Robust Equivariant Imaging on Colab

Bill Worstell

PicoRad -> MGH

4/9/2024

Chen, D., Tachella, J. and Davies, M.E., 2022. Robust equivariant imaging: a fully unsupervised framework for learning to image from noisy and partial measurements. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 5647-5656).

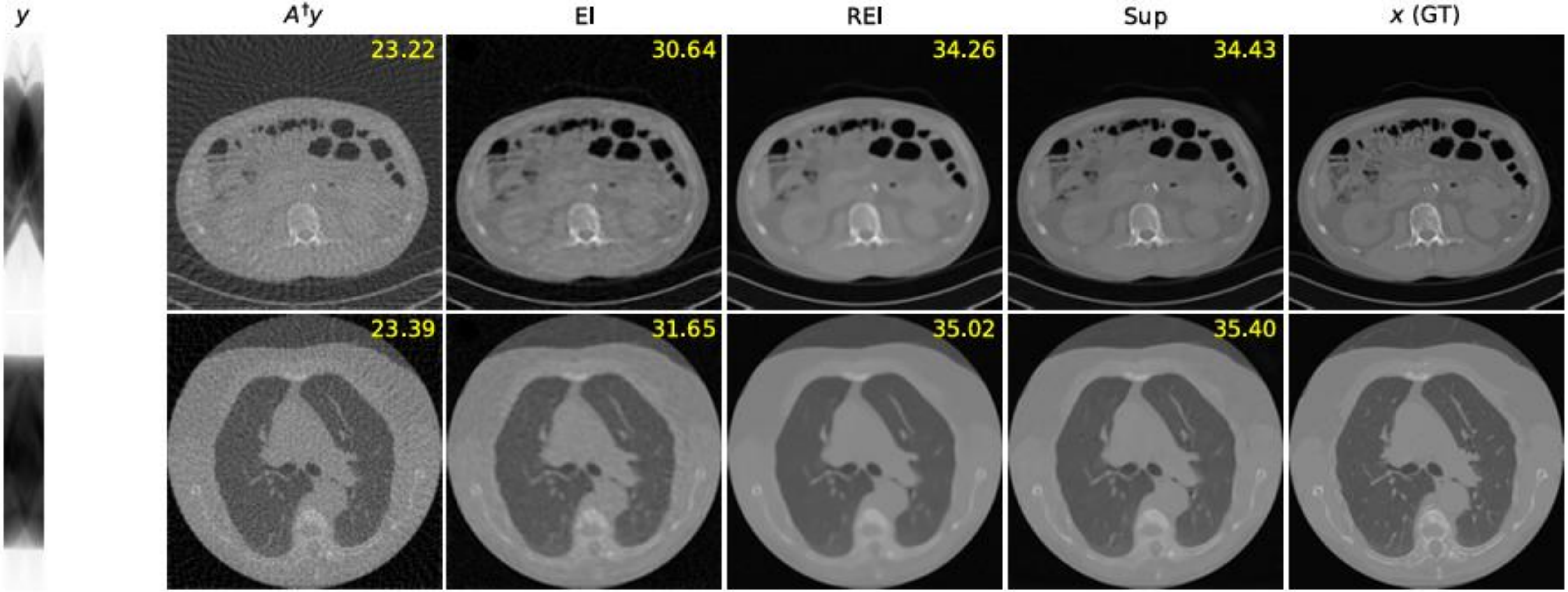
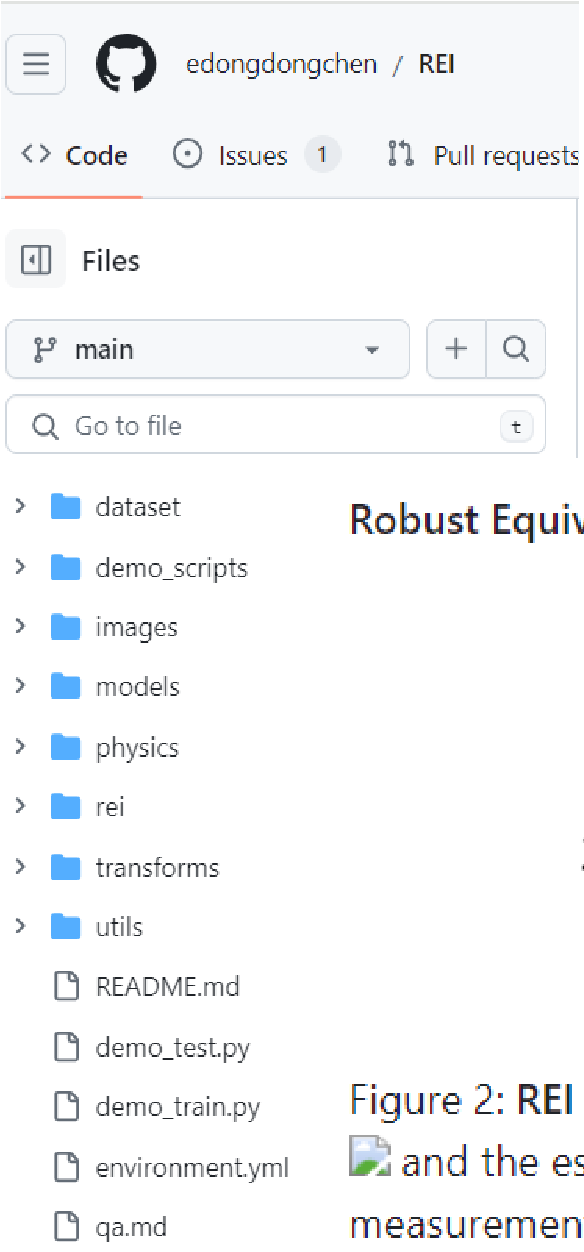


