SUPER RESOLUTION

Computer Vision Project

2021/2022

Fabrizio Rossi 1815023
Matteo Orsini 1795119

Super Resolution

- Process of recovering High-Resolution (HR) images from Low-Resolution (LR) images
- **SISR:** Super-Resolution (SR) image reconstructed from single LR

LR

SR

HR

Super Resolution

- SR problem can be modeled as a Supervised Learning task

LR = input

HR = labels

Paper - Contributions

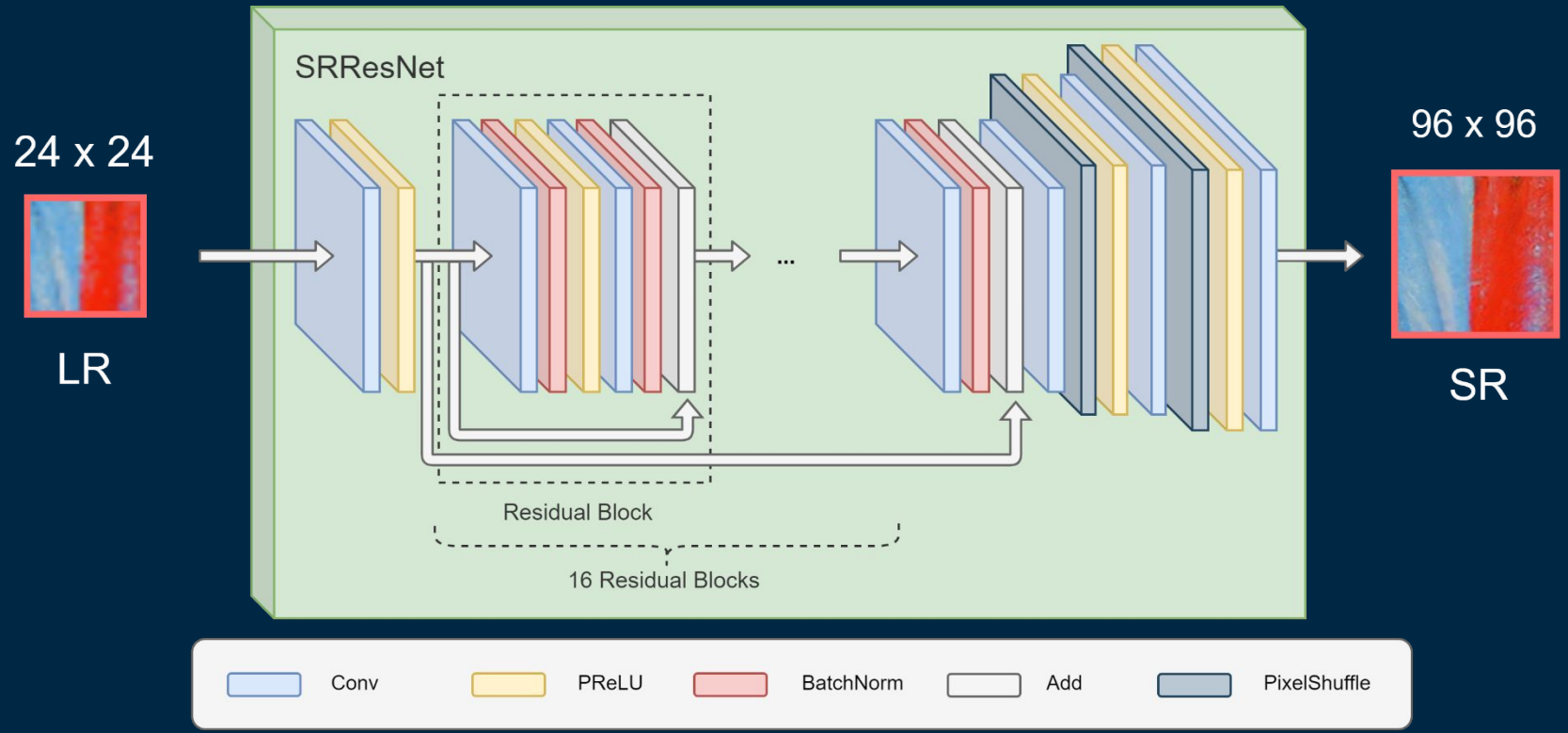
- *Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network*, Ledig et al. (2017)
- Contributions of the paper:
  - SRResNet
    - Baseline model
  - SRGAN
    - Improved model
    - Use of a VGG-based perceptual loss

