Image Super-Resolution Via a Convolutional Neural Network

Ben Garber
bgarber@stanford.edu

Aitan Grossman
aitan@stanford.edu

Sonja Johnson-Yu
sonja@cs.stanford.edu

Abstract

Super-resolution is the task of constructing a higher-resolution image from a lower-resolution image. While this task has traditionally been approached with non-neural methods such as bilinear and bicubic upsampling, neural networks offer an opportunity for significant improvements. We implement a Super-Resolution Convolutional Neural Network (SRCNN), as proposed by Dong *et al.* [2] and demonstrate improvements over the bicubic upsampling baseline test peak signal-to-noise ratio (PSNR) of 23.226 dB with a test PSNR of 26.442 dB.

1 Introduction

Image super-resolution is the task of recovering a high-resolution image from a lower-resolution image. This problem is notable for its applications in security as well as in medical imaging, especially since image reconstruction offers a methodology for correcting imaging system imperfections.

For our project, we implement SRCNN and refine the model in order to improve the quality of the output images, as measured by peak signal-to-noise ratio (PSNR). The input to our algorithm is a low-resolution image, which we feed through a convolutional neural network (CNN) in order to produce a high-resolution image.

Traditional methods for image upsampling rely on low-information, smooth interpolation between known pixels. Such methods can be treated as a convolution with a kernel encoding no information about the original photograph. Although they increase the resolution of an image, they fail to produce the clarity desired in the super-resolution task. Convolutional Neural Networks (CNNs) are a generalization of such algorithms, using learned kernels with nonlinear activations to encode general characteristics about photographs that can add structure lost in the low-resolution input. CNN architectures such as SRCNN [2] have been successfully applied to the super-resolution task.

2 Related Work

Yang *et al.* [7] give a detailed overview of the different approaches to single image super-resolution (SISR). Common SISR methods can be divided into three categories: interpolation-based methods, reconstruction-based methods, and learning-based methods, the last of which has received the majority of recent attention for its ability to yield accurate results given prior knowledge without the slow speed of reconstruction-based super-resolution methods. Very recently, deep learning-based SISR methods have shown the most promise.

Bicubic interpolation [3] is an exact, iterative, interpolation-based method commonly used because of its simplicity and speed. This is a good baseline with which to compare deep learning-based methods, and it also happens to be part of the required preprocessing for the SRCNN method. The key drawback of bicubic interpolation as super-resolution is that it does not make use of outside knowledge and thus does not lend any additional information to the upscaled image. It also has a smoothing effect, which ultimately limits the performance of SRCNN.

To circumvent the issues posed by the initial bicubic upscaling, Shi *et al.* [6] developed ESPCN (a more successful version of its predecessor, FSRCNN), which extracts feature maps in low-resolution space and introduces an efficient sub-pixel deconvolution layer whose learned filters upscale the previous layer (rather than the usual downscaling). ESPCN is also more efficient because it works on smaller-dimensional layers, and it can be used on real-time HD video.

A subsequent class of models expanded on the ESPCN approach, and took advantage of its superior performance, to increase the depth of super-resolution models and increase the solution space, which was found to greatly improve accuracy. Kim *et. al.*'s DRCN and VDSR models [4] achieve the increased depth by cascading small filters many times (both networks are roughly 20 layers), which has the result of preserving contextual information over large image regions.

The current state of the art is achieved by models like Lim *et al.*'s EDSR and MDSR [5], which take advantage of the insight that whereas many CNN architectures are designed to recognize highly abstract inner representations, the task of super-resolution is localized and largely scale-free. As such, the developers of these methods utilize a number of "tricks" to accelerate training and improve performance, including sharing weights between models designed for different scales.

Although more advanced methods than SRCNN exist, our aim by using this simpler model is to reimplement a performant super-resolution model, and by attempting to improve it, gain practical insight into how to better apply deep learning to the task of super-resolution.

3 Dataset and Features

We draw our dataset from DIV2K [1], a dataset of 1000 high-resolution images with diverse subjects. Due to the constraints on GPU resources, we used a subsample of 100 images, with a 60/20/20 train/val/test split. Because the images in the DIV2K dataset have varying sizes, we first center crop them to 800x800 and then downsize to 224x224. In order to generate our low-resolution images, we resize them to half that size and then resize them back up to 224x224, producing an image of lower resolution.

Figure 1: Example of input image (left) and target image (right).