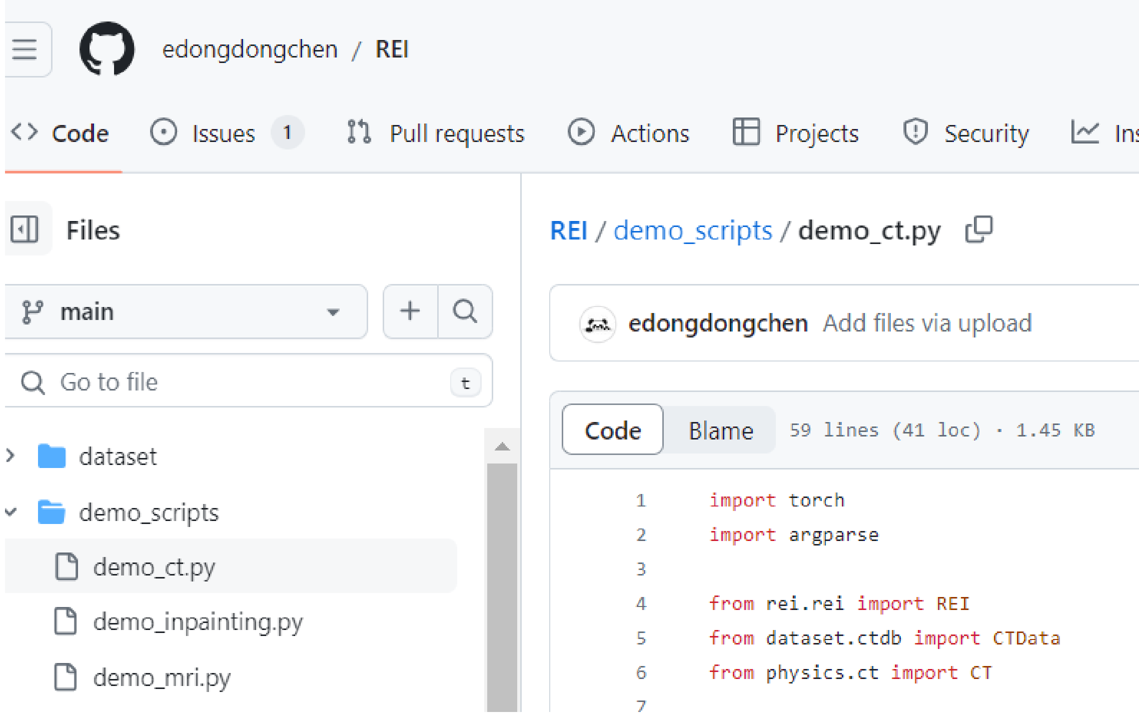
Figure 7. CT image reconstruction (50 views) on the test observations with mixed Poisson-Gaussian noise, $I_0 = 10^5$, $\sigma = 30$, $\gamma = 1$. PSNR values are shown in the top right corner of the images.

https://github.com/edongdongchen/REI



edongdongchen / REI

- dataset
- demo_scripts
- images
- models
- physics
- rei
- transforms
- utils
- README.md
- demo_test.py
- demo_train.py
- environment.yml
- qa.md



edongdongchen / REI

- dataset
- demo_scripts
  - demo_ct.py
  - demo_inpainting.py
  - demo_mri.py

REI / demo_scripts / demo_ct.py

edongdongchen Add files via upload

59 lines (41 loc) · 1.45 KB

```
import torch
import argparse

from rei.rei import REI
from dataset.ctdb import CTData
from physics.ct import CT

```

Robust Equivariant Imaging (REI)

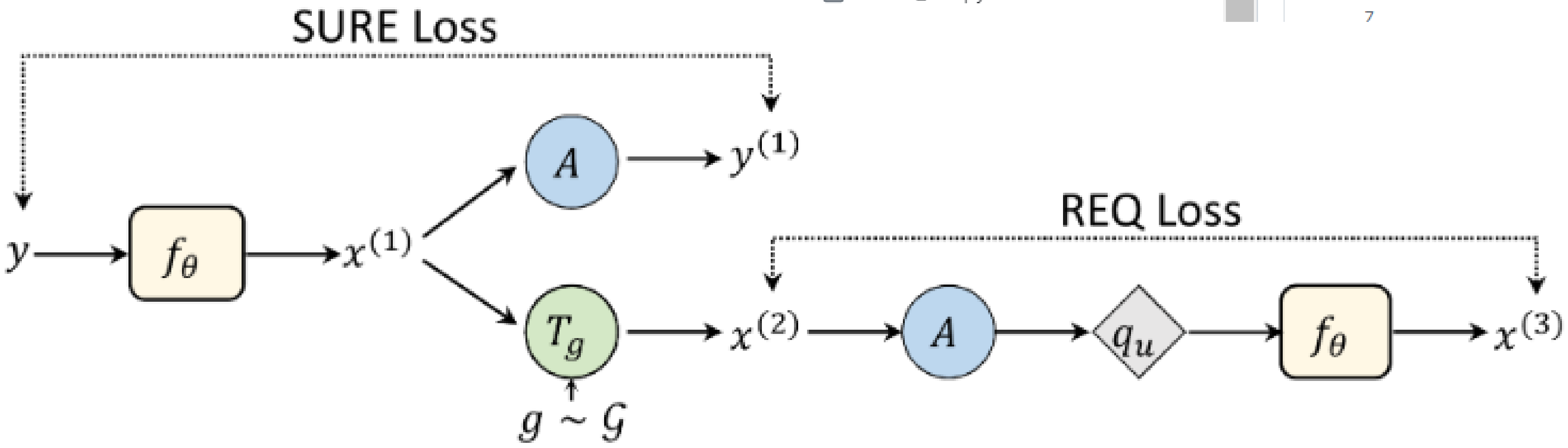


Figure 2: **REI training strategy.** represents the estimated image, is the transformation, while and represent and the estimate of from the (noisy) measurements respectively. The `SURE` loss aims to estimate the measurement consistency of clean measurement, `REQ` (robust equivariance) loss is the error (e.g. MSE) between and .

https://github.com/edongdongchen/REI/blob/main/demo_train.py

```
if args.task == 'ct':
    n_views = 50 # number of views
    tau = 10 # SURE

    epochs = 3000
    ckp_interval = 100
    schedule = [1000, 2000]

    batch_size = 2
    lr = {'G': 5e-4, 'WD': 1e-8}
    alpha = {'req': 1e3, 'sure': 1e-5}

    # define a MPG noise model
    I0 = 1e5
    noise_sigam = 30
    noise_model = {'noise_type': 'mpg', # mixed poisson-gaussian
                   'sigma': noise_sigam,
                   'gamma': 1}

    dataloader = torch.utils.data.DataLoader(
        dataset=CTData(mode='train'), batch_size=batch_size, shuffle=True)


    transform = Rotate(n_trans=2, random_rotate=True)

    physics = CT(256, n_views, circle=False, device=device, I0=I0,
                 noise_model=noise_model)

    rei = REI(in_channels=1, out_channels=1, img_width=256, img_height=256,
              dtype=torch.float, device=device)
```