SRResNet

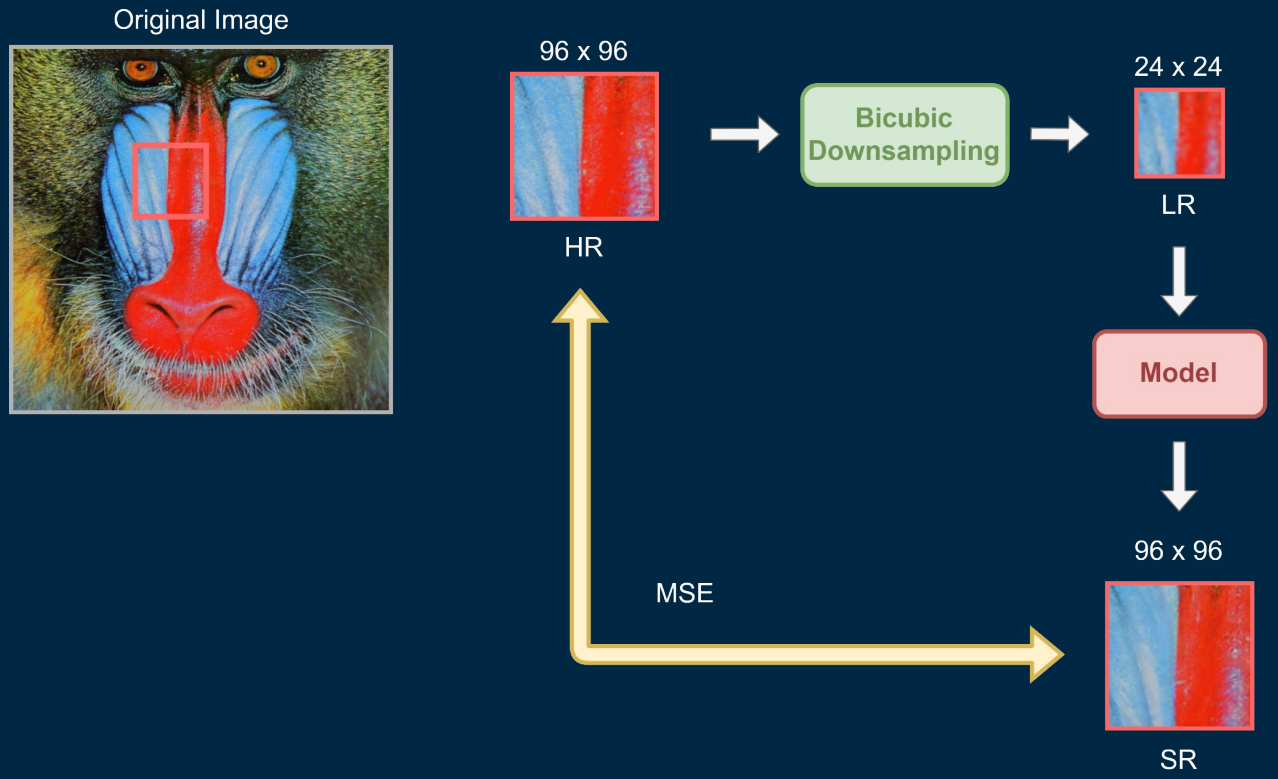
Baseline model

01

SRResNet - Architecture



SRResNet - Training

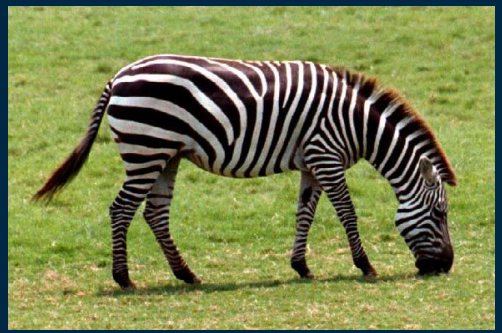


SRResNet – Results

LR

SR

HR



SRResNet – Results

LR

SR

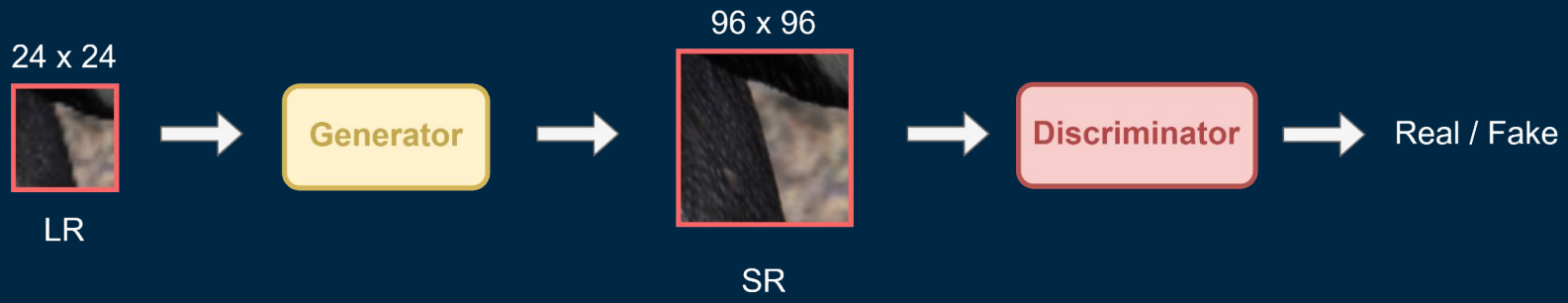
HR

SRGAN

