resolution)
show_image(LR_img)
#Super Resolution Image
show_image(SR_img)
```

10.References

- SRGAN PAPER
- Jaffe, Luke, Shiv Sundram, and Christian Martinez-Nieves. "Super-Resolution to Improve Classification Accuracy of Low-Resolution Images."
- Ledig, Christian, et al. "Photo-realistic single image super-resolution using a generative adversarial network." arXiv preprint (2016).
- Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.
- 1909.07113.pdf (arxiv.org)
- Image Super Resolution | Deep Learning for Image Super Resolution (analyticsvidhya.com)
- Using SRGANs to Generate Photo-realistic Images [Tutorial] | Packet Hub (packtpub.com)
- SR-Resnet
- Image Super Resolution
- SRGAN_Paper.pdf - Google Drive
- Photo-Realistic Single Image Super-Resolution
- https://developers.google.com/machine-learning/gan/loss
- https://www.mathworks.com/help/images/single-image-super-resolution-using-deep-learning.html
- https://link.springer.com/chapter/10.1007/978-3-319-10593-2_25
- https://web.cs.hacettepe.edu.tr/~erkut/cmp717.s18/materials/w09-presentation-a.pdf
- Aly, H.A., Dubois, E.: Image up-sampling using total-variation regularization with a new observation model.
- Glasner, D., Bagon, S., Irani, M.: Super-resolution from a single image. In: ICCV (2009)
- Image upsampling via texture hallucination. In: ICCP (2010)
- Irani, M., Peleg, S.: Improving resolution by image registration. CGVIP 53(3), 231–239 (1991) Google Scholar