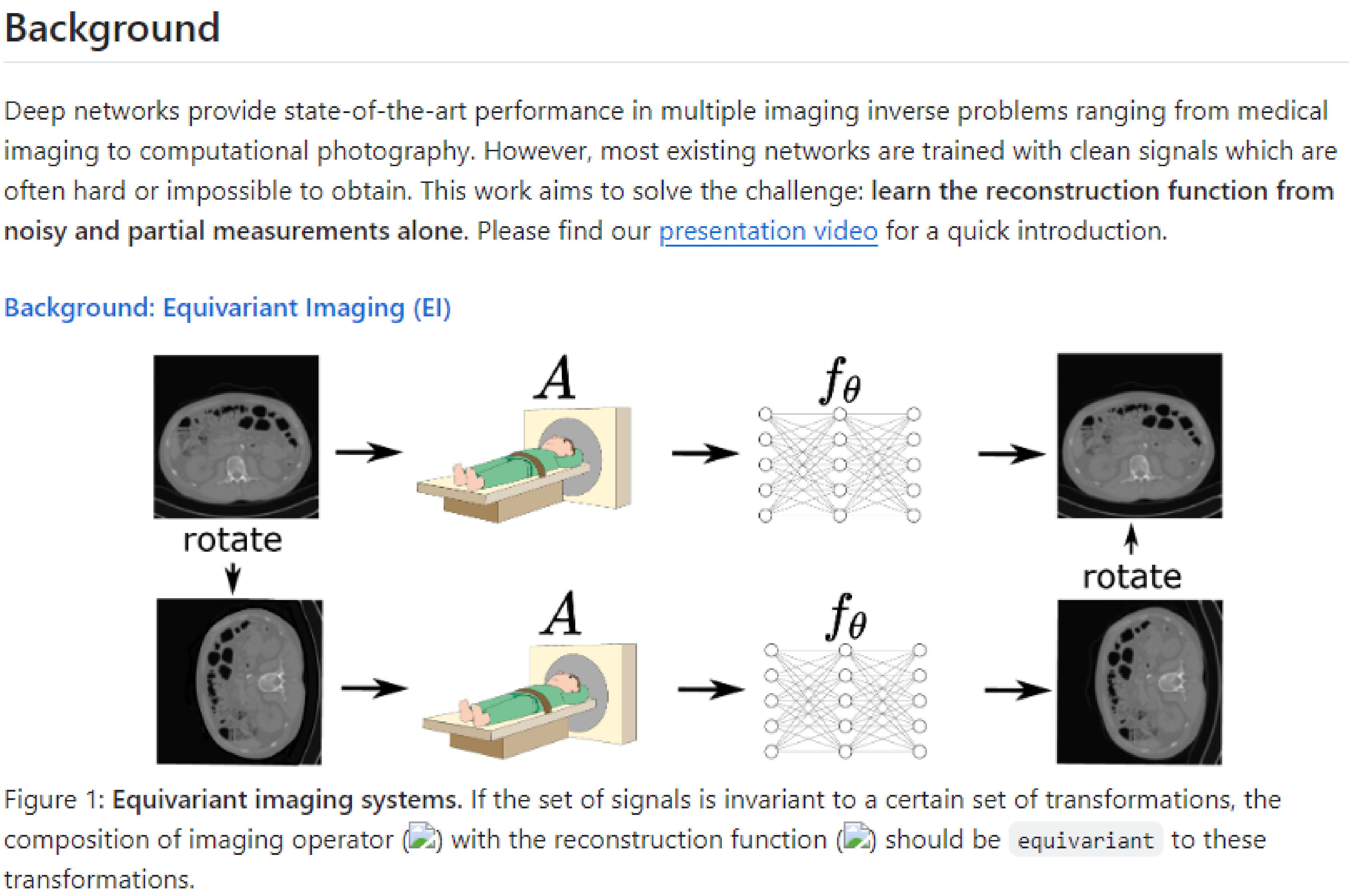
https://github.com/edongdongchen/REI/blob/main/demo_scripts/demo_ct.py

```
import argparse

from rei.rei import REI
from dataset.ctdb import CTData
from physics.ct import CT

from transforms.rotate import Rotate


parser =
argparse.ArgumentParser(description='REI')
# inverse problem configs:
parser.add_argument('--task', default='ct',
type=str,
                    help="inverse problems=['ct',
'inpainting', 'mri'] (default: 'ct')")

def main():
    args = parser.parse_args()

    device='cuda:1'

    pretrained = None
    lr_cos = False
    save_ckp = True
    report_psnr = True

    n_views = 50
    tau = 10

    epochs = 3000
    ckp_interval = 100
    schedule = [1000, 2000]

    batch_size = 2
    lr = {'G': 5e-4, 'WD': 1e-8}
    alpha = {'req': 1e3, 'sure': 1e-5}

    I0 = 1e5
    noise_sigam = 30
    noise_model = {'noise_type': 'mpg',
                   'sigma': noise_sigam,
                   'gamma': 1}

    dataloader = torch.utils.data.DataLoader(
        dataset=CTData(mode='train'),
batch_size=batch_size, shuffle=True)


    transform = Rotate(n_trans=2,
random_rotate=True)

    physics = CT(256, n_views, circle=False,
device=device, I0=I0,
                 noise_model=noise_model)

    rei = REI(in_channels=1, out_channels=1,
img_width=256, img_height=256,
              dtype=torch.float, device=device)

    rei.train_rei(dataloader, physics, transform,
epochs, lr, alpha, ckp_interval,
                  schedule, pretrained, lr_cos, save_ckp,
tau, report_psnr, args)

if __name__ == '__main__':
    main()
```

https://github.com/edongdongchen/REI

Fork to:

https://github.com/BillWorstell/REI



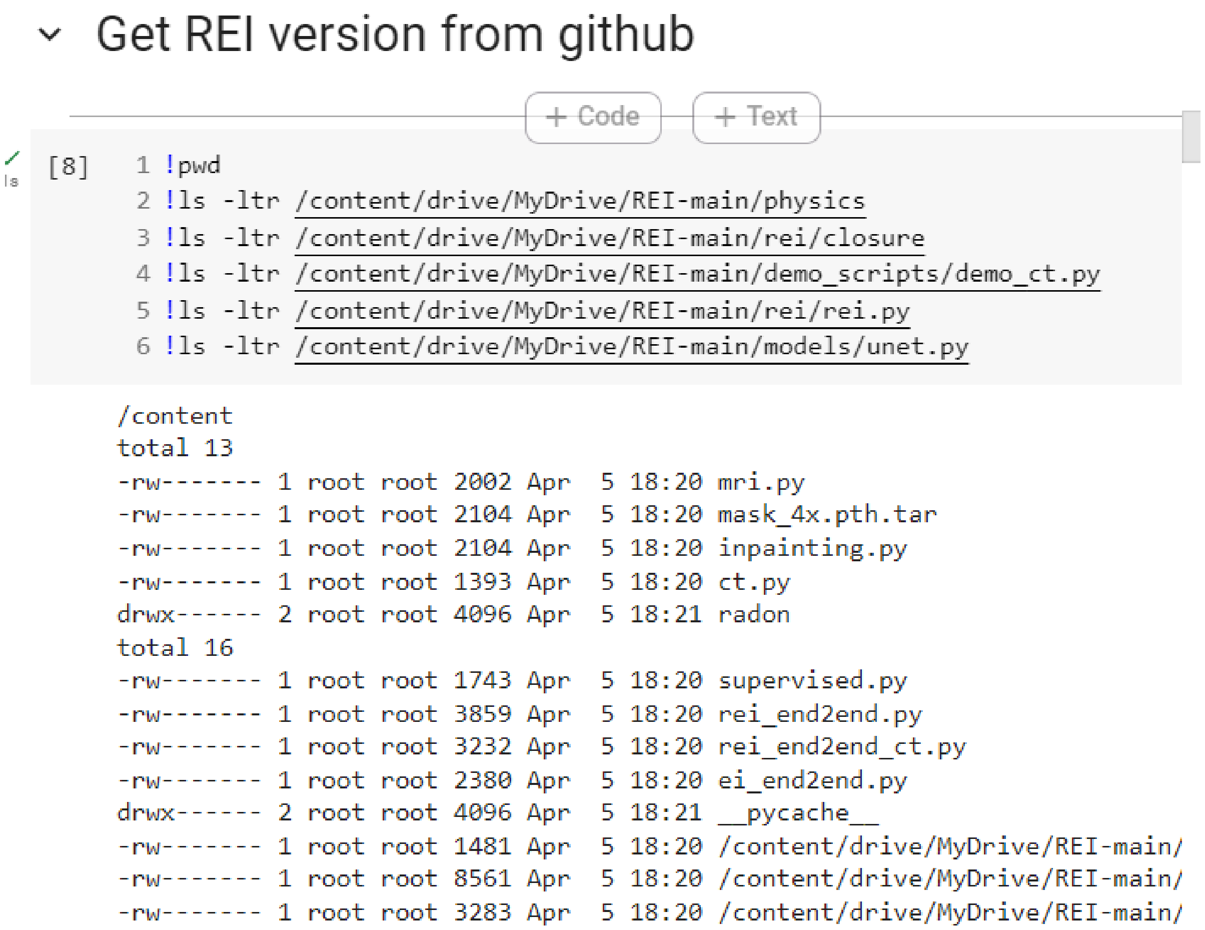
Background

Deep networks provide state-of-the-art performance in multiple imaging inverse problems ranging from medical imaging to computational photography. However, most existing networks are trained with clean signals which are often hard or impossible to obtain. This work aims to solve the challenge: **learn the reconstruction function from noisy and partial measurements alone**. Please find our presentation video for a quick introduction.

Background: Equivariant Imaging (EI)

Figure 1: **Equivariant imaging systems.** If the set of signals is invariant to a certain set of transformations, the composition of imaging operator () with the reconstruction function () should be `equivariant` to these transformations.

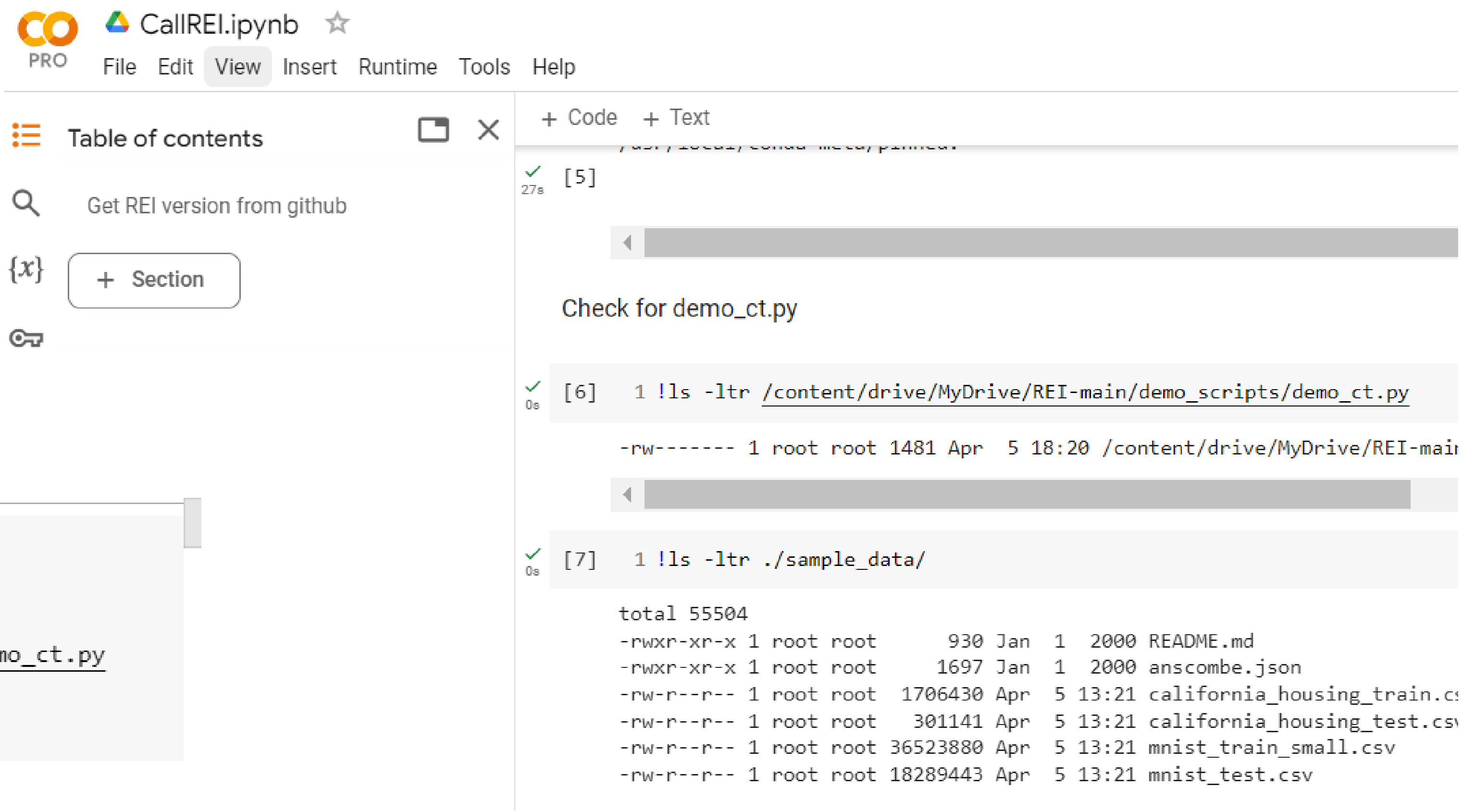
https://colab.research.google.com/drive/1yXWCPSHR2IRB0fjluz-l0xsqwa7u0wi1#scrollTo=nAiP9qnZdH3G&uniqifier=1



