
Image Super-Resolution Using Convolutional Neural Network

Yongming Ding 116039910001



Low resolution



After Super-Resolution

Abstract

With the current surge in popularity of image-based applications, improving content quality is vital. While hardware-based solutions do exist, an approach called image super-resolution adopts a more software-based approach. Specifically, the goal of image super-resolution is to sharpen or improve the quality of a low-resolution (LR) image input by outputting a super-resolved high-resolution (HR) image output. Deep learning techniques have been successfully applied in many areas of computer vision, including low-level image restoration problems. In this paper, we propose three convolutional neural network (CNN) models that takes the low-resolution image as the input and outputs the high-resolution one.

Keywords: Super-resolution, convolutional neural network

Introduction

Single image super-resolution (SR) aims at obtaining a high-resolution (HR) image from a low-resolution (LR) input image by inferring all the missing high frequency contents. With the known variables in LR images greatly outnumbered by the unknowns in HR images, SR is a highly ill-posed problem and the current techniques are far from being satisfactory for many real applications[1]. To regularize the solution of SR, people have exploited various priors of natural images. Analytical priors, such as bicubic interpolation, work well for smooth regions; while image models based on statistics of edges [2] and gradients [3] can recover sharper structures. In the patch-based SR methods, HR patch candidates are represented as the sparse linear combination of dictionary atoms trained from external databases [4], or recovered from similar examples in the LR image itself at different locations and across different scales [5, 6].

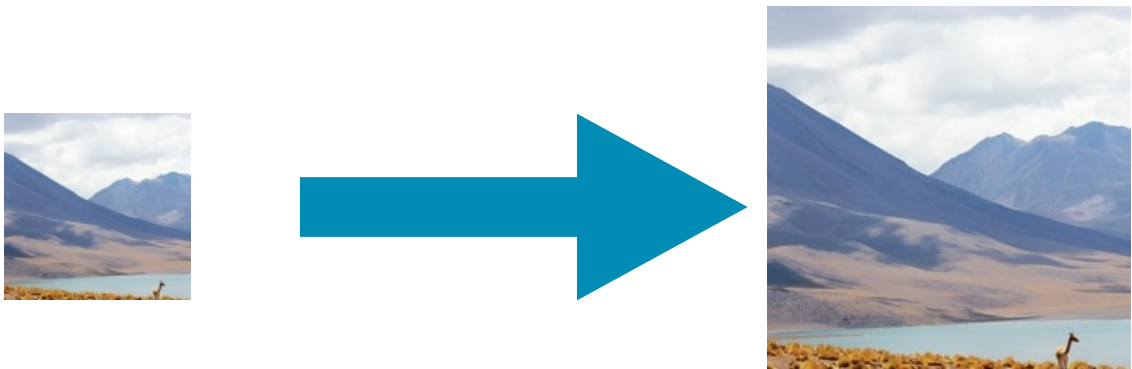


Fig.1 Super-resolution task

More recently, inspired by the great success achieved by deep learning [7, 8, 9] in other computer vision tasks, people begin to use neural networks with deep architecture for image SR. Multiple layers of collaborative auto-encoders are stacked together in [10] for robust matching of self-similar patches. Deep convolutional neural networks (CNN) [11] and deconvolutional networks [12] are designed that directly learn the non-linear mapping from LR space to HR space in a way similar to coupled sparse coding [13]. As these deep networks allow end-to-end training of all the model components between LR input and HR output, significant improvements have been observed over their shadow counterparts.

In this project, we have experimented with three different implementations of convolutional neural network models for image super-resolution. The first model called CNN_SMALL is the smallest one containing three layers. The second model called CNN_MEDIUM is larger having five layers. The final model CNN_DEEP is the largest one consisted of seven convolution layers. Besides, we implement waifu2x CNN model in a open source software [14] as for comparison.

Dataset

Training images may be more important than algorithm of deep learning to achieve high quality super resolution. Generally, training images dataset distributed for image classification task are not so high quality in terms of sharpness of the images. We choose PEXELS photos as training images in this project. This website provides high quality photos under Creative Commons Zero (CC0) license. Our training image dataset consists of 5000 medium size photos from PEXELS. These images ranged in content from flowers and animals to people, cars and other household objects. Theses photos ranged in size from 500*300 to 1500*1000. Some examples of images from dataset are shown below.

