A Document for the Codes



Yi Zhang

2018/09/13

[1. Directory 1-1](#_Toc524790004)

[1.1. “Code” Folder 1-1](#_Toc524790005)

[1.1.1. Contents of “DeepNet\_905” 1-1](#_Toc524790006)

[1.2. “Data” Folder 1-1](#_Toc524790007)

[2. Basic Usage 2-3](#_Toc524790008)

[2.1. Train the network 2-3](#_Toc524790009)

[2.1.1. Training of ResNet 2-3](#_Toc524790010)

[2.1.2. Training of Unfolded Net 2-7](#_Toc524790011)

[2.2. Evaluate the network 2-9](#_Toc524790012)

[2.2.1. Evaluation of ResNet 2-9](#_Toc524790013)

[2.2.2. Evaluation of Unfolded Net 2-10](#_Toc524790014)

[3. Some Information about Training Time of Unfolded Net 3-11](#_Toc524790015)

[3.1. The training time on CPU and GPU 3-11](#_Toc524790016)

[3.2. What if we run multiple tasks at the same time(on HAL’s GPU) 3-11](#_Toc524790017)

[4. Which tasks do we need to submit 4-13](#_Toc524790018)

[4.1. Get pre-trained Network 4-13](#_Toc524790019)

[5. Other Instructions 5-14](#_Toc524790020)

[5.1. Notice 5-14](#_Toc524790021)

[5.2. Functions I didn’t implement 5-14](#_Toc524790022)

[5.3. When meet errors or bad performance, please have a look here 5-14](#_Toc524790023)

# Directory

1. Matlab – codes for running the L+S FISTA codes, SVD and wall filtering.
2. Data – contains simulation data for training and evaluation.
3. Rest of the folders are needed for running the PyTorch codes for training and evaluating CORONA.

### Contents of “DeepNet\_905”

There are 10 folders in “DeepNet\_905”, the most important folders are:

1. Res3dC: contains the codes for training ResNet, and saved models, pictures, logs, etc.

2. Unfolded: contains the codes for training Unfolded Network, and saved models, pictures, logs, etc.

3. Unfolded\_MSElayer: contains the codes for training Unfolded Network and tracking loss between the output of each layer and the ground truth, as well as saved models, pictures, logs, etc.

4. eval\_Res3dC: contains the codes for evaluate trained ResNet model on simulated data or in-vivo data.

5. eval\_Unfolded: contains the codes for evaluate trained Unfolded Net model on simulated data or in-vivo data.

Other folders are used for assisting the codes described above:

6. classes: class for plotting figures, converting data formats, etc.

7. network: the definition for different DNN, such as ResNet, Unfolded Net.

8. saved: all of the good models should be copied and saved here. We do this so that we only need to change the name of model in the evaluating scripts.

9. SimPlatform: the codes for simulating(without GUI). But because we have already got the simulated data in “Data” folder, this folder won’t be used.

10. tools: some functions for assisting evaluating performance, save movies as gifs, etc.

## “Data” Folder

There are 5 folders in “Data” folder:

1. for\_eval: pictures or gifs or mat files generated from evaluating performance will be saved here

(1)gifs\_Res3dC/gifs\_Unfolded: gifs for evaluating ResNet and Unfolded Net will be saved in respective folder.

(2)output\_patches: when we want to process one dataset(test1 or test2) using one model, the output patches will be saved here temporarily.

(3)resources: the input data for evaluating performance is saved here.

(4)results\_Res3dC/results\_Unfolded: the output mat files of the model when evaluated will be saved here.

2. Invivo: the in-vivo dataset(before switching rats)

3. Invivo\_converse: the switched in-vivo dataset

4. Sim: the simulated dataset

5. Temp: some temporary output files

# Basic Usage

## Train the network

### Training of ResNet

1. Please go to “Code/DeepNet\_905/Res3dC/main.py”(main\_cpu.py is a cpu version for training, we won’t use it)
2. Here is a screen shot of the code:

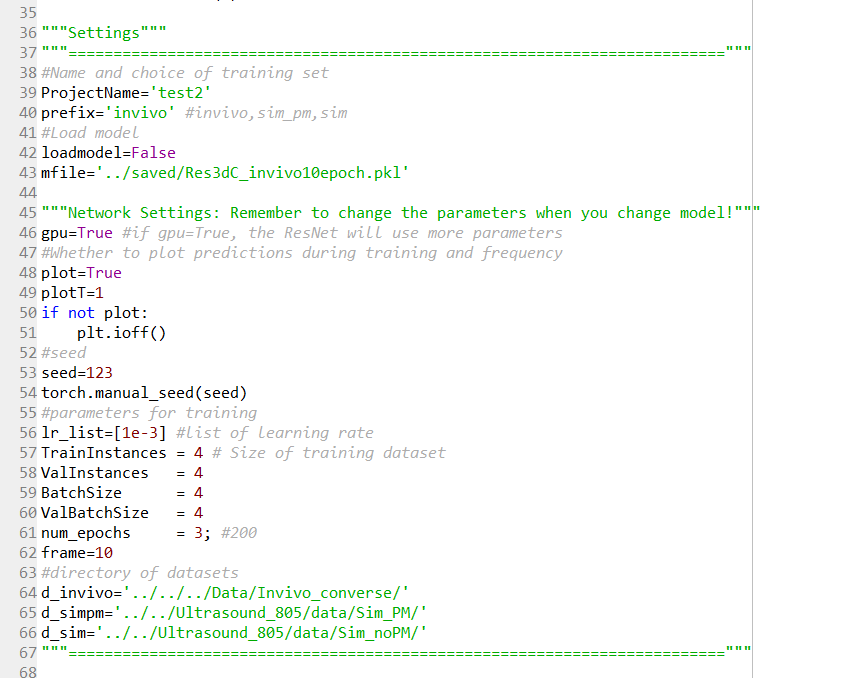


Figure 1

Only the “Settings” part shown above is needed to change in each training. The meaning of the parameters are as follows:

1. ProjectName: the parameter will be written in the names of output files, so that when we submit many tasks at the same time, the output files won’t overwrite.
2. prefix: we have “invivo/sim/sim\_pm” 3 options. The parameter will determine which dataset will be used for training. ‘invivo’ represents in-vivo data, ‘sim’ represents simulated data.(‘sim\_pm’ is another version of simulation, I didn’t use it in previous training, so it won’t be used later.)
3. loadmodel: decide whether to load a pre-trained model.(when training the network on in-vivo data, we need to load the model pre-trained on simulated data)
4. mfile: directory of model file, the model file should be pkl file or pth file.
5. gpu: the parameter won’t decide the location of calculation, it only represent different version of network parameters:

if gpu==True, the number of kernels and kernel size in each layer will be [16,8,8,8,8,8,1] and [5,3,3,3,3,3,3]

if gpu==False, the number of kernels and kernel size in each layer will be [16,4,4,4,4,4,1] and [3,3,3,3,3,3,3]

1. plot: whether to plot the results after several epochs. When train the network on GPU, please set the parameter to False.
2. plotT: if plot==True, then the parameter will decide we will plot performance after how many epochs.
3. seed: The manual seed, if the manual seed is fixed, we will be able to completely reproduce the training process.
4. lr\_list: the list of learning rate, usually we will only use one learning rate in one learning process(And I’m not sure if a bug will occur if we write more than one learning rate in the list because I didn’t check it)
5. TrainInstances, ValInstances, BatchSize, ValBatchSize, num\_epochs: they are easy to understand, so no explanations here
6. frame: when plotting the performance of model, which frame from the movie will be plotted.
7. d\_invivo, d\_sim, d\_simpm: the directory of in-vivo, simulated, another version of simulated(won’t be used) dataset.

p.s. The d\_invivo variable should be the directory of “Invivo\_converse” folder instead of “Invivo” folder, please be careful.

3. please set proper values for the variables above, then wait for running.

p.s. The command I use for submitting tasks to GPU is “screen -S [ProjectName]”, so that if any bugs occurred, we can still get the error information after detached.