Get REI version from github

```
[8]  !pwd
     !ls -ltr /content/drive/MyDrive/REI-main/physics
     !ls -ltr /content/drive/MyDrive/REI-main/rei/closure
     !ls -ltr /content/drive/MyDrive/REI-main/demo_scripts/demo_ct.py
     !ls -ltr /content/drive/MyDrive/REI-main/rei/rei.py
     !ls -ltr /content/drive/MyDrive/REI-main/models/unet.py
```

```
/content
total 13
-rw------- 1 root root 2002 Apr  5 18:20 mri.py
-rw------- 1 root root 2104 Apr  5 18:20 mask_4x.pth.tar
-rw------- 1 root root 2104 Apr  5 18:20 inpainting.py
-rw------- 1 root root 1393 Apr  5 18:20 ct.py
drwx------ 2 root root 4096 Apr  5 18:21 radon
total 16
-rw------- 1 root root 1743 Apr  5 18:20 supervised.py
-rw------- 1 root root 3859 Apr  5 18:20 rei_end2end.py
-rw------- 1 root root 3232 Apr  5 18:20 rei_end2end_ct.py
-rw------- 1 root root 2380 Apr  5 18:20 ei_end2end.py
drwx------ 2 root root 4096 Apr  5 18:21 __pycache__
-rw------- 1 root root 1481 Apr  5 18:20 /content/drive/MyDrive/REI-main/
-rw------- 1 root root 8561 Apr  5 18:20 /content/drive/MyDrive/REI-main/
-rw------- 1 root root 3283 Apr  5 18:20 /content/drive/MyDrive/REI-main/
```



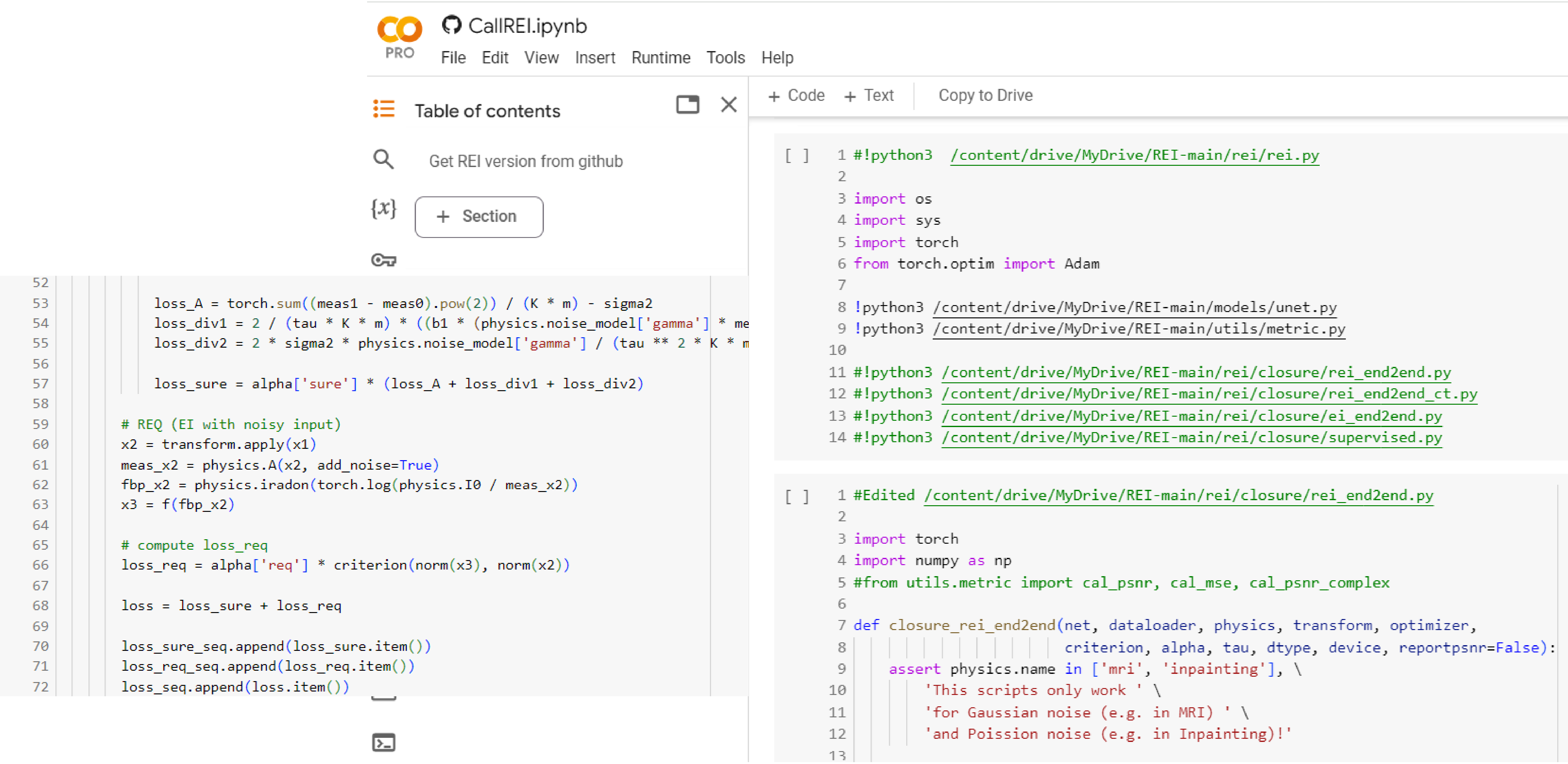
```
[5]
```

Check for demo_ct.py

```
[6]  !ls -ltr /content/drive/MyDrive/REI-main/demo_scripts/demo_ct.py

-rw------- 1 root root 1481 Apr  5 18:20 /content/drive/MyDrive/REI-mai
```

```
[7]  !ls -ltr ./sample_data/

total 55504
-rwxr-xr-x 1 root root      930 Jan  1  2000 README.md
-rwxr-xr-x 1 root root     1697 Jan  1  2000 anscombe.json
-rw-r--r-- 1 root root  1706430 Apr  5 13:21 california_housing_train.cs
-rw-r--r-- 1 root root   301141 Apr  5 13:21 california_housing_test.csv
-rw-r--r-- 1 root root 36523880 Apr  5 13:21 mnist_train_small.csv
-rw-r--r-- 1 root root 18289443 Apr  5 13:21 mnist_test.csv
```

https://colab.research.google.com/github/BillWorstell/REI/blob/main/CallREI.ipynb