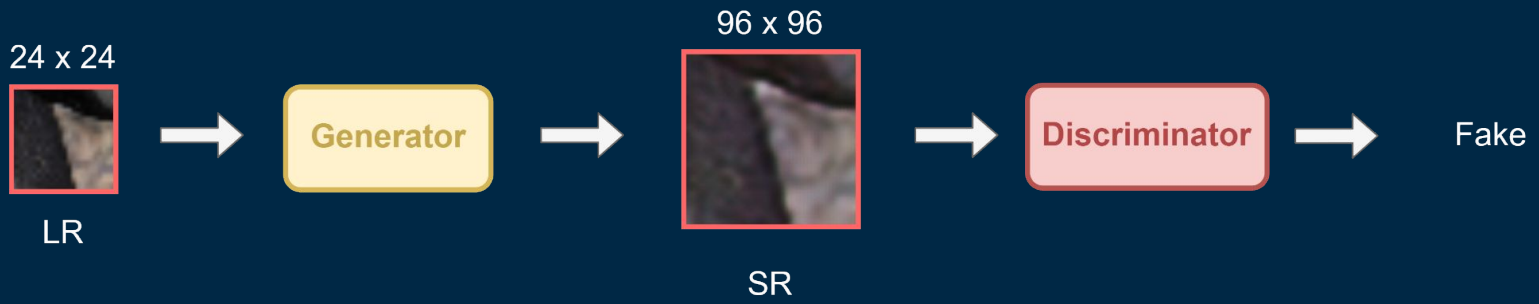
Improved model

02

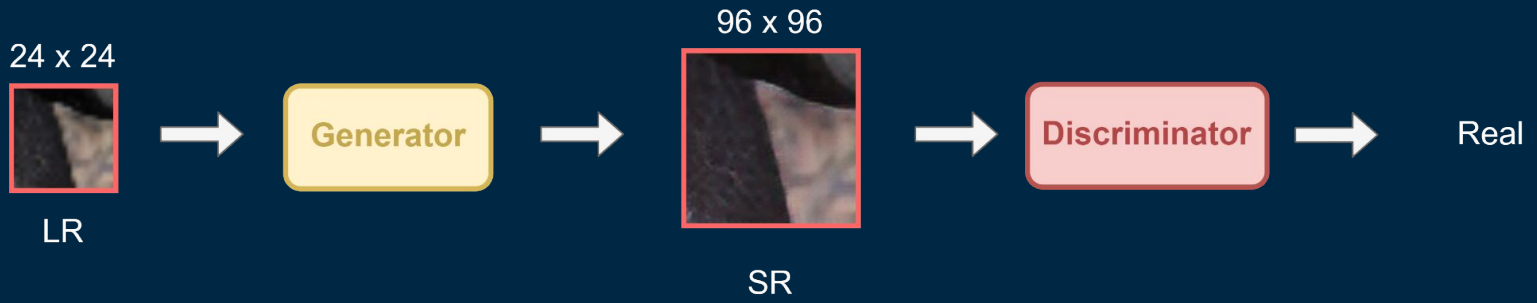
SRGAN - GAN



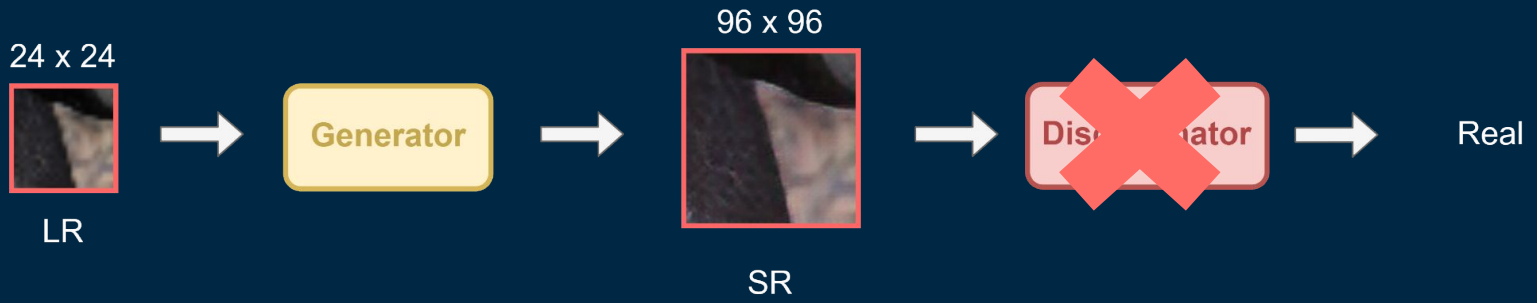
SRGAN - GAN



SRGAN - GAN

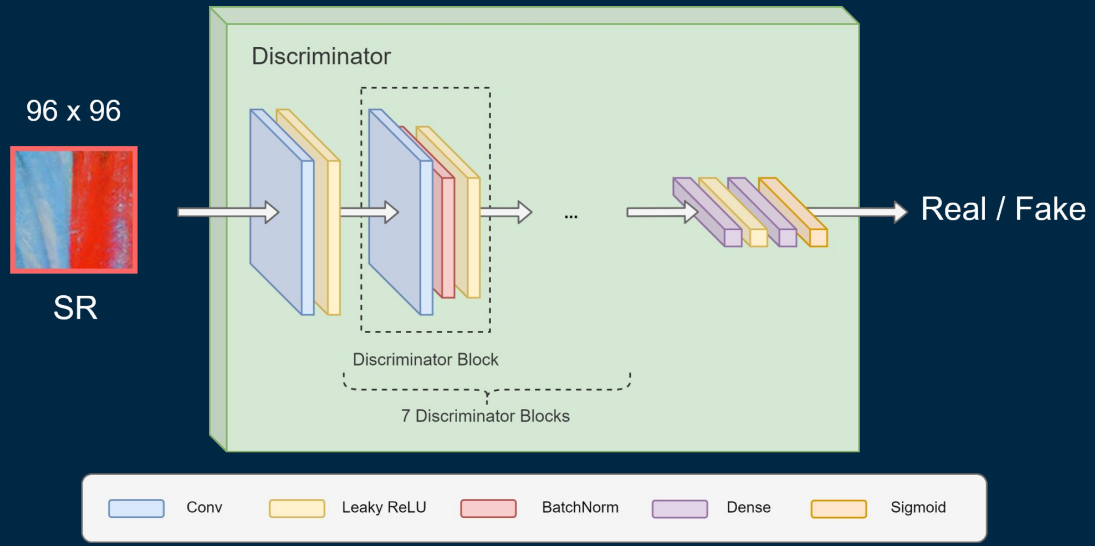


SRGAN - GAN



SRGAN - GAN Architecture

- Generator = SRResNet
- Discriminator

