Demo
End-to-End
Deep Learning Pipeline

Enhao Gong

07-13-2017

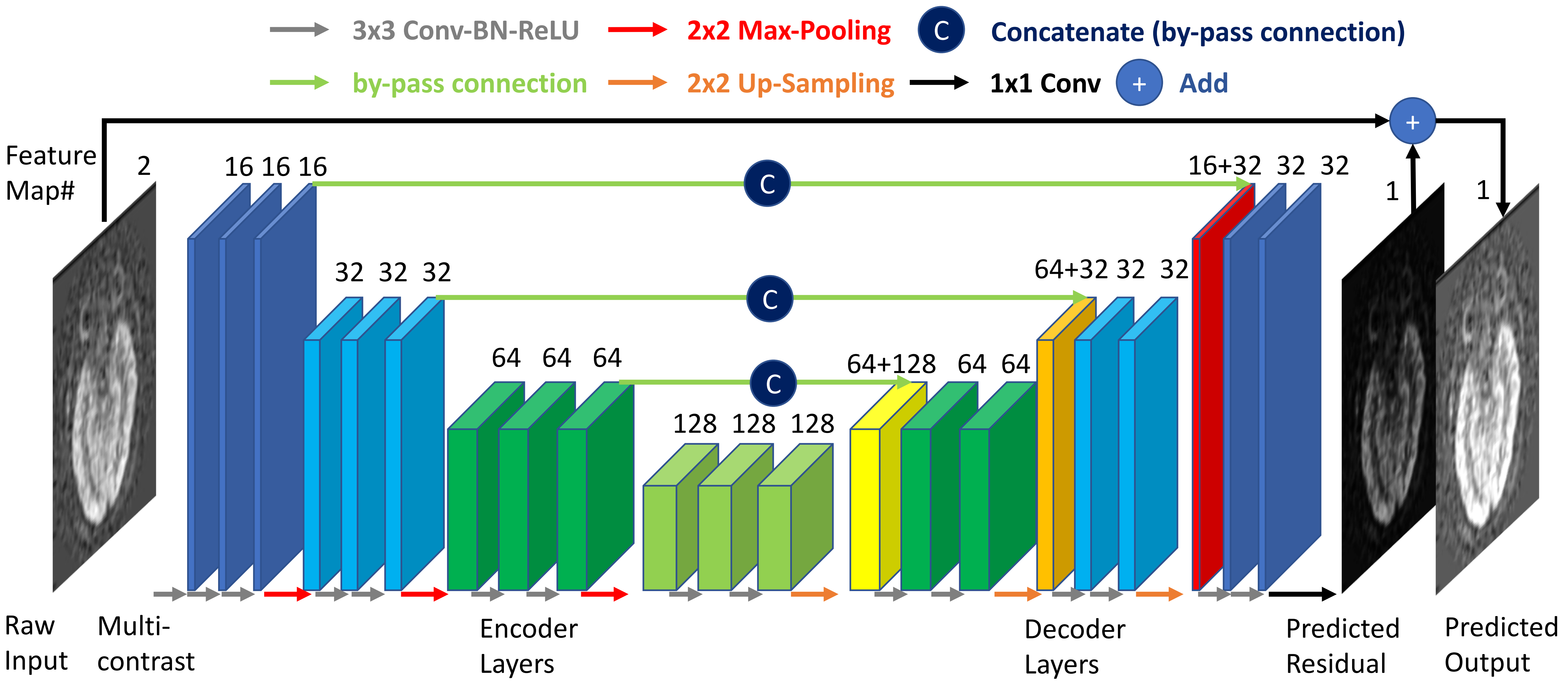
Frameworks and Repositary

- We are using Keras in this demo which uses TensorFlow as backend on Longo
- PyTorch is another framework that becomes more and more popular
- Repositaries
  - Super-Resolution with GAN using Keras: https://github.com/titu1994/Super-Resolution-using-Generative-Adversarial-Networks
  - Super-Resolution with GAN using TF: https://github.com/david-gpu/srez
  - Super-Resolution using py-torch: https://github.com/pytorch/examples/tree/master/super_resolution
  - Fast-Neural-Style using py-torch: https://github.com/jcjohnson/fast-neural-style https://github.com/bengxy/FastNeuralStyle
  - cycle-GAN from Junyan Zhu at Berkeley: https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix
  - ML/DL for MRI from Peng Cao at UCSF: https://github.com/peng-cao/mripy