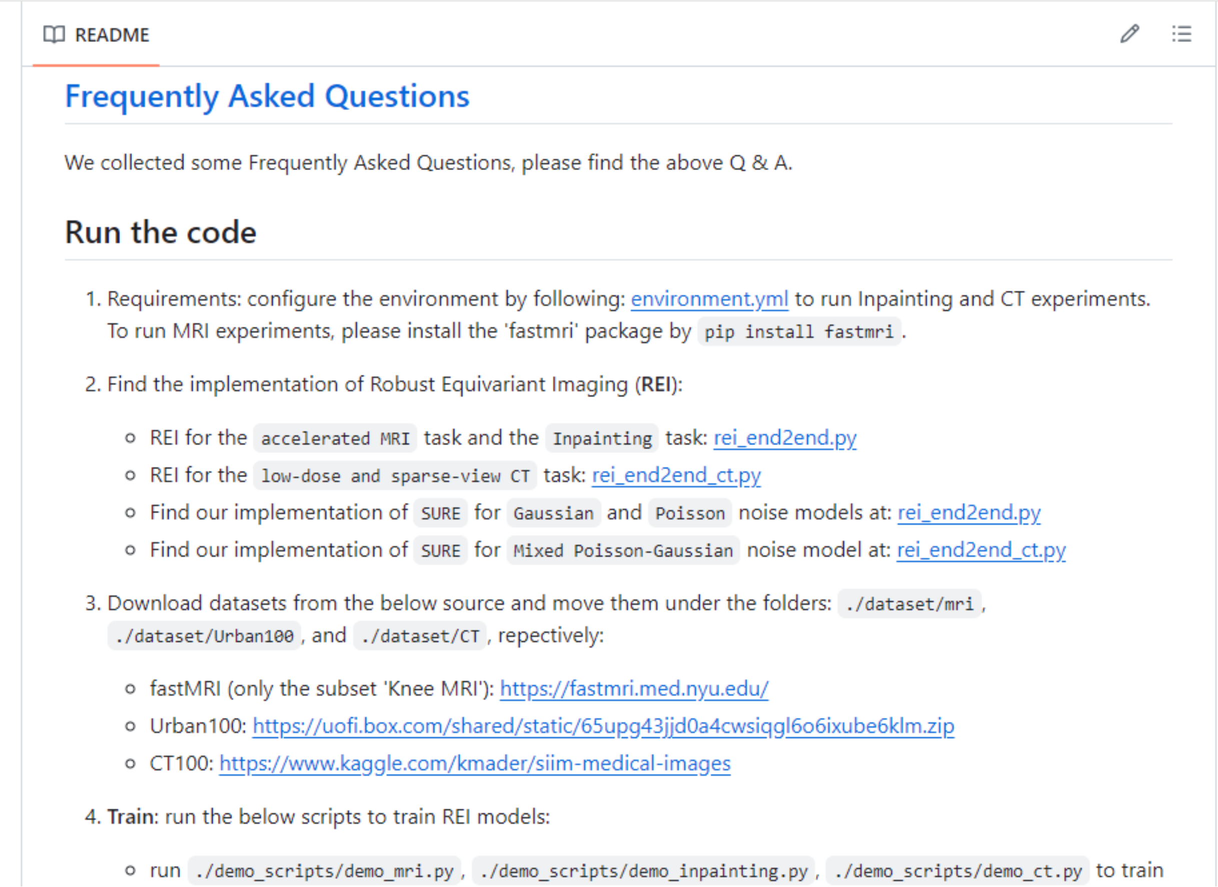

PRO
CallREI.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text Copy to Drive
Table of contents
Get REI version from github
+ Section
[ ]
[ ]

```
            loss_A = torch.sum((meas1 - meas0).pow(2)) / (K * m) - sigma2
            loss_div1 = 2 / (tau * K * m) * ((b1 * (physics.noise_model['gamma'] * me
            loss_div2 = 2 * sigma2 * physics.noise_model['gamma'] / (tau ** 2 * K * m

            loss_sure = alpha['sure'] * (loss_A + loss_div1 + loss_div2)

        # REQ (EI with noisy input)
        x2 = transform.apply(x1)
        meas_x2 = physics.A(x2, add_noise=True)
        fbp_x2 = physics.iradon(torch.log(physics.I0 / meas_x2))
        x3 = f(fbp_x2)

        # compute loss_req
        loss_req = alpha['req'] * criterion(norm(x3), norm(x2))

        loss = loss_sure + loss_req

        loss_sure_seq.append(loss_sure.item())
        loss_req_seq.append(loss_req.item())
        loss_seq.append(loss.item())
```

```
#!python3  /content/drive/MyDrive/REI-main/rei/rei.py

import os
import sys
import torch
from torch.optim import Adam

!python3 /content/drive/MyDrive/REI-main/models/unet.py
!python3 /content/drive/MyDrive/REI-main/utils/metric.py

#!python3 /content/drive/MyDrive/REI-main/rei/closure/rei_end2end.py
#!python3 /content/drive/MyDrive/REI-main/rei/closure/rei_end2end_ct.py
#!python3 /content/drive/MyDrive/REI-main/rei/closure/ei_end2end.py
#!python3 /content/drive/MyDrive/REI-main/rei/closure/supervised.py
```

```
#Edited /content/drive/MyDrive/REI-main/rei/closure/rei_end2end.py

import torch
import numpy as np
#from utils.metric import cal_psnr, cal_mse, cal_psnr_complex

def closure_rei_end2end(net, dataloader, physics, transform, optimizer,
                        criterion, alpha, tau, dtype, device, reportpsnr=False):
    assert physics.name in ['mri', 'inpainting'], \
        'This scripts only work ' \
        'for Gaussian noise (e.g. in MRI) ' \
        'and Poission noise (e.g. in Inpainting)!'
```

