Liam Pavlovic
CS7180: Project 1 Image Enhancement
09/23/2023

Abstract
For my project, I implemented a generative adversarial network for the super-resolution task. My network is based on the Super-Resolution GAN (SR-GAN)[1] and Enhanced Super-Resolution GAN (ESR-GAN)[2] papers. My implementation sits between these two papers, implementing some but not all of ESR-GAN's improvements. I tested my implementation with the PSNR metric and qualitative inspection.

Introduction
My implementation is primarily inspired by the SR-GAN [1] paper which introduced the first super-resolution technique that utilized a GAN framework. In this scheme, two models are trained, a generator and a discriminator. The generator learns to output realistic high-resolution upscales and the discriminator learns to differentiate between the generated upscales and the genuine high-resolution images. The adversarial loss used to train both models is based on how realistic the discriminator rated the generator's outputs.

In addition to this adversarial loss, the SR-GAN employs perceptual loss. Perceptual loss is computed as the difference between the feature representations of the generator's outputs and their corresponding references when sent through some pre-trained loss network. This loss is intended to encourage the generator to mimic the basic features (i.e. lines, textures) of the high-resolution image.

My implementation also takes some inspiration from the ESR-GAN [2] paper. The ESR-GAN paper applied an array of augmentations to the SR-GAN paper to improve performance. The basic convolutional block in the SR-GAN was replaced with a novel Residual-in-Residual block which has a more complex, hierarchical structure of skip and residual connections. For stability, the ESR-GAN utilizes residual scaling, rather than batch normalization, as the latter was empirically shown to hurt super-resolution performance.

The discriminator used relative, rather than absolute loss, to enable both the reference and generated images to contribute to the generator's training. Also to maximize information available for learning, perceptual loss was applied on features before, rather than after, activation layers. Lastly, the final model was an interpolation of the GAN generator and a PSNR-trained network to balance the performance advantages of the two approaches.

Method

As is standard for GAN methods, my model contains a generator network and a discriminator network. Like SR-GAN, the generator begins with a series of convolution blocks in the low-resolution space for feature extraction. Each of these blocks have an individual residual connection in addition to a shared residual connection that bypasses all blocks as a unit. The resulting features are then upscaled to the high-resolution space using bicubic interpolation. There are a few convolution layers in the high-resolution space, for refinement, before the final image is produced. All residual connections utilize residual scaling and the weights of the network are initialized to small values near zero for training stability.

