Super-Resolution Imaging

Whitney Long
wlong6@wisc.edu

Chenyang Jiang
cjiang77@wisc.edu

Fangying Zhan
fzhan4@wisc.edu

Abstract

In this work, we employ three deep learning techniques to reconstruct high resolution (HR) image based on low resolution (LR) image input. Specifically, we focus on single image super resolution (SISR). We employ two baseline methods - bilinear interpolation and a simple Convolutional Neural Network with Conv2DTrasnpose layers, to compare our models and results with. Then, we mainly explore the super resolution convolutional neural network (SRCNN). The results show that bilinear interpolation method and transposed convolution method give similar results, whereas SRCNN is superior, which gives significantly smaller MSE and MAE, and larger PSNR and SSIM. We conclude that SRCNN is the most powerful model in our experiment.

1. Introduction

Modern people love taking photos to record and share details of their daily lives. Looking at old pictures that we took would always help to refresh our memories. As technologies develop, in recent years, people are pursuing pictures of higher qualities. However, old pictures are often of comparatively low qualities. As a result, there is an increasing demand for improving old and unclear photos into clear and vivid ones.

We are motivated by the fact that HR images provide people with utmost convenience throughout many areas in life whenever they are needed. So far, there are already many applications appeared with advantages of using images with high resolution. For example, HR medical images help doctors to make diagnosis more accurately; HR images from surveillance camera allow users to tell the characteristics of criminals; it is even possible to distinguish similar objects using HR satellite images.

Our initial motivation of this project comes from the LR images from webcam. Webcam are widely used worldwide for all kinds of usage, such as health care, video security, video monitoring, etc. Due to the quality of webcam itself and some other reasons, the images from the webcam may not be clear enough to provide specific information as expected. For example, sometimes it is hard to recognize the registration plate of the illegal vehicles; or sometimes it cannot tell clear facial features of robbers. We start by asking if we can enhance the resolution of LR images via certain techniques. By developing such a tool, we may apply it into our daily life as well, even for enhancing the resolution of a selfie.

In this project, we try to generate high quality images based on its original low quality versions. The quality of an image depends on its resolution. The higher the resolution, the more information the image contains. The problem is how to create a high resolution (HR) image using a single low resolution (LR) image as input. Super resolution (SR) imaging is a class of techniques that increase the resolution of an imaging system. In other words, it is the predicted HR image based on the LR image we have.

To solve this problem, we train an SRCNN, which is well-known for enhancing LR images. We also make use of bilinear interpolation and train a simple CNN with Conv2DTrasnpose layers so that we can set these two methods as baselines to compare the results from SRCNN with.

2. Related Work

The purpose in this project is to use low resolution images as an input to obtain corresponding high resolution images. The deep learning techniques for super resolution is relatively new. Dong et al.[4] propose a deep learning method for single image super-resolution (SR), which is also known as SRCNN. The architecture mianly includes three phases: patch extraction and representation, non-linear mapping, and reconstruction. Unlike traditional methods that attempts to handle separate components, SRCNN suggests a joint optimization of all layers. We mainly employ SRCNN in our project.

According to Yang et al.[7], there are nowadays three mainstream algorithms of single image super-resolution (SISR) methods: interpolation-based, reconstruction-based, and learning-based methods. For our project, we choose to implement SRCNN, which belongs to the interpolation-based methods that is considered speedy and straightforward. However, it is also subject to the suffering from accuracy.

3. Proposed Method

In this section, we mainly discuss details of the super-resolution CNN (SRCNN) architecture. We will also discuss bilinear interpolation method and transposed convolution method. We set these two methods as our baseline, and compare them with SRCNN architecture.

