Effective Super-Resolution Methods for Paired Electron Microscopic Images

Data and Codes

All MATLAB codes are tested in MATLAB R2018b with Image Processing and Parallel Computing Toolboxes. The three deep learning methods (VDSR, EDSR and RCAN) are tested on the Texas A&M University HRPC Clusters under Ubuntu 14.04/16.04, Python3.6, PyTorch\_0.4.0, CUDA8.0, cuDNN5.1 with Titan X/1080Ti/Xp GPUs.

**1. Prepare Data**

1.1 Inputs Data

There are 22 pairs of 1280 × 1024 electronic microscopic (EM) images, indexed by #\_#. Take those images and put them in the ‘RawData’ folder. The low-resolution (LR) images, with × 10,000 magnification, are labeled with ‘\_10000’; while the high-resolution (HR) images, with × 20,000 magnification, are labeled with ‘\_20000’.

1.2 Global Registration

Run ‘Global\_Registration.m’ to output the overlapping areas of the pair images under the ‘SubImages’ folder. The registered 640 × 472 LR images are labeled with ‘\_LR’, while the 1280 × 944 HR images are labeled with ‘\_HR’.

1.3. Local Registration

Run ‘Local\_Registration.m’ to generate ‘#\_#\_Shift\_Map.mat’ for the image pair #\_# under the ‘SubImages’ folder.

1.4. Subimage Segmentation

Run ‘Segment\_Subimage.m’ to segment each pair into 12 subimages: 9 in-sample images and 3 out-of-sample ones. The LR (160 × 157) and HR (320 × 314) in-sample images are labeled with ‘\_InSample\_#’ and \_InSample\_#\_GT’ respectively. The LR and HR out-of-sample ones are labeled with ‘\_OutSample\_#’ and ’\_OutSample\_#\_GT’ respectively.

The outputs are stored under the ‘SubImages’ folder.

**2. Implement Super-Resolution Methods**

**2.1 LB-NLM methods**

2.1.1 Pooled Training

Run ‘Sample\_Library\_Paired\_Shift\_Pooled.m’ to generate a patch library using the left 75% area of all the training image pairs, named as ‘Library\_Pooled\_P9\_S80000\_paired\_shift.mat’ under the ‘Libraries’ folder.

2.1.2 Self Training

Run ‘Reconstruct\_LBNLM\_Self.m’ to generate a patch library using the left 75% area of each training image pair #\_#, named as ‘Library\_#\_#\_P9\_S80000\_paired\_shift.mat’ under the ‘Libraries’ folder.

2.1.3 Paired LB-NLM with Pooled Training

Run ‘Reconstruct\_LBNLM\_Pooled.m’. All the results are stored under the ‘Result/LBNLM’ folder. The reconstructed images are labeled as ‘\_LBNLM\_pooled.bmp’.

2.1.4 Paired LB-NLM with Self Training

Run ‘Reconstruct\_LBNLM\_Self.m’. The reconstructed results are also stored under the ‘Result/LBNLM’ folder, labeled as ‘\_LBNLM\_self.bmp’.

2.1.5 Original LB-NLM with Pooled Training

Run ‘Reconstruct\_LBNLM\_or\_Pooled.m’. All the results are stored under the ‘Result/LBNLM\_or’ folder. The reconstructed images are labeled as ‘\_LBNLM\_or\_pooled.bmp’.

2.1.6 Original LB-NLM with Self Training

Run ‘Reconstruct\_LBNLM\_ or\_Self.m’. The reconstructed results are also stored under the ‘Result/LBNLM\_or’ folder, labeled as ‘\_LBNLM\_or\_self.bmp’.

**2.2 ScSR Method**

Copy all in-sample subimages (‘#\_#\_InSample\_#.bmp’ and ‘#\_#\_InSample\_#\_GT.bmp’) from the ‘SubImages’ folder into the ‘ScSR/Data/Traning\_All’ folder.

2.2.1 Train Dictionary with Pooled Training

Run ‘ScSR/Dictionary\_Training\_Pooled.m’ to learn a dictionary using all training images. The trained dictionary is stored in ‘ScSR/Dictionary/D\_1024\_0.15\_5\_Pooled.mat’.

2.2.3 Train Dictionary with Self Training

Run ‘ScSR/Dictionary\_Training\_Self.m’ to learn a dictionary from each image pair #\_#. The trained dictionary is stored in ‘ScSR/Dictionary/D\_1024\_0.15\_5\_#\_#.mat’

2.2.3 ScSR with Pooled Training

Run ‘ScSR/ScSR\_paired\_Pooled.m’. All the results are stored under the ‘Result/ScSR/Pooled’ folder. The reconstructed images are labeled as ‘\_ScSR.bmp’.

2.2.4 ScSR with Self Training

Run ‘ScSR/ScSR\_paired\_Self.m’. All the results are stored under the ‘Result/ScSR/Self’ folder. The reconstructed images are also labeled as ‘\_ScSR.bmp’.

**2.3 SRSW Method**

Copy all registered image pairs (‘#\_#\_HR.bmp’ and ‘#\_#\_LR2.bmp’) from the ‘SubImages’ folder into the ‘SRSW/Images/Train’ folder.