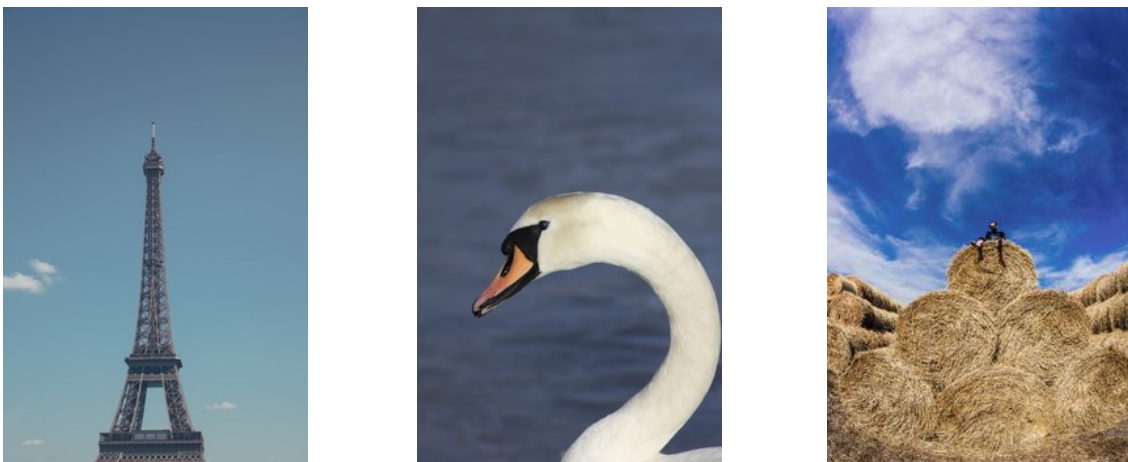


Fig.2 Image dataset examples

Convolutional Neural Networks for Super-Resolution

Convolutional neural networks (CNNs) have been used for many image processing tasks such as classification, and have recently been applied to image super-resolution. CNNs typically include convolutional layers, pooling layers, and fully-connected layers, but super-resolution literature tends to rely on architectures that only include the first type. In this paper, we propose three convolutional neural network (CNN) models: CNN_SMALL, CNN_MEDIUM and CNN_DEEP.

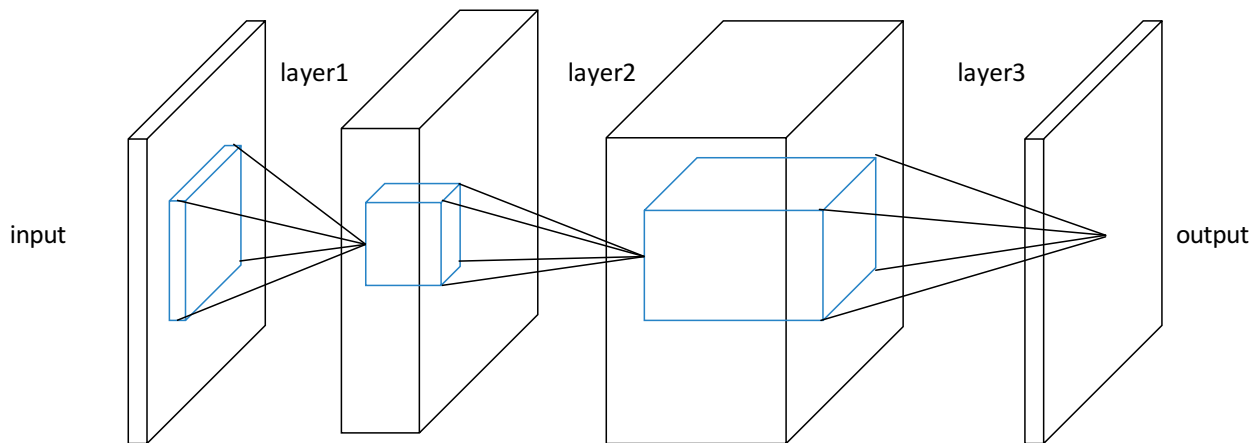


Fig.3 CNN_SMALL model

CNN_SMALL has three convolutional layers. The attributes of each layer are listed in table below.