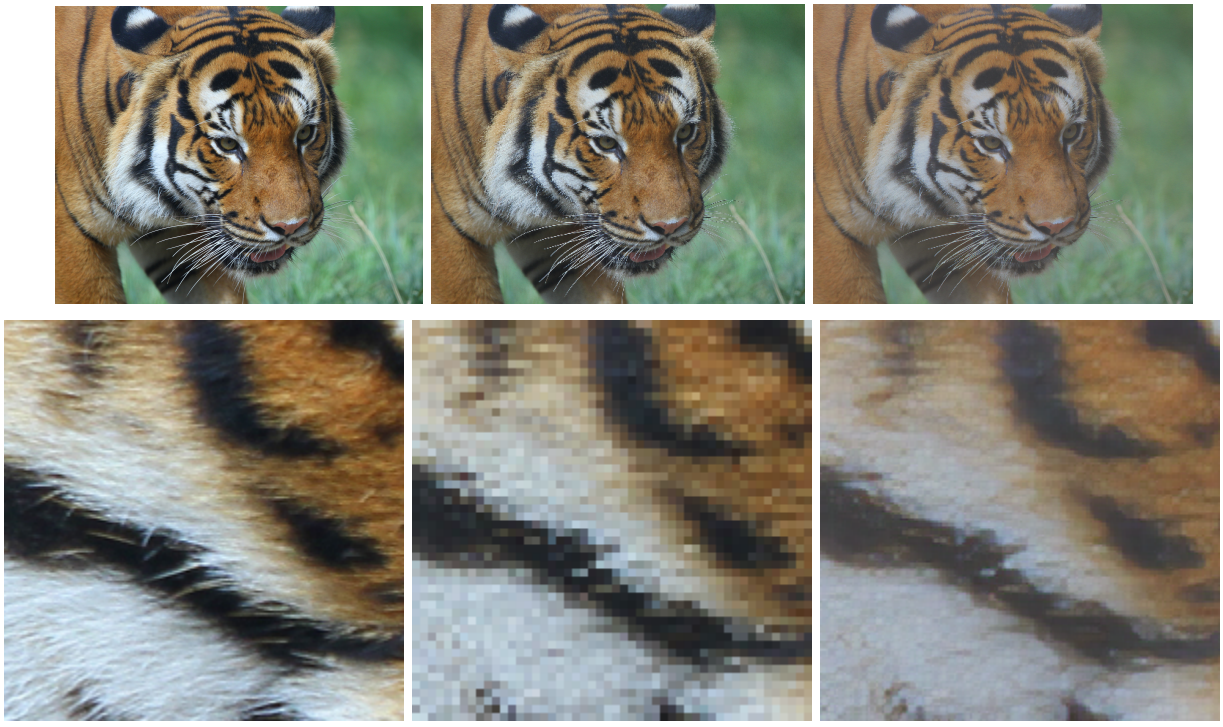
The structure of the discriminator is identical to SR-GAN. A series of convolution blocks with batch normalization are employed and the resulting features are flattened and sent through a dense net to produce the discriminator's rating of the image. I kept batch normalization in the discriminator because it does not have any residual connection with which residual scaling can be applied. For training stability, the discriminator uses the same initialization scheme as the generator.

The networks were trained on a weighted sum of adversarial loss and perceptual loss, with greater emphasis placed on the latter. The network used for perceptual loss was a VGG-19 network pre-trained on image classification, using features from its convolutional layers pre-activation. The networks were trained on the BSDS300 dataset for 1,500 epochs.

Results

BSDS300 Test Set	Bicubic	GAN
PSNR	21.785	21.637

My implementation, like the original SR-GAN, was slightly outperformed by bicubic interpolation on the PSNR metric for the BSDS300 test dataset. This is expected because the model was not trained to perform well on this metric and avoids the over-smoothing behavior that PSNR rewards. However, this is a good argument for the model interpolation feature of ESR-GAN (which I did not implement), since it incorporates the fidelity and noise-reduction of PSNR in a balanced manner.



From left to right, these images are the original high-resolution image, a bicubic interpolation upscale, and a GAN generated super-resolution image. This qualitative comparison showcases the strengths of the GAN approach. Where the bicubic upscale is very pixelated, the GAN super-resolution image manages to somewhat replicate the fur texture, albeit with duller lines.

Reflection and Acknowledgements

I presented the ESR-GAN paper in class and I believe that implementing the technique has really solidified my understanding of it. In particular, I think I have a stronger understanding of how it differs from the SR-GAN paper and why the specific modifications the ESR-GAN authors employed were chosen. This was my first time implementing a GAN and I have become more familiarized with the intricacies of training such a model, such as balancing the performance of the discriminator and generator models to keep the training gradients informative.

I referenced the SR-GAN and ESR-GAN papers during this project, but not their corresponding implementations. My implementation was produced from scratch with the exception of a couple code snippets which are cited in the comments. I also referenced the PyTorch DCGAN tutorial [3] for the basics on GAN implementation.

Code Link: https://github.com/liampav3/SuperRes-GAN

Sources:

[1] Ledig, Christian, et al. "Photo-realistic single image super-resolution using a generative adversarial network." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

[2] Wang, Xintao, et al. "Esrgan: Enhanced super-resolution generative adversarial networks." *Proceedings of the European conference on computer vision (ECCV) workshops*. 2018.

[3] https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html