Advancing X-ray Tomography using AI (TomoGAN)

Zhengchun Liu

Assistant Computer Scientist at Data Science and Learning Division

Mar. 2nd, 2020
APS XSD Coffee Talk

About me and my work

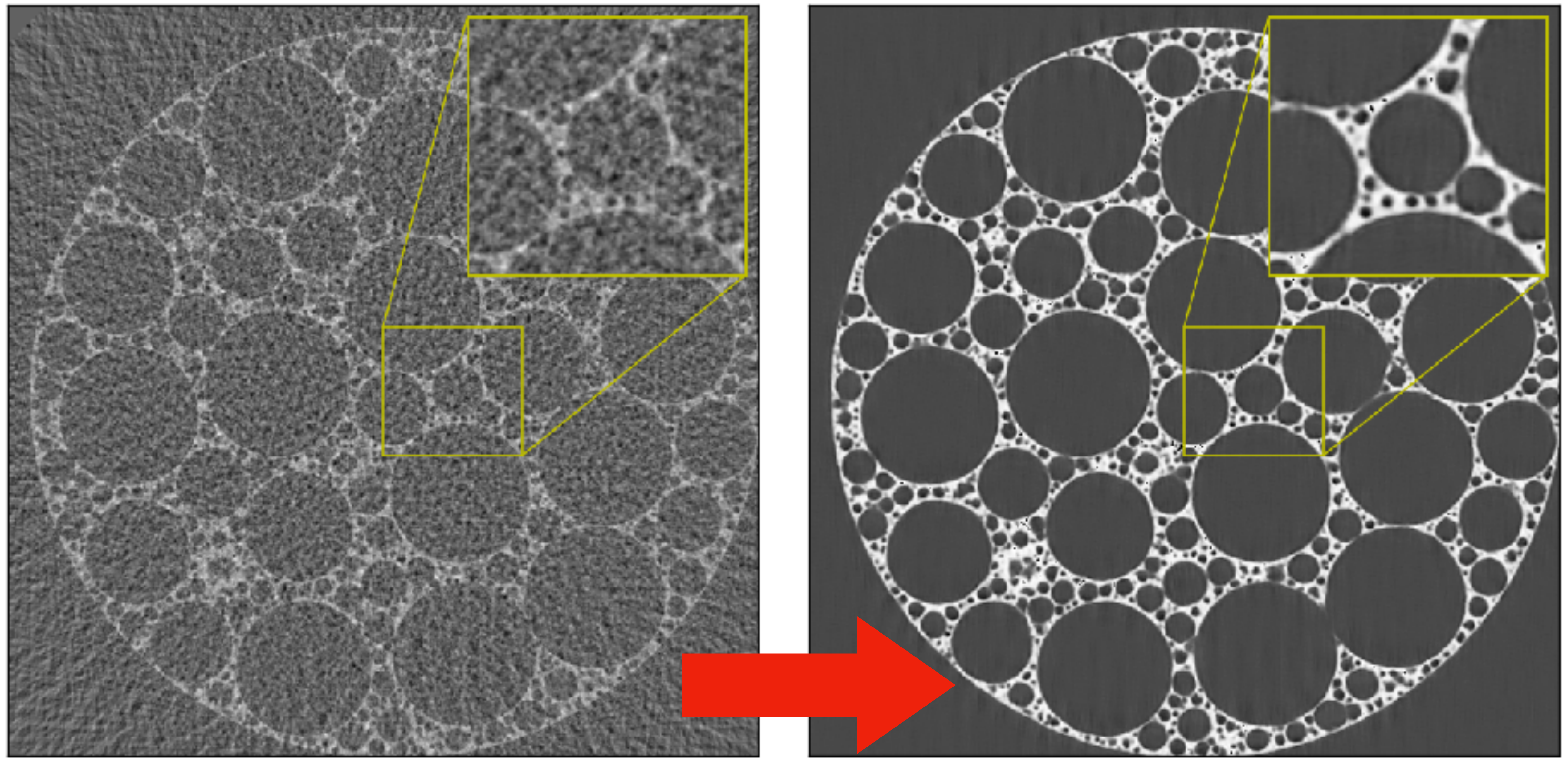
- Assistant Computer Scientist at Data Science and Learning Division
- High Performance Wide Area Data Transfer
- Data science and machine learning for computing system, e.g., performance modeling, bottleneck detection and reasoning.
- AI for Science, e.g., **X-ray at APS**, Climate Simulation, Accelerator at APS etc.
- Looking for collaboration on applying AI for more other domains.