Summary

- Copy the demo folder on Longo: **/data/enhaog/data_lowdose/**
  - Include dataset and script
  - Run from directory: **/data/enhaog/data_lowdose/scripts**
- Deep Learning framework: Keras + TF
- Input/Output:
  - Low-dose and Normal-dose PET
  - DICOM→NIfTi→ using ***dicom2nifti*** and ***nibabel***
    - https://github.com/icometrix/dicom2nifti
    - http://nipy.org/nibabel/nifti_images.html
  - Input and output volumes are normalized by norm and re-scaled
- Network structure
  - Encoder-Decoder with by-passes
  - Control number of poolings and convolution layers between poolings
  - Control input size, 8x augmentations and number of epochs

network structure



Code Structure

- Modules
  - Compute image metrics: Cafndl_metrics.py
  - Define network structures: Cafndl_network.py
  - Define fileio with nifti: Cafndl_fileio.py,
  - Utility functions (e.g. augmentations): Cafndl_utils.py
- Scripts
  - Script_demo_train.py:
    - train on 1 dataset 256x256 with 89 slices and x8 augmentations
    - Network weights 409729
    - Train on 640 samples, validate on 72 samples
    - 100 epochs w.r.t. L1 loss, taking 30min on Longo
    - Save optimal network weights to checkpoint file, export figure and json file for loss evolution
  - Script_demo_test.py:
    - test on 2 datasets, 256x256 each with 89 slices, no augmentations
    - predict on data size (178, 256, 256, 1) using time 0:00:02.464291
    - Export image comparison for testing slices and export error to json file

Specify for training

```
filename_checkpoint = '../ckpt/model_demo.ckpt’

filename_init = ””

list_dataset_train =
[
 {'input':'/data/enhaog/data_lowdose/GBM_Ex1496/DRF100_nifti/803_.nii.gz',
'gt':'/data/enhaog/data_lowdose/GBM_Ex1496/DRF001_nifti/800_.nii.gz’}
]
```

Specify for testing

```
filename_checkpoint = '../ckpt/model_demo.ckpt’

list_dataset_test =
[
{ 'input':'/data/enhaog/data_lowdose/GBM_Ex1496/DRF100_nifti/803_.nii.gz',
'gt':'/data/enhaog/data_lowdose/GBM_Ex1496/DRF001_nifti/800_.nii.gz'},
{ 'input':'/data/enhaog/data_lowdose/GBM_Ex842/DRF100_nifti/803_.nii.gz',
'gt':'/data/enhaog/data_lowdose/GBM_Ex842/DRF001_nifti/800_.nii.gz'}
]

filename_results = '../results/result_demo_0001'
```

Data format

- Conversion from DICOM to NIFTI
  - $ mkdir DRF100_nifti
  - $ dicom2nifti DRF100 DRF100_nifti

- Conversion from .mat to NIFTI
  - http://www.mathworks.com/matlabcentral/fileexchange/8797-tools-for-nifti-and-analyze-image
  - https://sites.google.com/site/kittipat/mvpa-for-brain-fmri/convert_matlab_nifti

Dependencies

```
import numpy as np
import os
import datetime
from cafndl_fileio import *
from cafndl_utils import *
from cafndl_network import *
from keras.callbacks import ModelCheckpoint
From keras.optimizers import Adam
```

Augmentation

```
'''augmentation'''
list_augments = []
num_augment_flipxy = 2
num_augment_flipx = 2
num_augment_flipy = 2
num_augment_shiftx = 1
num_augment_shifty = 1
```

Network setup

```
'''setup parameters'''
# related to model
num_poolings = 3
num_conv_per_pooling = 3
```

```
# related to training
lr_init = 0.001
num_epoch = 100
batch_size = 4
ratio_validation = 0.1
```