https://github.com/edongdongchen/REI



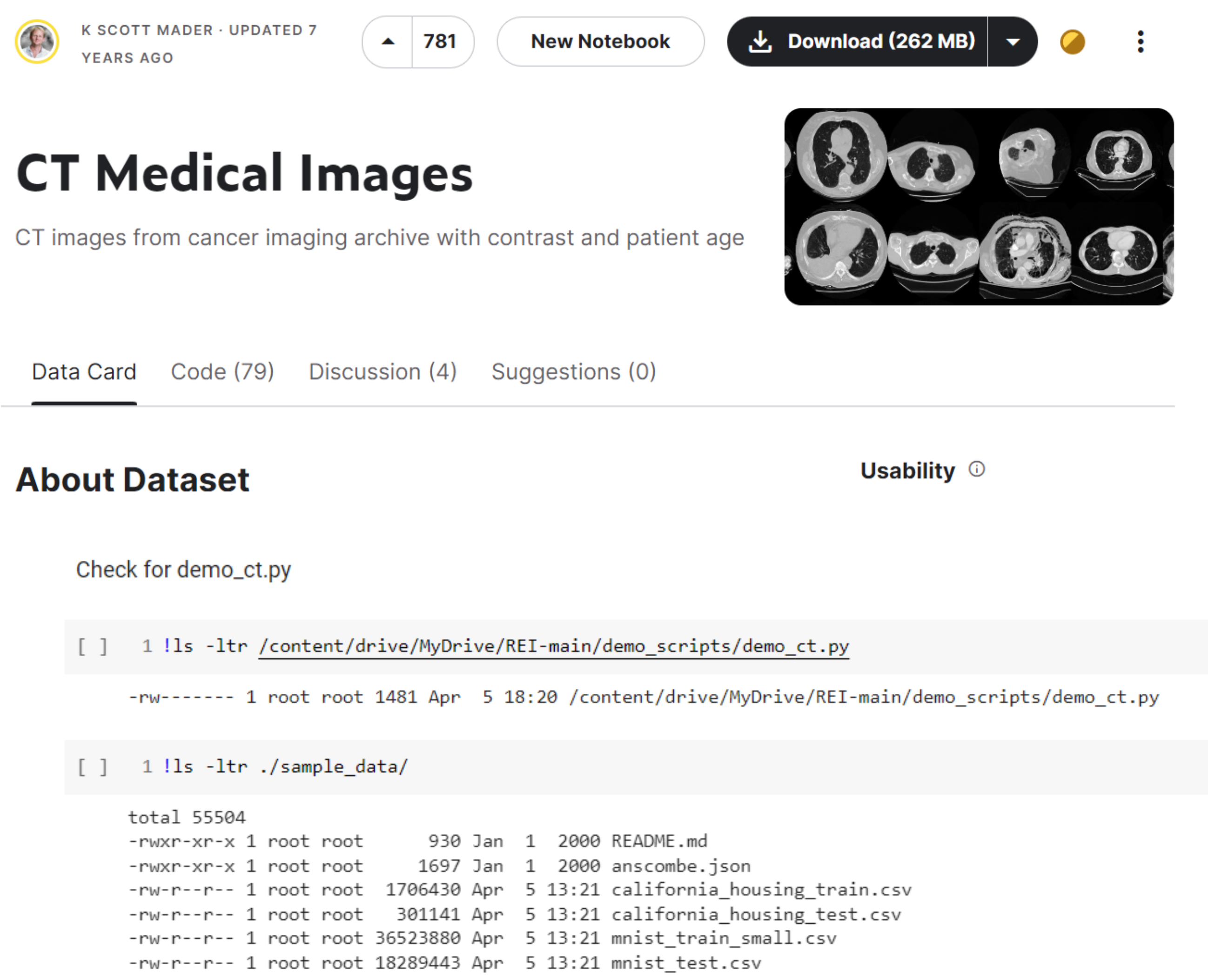
Frequently Asked Questions

We collected some Frequently Asked Questions, please find the above Q & A.

Run the code

1. Requirements: configure the environment by following: environment.yml to run Inpainting and CT experiments. To run MRI experiments, please install the 'fastmri' package by `pip install fastmri`.
2. Find the implementation of Robust Equivariant Imaging (**REI**):
   - REI for the `accelerated MRI` task and the `Inpainting` task: rei_end2end.py
   - REI for the `low-dose and sparse-view CT` task: rei_end2end_ct.py
   - Find our implementation of `SURE` for `Gaussian` and `Poisson` noise models at: rei_end2end.py
   - Find our implementation of `SURE` for `Mixed Poisson-Gaussian` noise model at: rei_end2end_ct.py
3. Download datasets from the below source and move them under the folders: `./dataset/mri`, `./dataset/Urban100`, and `./dataset/CT`, repectively:
   - fastMRI (only the subset 'Knee MRI'): https://fastmri.med.nyu.edu/
   - Urban100: https://uofi.box.com/shared/static/65upg43jjd0a4cwsiqgl6o6ixube6klm.zip
   - CT100: https://www.kaggle.com/kmader/siim-medical-images
4. **Train**: run the below scripts to train REI models:
   - run `./demo_scripts/demo_mri.py`, `./demo_scripts/demo_inpainting.py`, `./demo_scripts/demo_ct.py` to train

https://www.kaggle.com/datasets/kmader/siim-medical-images?resource=download



CT Medical Images

CT images from cancer imaging archive with contrast and patient age

About Dataset

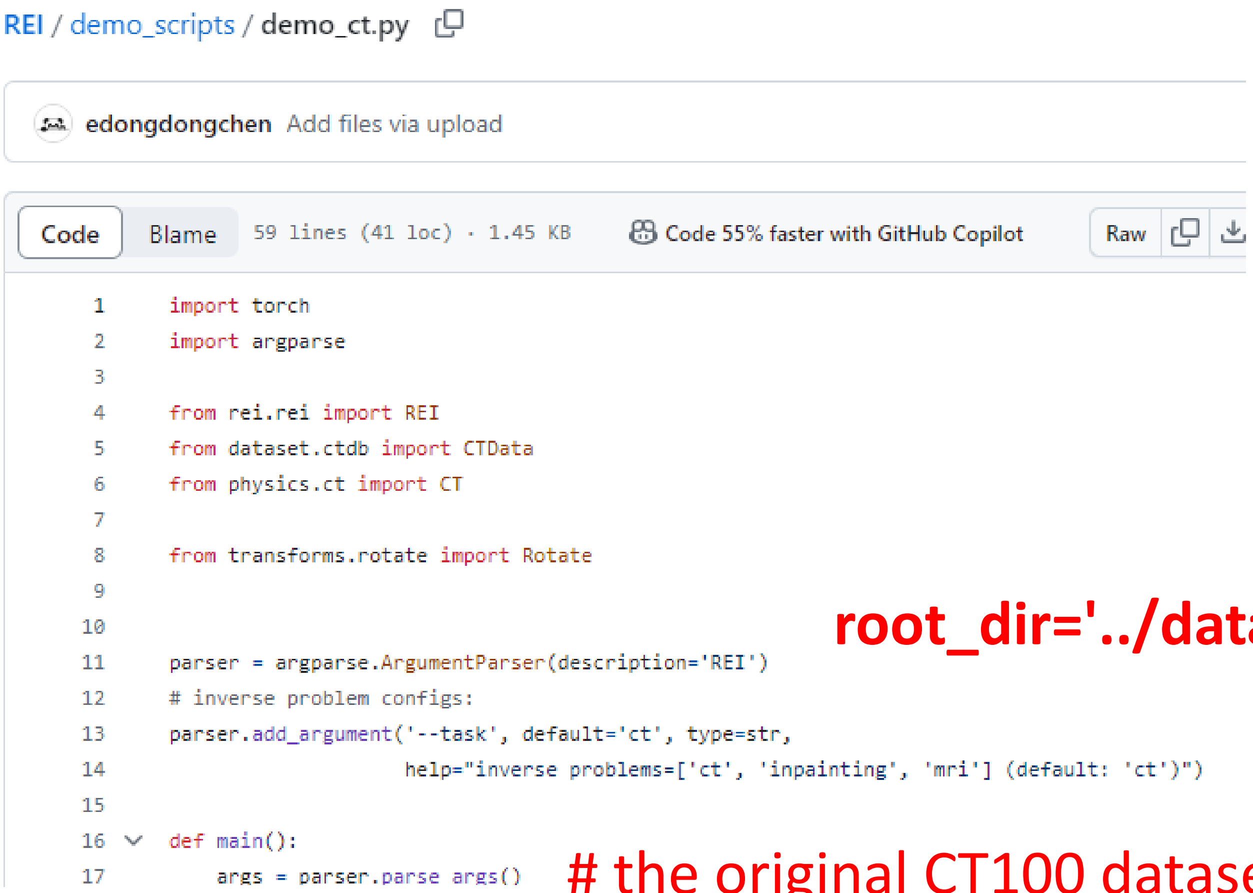
Check for demo_ct.py

```
[ ]  !ls -ltr /content/drive/MyDrive/REI-main/demo_scripts/demo_ct.py

-rw------- 1 root root 1481 Apr  5 18:20 /content/drive/MyDrive/REI-main/demo_scripts/demo_ct.py
```

```
[ ]  !ls -ltr ./sample_data/

total 55504
-rwxr-xr-x 1 root root      930 Jan  1  2000 README.md
-rwxr-xr-x 1 root root     1697 Jan  1  2000 anscombe.json
-rw-r--r-- 1 root root  1706430 Apr  5 13:21 california_housing_train.csv
-rw-r--r-- 1 root root   301141 Apr  5 13:21 california_housing_test.csv
-rw-r--r-- 1 root root 36523880 Apr  5 13:21 mnist_train_small.csv
-rw-r--r-- 1 root root 18289443 Apr  5 13:21 mnist_test.csv
```