Motivation

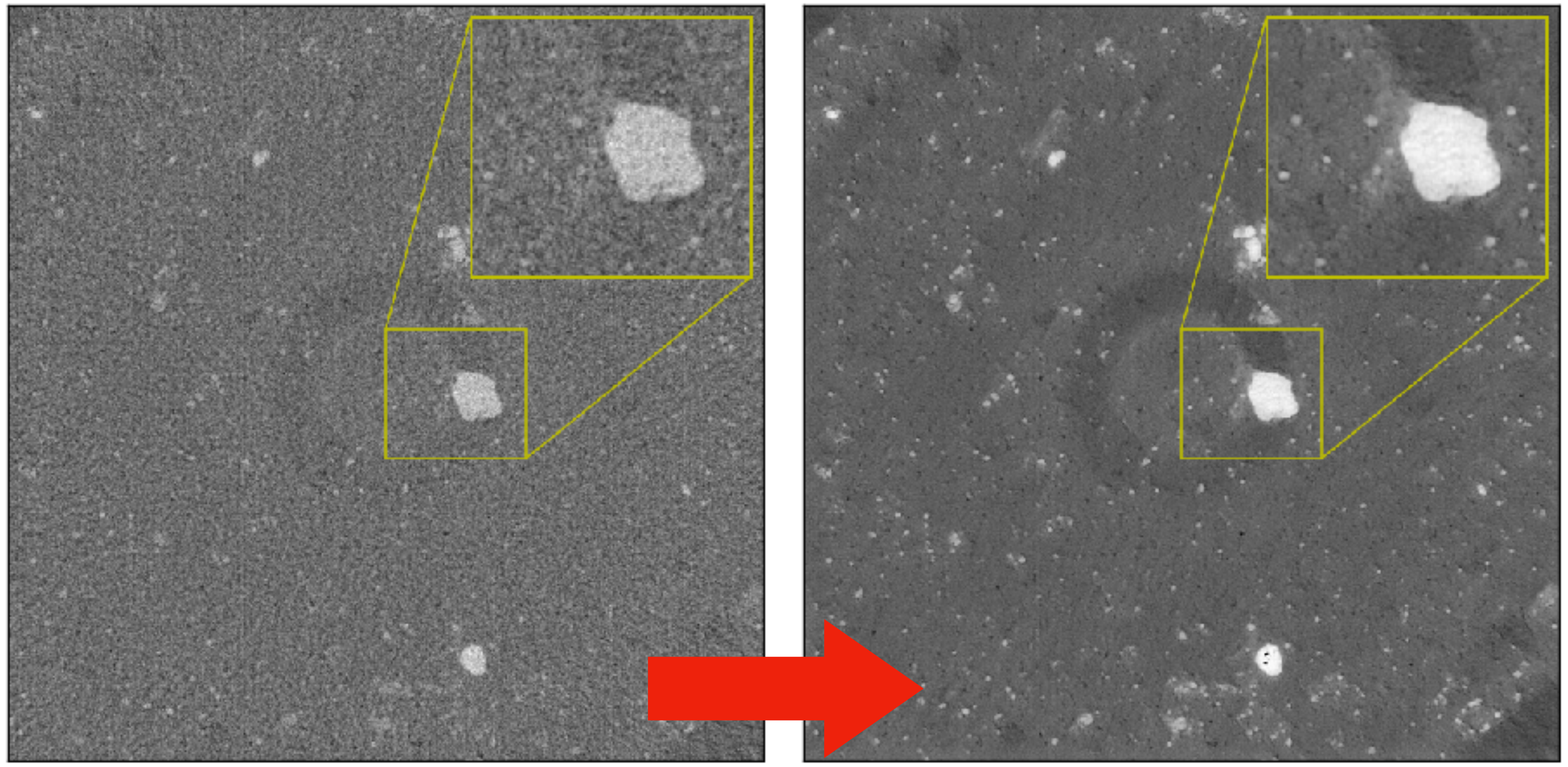
(1) lower X-ray dosage for sensitive sample like bio-sample;

(2) faster experiment to capture dynamic features, like in fast chemical reaction processes;

(3) smaller dataset and less computation for [near] realtime tomography imaging.