3.1. SRCNN Architecture

The sketch of SRCNN architecture is shown in figure 1. It has four operations in total. To begin with, using bicubic interpolation method to get the input image from the LR image. The input image has our desired HR size. Then, the first layer extracts sets of feature maps from the input image. After that, the second layer maps those feature maps to the HR patches using nonlinear method. At last, the third layer produces the HR image by reconstruction. It is a three-layer CNN, where the parameters of each layer are (input channel)×(output channel)×(kernel size)×(kernel size), where the first layer is $3 \times 64 \times 9 \times 9$, the second layer is $64 \times 32 \times 5 \times 5$, and the third layer is $32 \times 3 \times 5 \times 5$. We use ReLu function as activation function for the first and second layer, use Tanh function as activation function for the last layer. ReLU and Tanh are defined as:

$$ReLU(x) = max(x, 0) \tag{1}$$

$$Tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2}$$

In order to compare with bilinear interpolation method, we use bilinear interpolation method to up-scale LR images in our first step instead of bicubic method. We use two different loss functions: Mean Square Error (MSE) and Mean Absolute Error (MAE) to compare their results. MSE and MAE are defined as:

$$MSE = \frac{1}{N}\|I - \hat{I}\|^2 \tag{3}$$

$$MAE = \frac{1}{N}\|I - \hat{I}\|_1 \tag{4}$$

where N is the number of total pixels, I and $\hat{I}$ are two given images, $\|\cdot\|$ is the regular norm, $\|\cdot\|_1$ is the l_1 norm.

Optimizing (3) can be interpreted in a probabilistic way by assuming Gaussian white noise independent of the image in the regression model, and then, the conditional probability of y given x becomes a Gaussian distribution[7]:

$$p(I|I_0) = N(I; \hat{I}(I_0, \theta), \sigma^2 E) \tag{5}$$

where I_0 is the input LR image, I is the HR image, $\hat{I}$ is the training output. $\sigma^2 E$ is the diagonal covariance matrix, E is the identity matrix. Then, optimizing (3) is to find the maximum likelihood estimation (MLE) of (5).

Optimizing (4) can be interpreted in a probabilistic way by assuming Laplacian white noise independent of the image in the regression model, and then, the conditional probability of y given x becomes a Laplacian distribution:

$$p(I|I_0) = Laplace(I; \hat{I}(I_0, \theta), bE) \tag{6}$$

where I_0 is the input LR image, I is the HR image, $\hat{I}$ is the training output. b is the parameter of Laplacian distribution, E is the identity matrix. Then, optimizing (4) is to find the maximum likelihood estimation (MLE) of (6).

3.2. Bilinear Interpolation Method

This is an extension of linear interpolation for interpolation functions of two variables(e.g. x and y) on a rectilinear 2D grid. It is one of the basic resampling techniques in computer vision and image processing, where it is also called bilinear filtering or bilinear texture mapping.[5] Suppose we have 4 points: $Q_{11}(x_1, y_1), Q_{12}(x_1, y_2), Q_{21}(x_2, y_1), Q_{22}(x_2, y_2),$ and we want to know the value of $f(x, y)$. We can do linear interpolation in x-direction:

$$f(x, y_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \tag{7}$$

$$f(x, y_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \tag{8}$$

Then, we can get the result by doing linear interpolation in y-direction:

$$\begin{aligned} f(x, y) &\approx \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2) \\ &= \frac{f(Q_{11})(x_2 - x)(y_2 - y) + f(Q_{12})(x_2 - x)(y_1 - y)}{(x_2 - x_1)(y_2 - y_1)} \\ &+ \frac{f(Q_{21})(x_1 - x)(y_2 - y) + f(Q_{22})(x_1 - x)(y_1 - y)}{(x_2 - x_1)(y_2 - y_1)} \end{aligned} \tag{9}$$

For the LR images, we can use bilinear interpolation method to compute the value for unknown pixels using the 4 nearest known pixels. It then gives the up-scaled images. Figure 2 gives a simple example.