4. please check the output files in “Res3dC/Results”:

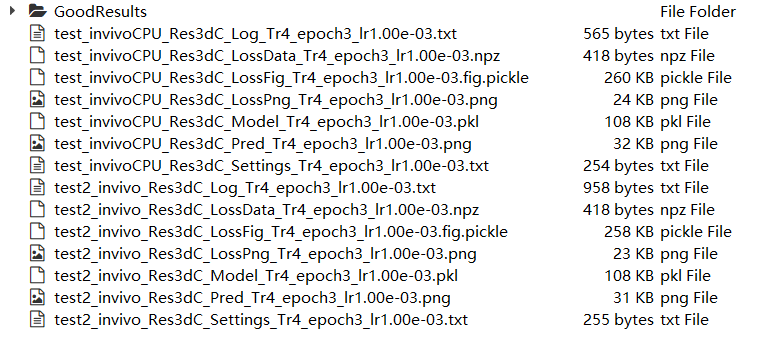


Figure 2

(1)The format of file name is as follows:

[ProjectName]\_[dataset name][CPU: if the file is generated by main\_cpu.py, there will be a ‘CPU’ word here]\_[network name]\_[file type]\_Tr[TrainInstances]\_epoch[num\_epochs]\_lr[learning rate]

(2)\*how to check the output files:

before saving the output files into “GoodResults”, please open the png file which has “Pred” in its name, it will show you how’s the performance in the last epoch, if it looks good, then there’s no problem, you can also check the png file which has “LossPng” in its name, and also the “Log” file.

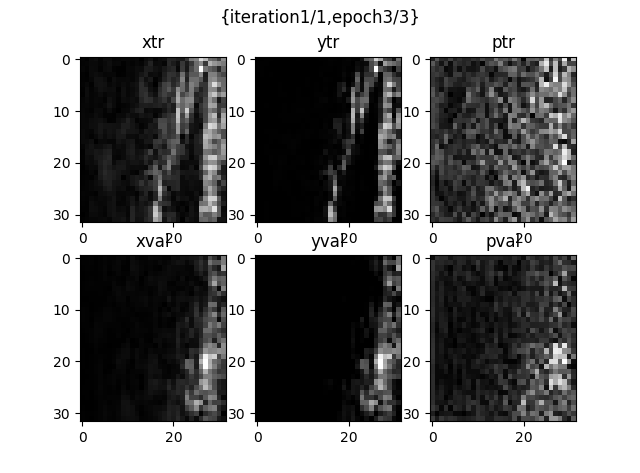


Figure 3: “Pred” file.

xtr: input of training data, ytr: ground truth of training, ptr: prediction of training

xval: input of validation data, yval: ground truth of validation, pval: prediction of validation

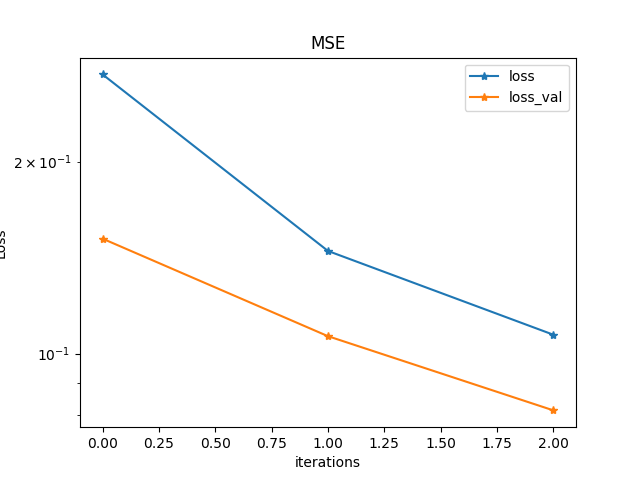


Figure 4: “LossPng” file.