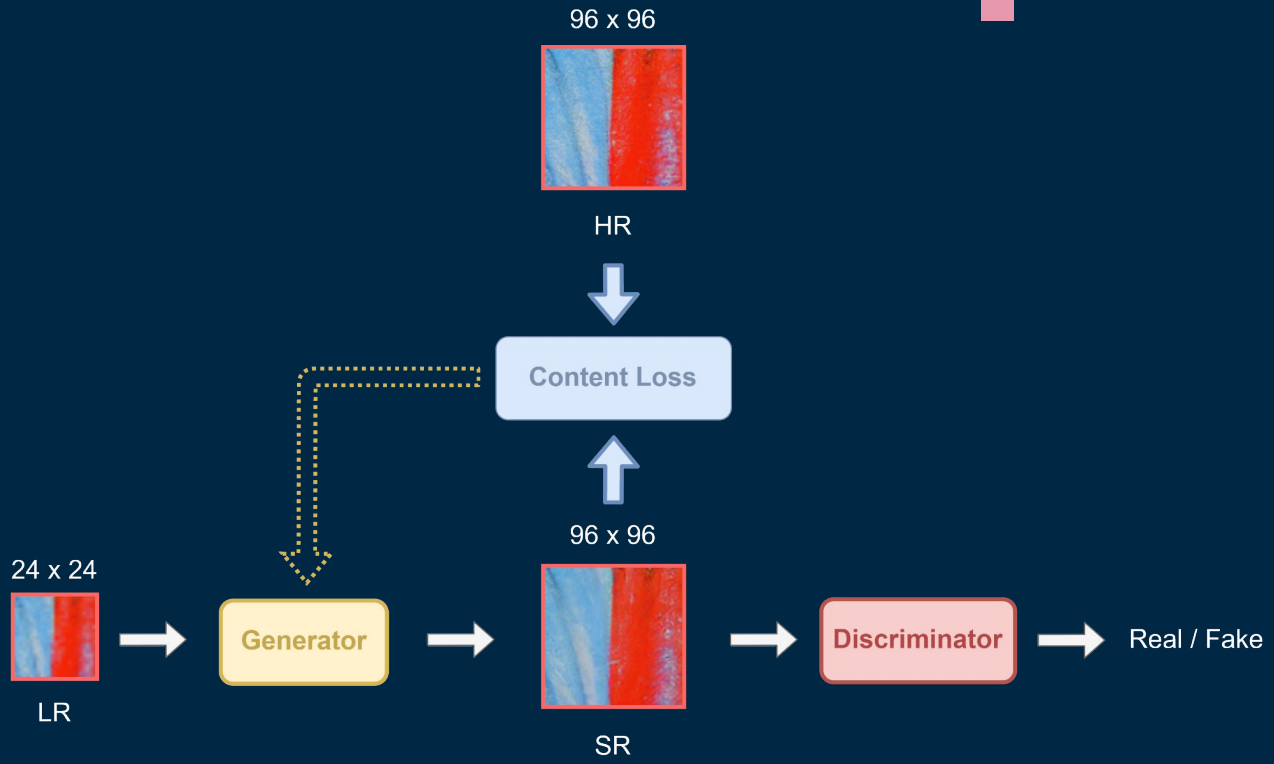


SRGAN – Perceptual Loss

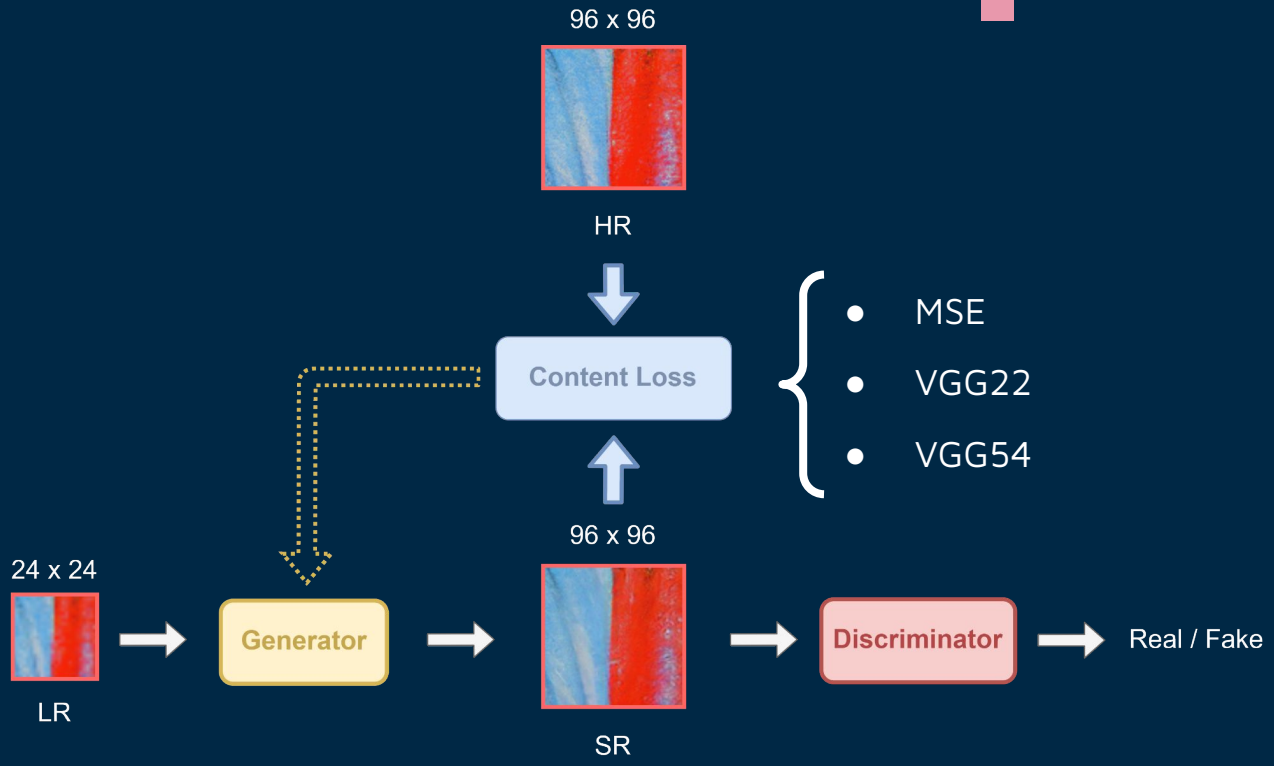
Perceptual Loss = Content Loss + Adversarial Loss

- Content Loss
  - Perceptually similar images to HR
- Adversarial Loss
  - GAN mechanism, force output to more natural images