3.3. Transposed Convolution Method

Transposed convolution method is another way to do up-sampling. The sketch of transposed convolution is shown in figure 3. We can use a simple CNN with just Conv2DTranspose layer to realize it. The parameters of this layer can be computed by the formula:

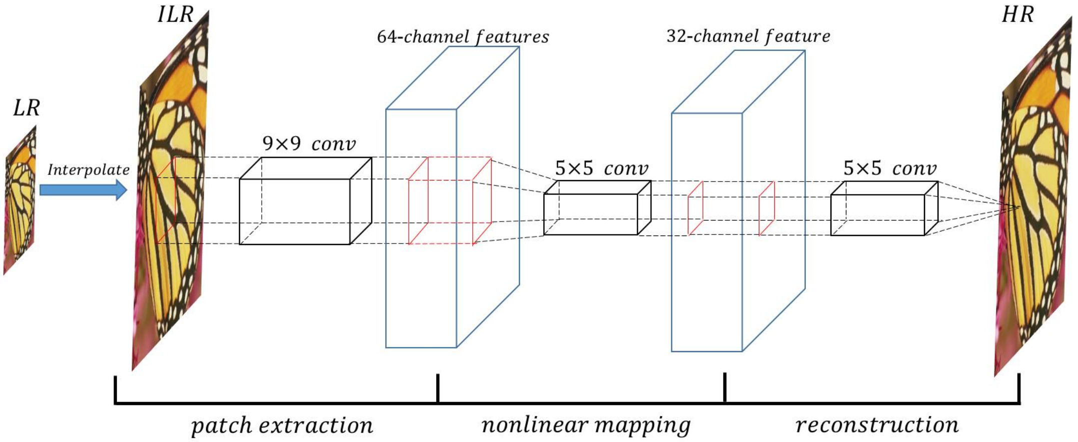
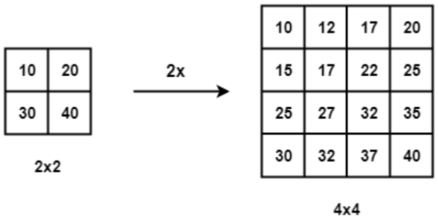


Figure 1. Sketch of SRCNN Architecture[7]



10	20
30	40

10	12	17	20
15	17	22	25
25	27	32	35
30	32	37	40

Figure 2. An Example of Bilinear Interpolation Method[1]

$$o = \frac{i + 2p - k}{s} + 1 \tag{10}$$

where o is the output size, i is the input size, p is the number of paddings, k is the kernel size, s is the stride size. We utilize this simple method to compare with complicated SCRNN model to check whether the result could be better.

4. Experiments

4.1. Dataset

The DIV2K dataset, which includes 900 2K resolution images and the corresponding low-resolution versions with downscaling factors of 2, 3, 4 and 8 enables us to take low-resolution images as inputs and compare outputs with the high-resolution versions. But our computers don't have enough memories, so we decide to use the original LR versions with downscaling factors of 8 as our HR images. We resize these HR images into 256×256. Then we resize the transformed images into 64×64 as our LR images. We take 800 images as our training data, 50 as validation data, and 50 as testing data. [3]

4.2. Software

The tools we mainly use are Jupyter and PyTorch based on Python. The packages we use are *torch, os, torchvision, pandas, numpy, matplotlib, time* and *random*.

4.3. Hardware

Most experiments will be running on a 4×2080Ti server. The server is sufficient to process all images from our dataset.

4.4. Details of Experiments

For SRCNN model, its architecture is shown in table 1. We set batch size to be 64, number of epochs to be 64 and learning rate to be 0.001.

Layer	Input	Output	Kernel	Stride	Padding
Conv2d	3	64	9×9	1	4
ReLU					
Conv2d	64	32	5×5	1	2
ReLU					
Conv2d	32	3	5×5	1	2
Tanh					

Table 1. Architecture of SRCNN Model

For transposed convolution model, we set batch size to be 64, number of epochs to be 64 and learning rate to be 0.001. In its Conv2DTranspose layer, the parameters are $3 \times 3 \times 6 \times 6$. The activation function is Tanh.

Next, we use MAE and MSE as loss function in both models and compare their results.

5. Results and Discussion

5.1. Results

Figure 4 shows the visual results. We can see that bilinear interpolation and transposed convolution method give similar results, and SRCNN model gives much better results. This is what we expected.