Copy all subimages (‘#\_#\_InSample\_#.bmp’, ‘#\_#\_InSample\_#\_GT.bmp’, ‘#\_#\_OutSample\_#.bmp’, ‘#\_#\_OutSample\_#\_GT.bmp’) from the ‘SubImages’ folder into the ‘SRSW/Images/Test’ folder.

2.3.1 Construct Database with Pooled Training

Run ‘SRSW/data\_construction\_Pooled.m’ to learn a database using all training images. The paired database is stored in ‘ScSR/Database/HR\_Data\_SR\_Pooled2.txt’ and ‘ScSR/Database/LR\_Data\_SR\_Pooled2.txt’.

2.3.2 Construct Database with Self Training

Run ‘SRSW /data\_construction\_Self.m’ to learn a database for each image pair #\_#. The paired database is stored in ‘ScSR/Database/ HR\_Data\_SR\_#\_#.txt’ and ‘ScSR/Database/ LR\_Data\_SR\_#\_#.txt’.

2.3.3 SRSW with Pooled Training

Run ‘SRSW /Demo\_SR\_Pooled.m’. All the results are stored under the ‘Result/SRSW/Pooled’ folder. The reconstructed images are labeled as ‘\_ScSR.bmp’.

2.3.4 SRSW with Self Training

Run ‘SRSW /Demo\_SR\_Self.m’. All the results are stored under the ‘Result/SRSW/Self’ folder. The reconstructed images are labeled as ‘\_ScSR.bmp’.

**2.4 Deep Learning Methods**

The procedure to use any one of the deep learning methods is as follows:

1. Make sure all steps of Section 1 “Prepare Data” are completed, so that the subimages are generated and available in the folder of *SubImages*.
2. Go to the folder of *VDSR\_EDSR\_RCAN*, run *build\_dataset\_DL.py* to generate required datasets for use by the deep learning methods. The dataset folder is *DIV2K* under *VDSR\_EDSR\_RCAN*.
3. Copy the whole folder of (and the files within) *VDSR\_EDSR\_RCAN* to HPRC or to whichever place you plan to run the deep learning code.
4. Train: run the training code for a specific deep learning method. The code is in the folder of *VDSR\_EDSR\_RCAN/RCAN\_TrainCode/code*, and the command to use is in the following subsections. The training results will be saved in the folder of *VDSR\_EDSR\_RCAN/RCAN\_TrainCode/experiment*.
5. Test: run the test/inference code for a specific deep learning method. The code is in the folder of *VDSR\_EDSR\_RCAN/RCAN\_TestCode/code*, and the command to use is in the following subsections. The test results will be saved in the folder of *VDSR\_EDSR\_RCAN/RCAN\_TestCode/SR/BI*.
6. Go to the folder of *VDSR\_EDSR\_RCAN/RCAN\_TestCode/SR/BI*. Run *rename\_pooled.py* and *rename\_self.py* to reorganize and rename the result files to be consistent with those output images processed by other methods. The detailed instruction of running those two python codes can be found in a README in the same folder.
7. Copy the organized/renamed final results from the folder of *VDSR\_EDSR\_RCAN/RCAN\_TestCode/SR/BI* to the respective subfolders under the folder of *Result*.

**The following are the python commands used in Steps 4 (for training) and 5 (for test/inference).**

2.4.1 VDSR with Pooled-Training

Training:

python vdsr\_main\_pooled.py --model VDSR --save *Training\_Result\_Folder* --scale 1 --reset --chop --save\_results --patch\_size 41 --n\_train 198 --n\_val 66 --batch\_size 16 --offset\_val 198

Inference:

CUDA\_VISIBLE\_DEVICES=0 python main.py --data\_test MyImage --scale 1 --model VDSR --n\_resgroups 10 --n\_resblocks 20 --n\_feats 64 --pre\_train ../../RCAN\_TrainCode/experiment/*Training\_Result\_Folder*/model/model\_best.pt --test\_only --save\_results --chop --save *Test\_Result\_Folder* --testpath ../../DIV2K/pooled/BI --testset imagepooled

2.4.2 VDSR with Self-Training for the image pair #\_# (replace the index of the image pair accordingly)

Training:

python vdsr\_main\_#\_#.py --model VDSR --save *Training\_Result\_Folder* --scale 1 --reset --chop --save\_results --patch\_size 41

Inference:

CUDA\_VISIBLE\_DEVICES=0 python main.py --data\_test MyImage --scale 1 --model VDSR --n\_resgroups 10 --n\_resblocks 20 --n\_feats 64 --pre\_train ../../RCAN\_TrainCode/experiment/*Training\_Result\_Folder*/model/model\_best.pt --test\_only --save\_results --chop --save *Test\_Result\_Folder* --testpath ../../DIV2K/#\_#/BI --testset image#\_#

2.4.3 EDSR with Pooled-Training

Training:

python main\_pooled.py --model EDSR --save *Training\_Result\_Folder* --scale 2 --reset --chop --save\_results --n\_train 198 --n\_val 66 --batch\_size 16 --offset\_val 198