For main\_cpu.py, the x label will be iterations, but for main.py it will be epochs

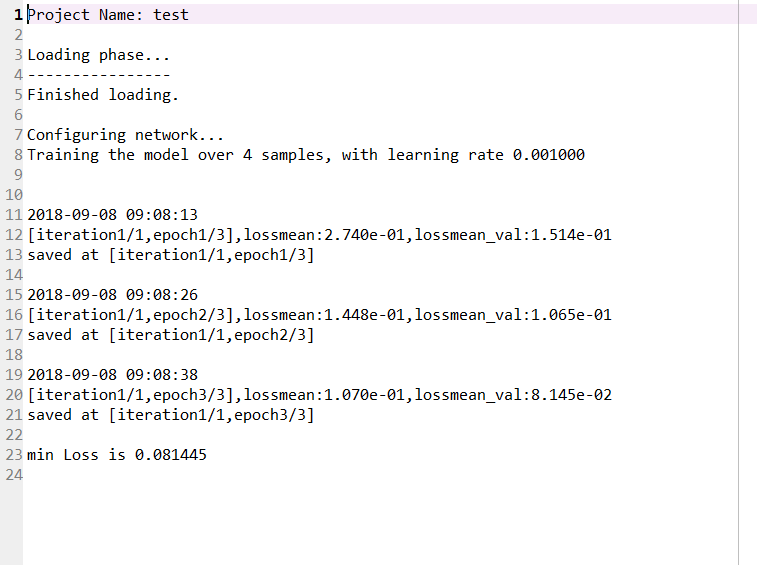


Figure 5: “Log” file

5. If all of the files above have no problems, please save them in GoodResults folder.

### Training of Unfolded Net

1. Please go to “Unfolded/main.py” script.
2. The parameters need to be changed are as follows:



Figure 6

Most of the parameters are the same as ResNet, I only explain the “Network Settings” here:

(1)”layers”: number of layers

(2)”kernel”: the list for kernel size of each layer. If kth tuple in the list is (a,b), then the kth convolutional block has the 3 dimensional complex kernel of size a\*a\*b. When b==1, the convolution will be 2 dimensional convolution.

(3)”coef\_L”: the threshold of L in each layer will be exp\_L[i]\*coef\_L\*S[0]. “exp\_L[i]” is a trainable parameter and its value is between 0 and 1. S[0] is the biggest singular value of L.

p.s. coef\_L=0.4 is a satisfying value, which lead to the performance I showed you.

(4)”coef\_S”: the threshold of S in each layer will be exp\_S[i]\*coef\_S\*mean[S]. “exp\_S[i]” is a trainable parameter and its value is between 0 and 1.

p.s. coef\_S=1.8 is a satisfying value, which lead to the performance I showed you.

(5)”CalInGPU”: whether to permit the network calculating in GPU. (because there are some cases when calculating in CPU is faster than GPU)

3. check output files: in addition to Log, Pred, LossPng, we should also check expLPng and expSPng files.

p.s. The usage of script in Unfolded\_MSElayer folder is similar.

## Evaluate the network

### Evaluation of ResNet

1. Please go to the folder “Code/DeepNet\_905/eval\_Res3dC/”
2. There are 4 scripts for evaluating ResNet:
3. eval\_invivo\_dataset.py: this script will process one dataset with specified model file and save all of the output patches into the specified directory.(The parameters are all easy to understand, so no explanations here)

p.s. The python version of PatchManipulator\_V2 is in “Code/DeepNet\_905/tools/combine.py”, please use it when you want to combine the output patches.

1. eval\_invivo\_patch: this script will process one patch from the whole movie.(The ResNet and Unfolded Net can process input data of arbitrary size, so you can set any size you want)



Figure 7

“arrange”: if arrange=[a,b,c,d,e,f], then the patch will be movie[a:b,c:d,e:f]

“data\_dir”: the directory of input whole movie(There are other movie files in the same folder as default directory)

“Sfile”: the directory of ground truth whole movie(There are other movie files in the same folder as default directory)

1. eval\_sim\_file.py: This script will process a simulated movie from the specified directory.(In fact, it’s not necessary to change the input data directory). The parameters are easy to understand.
2. eval\_sim\_gen.py: This script will generate a simulated movie, then process it with specified model file.(It can be used to study how parameters of simulated data influence the performance, but we don’t need to use if recently)

### Evaluation of Unfolded Net

1. Please go to the folder “Code/DeepNet\_905/eval\_Unfolded/”

2. There are 4 scripts for evaluating Unfolded Net:

(1)eval\_invivo\_dataset.py: It has the same function as in eval\_Res3dC.

p.s. But do remember to set the network parameters the same value as when they were trained.

