Report: Super-Resolution for Downscaling on Oceanographic Fields

Kewei XU - kewei.xu@etu.sorbonne-universite.fr

February 13, 2022

Abstract

We address the problem of retrieving a geophysical field at a higher resolution from various low-resolution satellite data. The downscaling of oceanographic fields may be viewed as a super-resolution problem. In this project, we use a deep learning approch [6] [5] to solve the problem. I chose a classical super-resolution network, the SRCNN [1]. And optimized it based on its network architecture and the structure of data used. Finally tested on dataset NATL60 and obtained satisfactory results.

Keywords
Downscaling, Super-resolution, Oceanographic

1 Introduction

Basically, this report will be divided into four main parts. The first part will introduce about the dataset and the pre-processing that makes data more suitable for training. Then a detailed introduction to the network I use, the SRCNN [1]. This will be followed by an optimization of the network and training data to improve the super-resolution results. Finally, a brief description of the training and testing methods will be given and comparison results will be shown.

2 Dataset and Pre-processing

In this section, I will cover the introduction about the dataset and the pre-processing I apply to the dataset.

2.1 Dataset

The dataset I use is NATL60. It contains ocean Monitoring Information of North Atlantic Ocean in a period between 2006 and 2017. The information is recorded at one day intervals. We can get 3734 images from this dataset. In this project, we mainly consider two types of images in this dataset, Sea Surface Temperature (SST) and Sea Surface Height (SSH).

2.2 Pre-processing

Since we are only trying to super-resolve the data on the ocean field. However, the image contains a lot of continental data, so I want to keep as much of the ocean data as possible while eliminating the continental data. So the image was cut into 32 small patches (Figure 1). Although some image segmentation algorithms can be used to achieve the purpose of culling data based on the grayscale relationship of the image. But since all the images were recorded at the same location, so theoretically, it is possible to cut for all the images as long as one cut was done. So in the end, I chose to do the cut manually, which also ensures that all patches correspond to the same geographic location information. This also makes it easier to train and test the data if it is for a specific field range.

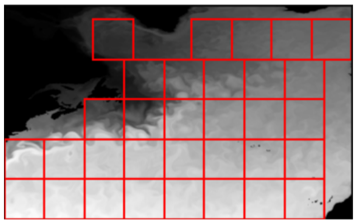

Figure 1: Cut in 32 patches, patch size: 90 x 90

To acquire a low-resolution image (Figure 2), a simple nearest-neighbor interpolation is used to downsample the original resolution image by a factor of three.

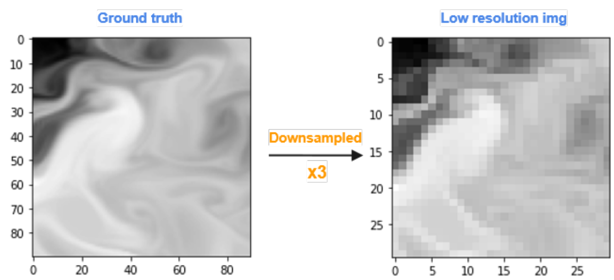

Figure 2: Downsampled by factor 3

3 Super-Resolution Using Convolution Networks

In this section, I will explain a classical convolutional neural network for image super-resolution, which is also the main network used in this project. The Super-Resolution Convolutional Neural Network (SRCNN) [1]. The architecture of this network is quite simple, it contains three convolutional layers (Figure 3) :

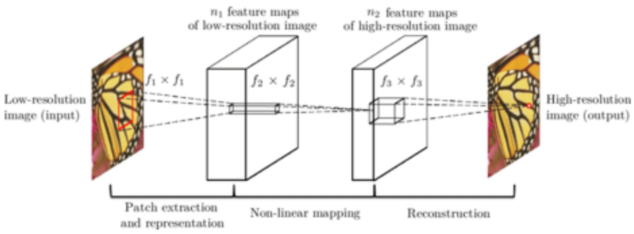

Figure 3: SRCNN architecture [Dong et al. 2015]

3.1 Three convolution layers

Since there is no deconvolution layer in the network, SRCNN requires a pre-processing, it first uses bicubic interpolation to enlarge the low-resolution image to the target size (size after super-resolution). Here, we define $\mathbf{Y}$ as low-resolution image after interpolation.