Table.1 CNN_SMALL layers attribute

Attribute	layer1	layer2	layer3
In channels	1	8	16
Out channels	8	16	3
Kernel size	3	3	3
Parameter numbers	80	1168	435

- In channels: channel number input this layer
- Out channels: channel number output this layer
- Kernel size: the number of convolutional kernels in this layer
- Parameter numbers: the number of parameters in this layer, it equals to In channel * Out channel * (kernel size)^2 + Out channel

The activation function used is relu in all layers.

Besides, we use Mean Squared Error (MSE) as the loss function:

$$L(\Theta) = \frac{1}{n} \sum_{i=1}^n ||F(\mathbf{Y}_i; \Theta) - \mathbf{X}_i||^2$$

where n is the number of training samples. The loss is minimized using stochastic gradient descent with the standard back propagation.

Similarly, we propose a little larger convolutional neural network, CNN_MEDIUM. It has five convolutional layers. The attributes of each layer are listed in table below.

Table.2 CNN_MEDIUM layers attribute

Attribute	layer1	layer2	layer3	layer4	layer5
In channels	1	16	32	64	32
Out channels	16	32	64	32	3
Kernel size	3	3	3	3	3
Parameter numbers	160	4640	18496	18464	867

The activation function used is relu and the loss function is MSE, as same as CNN_SMALL.

Finally, we build a deep convolutional neural network, CNN_DEEP. It has seven convolutional layers. The attributes of each layer are listed in table below.

Table.3 CNN_DEEP layers attribute

Attribute	layer1	layer2	layer3	layer4	layer5	layer6	layer7
In channels	1	32	64	128	128	64	32
Out channels	32	64	128	128	64	32	3
Kernel size	3	3	3	3	3	3	3
Parameter numbers	320	18496	73856	147584	73792	18464	867

The activation function used is relu and the loss function is MSE, as same as CNN_SMALL and CNN_MEDIUM.

Additionally, we implement the waifu2x CNN model. Waifu2x is an open source software, originally published to enlarge Anime-style art image and it also supports picture style or real photos now.

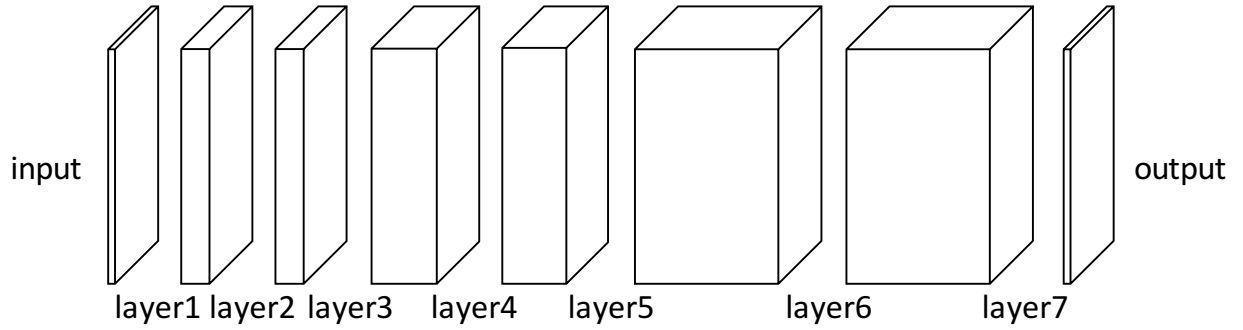


Fig.4 waifu2x CNN model

As shown in Fig.4, Waifu2x CNN model contains seven convolutional layers too and detailed parameters are listed in Table.4.

Table.4 waifu2x CNN layers attribute

Attribute	layer1	layer2	layer3	layer4	layer5	layer6	layer7
In channels	1	32	32	64	64	128	128
Out channels	32	32	64	64	128	128	3
Kernel size	3	3	3	3	3	3	3
Parameter numbers	320	9248	18496	36928	73856	147584	3459

The activation function used is relu and the loss function is MSE, as same as CNN_SMALL, CNN_MEDIUM and CNN_DEEP.

Code implementation

We use Python to implement these CNN models and train models on the server 202.121.181.3. The project structure is shown in shown in Table.5.