Inference:

CUDA\_VISIBLE\_DEVICES=0 python main.py --data\_test MyImage --scale 2 --model EDSR --n\_resgroups 10 --n\_resblocks 20 --n\_feats 64 --pre\_train ../../RCAN\_TrainCode/experiment/*Training\_Result\_Folder*/model/model\_best.pt --test\_only --save\_results --chop --save *Test\_Result\_Folder* --testpath ../../DIV2K/pooled/LR --testset imagepooled

2.4.4 EDSR with Self-Training for the image-pair #\_# (replace the index of the image pair accordingly)

Training:

python main\_#\_#.py --model EDSR --save *Training\_Result\_Folder* --scale 2 --reset --chop --save\_results

Inference:

CUDA\_VISIBLE\_DEVICES=0 python main.py --data\_test MyImage --scale 2 --model EDSR --n\_resgroups 10 --n\_resblocks 20 --n\_feats 64 --pre\_train ../../RCAN\_TrainCode/experiment/*Training\_Result\_Folder*/model/model\_best.pt --test\_only --save\_results --chop --save *Test\_Result\_Folder* --testpath ../../DIV2K/#\_#/LR --testset image#\_#

2.4.5 RCAN with Pooled-Training

Training:

python main\_pooled.py --model RCAN --save *Training\_Result\_Folder* --n\_GPUs 2 --scale 2 --n\_resgroups 10 --n\_resblocks 20 --n\_feats 64 --n\_colors 1 --reset --chop --save\_results --patch\_size 96 --n\_train 198 --n\_val 66 --batch\_size 16 --offset\_val 198

Inference:

CUDA\_VISIBLE\_DEVICES=0 python main.py --data\_test MyImage --scale 2 --model RCAN --n\_resgroups 10 --n\_resblocks 20 --n\_feats 64 --patch\_size 96 --pre\_train ../../RCAN\_TrainCode/experiment/*Training\_Result\_Folder*/model/model\_best.pt --test\_only --save\_results --chop --save *Test\_Result\_Folder* --testpath ../../DIV2K/pooled/LR --testset imagepooled

2.4.6 RCAN with Self-Training for the image-pair #\_# (replace the index of the image pair accordingly)

Training:

python main\_#\_#.py --model RCAN --save *Training\_Result\_Folder* --scale 2 --n\_resgroups 10 --n\_resblocks 20 --n\_feats 64 --n\_colors 1 --reset --chop --save\_results --patch\_size 96

Inference:

CUDA\_VISIBLE\_DEVICES=0 python main.py --data\_test MyImage --scale 2 --model RCAN --n\_resgroups 10 --n\_resblocks 20 --n\_feats 64 --pre\_train ../../RCAN\_TrainCode/experiment/*Training\_Result\_Folder*/model/model\_best.pt --test\_only --save\_results --chop --save *Test\_Result\_Folder* --testpath ../../DIV2K/#\_#/LR --testset image#\_#

**3 Reproducing Tables and Figures**

There can be small differences between the reproduced results and those published in our paper due the randomness in the training process, e.g., random sampling of image patches in the ScSR, SRSW and LB-NLM, and the stochastic gradient descent in the deep learning methods.

3.1 Table I

Run ‘Table1\_XXXX.m’ to show the its performance with self-training and pooled training, where ‘XXXX’ indicates that method’s name.

3.2 Figure 7

The figure includes four in-sample subimages:

First row: 0\_1\_InSample\_1

Second row: 15\_3\_InSample\_4

Third row: 45\_3\_InSample\_5

Fourth row: 60\_2\_InSample\_1

The results can be found under the ‘Result/VDSR/Pooled Training’, ‘Result/VDSR/Self Training’, and ‘Result/LBNLM’ folders.

3.3 Figure 8

The figure includes four out-of-sample subimages:

First row: 0\_1\_OutSample\_3

Second row: 5\_2\_OutSample\_1

Third row: 45\_3\_OutSample\_3

Fourth row: 60\_4\_OutSample\_2

The results can be found under the ‘Result/VDSR/Pooled Training’, ‘Result/VDSR/Self Training’, and ‘Result/LBNLM’ folders.

3.4 Figure 9

Run ‘Segment\_FG.m’ to identify the background/foreground for each image pair. The binary images are stored under the ‘SubImage’ folder, labelled with ‘\_binary.bmp’.

The figure includes two subimages: 0\_1\_InSample\_1 and 0\_1\_OutSample\_3, which can be found under the ‘SubImage’ folder.

3.5 Table III

Run ‘Table3’ to show the foreground/background improvements of the four SR methods.

3.6 Figure 10

Run ‘Table4.m’. The detection results are stored under ‘Result/Edge’ folder. The three detection results are labelled with ‘\_Edge\_HR’, ‘\_Edge\_BI’, and ’\_Edge\_SR’ respectively. The figure demonstrates the results of the two subimages, 0\_1\_InSample\_1 and 0\_1\_OutSample\_3.

3.7 Table IV

From the outputs of ‘Table4.m’.