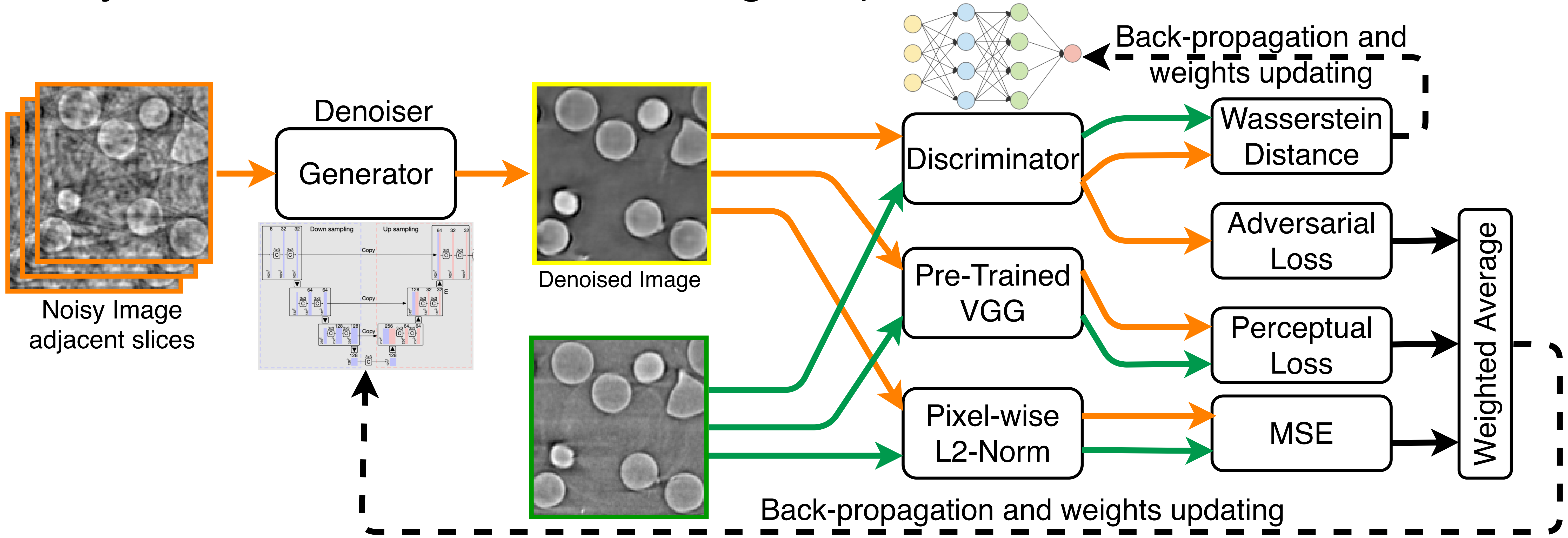
On the left, the results of conventional reconstruction, which are highly noisy. On the right, those same results after denoising with TomoGAN.



Model is trained with one shale sample imaged at APS and tested with **another** shale sample imaged at Swiss Light Source (SLS).