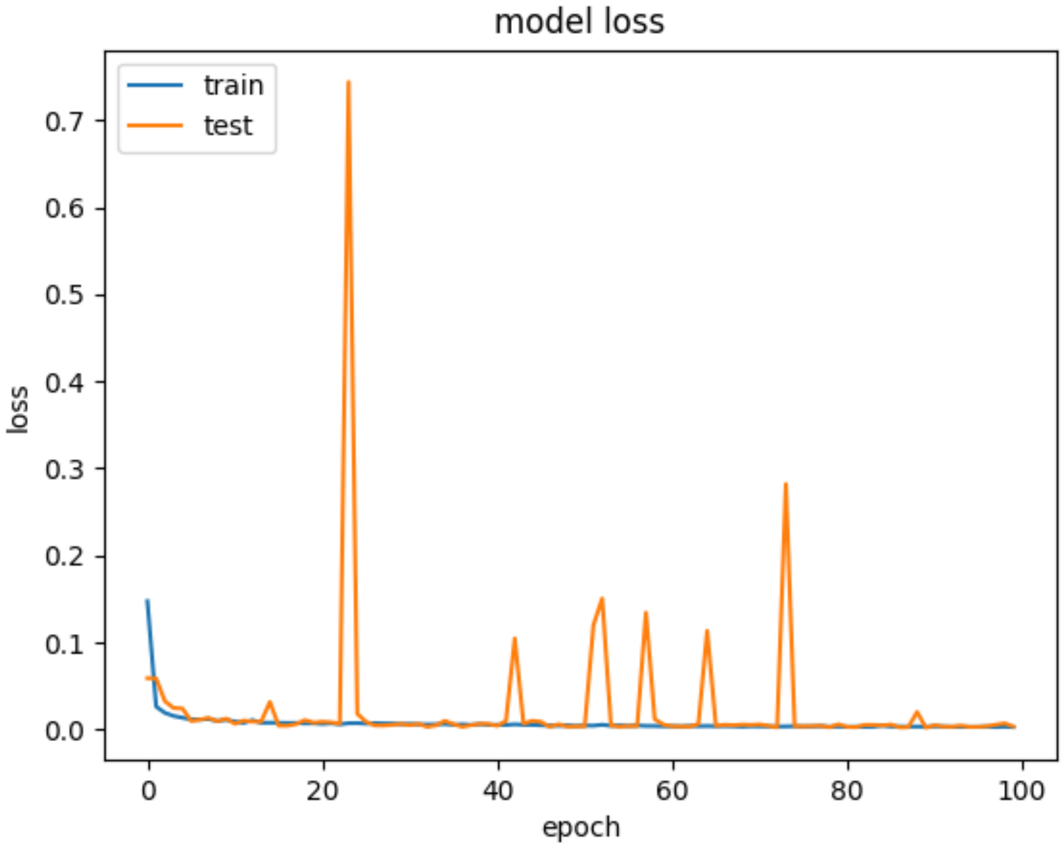
```
# default settings
num_channel_input = data_train_input.shape[-1]
num_channel_output = data_train_gt.shape[-1]
img_rows = data_train_input.shape[1]
img_cols = data_train_gt.shape[1]
# Be nice to not use all GPU resources
keras_memory = 0.4
keras_backend = 'tf'
with_batch_norm = True
```

1st: Just run with specified input and output NIFTI filepaths using default parameters