Table.5 Project code structure

Function Name	Job of Function
<code>train.py</code>	Train CNN models according to parameters
<code>evaluate.py</code>	Do super resolution on specific image to evaluate CNN models
<code>draw_lossplot.py</code>	Draw the loss-step plot during CNN models training process
<code>arch/cnn_small.py</code>	CNN_SMALL model implementation
<code>arch/cnn_medium.py</code>	CNN_MEDIUM model implementation
<code>arch/cnn_deep.py</code>	CNN_DEEP model implementation
<code>arch/cnn_waifu2x.py</code>	Waifu2x CNN model implementation
<code>tools/*</code>	Image processing, resize images and prepare data for training

Results and Analyze

For each CNN model: CNN_SMALL, CNN_MEDIUM, CNN_DEEP and waifu2x CNN, we use the same training image dataset containing 5000 images and iterate 100 epoch. Fig.5 - Fig.8 show the loss-epoch plot during training process for four models.

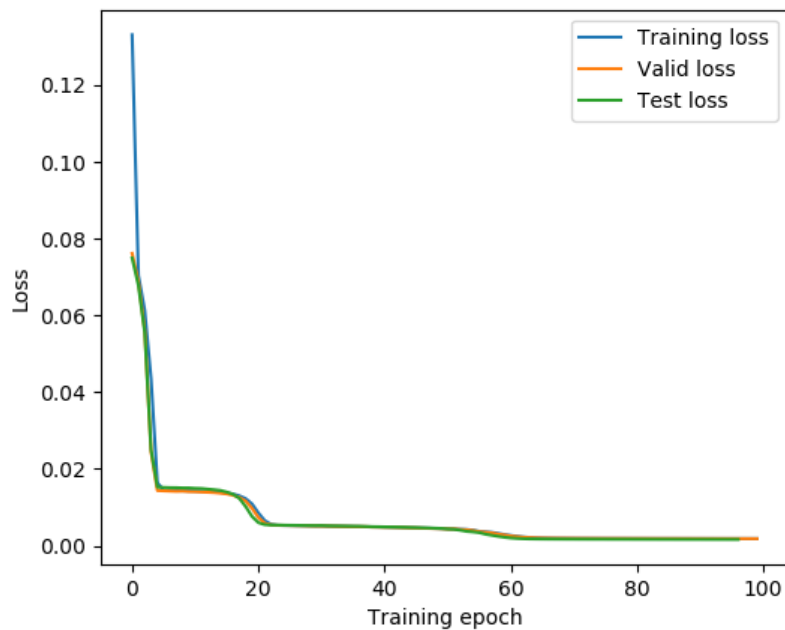


Fig.5 CNN_SMALL loss-epoch plot

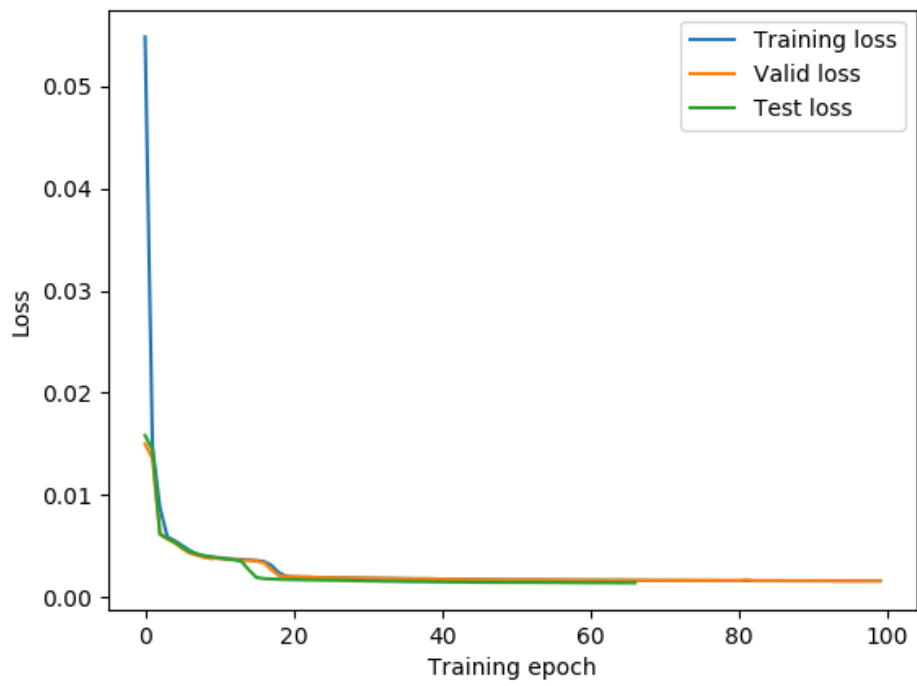


Fig.6 CNN_MEDIUM loss-epoch plot

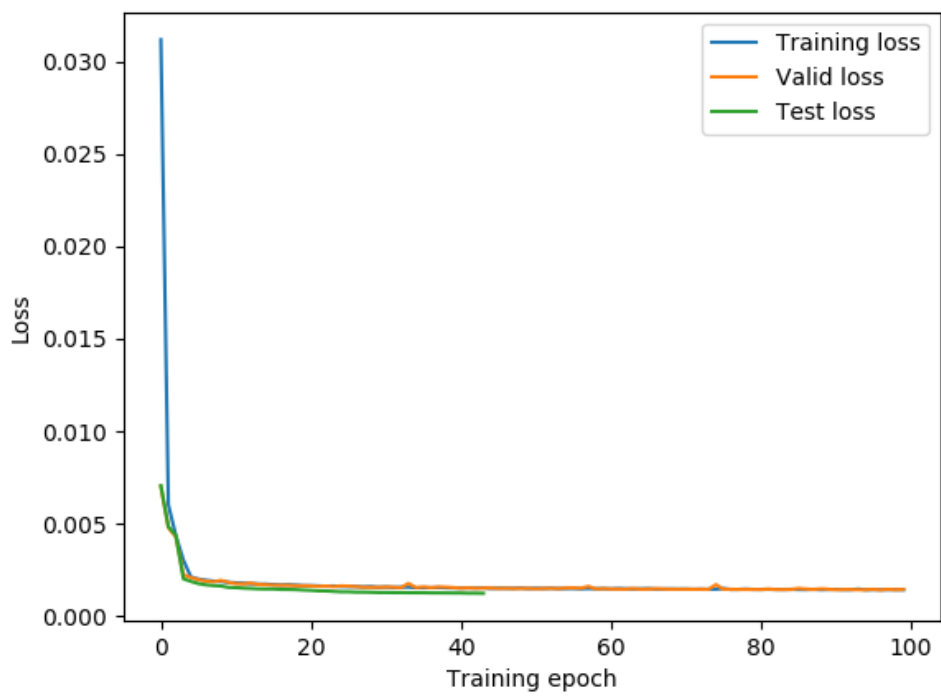


Fig.7 CNN_DEEP loss-epoch plot

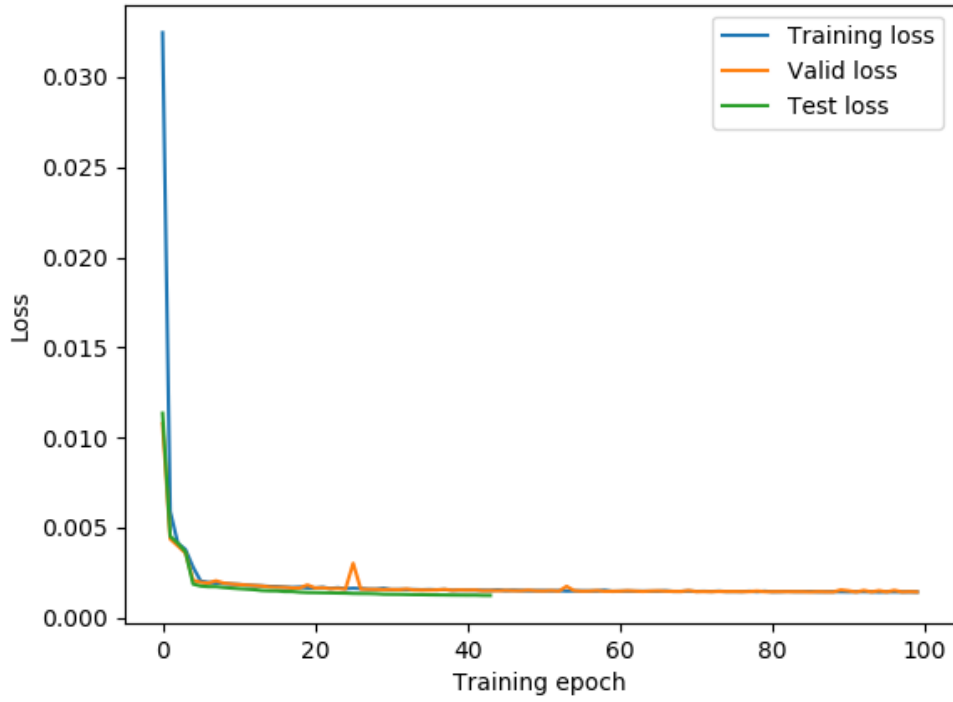


Fig.8 waifu2x CNN loss-epoch plot

After 100 training epoch, the final training loss, validation loss, test loss for four models are listed in Table.6:






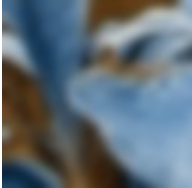










Table.6 Loss value comparison for four models

	CNN_SMALL	CNN_MEDIUM	CNN_DEEP	Waifu2x CNN
Training loss	0.001820	0.001583	0.001422	0.001425