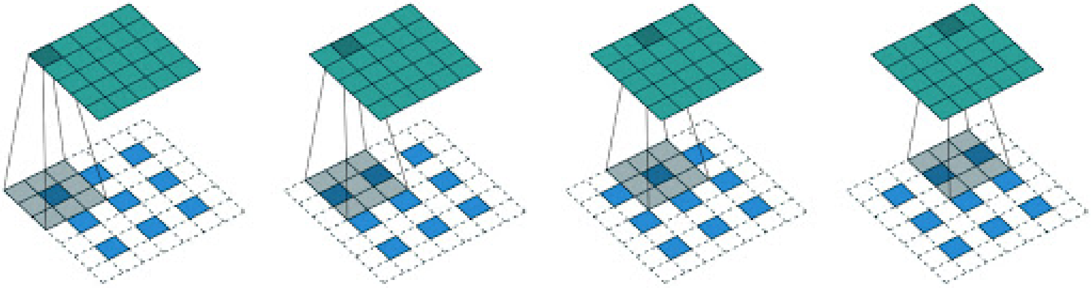
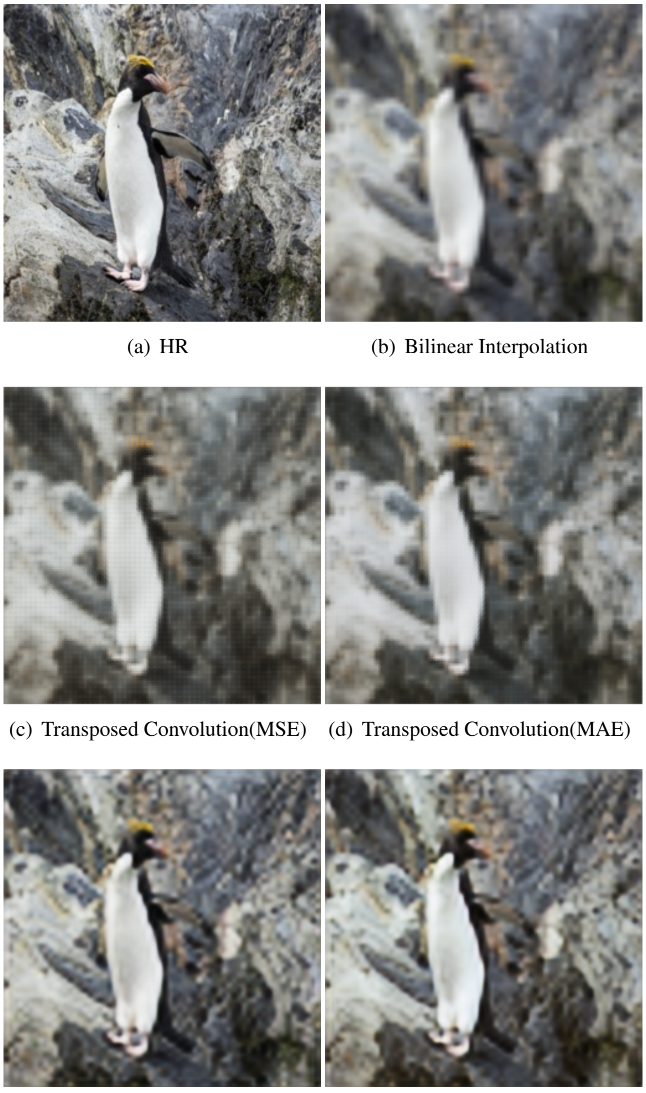


Figure 3. Sketch of Transposed Convolution[2]



(a) HR

(b) Bilinear Interpolation

(c) Transposed Convolution(MSE)

(d) Transposed Convolution(MAE)

(e) SRCNN(MSE)

(f) SRCNN(MAE)

Figure 4. Visual Comparison among Different Models

We can compare the MSE and MAE loss of these results, a more common criteria is peak signal-to-noise ratio (PSNR), which is defined as:

$$PSNR = 10log_{10}(\frac{L^2}{MSE}) \tag{11}$$

where L is maximum possible pixel value of the image, MSE is mean square error as mentioned before. In our experiment, the value of L is 255.