3.1.1 Patch extraction and representation

$$F_1(\mathbf{Y}) = \max(0, W_1 * \mathbf{Y} + B_1)$$

Here max() is the non-linear activation function **ReLU**. W_1 and B_1 are parameters of convolution function of n_1 filters $(c \times f_1 \times f_1)$, where c is the number of channel of low-resolution image, f_1 is the height and width of filters, and n_1 represent the number of output channel.

3.1.2 Non-linear mapping

$$F_2(\mathbf{Y}) = \max(0, W_2 * F_1(\mathbf{Y}) + B_2)$$

Here max() is the nonlinear activation function **ReLU**. W_2 and B_2 are parameters of convolution function of n_2 filters $(n_1 \times f_2 \times f_2)$, where n_1 is the number of channel of out put feature maps in the layer before, f_2 is the height and width of filters, and n_2 represent the number of output channel. In fact here is just a non-linear mapping layer to transform feature maps from low-resolution to high-resolution, we can set the size of filter to 1, $f_2 = 1$.

3.1.3 Reconstruction

$$F_3(\mathbf{Y}) = W_3 * F_2(\mathbf{Y}) + B_3$$

W_3 and B_3 are parameters of convolution function of c filters $(n_2 \times f_3 \times f_3)$, where n_2 is the number of channel of out put feature maps in the layer before, f_3 is the height and width of filters, and c represent the number of output channel, here we reconstruction the high-resolution image, the output channel c need to be equal to the channel of low-resolution image that as the input of the network.

3.2 Training

Loss function, use simple Mean Squared Error (MSE Loss).

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2$$

Here, I stand for the high-resoltion image, K stand for the predicted image by network. So basically the loss function is compute the gray value distance between every pixels of the same locations from the predicted image and the ground truth.

For the learning rate, a little change was made. For the first two layers of the network, the original initial learning rate is used, and in the last layer, the learning rate is reduced to 10% of the original value.

3.3 Testing

For testing the model, I use Peak Signal to Noise Ratio (PSNR). This is the function to determine the similarity of two images.

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

Here MAX_I stand for the max value of image, if the image is coding in 8 bits, it will be 255. Normally $PSNR = 50$. It is basically difficult to distinguish the two images.

The entire super-resolution processing pipeline is shown in the following figure (Figure 4).

Figure 4: Super-resolution pipeline

4 Improving Results

In this section, I will explain some improvements based on the network's own architecture and the structure of the data itself.

4.1 Add additional data

In fact we can use high resolution SST images to assist in reconstructing low resolution SSH images.

The steps are as follows: for the input interpolated low-resolution SSH image, an additional channel is added to it to add the high-resolution SST image. In this way the input of the network becomes a two-channel image, so we duplicate one channel of the high-resolution SSH image (The ground truth) and take the first channel of the predicted image as the result (Figure 5).

Figure 5: Add additional data

4.2 Sequence processing

The next improving is adding time sequence processing. Because the dataset is recorded by day, so the images is actually a sequence just like video, only difference is the time interval between frames is one day.

I implement a video super resolution network based on SRCNN [1] which called VSRnet [3]. The whole pipeline is shown as follows (Figure 6):

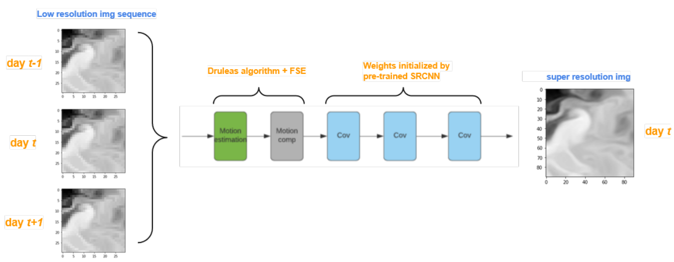

Figure 6: Sequence process

The three input frames are concatenated along the first dimension before first convolutional layer is applied. The new input image for layer 1 in SRCNN [1] is 3 channels. The step follows is a motion processing module consisting by motion estimation and motion compensation. Here, for motion estimation use Druleas algorithm [2]. For motion compensation, use an adaptive motion compensation (AMC) which can reduces the influence of neighboring frames for the reconstruction in case of misregistration.

$$y_{t-T}^{amc}(i,j) = (1 - r(i,j))y_t(i,j) + r(i,j)y_{t-T}^{mc}(i,j)$$

$$r(i,j) = \exp(-ke(i,j))$$

where y_t is the center frame, y_{t-T}^{mc} is the motion compensated neighboring frame and y_{t-T}^{amc} is the neighboring frame after applying adaptive motion compensation. k is a constant, $e(i,j)$ is the misregistration error. If $e(i,j)$ is large which makes $r(i,j)$ small, then the adaptively motion compensated pixel is just depend on the pixel in the current frame y_t.

