Superresolution with GANs

Emre Inceoglu
Oguzhan Atakan

Introduction

This project aims at developing a superresolution model that is implemented through using generative adversarial networks (GANs) using minimal architecture. Following the recent advances in deep learning, superresolution has been improved as an efficient and effective way to increase the resolution of images without experiencing quality loss as from conventional upscaling methods as well as by performing with high speed. In our model, we follow the state-of-the-art architectures proposed in three papers, which propose the models EnhanceNet, UNet, and VGGNet. We wanted to produce similar results by combining these approaches with a smaller network architecture. Our architecture follows the generator/discriminator approach of GANs, where the generator takes a downsampled image and generates the upsampled version, after which the discriminator decides whether the image is fake or real. In this case, the generator is designed in a way where the image is encoded and then decoded to its final representation with residual connections between the encoding and decoding layers. Finally, we experimented with various loss functions that use the VGG network and TVLoss, which in the end improved our results. After the training, our model can predict the target using a downsampled image well and also can enhance any image with three channels.

Problem Statement and Motivation

Superresolution offers many benefits to various fields including medical image analysis and security related fields. The field of deep learning has experienced significant advances in the past 10 years. Following these advances, superresolution models have been developed to provide both efficient and effective ways to upsample images with minimum quality loss. Superresolution models do this in a much faster way than conventional upscaling methods, which makes them tempting options for being used in graphics processing related tasks like customizing resolution in GPUs. However, the state of the art models consist of very deep convolutional models, so we wanted to use the best parts of these approaches and combine them in a minimalist network.

Related Works

In implementing our model, we followed three state-of-the-art architectures proposed in three papers [2], [3], and [4], which propose the models EnhanceNet, UNet, and VGGNet. EnhanceNet proposes an architecture that uses GANs. Their general approach is in a way where in training, images are downsampled and are feeded into the generator, which then produces the upsampled version of the image. Then, the discriminator decides whether the produced image is fake or real. The distribution of the dataset takes into account the high resolution aspects of the images and thus when trained, the generator produces realistic upsampled images. We essentially followed this architecture in our model.

The second model that we integrated into our model was the UNet architecture. UNet was originally proposed as an image segmentation model that was intended for biomedical purposes. The architecture consists of residual encoding and decoding layers, where there are residual connections between the same sized encoding and decoding layers to finally output the resulting two dimensional segmented image in the final layer.

We finally included the VGGNet model in our model design. We used this in our training, where the VGG outputs a loss depending on the features that are outputted from the inputted, downsampled image and those from the real, high resolution image before the downsampling operation. This approach was useful because the model learns the features that are embedded in high resolution images, and produces good results with high quality.

General Approach and Methodology

The general framework that we used to implement our model was briefly described in the Related Works section. However, to explain in a stepwise manner, our general methodology is as follows: we used a generator that takes a downsampled image as input and generates the enhanced image. Meanwhile, the discriminator is trained to distinguish between fake and real images to achieve better results. We used the VGG network and TVLoss to calculate the loss and also experimented with custom color saturation and ratios in the loss function. When training our model, we trained the generator separately than the discriminator in one portion. We downsampled the input images by a factor of 2 and used our generator architecture to calculate the loss between the output images that the generator produces and their corresponding high resolution samples. This way, there exists a supervised relationship between the downsampled images and their corresponding high resolution images when training the generator. The rest of the training for the generator and discriminator follows the conventional GAN training methodology where the generator tries to fake the discriminator and the discriminator tries to distinguish between generator produced images and real images.

Output	Layers
h x w x 3	Input
h x w x 64	Initial Convolution
2h x 2w x 64	Bilinear Upsampling
2h x 2w x 64	7 x Res Block: Conv, LReLU, BN...
2h x 2w x 64	Skip Con: Res+Upsample
2h x 2w x 64	Final Convolution
2h x 2w x 3	Output

Table 1: Simple SR Model Layer Summary

Model Training

In the training we used the DIV2K dataset. We wanted to train the model with a small crop of an image, downsample the image and use it as our input and try to predict the original cropped image. We used SGD and cycling learning rates for better convergence. In order to achieve better training we developed a simple convolutional classifier as a discriminator for better looking and non easily distinguishable outputs. We created a function that finds the mean squared error between the target image and our output. Also, we utilized VGG16 for the loss function. We wanted the VGG16's feature extractor and classifier to perceive and classify our image and the target image as close to gather as possible. Also, because of the structure of the model, we also wanted to decrease the deviations between the color ratios of the output and target images. So we calculated color ratios for each pixel and scored them. So we find the ratio between red/blue, blue/green and green/red ratios and find the mean squared error of these ratios between the original image and the output image. We combined them in our custom loss function and trained based on these values. It took around 3 hours using google colab. After 100 epochs the model was not improving so we decided to stop the training after 120 epochs .