4 Methods

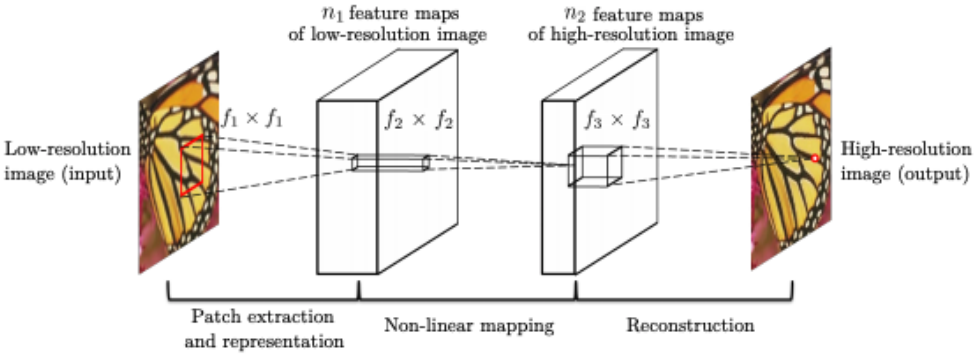

Figure 2: SRCNN architecture

4.1 Architecture

SRCNN comprises three convolutional blocks corresponding to patch extraction, non-linear mapping, and reconstruction. The first layer convolves the input image with a 9x9 kernel with padding and expands the three-channel image into 64 feature maps. The second layer applies a 1x1 kernel to condense to 32 feature maps. The third layer applies a 5x5 kernel to generate the output image. Both the first and second convolutional layers are succeeded by a ReLU activation function.

Convolutional blocks can be treated as a subset of linear models with a small number of parameters. The 9x9 kernels of the first SRCNN layer apply the same 81 parameters (values for each pixel of the kernel) to a full image, such that each pixel of the resulting image corresponds to the level of similarity between the corresponding 9x9 pixel patch of the input image and that kernel. The nonlinear mapping step connects these input feature maps to a reduced set of output feature maps. The 5x5 kernels can be viewed as higher-resolution/granularity image features represented by the pixels of the output feature maps.

4.2 Metrics

For training and validation, we used mean squared error (MSE) for our loss function.

$$L(\ ) = \frac{1}{n}\sum_{i=1}^{n}\left(h(x^{(i)};\ ) - y(i)\right)^2$$

In order to evaluate the performance of the model, we also used peak signal-to-noise ratio (PSNR), a common metric for evaluating the quality of the image restoration,

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right)$$

where MAX_I^2 is the maximum possible pixel value of the image. While the numerator captures the signal strength, the denominator represents the noise strength. The higher the peak signal-to-noise ratio, the better the reconstruction. Consequently, minimizing MSE leads to a maximization of PSNR.

5 Experiments

5.1 Baseline

We implemented three different non-neural baselines: nearest neighbor, bilinear, and bicubic. While the loss of bilinear and bicubic upsampling methods is comparable to the minimum validation loss of our best model (learning rate = 0.001), the validation PSNR of our model greatly exceeds the PSNR of the output images from the bilinear and bicubic methods (see Figure 4). This indicates a higher-quality image reconstruction, as shown in Figure 3. Our SRCNN model significantly improved over the baseline validation and test PSNR, as shown in Table 1.

Figure 3: Image quality comparison of bilinear upsampling (left) and SRCNN (right).

5.2 SRCNN

We trained a variety of SRCNN models using the 60 image training dataset, using the performance on the validation set in order to tune hyperparameters. The PSNR and losses for the training and validation sets were comparable, with a max train PSNR of 28.03 dB and a max val PSNR of 27.46 dB, indicating that we did not overfit our model. Indeed, when we ran our best model on the final test set, we produced images with an average loss of 0.00643 and an average PSNR of 26.44 dB. Our model performances are summarized in Table 1.

Learning rate: When tuning the hyperparameters of our model, we first performed a grid search to find an optimal learning rate of 0.001 (see Figure 4). Because we were testing locally due to GCloud resource shortages, our mini-batch size was limited to 2.

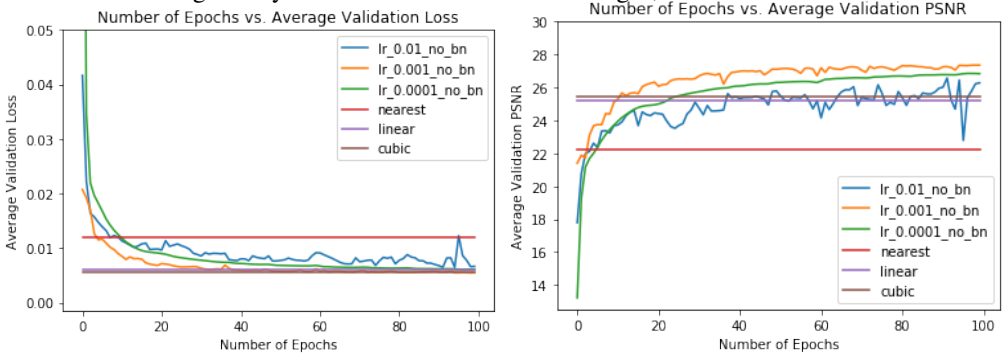

Figure 4: Comparison of validation loss and PSNR with various learning rates (and nearest neighbor, bilinear, and bicubic baseline)

Tanh: We found in early experiments that the output range sometimes caused color saturation where overloaded pixels would come out as a primary color (red, green, or blue). In order to counteract this effect, we applied a tanh activation function to the output pixels before rescaling. We found that this worsened the validation PSNR, as in Figure 5.