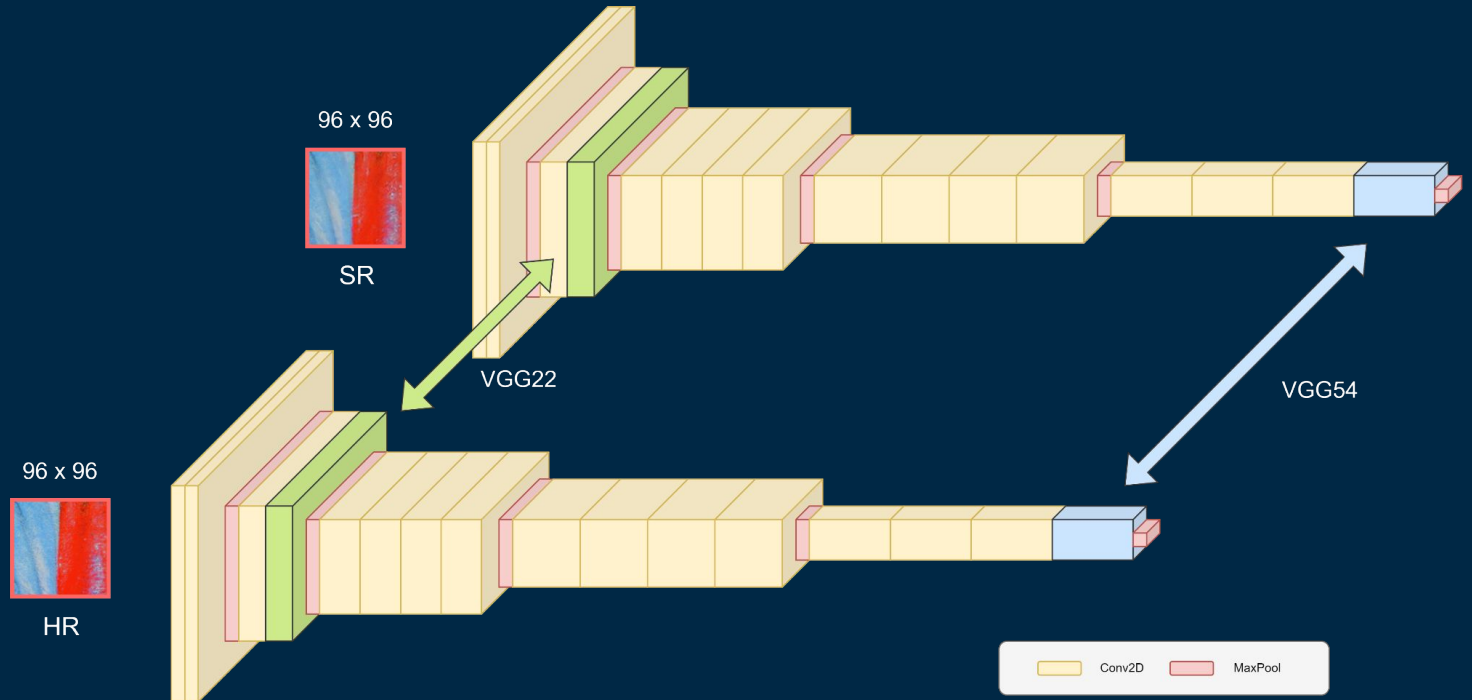
SRGAN – Content Loss



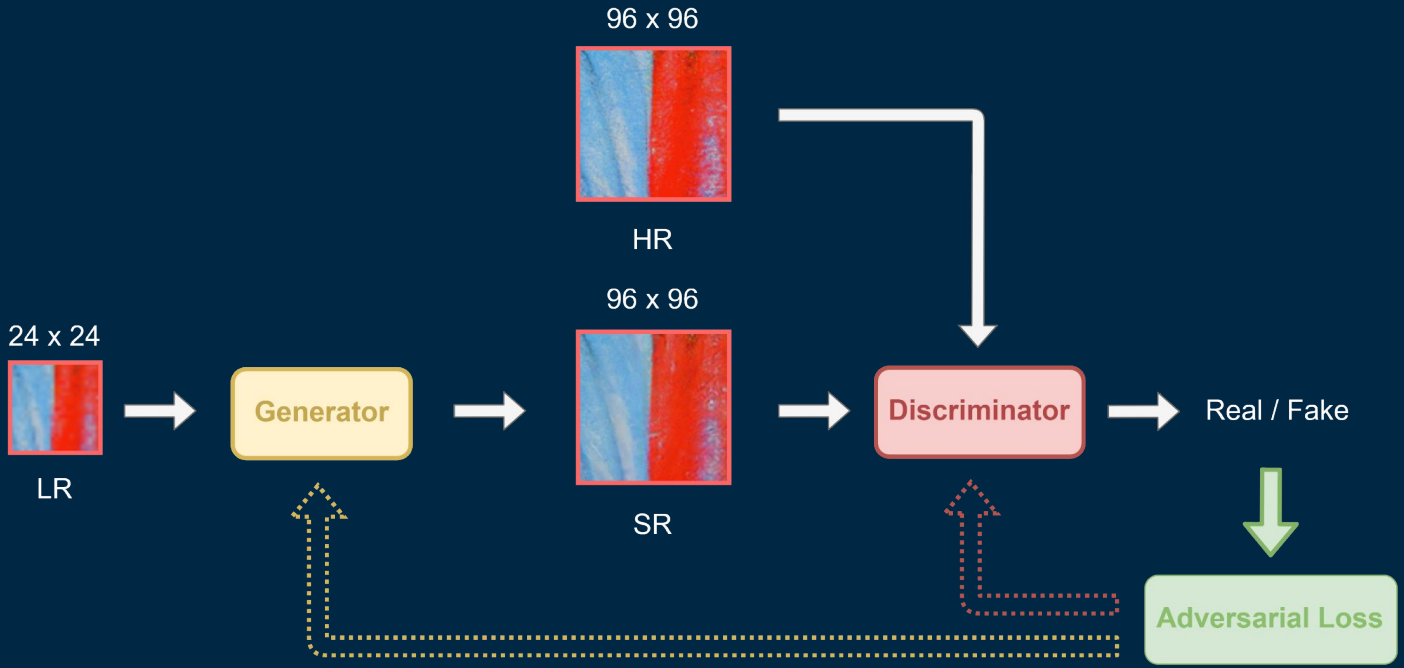
SRGAN - Content Loss



SRGAN – Content Loss

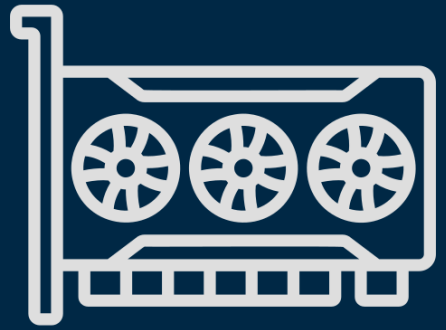


SRGAN - Adversarial Loss



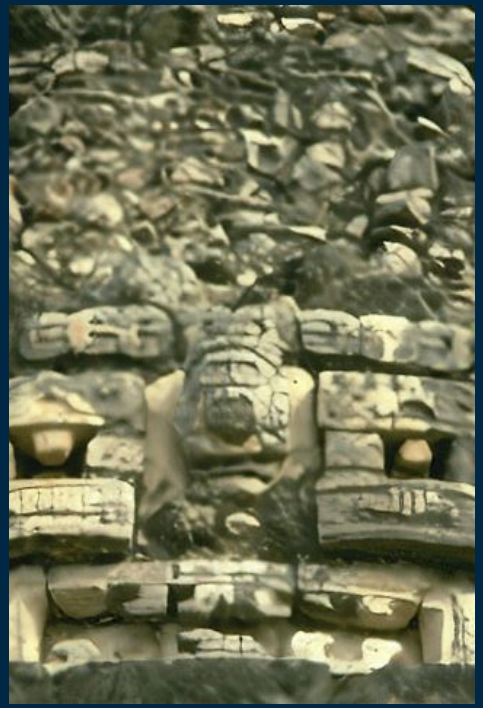
SRGAN - Training

- 350k images from ImageNet
- Training split into three phases:
  1. Initialize generator from pre-trained SRResNet (1mil iterations)
  2. Train GAN for 100k iterations with learning rate 10^{-4}
  3. Additional 100k iterations with learning rate 10^{-5}
- Total of **1.200.000** iterations