(2) eval\_invivo\_patch.py: It has the same function as in eval\_Res3dC.

p.s. But do remember to set the network parameters the same value as when they were trained. And you also need to set the directory of ground truth for L.

(3) eval\_sim\_file.py: It has the same function as in eval\_Res3dC.

p.s. But do remember to set the network parameters the same value as when they were trained.

(4) eval\_sim\_gen.py: It has the same function as in eval\_Res3dC.

p.s. But do remember to set the network parameters the same value as when they were trained.

# Some Information about Training Time of Unfolded Net

Here are some results about the experiments I did on September 8.

## The training time on CPU and GPU

1. Experiment 1:

TrainInstances=24, ValInstances=8, BatchSize=4, ValBatchSize=4, num\_epochs=5

I take a typical time for training and validation of each epoch here(There is only one task running on HAL when I did the experiment)

|  |  |  |
| --- | --- | --- |
| on Hal’s CPU | | |
| Training | Validation | Total |
| 30s | 4s | 34s |
| on Hal’s GPU | | |
| Training | Validation | Total |
| 19.1s | 1.1s | 21s |
| on Hanqing’s computer’s CPU | | |
| Training | Validation | Total |
| 11.4s | 1.2s | 12s |

So it’s fastest if you train the Unfolded Net on CPU of computers in the lab.

1. Experiment 2:

To further verify that it’s almost 2 times faster to train Unfolded Net on CPU than on HAL’GPU. I did another experiment.

TrainInstances=240, ValInstances=80, BatchSize=4, ValBatchSize=4

|  |  |  |
| --- | --- | --- |
| on Hal’s GPU | | |
| Training | Validation | Total |
| 194s | 11s | 206s |
| on Hanqing’s computer’s CPU | | |
| Training | Validation | Total |
| 116s | 11.6s | 128s |

So it will be best if you have redundant CPU resources and can train the Unfolded Net on them.

## What if we run multiple tasks at the same time(on HAL’s GPU)

1. If we run 1 tasks at the same time, the time consumption in each epoch will be:

|  |  |  |
| --- | --- | --- |
| on Hal’s GPU | | |
| Training | Validation | Total |
| 19.1s | 1.1s | 21s |

1. If we run 2 tasks at the same time, the time consumption in each epoch will be:

|  |  |  |
| --- | --- | --- |
| on Hal’s GPU | | |
| Training | Validation | Total |
| 34.5s | 2.5s | 37s |

1. If we run 3 tasks at the same time, the time consumption in each epoch will be:

|  |  |  |
| --- | --- | --- |
| on Hal’s GPU | | |
| Training | Validation | Total |
| 53s | 4s | 57s |

At present, I can’t see too much disadvantages if we run multiple tasks at the same time, but I think it’s still not proposed to do such things.

# Which tasks do we need to submit

## Get pre-trained Network

The basic tasks are to get models trained on “simulated data+in-vivo data”.

1. Train all the network we are interested in on simulated data:

(So far, I have trained the ResNet and 10 layer-Unfolded Net and 8 layer-Unfolded Net on HAL’GPU, please check the output files and save them)

1. Train all the pre-trained models on in-vivo data:

(set “loadmodel” parameter to True and write the right directory of model file)

1. Evaluate the models we get on simulated and in-vivo data.

(At present, we only need to use eval\_invivo\_patch and eval\_sim\_file script, so we can get the movies I showed before. But it’s necessary to change the input file in eval\_invivo\_patch to a new file from rat1.)

# Other Instructions

## Notice

1. The ResNet and Unfolded Net can process data of arbitrary size(not only 32x32x20), so in the evaluating scripts, the data patch can be arbitrary size.

## Functions I didn’t implement

1. I wrote a python version of FISTA function, but it’s quite simple. So when we want to compare performance between Unfolded Net and FISTA, please use the old Matlab version.

## When meet errors or bad performance, please have a look here

1. If you got bad performance when you process a dataset with specified model, then you can try to use a different patch size to process the whole movie.(Because larger patches may get better performance because they contain more information in each patch)
2. When evaluating Unfolded Net, be careful about the network parameters. If you forget the parameters of model, please go to the folder which contain the output files, and the parameters can be seen in “Settings” txt file.