Another criteria is structural similarity index(SSIM), which is based on three comparison measurements between two images: luminance(L), constant(C) and structure(S).[6] They are defined as:

$$L(I, \hat{I}) = \frac{2\mu_I \mu_{\hat{I}} + k_1}{\mu_I^2 + \mu_{\hat{I}}^2 + k_1} \tag{12}$$

$$C(I, \hat{I}) = \frac{2\sigma_I \sigma_{\hat{I}} + k_2}{\sigma_I^2 + \sigma_{\hat{I}}^2 + k_2} \tag{13}$$

$$S(I, \hat{I}) = \frac{\sigma_{I\hat{I}} + k_3}{\sigma_I \sigma_{\hat{I}} + k_3} \tag{14}$$

where $\mu_I, \mu_{\hat{I}}$ are the mean of I and $\hat{I}$, $\sigma_I, \sigma_{\hat{I}}$ are the standard deviation of I and $\hat{I}$, $\sigma_{I\hat{I}}$ is the covariance between I and $\hat{I}$, k_1, k_2 are constant to stabilize the division with weak denominator, $k_3 = \frac{k_2}{2}$. Then, we can compute SSIM by combining these three measurements together.

$$\begin{aligned} SSIM(I, \hat{I}) &= L(I, \hat{I}) \times C(I, \hat{I}) \times S(I, \hat{I}) \\ &= \frac{2\mu_I \mu_{\hat{I}} + k_1}{\mu_I^2 + \mu_{\hat{I}}^2 + k_1} \cdot \frac{2\sigma_{I\hat{I}} + k_2}{\sigma_I^2 + \sigma_{\hat{I}}^2 + k_2} \end{aligned} \tag{15}$$

From table 2, we can find that bilinear interpolation method and transposed convolution method give similar results. They are almost the same in all four criteria. SRCNN model is much better than the other two methods, which gives smaller MSE and MAE, and larger PSNR and SSIM.

5.2. Discussion

We basically complete the experiment as planned, but there are still some shortcomings that could be further improved. One problem is that the sizes of our HR images are 255×255, which are considerably small compared with

Model	MSE	MAE	PSNR	SSIM
BI*	0.0065	0.0602	70.0017	0.9237
TC(MSE)**	0.0068	0.0620	69.8058	0.9236
TC(MAE)	0.0066	0.0608	69.9354	0.9285
SRCNN(MSE)	0.0046	0.0511	71.5032	0.9514
SRCNN(MAE)	0.0047	0.0507	71.4098	0.9507

Table 2. Comparison among Different Models

* BI means bilinear interpolation.

** TC means transposed convolution.

the real HR images. If we can implement for the original 2040 × 2040 HR image, the result might be more obvious and thus our experiment could be much better.

From figure 5 and 6[1], we found that the results can potentially be improved because it is still underfitted. In our experiment, the learning rate is 0.001 and the number of epochs is 64. We only used one set of hyperparameters. The results might be better if we do a grid hyperparameter search and select the best performing model.

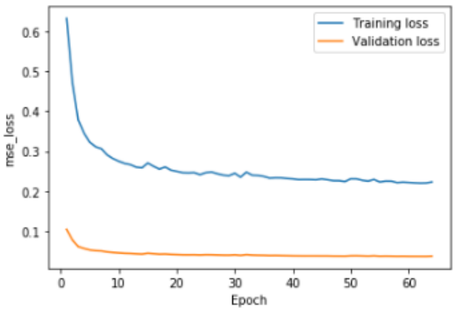


Figure 5. Loss Curve of SRCNN Model(MSE)

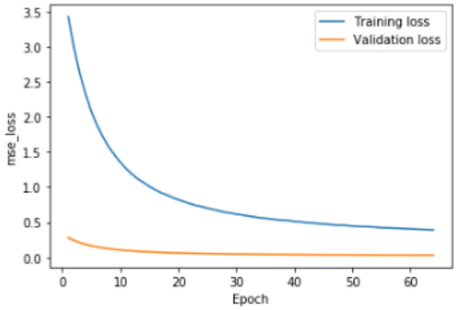


Figure 6. Loss Curve of Transposed Convolution Model(MSE)

SRCNN is a very classical technique to do super-resolution. However, there are also many advanced techniques emerging recently, such as Efficient Sub-Pixel Convolution Neural Network (ESPCN), and Fast Super-Resolution CNN (FSRCNN). Figure 7 and 8 give a detailed sketch of these two techniques. A further improvement to our project might be that we use these advanced techniques to train our models to process images and then compare the results across different approaches.