2nd: (if want to explore more) setup these parameters to change model complexity

3rd: (Optionally) Setup these parameters to change model training

Error Evolution



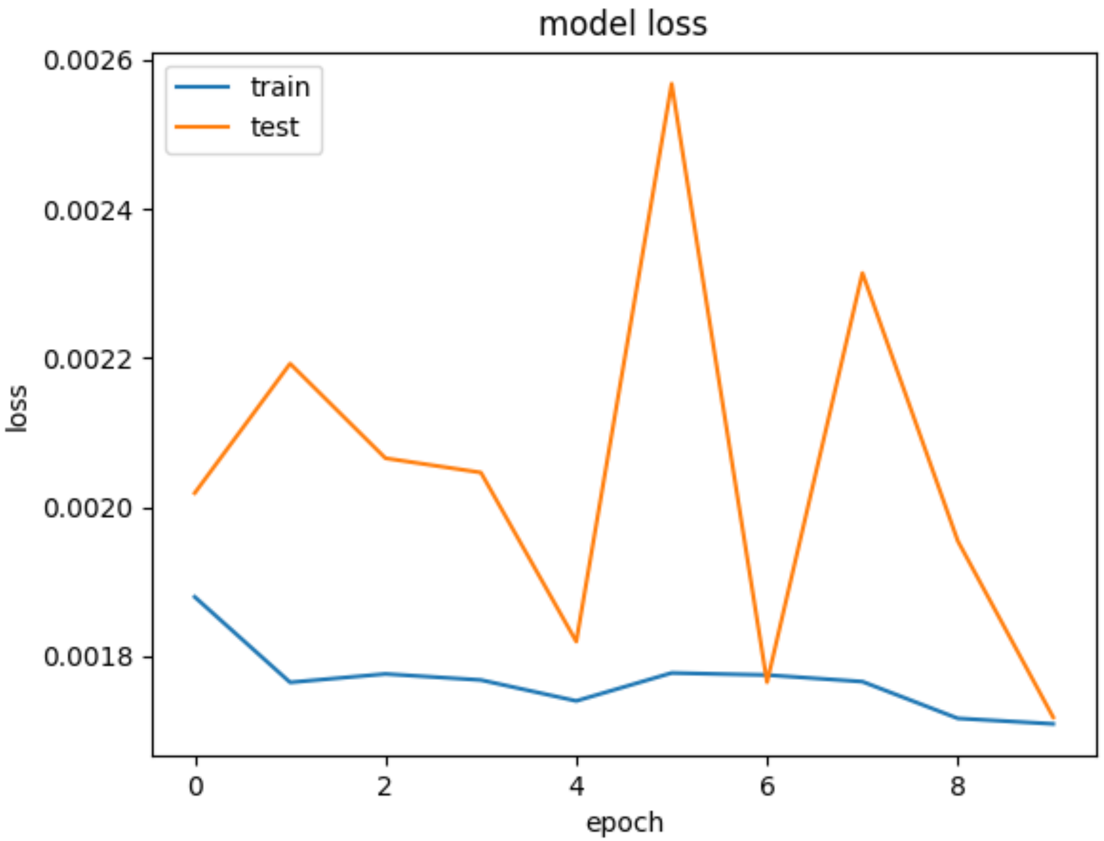
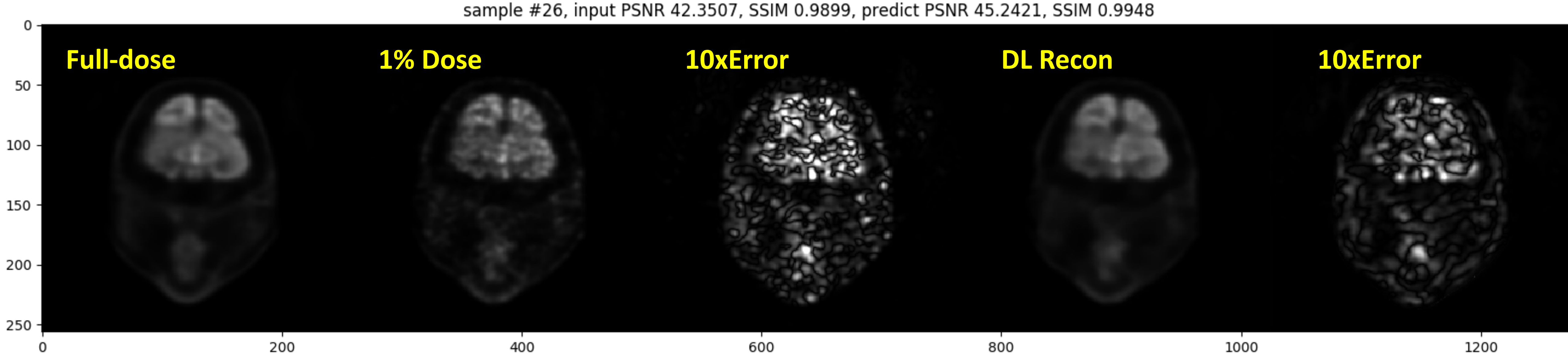
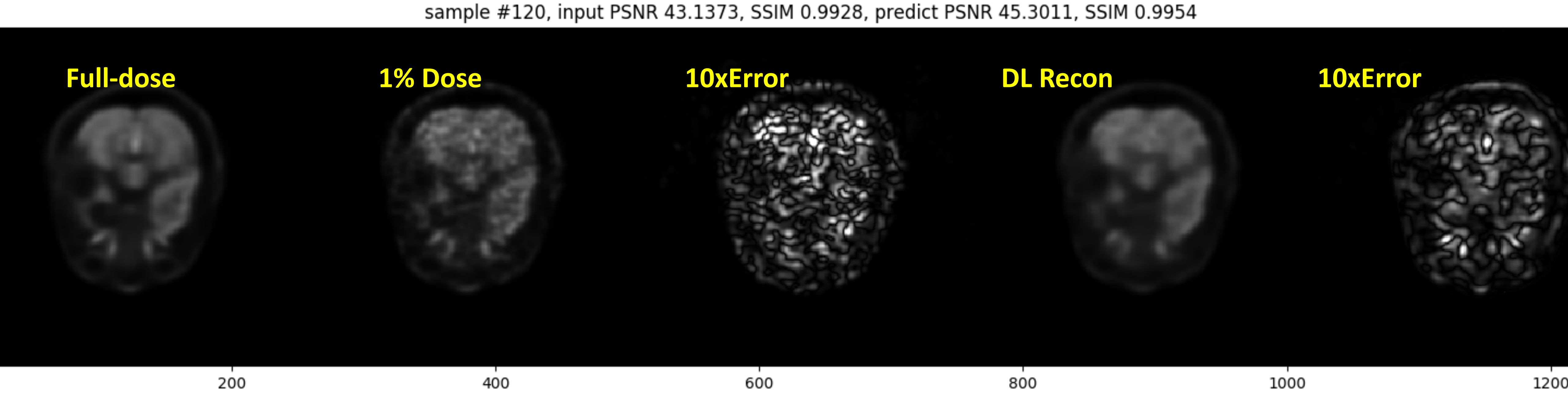
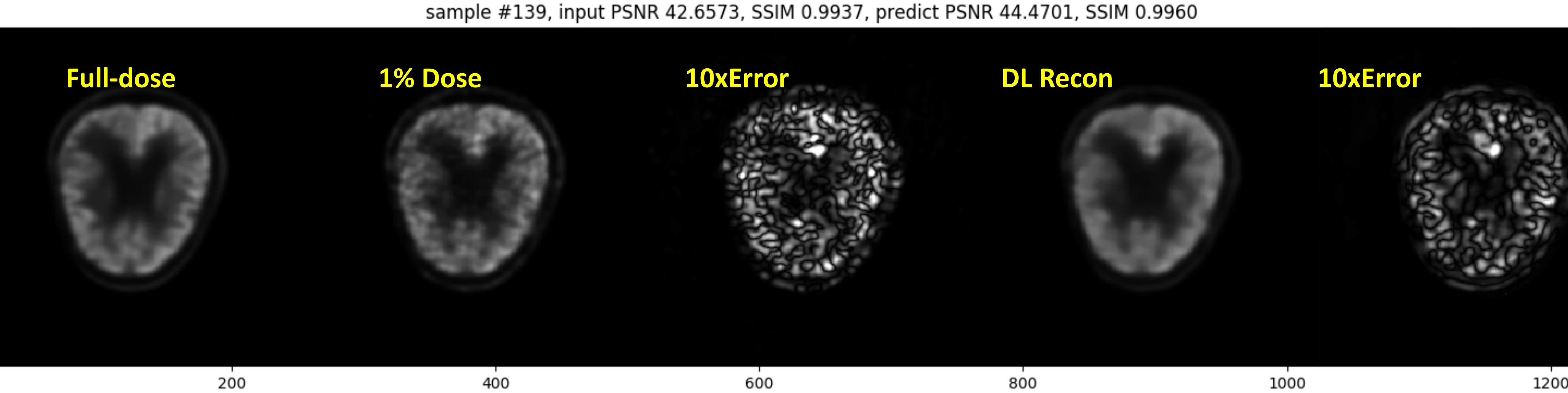


