Single Image Super Resolution:Analysis

Github(https://github.com/aba450/Super-Resolution)
Ammar bin Ayaz (aba450), Umar Farooq (uf247)

1 INTRODUCTION

Image super-resolution (SR), which refers to the process of recovering high- resolution (HR) images from low-resolution (LR) images, is an important class of image processing techniques in computer vision. In general, this problem is very challenging and inherently ill posed since there are always multiple HR images for a single LR image but with the rapid development of deep learning techniques, Super-resolution models based on Deep learning have been extensively explored and they often achieve state-of-the-art performance on different benchmarks of Super-Resolution.

1.1 MODELS WE HAVE EXPLORED

SRCNN— A Convolution neural network based approach of super resolving a low resolution image.
SRGAN/SRResNet— A residual block based approach with variations of using a GAN based architecture and a NoGAN based one.
ESRGAN— A variation in the architecture of SRGAN with introduction of Residual-in-Residual Dense blocks and some other tweaks.
Evaluation Metric— PSNR(Peak Signal to Noise Ratio)

2 SRGAN/SRResNet

2.1 INTRODUCTION

SRGAN proposes an adversarial objective function that promotes super resolved image that lie close to the manifold of natural images.

The main highlight of the approach taken is multi-task loss formulation that consists of three main parts: 1) a MSE loss that encodes pixel wise similarity, 2) a perceptual similarity metric in terms of distance metric defined over high-level image representation, and 3) an adversarial loss that balances a min-max game between a generator and a discriminator.

2.2 Architecture

Generator:

The core of the network is a number of residual blocks which have identical layout. There are two convolutional layers with small 3x3 kernels and 64 feature maps followed by batch normalization layers. The activation function used is ParamtericRELU

Discriminator:

It contains eight convolutional layers with an increasing number of **3x3** kernels which is similar to a **VGG** network. The increment is by a factor of 2 from **64** to **512** kernels. Strides are used to reduce resolution when the number of features is doubles. The activation used in the beginning is **leakyRELU** and in the end the network has two dense layers with a final **sigmoid** activation function.

2.3 Key takeaways

1. GAN based approach for Super resolution, with modifications in loss function
2. New State of the Art with SRResNet which is basically using the generator as the model, meaning a NoGAN approach.
3. Highlighting the limitations of PSNR metrics for evaluation and using Mean Opinion Score (MOS) for evaluation.

2.4 Training/Testing

We have trained both SRResNet and SRGAN on Flickr8k dataset as ImageNet used in the paper is very big and we were unable to train it in google colab. We have tested on benchmark dataset **Set5** and **Set14**. We will use the **BSD100** dataset in future.

Training Details:

16 residual blocks for generator

Batch Size of 16

100,000 iterations for SRResNet and 10,000 for SRGAN on the batch

A target size of (96 * 96)

Adam optimizer with learning rate of 1e-4

4x downsampling of images

Results:

SRResNet — A PSNR of 27.40 on **Set5** and 25.2 on **Set14** which is less than the original reported score, this is quite expected as we are training on much smaller

dataset and for less number of iterations. PSNR (32.05 and 28.49) for **Set5** and **Set14** in paper.

SRGAN — A PSNR of 26.40 on **Set5** and 24.2 on **Set14** which is less than the original reported score, this is quite expected as we are training on much smaller dataset and for less number of iterations. PSNR (29.40 and 26.509) for **Set5** and **Set14** in paper.

3 ESRGAN

3.1 INTRODUCTION

This paper really takes the concept of SRGAN and tries to improve the model by incorporating the following changes. 1) Network structure introduces Residual-in-Residual Dense Block (RDDB), which has higher capacity and easier to train.
2) Removal of Batch normalization layers, and using residual scaling and smaller initialization to help training a very deep network.
3) Introduction of RaGAN (Relativistic average GAN), which focusses on identifying more realistic images, rather than if an image is fake or real.
4) Improvement in perceptual loss by using VGG features before activation, which was done after activation in SRGAN.