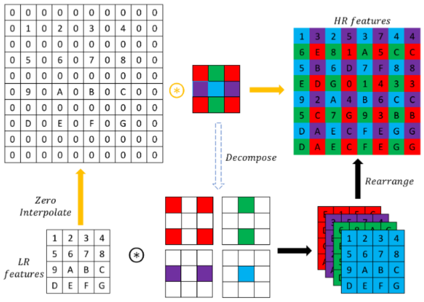


Figure 7. Sketch of ESPCN[7]

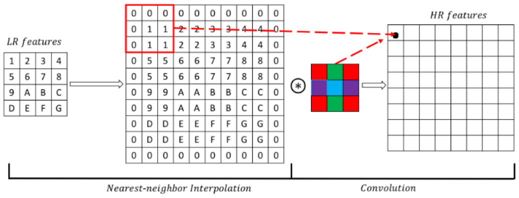


Figure 8. Sketch of FSRCNN[7]

6. Conclusions

The initial motivation of our experiment is to increase the resolution of single image input, based on the fact that high resolution (HR) images can provide more information for their users and therefore be more useful with different aspects in our daily lives. According to the experiment, we consider that our goal is not completely achieved, since the results are not very desirable. In this project, we discuss three methods: super-resolution convolutional neural network (SRCNN), bilinear interpolation, and a simple convolutional neural network (CNN) with one layer. We have two loss functions: mean square error and mean absolute error, which are used and compared as the optimization objectives of the neural networks in image processing. We select MSE loss used in SRCNN as the benchmark. Besides two optimization objectives, we have two approaches: Peak Signal to Noise Ratio (PSNR) and structural similarity index (SSIM). PSNR is the widely used metric for quantitatively evaluating image restoration quality, whereas SSIM is used to evaluate models from three aspects: luminance, contrast, and structure. Same as PSNR, SSIM measures reconstruction quality and compare the similarities of two images comprehensively. From the results of this experiment, SRCNN outperforms the single-layer convolutional neural network and the bilinear interpolation method. Thus, we

[1] The curves show the sum of MSE of each images in train set and validation set, so the MSE loss of train set is higher than MSE loss of validation set.

believe that SRCNN is the most useful model among three. We implement three methods to the best of our knowledge. In the future, we expect that further approaches may be explored with more complicated neural network models and different hyperparameters.

7. Contributions

For the computational part, Whitney is responsible for developing data loader and augmenting the data. Chenyang is responsible for the experimental part and developing deep learning model. Fangying is responsible for developing an evaluation method. For the writing part, each of is actively sharing ideas and contributing to improve every part of the final report. Specifically, Fangying is mainly responsible for introduction and related works. Chenyang is responsible for method and experiment parts. Whitney is responsible for results and discussion parts.

References

[1] https://theailearner.com/2018/12/29/image-processing-bilinear-interpolation/.

[2] https://buptldy.github.io/2016/10/29/2016-10-29-deconv/.

[3] E. Agustsson and R. Timofte. *NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study*. July 2017.

[4] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, 2016.

[5] Wikipedia contributors. Bilinear interpolation — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Bilinear_interpolation&oldid=953557436, 2020. [Online; accessed 3-May-2020].

[6] Wikipedia contributors. Structural similarity — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Structural_similarity&oldid=953403184, 2020. [Online; accessed 3-May-2020].

[7] W. Yang, X. Zhang, Y. Tian, W. Wang, J. Xue, and Q. Liao. Deep learning for single image super-resolution: A brief review. *IEEE Transactions on Multimedia*, 21(12):3106–3121, Dec 2019.