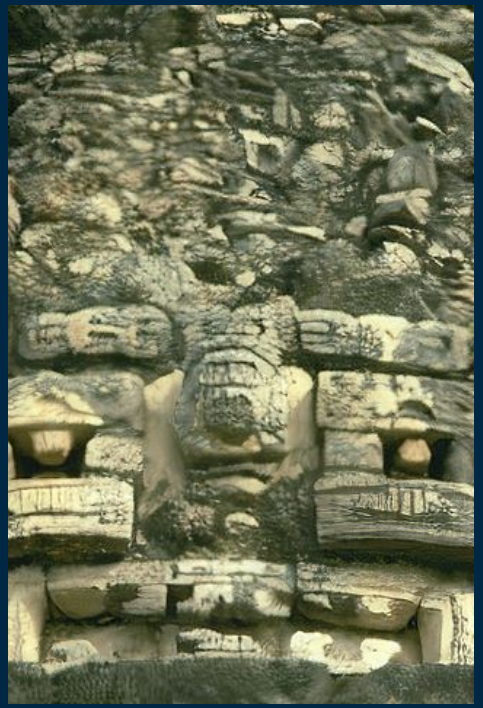


SRGAN - Results

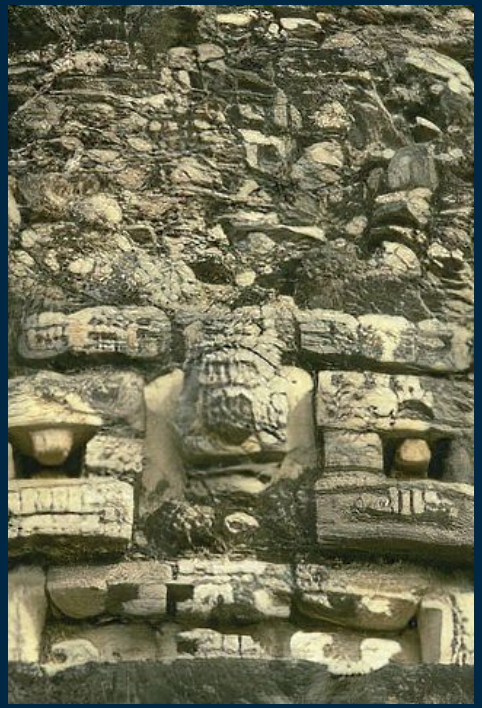
SRGAN-MSE



SRGAN-VGG22



SRGAN-VGG54



SRGAN - Results

SRGAN-MSE

SRGAN-VGG22

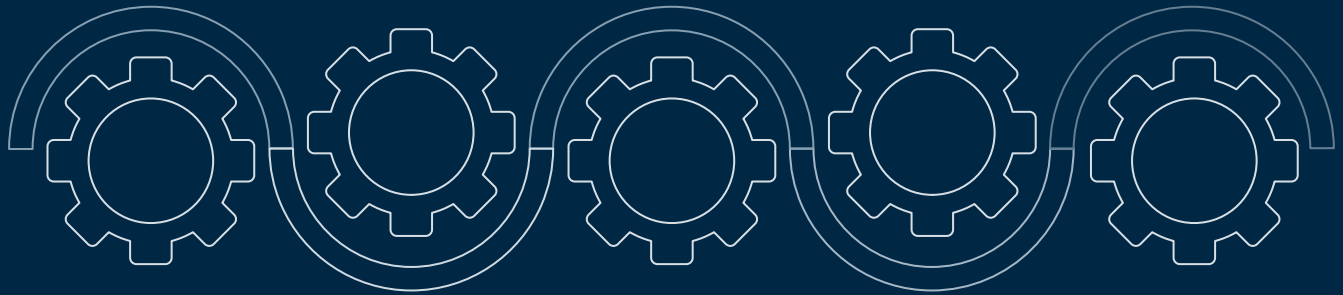
SRGAN-VGG54

OUR CONTRIBUTION

03

Our Contribution

- We wanted to improve their model and check the results
- Original training impossible to reproduce with our limited resources
- Worse results even if method is better
- For a fair comparison, we implemented the SRGAN ourselves



Our Implementation

- **Python** and **TensorFlow** for the model
- **Colab** environment for development
  - Free plan with limited execution time
- Integration with **Weights and Biases** (wandb)
  - Resume interrupted runs
  - Save best model
  - Save images for inspection
  - Comparison among runs

Our Dataset

- DIV2K dataset
  - 1.000 HR images (splitted in 800 train, 200 validate)
  - LR version generated with bicubic downsampling (4x scaling factor)

Our Training

Phase	Our Iterations	Their Iterations
Pre-train SRResNet	62.500	1.000.000
Train SRGAN LR 10^{-4}	6.250	100.000
Train SRGAN LR 10^{-5}	6.250	100.000
TOTAL ITERATIONS	**75.000**	**1.200.000**