3.2 Architecture

Generator:

The main changes in the structure of the generator of SRGAN are:
1) Remove all the Batch Normalization layers
2) Replace the original basic block with the proposed Residual-in-Residual Dense Block(RDDB).

Discriminator:

Apart from making changes in the generator architecture they also changed the structure of the discriminator by introducing relativistic GAN, which focusses on identifying the relative difference between a realistic image and a fake one. The below figure depicts the difference of this from a standard discriminator.

Perceptual Loss:

A more effective perceptual loss by constraining on features before activation rather than after activation as practiced in SRGAN. The advantage of using the features before activation overcomes drawbacks in the original design like having sparse

activated features in a very deep network, using features after activation also causes inconsistent reconstructed brightness compared to the ground truth image.

3.3 Inference

We used a pertained model from tensor flow hub, will provide the link to the model in references. We used **Set5** and **Set14** benchmark datasets to evaluate the performance of the model. We used a 4x downsampling factor on the high resolution images as part of the downsampling process. We tried to train the model on colab but due to the complexity of the model and limitations of memory on colab we got out of memory error. In future we will try to find a work around, otherwise we will provide the training code for the same.

Results:

ESRGAN — A **PSNR** of 29.006 and **SSIM** of 0.828 on **Set5** and **PSNR** of 26.568 and SSIM of 0.744 on **Set14**. **PSNR** (32.5 and 28.99) and **SSIM**(0.9 and 0.79) for **Set5** and **Set14** in the GitHub repo of the paper.

4 SRCNN

4.1 INTRODUCTION

SRCNN is a shallow deep learning model with only three layers.Layers used in the network are described below:

- Patch Extraction and representation: This layer extracts over-lapping patches from the low- resolution image and upscales it to the desired size using bicubic interpolation i.e., represents each patch as a high dimensional vector. These vectors comprise a set of feature maps.
- Non-linear mapping: This layer performs non-linear mapping of the high dimensional vectors(n1) into other high dimensional vectors(n2) where n1 > n2, something like PCA but in a non-linear way. Each mapped vector represents the high-resolution patch.
- Reconstruction: After the mapping, the high-resolution image is reconstructed by patch-wise aggregation of high-resolution patches. This reconstructed image is expected to be similar to the ground truth image.

4.2 Training and Architecture

SRCNN is a 9-5-5 model i.e., 3 convolutional layers with kernel sizes 9,5,5 in their respective layers. The first layer outputs 64 feature maps, the second layer outputs 32 feature maps and the last layer outputs the high-resolution image
Training Details:
SRCNN was trained on T-91 image dataset and evaluated on Set 5 and Set 14.
Input: Images downscaled by a factor 2x,3x

Output: An up-scaled high-resolution image
Epochs: 400
Optimizer: Adam with rate 1e-4
Batch size: 16

4.3 Results

Eval. Mat	Scale	SRCNN	Dataset
PSNR	2	36.99	Set5
PSNR	2	32.34	Set14

5 Future Goals

- A more detailed description of the models proposed in the papers
- Evaluation on other benchmarks like BSD100
- If time permits, a basic introduction to a new technique using fastai.

6 References:

Paper:

- Photo-Realistic Image Super-Resolution Using GAN - https://arxiv.org/abs/1609.04802
- Image SuperResolution Using Convolutional Networks-https://arxiv.org/abs/1501.00092
- ESRGAN : Enhanced Super-Resolution Generative Adversarial Networks - https://arxiv.org/abs/1809.00219

Code:

- SCRNN Implementation - https://github.com/yjn870/SRCNN-pytorch
- ESRGAN : https://github.com/xinntao/ESRGAN,https://www.tensorflow.org/hub/tutorials/image_enhancing
- SRResNet : https://github.com/PacktPublishing/Generative-Adversarial-Networks-Projects, https://github.com/JGuillaumin/SuperResGAN-keras