Image Comparison




sample #120, input PSNR 43.1373, SSIM 0.9928, predict PSNR 45.3011, SSIM 0.9954
Full-dose
1% Dose
10xError
DL Recon
10xError
200
400
600
800
1000
1200


sample #139, input PSNR 42.6573, SSIM 0.9937, predict PSNR 44.4701, SSIM 0.9960
Full-dose
1% Dose
10xError
DL Recon
10xError
200
400
600
800
1000
1200

Metrics: 3dB PSNR gain, 40% RMSE reduction

- err=json.load(open('../results/result_demo_0713_error.json','r'))
- >>> np.mean([x['psnr'] for x in err['err_input']])
- 49.504403889841711
- >>> np.mean([x['psnr'] for x in err['err_pred']])
- 52.427994401663852
- >>> np.mean([x['ssim'] for x in err['err_pred']])
- 0.99537973032124527
- >>> np.mean([x['ssim'] for x in err['err_input']])
- 0.99124708525122274
- >>> np.mean([x['rmse'] for x in err['err_input']])
- 0.81647482010468753
- >>> np.mean([x['rmse'] for x in err['err_pred']])
- 0.47915047153748297

	Raw Input	DL Output
PSNR	49.5dB	52.4dB
SSIM	0.9912	0.9954
RMSE	81.65%	47.92%

Next release

- Input with multi-contrast and NLM feature augmentation
- More CPU memory friendly Patch augmentation
- Consider masking for sampling and error metrics
- Better interfaces to set parameters from commend-line