- More than 6 hours for each experiment

Our Results

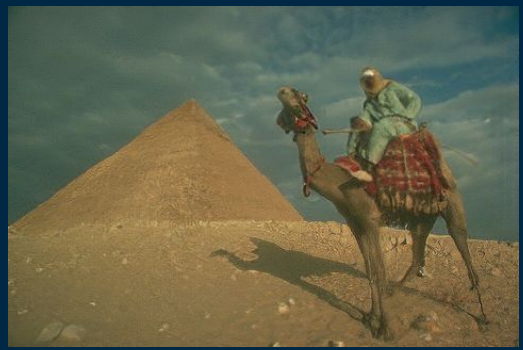
LR

Our SRGAN-VGG54

Their SRGAN-VGG54





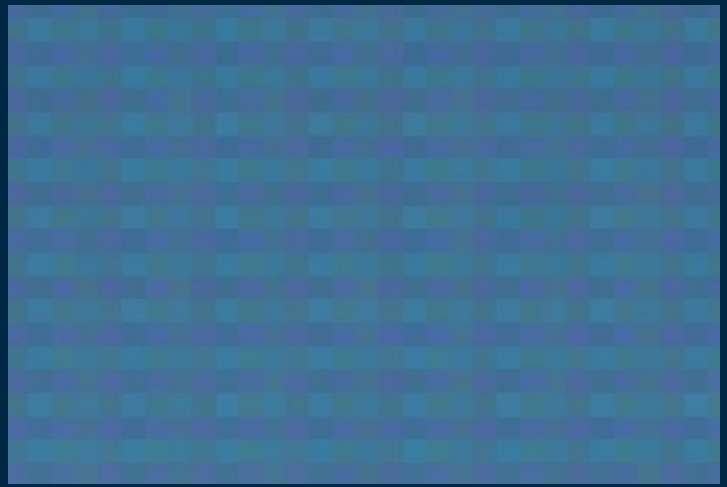
ARTIFACTS

The enemy

03

Artifacts - Analysis

- Checkerboard pattern that reduces naturalness
- No oversights or errors in the code



Artifacts - Analysis

- Known problem in literature
- Different hypothesis:
  - Conv2DTranspose layers
    - Replaced by PixelShuffle layers in SRGAN
  - Batch Normalization layers
    - ESRGAN authors advises to remove them
    - Doesn’t seem to affect our artifacts

Artifacts - Analysis

- Artifacts are affected by content loss with VGG

SRGAN-MSE

SRGAN-VGG22

SRGAN-VGG54

Artifacts - Analysis

- Artifacts are affected by content loss with VGG

SRGAN-MSE

SRGAN-VGG22

SRGAN-VGG54

Artifacts - Analysis

- We also analyzed the images attached to the original paper
- SRResNet-MSE doesn’t produce checkerboard artifacts…
- But they are present in the SRResNet-VGG22

SRResNet-MSE

SRResNet-VGG22

Artifacts - Analysis

- However, they are not present in the SRGAN variants...
- Even when using VGG losses!

SRGAN-MSE

SRGAN-VGG22

SRGAN-VGG54

Artifacts - Analysis

Why are there no checkerboard artifacts, then?

- Our hypothesis:
  - GAN removes artifacts added by the VGG loss in early stages
- Longer training time, bigger and different dataset may produce less artifacts
- Hit max reasonable training time, so we focused on our method
  - Artifacts will not affect comparison