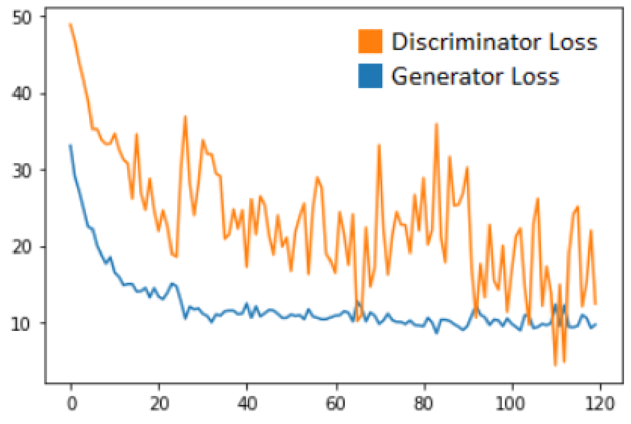

Figure 1: Training Loss

Results

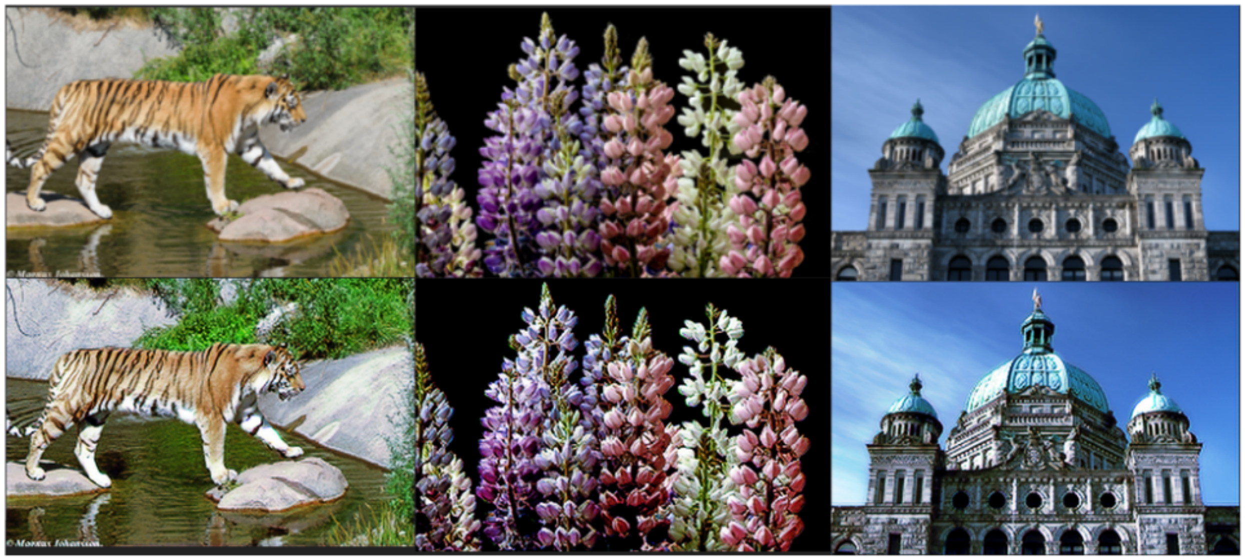

Figure 2: Top row: input image, Bottom row: output image

As can be seen from the images in figure 2, our model did improve the quality of the images. The blurriness is decreased, edges and sharp figures became more visible. However, there are also some shortcomings that are appaerent. Firstly the colors of especailly flat surfaces are a little bit off. It can be observed in the first image of the figure 2 where the ground rock behind the tiger became more blueish on the output image. We observed this behavior in the training process before we implemented the color loss function. After the implementation the color deviation decreased and became less noticeable, yet it is still a problem. For future works, this issue can be addressed. The second problem is the focus problem, in which an out of focus part on a picture is perceived as an important feature by the model, thus highly colorised and made more apparent than it should be. This can be observed from the cloud formation behind the building in the third picture of figure 2. We believe this is because the pattern of an out of focus part on a low resolution image is very similar to blurred out details. And since we are using a very simple architecture, the model is not suited enough to detect and act accordingly for such a subtle pattern.

The SSIM metrics of our models compared to other models trained on the same dataset can be seen in Table 2. Given that our model takes a more simplistic approach compared to the architectures of the other models, these results indicate that our model performs well and performs on a level that is a tier below the level that the models tested in [1] receive.

Model	**SSIM**
Bicubic	0.904
DRNN	0.940
MDSR	0.9692
CARN	0.9451
Our Model	0.7265

Table 2: Sample Models and their resulting average SSIM scores on the DIV2K dataset tested on x2 superresolution. More information on the models and their accuracies can be found on [1].

Project Repository & Files: https://github.com/slp-oozzyy/simple-sr-with-gan

References

[1] Anwar, Saeed, et al. “A Deep Journey into Super-Resolution: A Survey.” *ACM Computing Surveys*, vol. 53, no. 3, 2020, pp. 1–34, https://doi.org/10.1145/3390462.

[2] Ronneberger, Olaf, et al. “U-Net: Convolutional Networks for Biomedical Image Segmentation.” *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Springer International Publishing, 2015, pp. 234–41, https://doi.org/10.1007/978-3-319-24574-4_28.

[3] Sajjadi, Mehdi S. M., et al. *EnhanceNet: Single Image Super-Resolution Through Automated Texture Synthesis*. 2016, https://doi.org/10.48550/arxiv.1612.07919.

[4] Simonyan, Karen, and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014, https://doi.org/10.48550/arxiv.1409.1556.