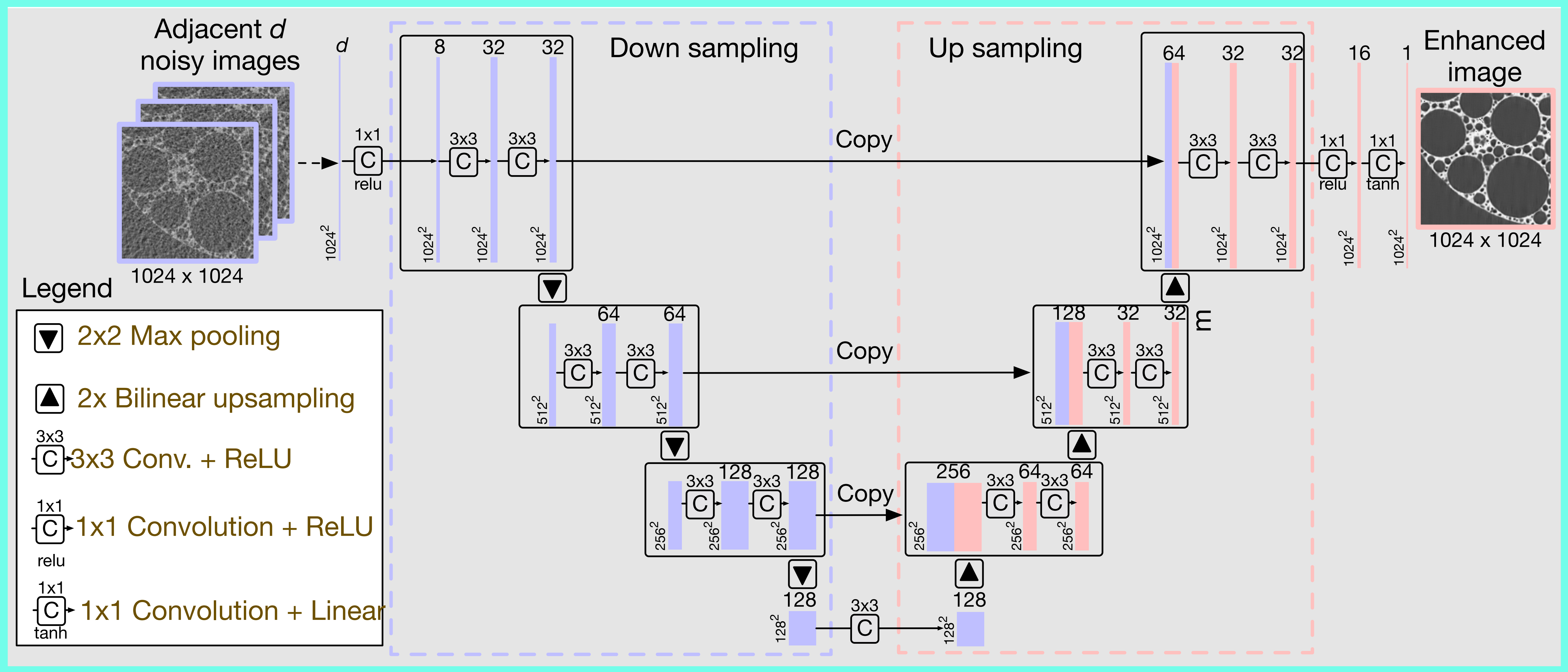
Method

A generative adversarial network (GAN) is a class of machine learning systems in which two neural networks, generator (G) and discriminator (D), contest with each other in a game (in the sense of game theory, often but not always in the form of a zero-sum game).



In our model, the discriminator's job remains unchanged, but the generator is tasked not only with fooling (indistinguishable) the discriminator but also with being near the ground truth output in an L2 sense.

The discriminator works as a helper to train the generator that we need to denoise images.



Our Generator Architecture

Training

Discriminator

Wasserstein GAN [1] + gradient penalty [2]

$$L\left(\theta_D\right) = \frac{1}{m}\sum_{i=1}^{m}\left[D\left(G\left(I_{LD}^{i}\right)\right) - D\left(I_{ND}^{i}\right)\right] + \lambda_D \frac{1}{m}\sum_{i=1}^{m}\left[\left(\left\| \nabla_{\bar{I}} D\left(\bar{I}^{i}\right)\right\|_2 - 1\right)^2\right],$$

Generator

Weighted average of Adversarial loss, Perceptual loss, and Pixel-wise MSE

$$\ell^{G} = \lambda_g \ell_{adv} + \lambda_p \ell_{mse} + \lambda_v \ell_{vgg}$$

$$\ell_{adv}\left(\theta_G\right) = -\frac{1}{m}\sum_{i=1}^{m} D\left(G\left(I_{LD}^{i}\right)\right)$$

$$\ell_{vgg} = \sum_{i=1}^{W_f}\sum_{i=1}^{H_f}\left(V_{\theta_{vgg}}\left(I^{ND}\right)_{i,j} - V_{\theta_{vgg}}\left(G_{\theta_G}\left(I^{LD}\right)\right)_{i,j}\right)^2$$

$$\ell_{mse} = \sum_{c=1}^{W}\sum_{r=1}^{H}\left(I_{c,r}^{ND} - G_{\theta_G}\left(I^{LD}\right)_{c,r}\right)^2$$

[1] Wasserstein GAN. M. Arjovsky, S. Chintala, L. Bottou. arXiv:1701.07875

[2] Improved Training of Wasserstein GANs. I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. Courville. arXiv:1704.00028