Validation loss	0.001832	0.001590	0.001451	0.001429
Best test loss	0.001625	0.001387	0.001234	0.001236
Training time (second)	678	1509	3921	4468

As we can see, with the CNN layer number increasing, the loss decrease significantly while the training time cost increased a lot. Also, we can notice that **our proposed CNN_DEEP model get lower loss than Waifu2x CNN model while it cost 10% less training time too.**

To show the training process more intuitively, I use the same low-resolution image as the input test image for each model during training process. And, at certain epoch, perform super-resolution for this test image. The result is shown in Table.7:

Table.7 Super-Resolution Performance during Training Process

Model Name	low-resolution image input	Super- Resolution at Epoch 5	Super- Resolution at Epoch 50	Super- Resolution at Epoch 100
CNN_SMALL				
CNN_MEDIUM				
CNN_DEEP				
Waifu2x CNN				

As we can see, CNN_DEEP and Waifu2x CNN can get much better performance than CNN_SMALL and CNN_MEDIUM at the same epoch. This result is consistent with the loss value and loss-epoch plots shown before.

Conclusion

In this paper, we have experimented with three different implementations of convolutional neural network models for image super-resolution: CNN_SMALL, CNN_MEDIUM and CNN_DEEP. Also, Waifu2x CNN model used in a open source software is built as for comparison. As a result, our 7-layers CNN_DEEP model get best loss value and its training time is 10% less than the Waifu2x CNN model — a 7-layers CNN model too.