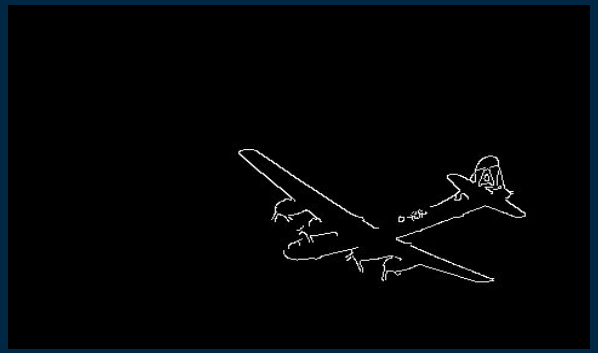
SRGAN-CANNY

Our approach

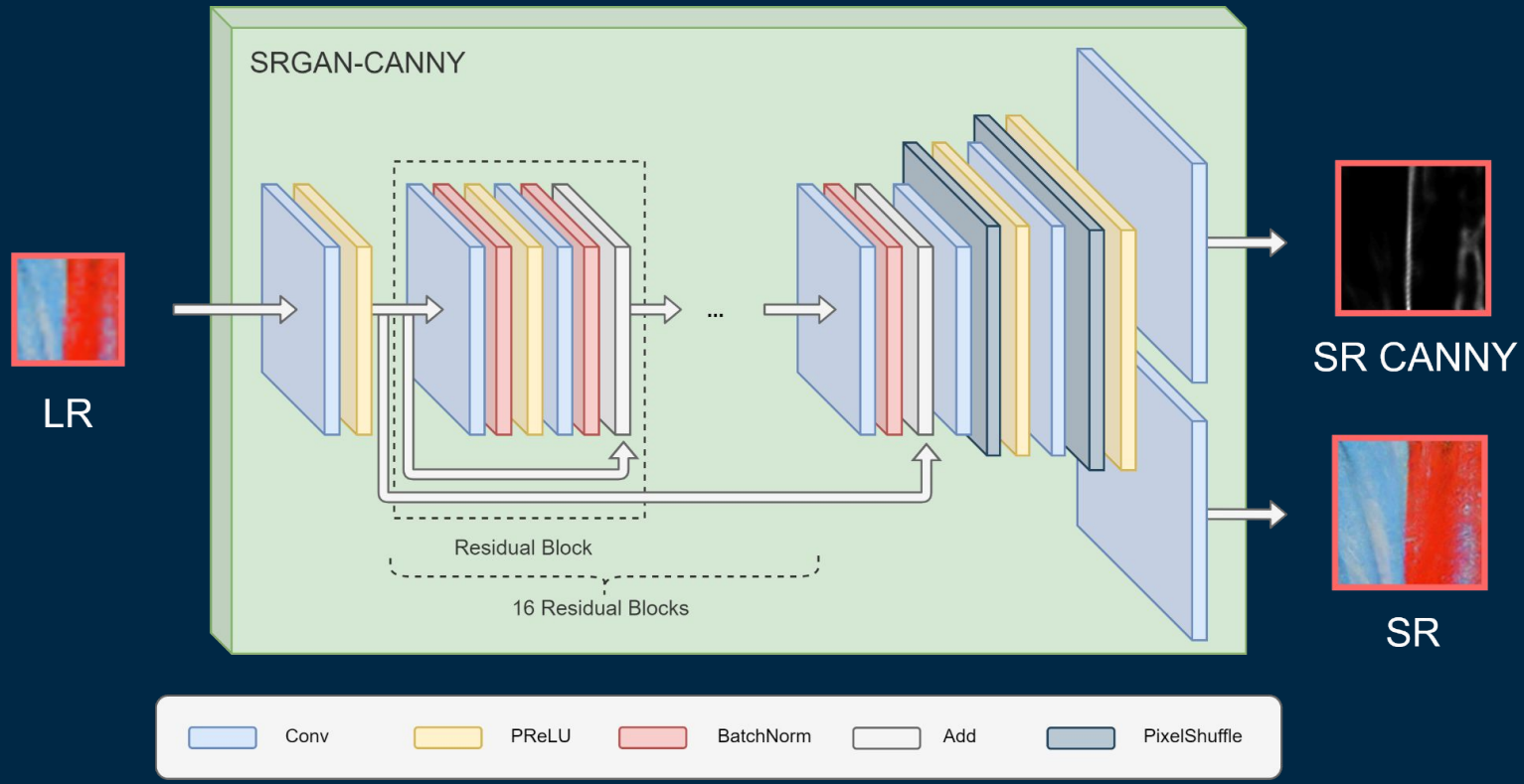
04

SRGAN-Canny – Edge detection

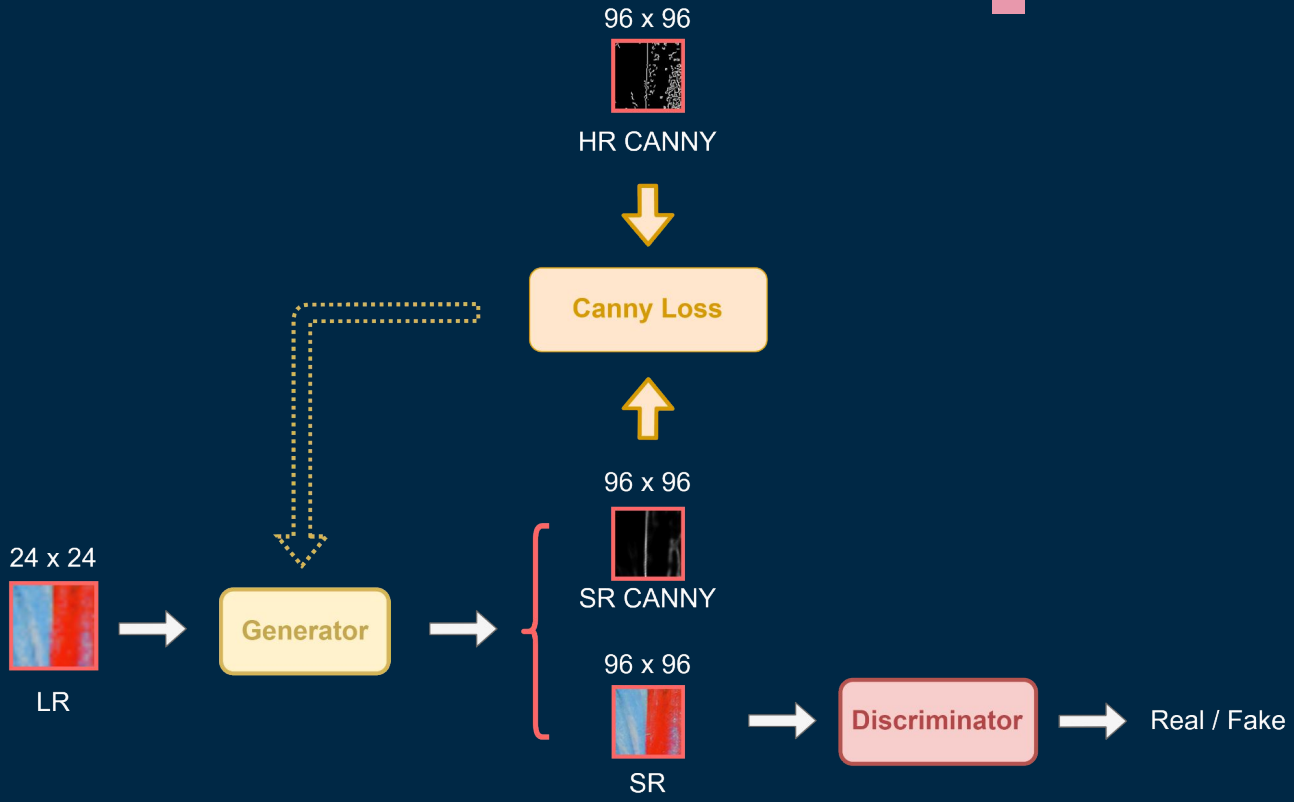
- In SR objects details and textures are really important
- Edge detection could refine contours and enhance details
- **Multi-task Learning**: multiple tasks solved simultaneously
- Model reproduces the Canny edge detection operator



SRGAN-Canny - Architecture



SRGAN-Canny - Canny loss

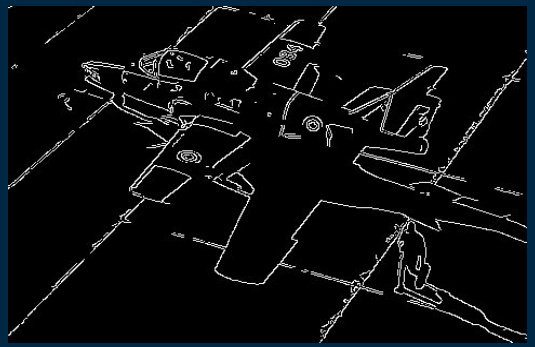


SRGAN-Canny - Results

LR

SRGAN-CANNY

HR



SRGAN-Canny - Results

LR

SRGAN-CANNY

HR

SRGAN-Canny vs SRGAN

SRGAN-CANNY

SRGAN-VGG54

SRGAN-Canny vs SRGAN

SRGAN-CANNY

PSNR: **23,2063** SSIM: **0,6036**

SRGAN-VGG54

PSNR: 22,7810 SSIM: 0,5950

SRGAN-Canny vs SRGAN

SRGAN-CANNY

PSNR: **23,2063** SSIM: **0,6036**

SRGAN-VGG54

PSNR: 22,7810 SSIM: 0,5950

SRGAN-Canny vs SRGAN

SRGAN-CANNY

SRGAN-VGG54

SRGAN-Canny vs SRGAN

SRGAN-CANNY

PSNR: 30,6772 SSIM: 0,6151

SRGAN-VGG54

PSNR: **30,8736** SSIM: **0,6400**

SRGAN-Canny vs SRGAN

- Test Sets: **Set5**, **Set14**, **BSD100** as original paper

	AVG MSE ▼	AVG SSIM ▲	AVG PSNR ▲
SRGAN-VGG54	0,01971	0,59896	23,87128
SRGAN-CANNY	**0,01808**	**0,61142**	**24,25129**

Thanks for the attention.