The rest part is exactly the same SRCNN [1], we can initialized the weights by a pre-trained SRCNN for saving time and getting better results.

5 Experiment

In this section, I will show the final super-resolution results. With SSH data, where we compare the results using the basic SRCNN [1] and the SRCNN with SST data added. With SST data, where we compare the results using the basic SRCNN and VSRnet.

5.1 Testing on SSH:

SRCNN parameters: layer 1: $f_1 = 5, n_1 = 64$, layer 2: $f_2 = 1, n_2 = 32$, layer 3: $f_3 = 5$. Using 32000 patches for training and 320 patches for testing, the average PSNR value is shown follows:

Table 1: Testing on SSH

Method	Interpolations	SRCNN	SRCNN+SST
PSNR avg	41.88	51.45	54.50

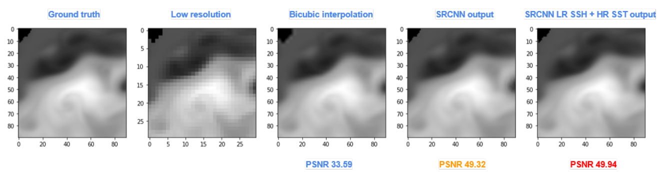

Figure 7: Example of testing on SSH

5.2 Testing on SST:

SRCNN parameters: layer 1: $f_1 = 5, n_1 = 64$, layer 2: $f_2 = 1, n_2 = 32$, layer 3: $f_3 = 5$. Image sequence size: 5. Using 32000 patches for training and 320 patches for testing, the average PSNR value is shown follows:

Table 2: Testing on SST

Method	Interpolations	SRCNN	VSRnet
PSNR avg	41.39	44.94	46.24

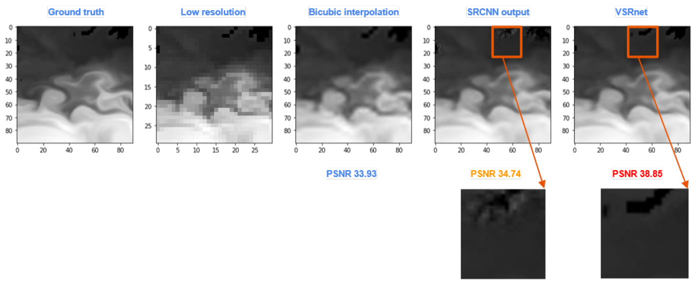

Figure 8: Example of testing on SST

Here is the one thing I noticed from the example, the SRCNN can't reconstruct well the edges between the land and the ocean, but VSRnet can fix this problem.

6 Conclusion

As the results I've shown before. These two ways are both good directions to improve the performance of the convolutional network. For the future implementation, maybe can increase the layers in the network or using more complex architecture of network like SRGAN [4] or maybe find out some specific data augmentation methods for oceanographic field images.

References

[1] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

[2] Marius Drulea and Sergiu Nedevschi. Total variation regularization of local-global optical flow. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 318–323. IEEE, 2011.

[3] Armin Kappeler, Seunghwan Yoo, Qiqin Dai, and Aggelos K Katsaggelos. Video super-resolution with convolutional neural networks. *IEEE transactions on computational imaging*, 2(2):109–122, 2016.

[4] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[5] Hongying Liu, Zhubo Ruan, Peng Zhao, Chao Dong, Fanhua Shang, Yuanyuan Liu, and Linlin Yang. Video super resolution based on deep learning: A comprehensive survey. *arXiv preprint arXiv:2007.12928*, 2020.

[6] Zhihao Wang, Jian Chen, and Steven C. H. Hoi. Deep learning for image super-resolution: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(10):3365–3387, 2021.