Batch Normalization: We experimented with the inclusion of batch normalization layers after the first and second convolutional layers (before the ReLUs), but we found that the model performed better without batch normalization, as in Figure 6.

Regularization: We added regularization with various scaling factors of the regularization term [0.01, 0.001, 0.0001, 0.00001, 0.000001] but found that regularization did not improve the performance of the model and in fact degraded it when the regularization scaling factor was too high, as in Figure 7.

6 Conclusion

In our experiments, we have demonstrated that SRCNN surpasses non-neural methods for the task of super-resolution, achieving a test PSNR of 26.442 dB, surpassing the baseline bicubic PSNR of 23.226 dB. In the output images, Figure 8, we see that the model increases

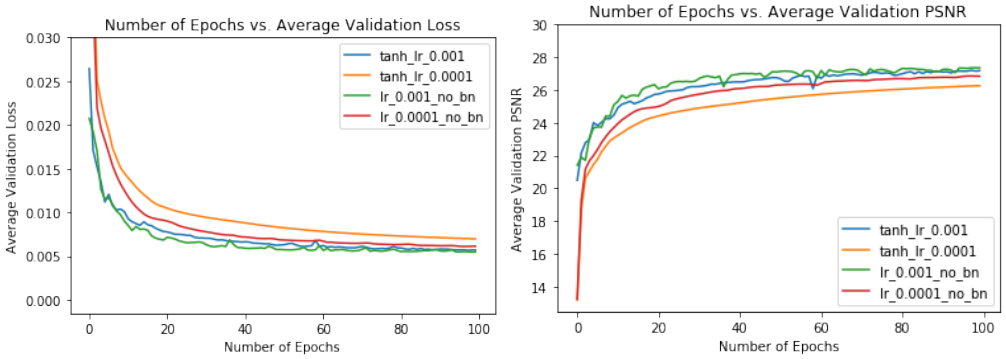

Figure 5: Comparison of validation loss and PSNR with tanh and no tanh activation functions

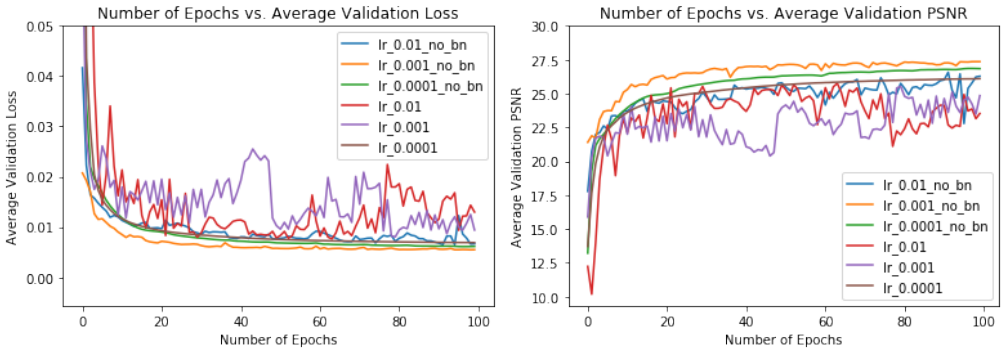

Figure 6: Comparison of validation loss and PSNR with batchnorm and no batchnorm. The experiments with batchnorm have much higher jitter and perform worse than the corresponding versions without.

sharpness and adds naturalistic detail. There are some remaining issues with pixels that are very different in color than they should be, possibly due due overflow or clipping. We also see some remaining pixelation, for instance, in the corn leaves at the upper right corner of the first photo in Figure 8.

Our work demonstrates the promise of deep neural architectures for achieving state-of-the-art performance on the task of image super-resolution. We also expect that having the computing resources and time to train on the full DIV2K dataset of 800 images would improve the model's performance. The lack of availability of GPUs also limited us to a relatively shallow neural network.

While the SRCNN architecture offers significant improvements over the baseline, other models such as Fast SRCNN (FSRCNN) as well as adversarial networks improve over SRCNN's performance. In future work, we would like to approach the super-resolution task using a UNet++ architecture as well as a generative adversarial network.

7 Contributions

We performed the coding, analysis, and writing as a team.

We would like to thank our mentor Guanzhi Wang for his advice.

Experiment	Max Val PSNR (dB)	Test Loss	Test PSNR (dB)
Basic SRCNN	27.455	**0.0064324**	**26.442**
+ Regularization	**27.459**	0.0064412	26.429
+ Tanh	27.336	0.0066215	26.282
+ Batch Norm	25.530	0.0100244	24.373
Bicubic	25.433	0.0066051	23.226
Bilinear	25.220	0.0074545	23.182
Nearest neighbor	22.198	0.014747	20.551

Table 1: Comparison of model performance, where learning rate is 0.001.