Results - Adjacent slices

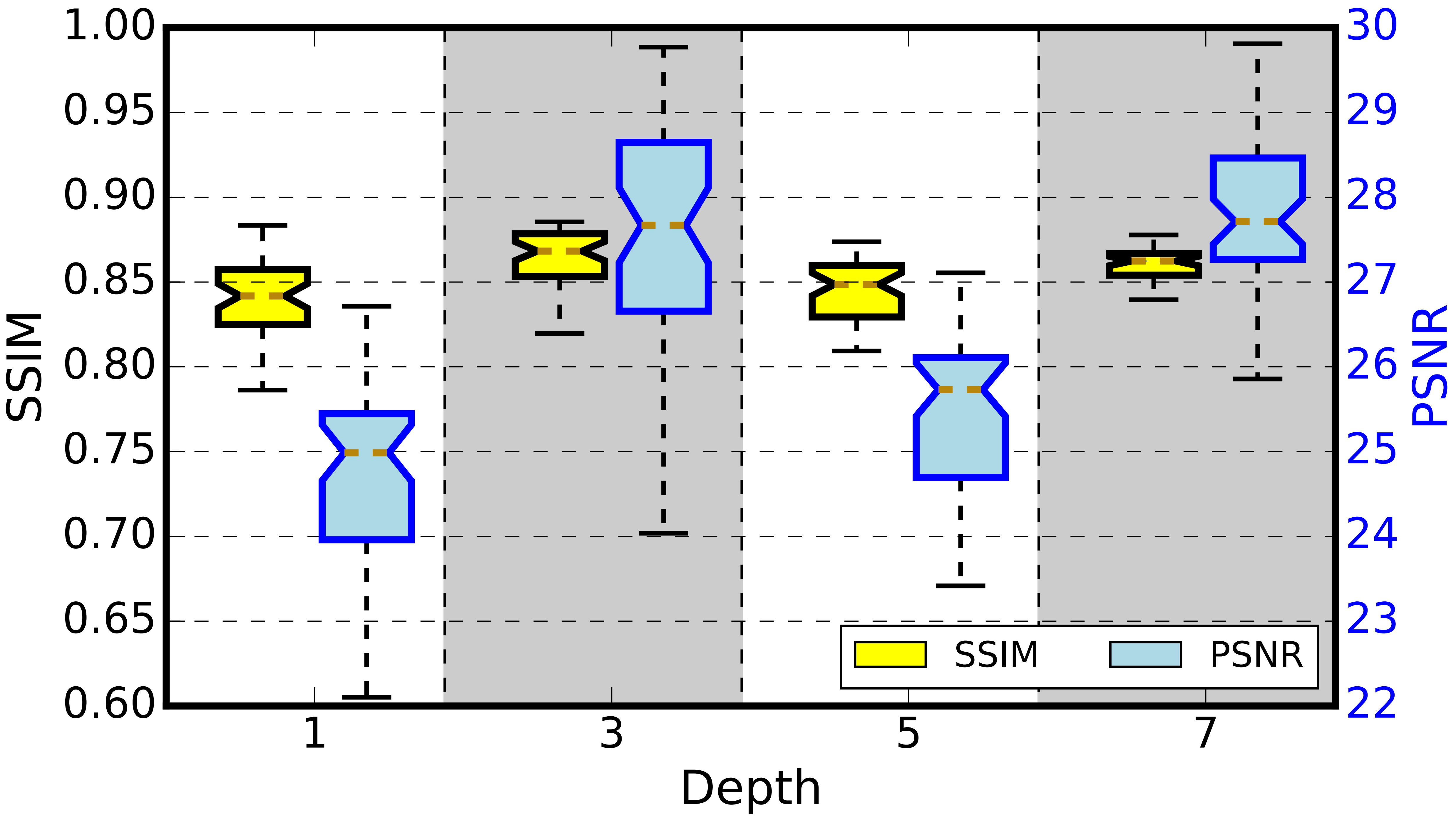
Effectiveness of using adjacent slices in image enhancement

SSIM: 0.843, PSNR: 25.5 | **SSIM:** 0.850, **PSNR:** 27.0 | **SSIM:** 0.831, **PSNR:** 25.9 | **SSIM:** 0.830, **PSNR:** 26.7

(a) Depth = 1 (b) Depth = 3 (c) Depth = 5 (d) Depth = 7 (d) Ground Truth

The input depth *d* has big influence on mode performance, and that *d*=3 gets the best quality, especially when the original feature edge is not sharp (e.g., the center circle).

We note that the best depth *d* depends on dataset characteristics such as feature resolution. *d*=3 may not be the best for other datasets where feature sizes change slowly across slices.