https://colab.research.google.com/github/BillWorstell/REI/blob/main/CallREI.ipynb https://github.com/edongdongchen/REI/blob/main/demo_scripts/demo_ct.py

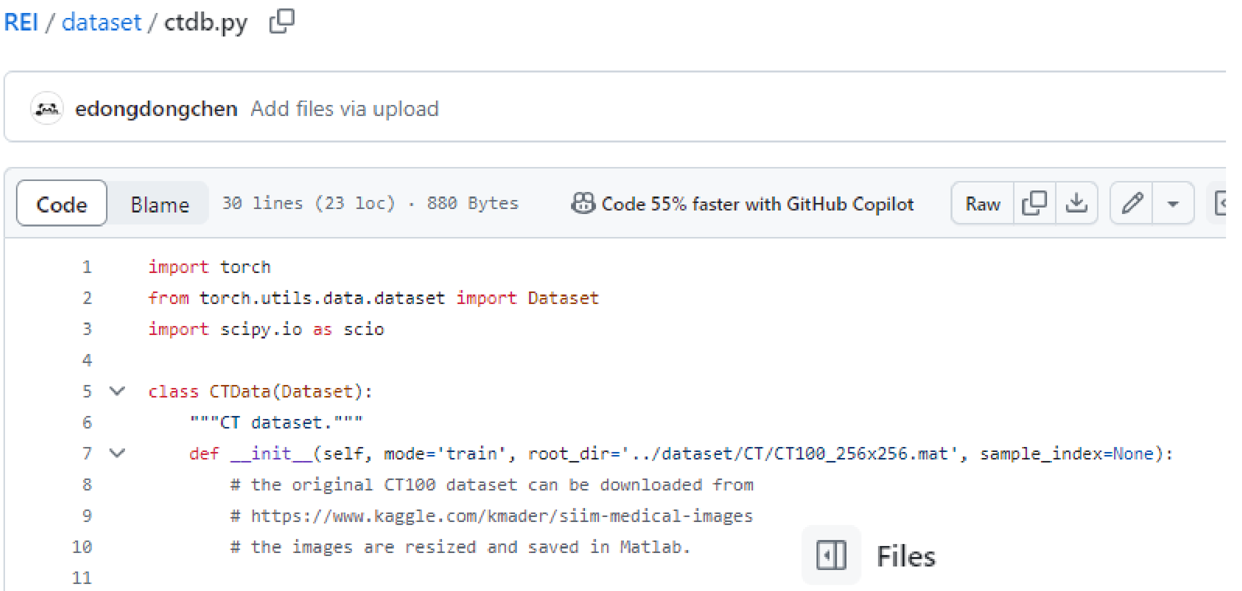
https://github.com/edongdongchen/REI/blob/main/dataset/ctdb.py


REI / demo_scripts / demo_ct.py

edongdongchen Add files via upload

Code
Blame
59 lines (41 loc) · 1.45 KB
Code 55% faster with GitHub Copilot
Raw

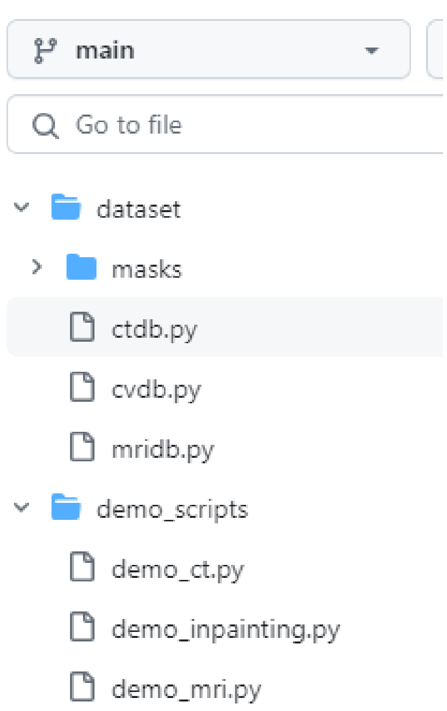
```
import torch
import argparse

from rei.rei import REI
from dataset.ctdb import CTData
from physics.ct import CT

from transforms.rotate import Rotate


parser = argparse.ArgumentParser(description='REI')
# inverse problem configs:
parser.add_argument('--task', default='ct', type=str,
                    help="inverse problems=['ct', 'inpainting', 'mri'] (default: 'ct')")

def main():
    args = parser.parse_args()
```


REI / dataset / ctdb.py

edongdongchen Add files via upload

Code
Blame
30 lines (23 loc) · 880 Bytes
Code 55% faster with GitHub Copilot
Raw

```
import torch
from torch.utils.data.dataset import Dataset
import scipy.io as scio

class CTData(Dataset):
    """CT dataset."""
    def __init__(self, mode='train', root_dir='../dataset/CT/CT100_256x256.mat', sample_index=None):
        # the original CT100 dataset can be downloaded from
        # https://www.kaggle.com/kmader/siim-medical-images
        # the images are resized and saved in Matlab.

```



root_dir='../dataset/CT/CT100_256x256.mat'

the original CT100 dataset can be downloaded from
https://www.kaggle.com/kmader/siim-medical-images
the images are resized and saved in Matlab.

Bottom Line

- To run the demo for the CT case, need input files in MATLAB format
- To generate input files in MATLAB format need to download data (which is still posted) then resize and save after reading in using MATLAB or OCTAVE
- Next task- download a file and write OCTAVE code to save it in a form where it can be read for the demo

https://github.com/edongdongchen/REI/blob/main/dataset/ctdb.py

```
class CTData(Dataset):
  """CT dataset."""
  def __init__(self, mode='train',
root_dir='../dataset/CT/CT100_256x256.mat',
sample_index=None):
    # the original CT100 dataset can be downloaded from
    # https://www.kaggle.com/kmader/siim-medical-images
    # the images are resized and saved in Matlab.

    mat_data = scio.loadmat(root_dir)
    x = torch.from_numpy(mat_data['DATA'])
```