Reference

1. S. Baker and T. Kanade. Limits on super-resolution and how to break them. *IEEE TPAMI*, 24(9):1167–1183, 2002.
2. R. Fattal. Image upsampling via imposed edge statistics. In *ACM Transactions on Graphics*, volume 26:3, page 95, 2007.
3. K. I. Kim and Y. Kwon. Single-image super-resolution using sparse regression and natural image prior. *IEEE TPAMI*, 32(6):1127–1133, 2010.
4. J. Yang, Z. Wang, Z. Lin, S. Cohen, and T. Huang. Coupled dictionary training for image super-resolution. *IEEE TIP*, 21(8):3467–3478, 2012.
5. G. Freedman and R. Fattal. Image and video upscaling from local self-examples. *ACM Transactions on Graphics*, 30(2):12, 2011.
6. D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *ICCV*, 2009.
7. A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.
8. P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv: 1312.6229*, 2013.
9. P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, pages 1096–1103, 2008.
10. Z. Cui, H. Chang, S. Shan, B. Zhong, and X. Chen. Deep network cascade for image super-resolution. In *ECCV*, pages 49–64, 2014.
11. C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, pages 184–199, 2014.
12. C. Osendorfer, H. Soyer, and P. van der Smagt. Image super-resolution with fast approximate convolutional sparse coding. In *NIPS*, pages 250–257. Springer, 2014.
13. J. Yang, Z. Wang, Z. Lin, S. Cohen, and T. Huang. Coupled dictionary training for image super-resolution. *IEEE TIP*, 21(8):3467–3478, 2012.
14. <http://waifu2x.udp.jp/>