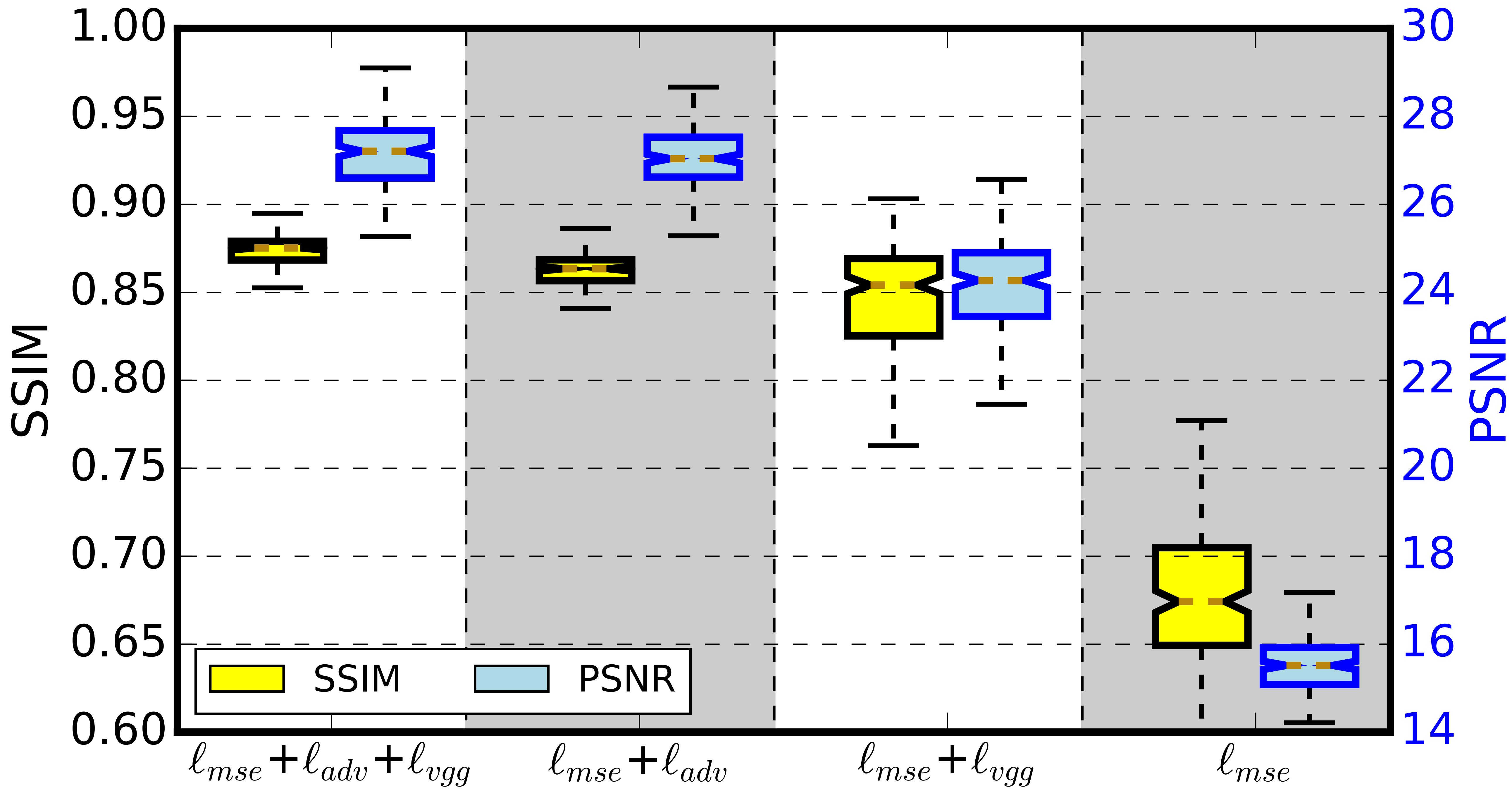


Results - Loss

Importance of the various loss terms

SSIM: 0.868, PSNR: 26.84 | SSIM: 0.842, PSNR: 26.79 | SSIM: 0.864, PSNR: 25.9 | SSIM: 0.811, PSNR: 24.5

$\ell_{mse} + \ell_{vgg} + \ell_{adv}$ | $\ell_{mse} + \ell_{adv}$ | $\ell_{mse} + \ell_{vgg}$ | ℓ_{mse} | Ground Truth



MSE is necessary to enforce correctness of low-frequency structures but MSE alone is not enough.

The adversarial and perceptual loss terms each provide considerable improvements when used in isolation.

The two together are only slightly better than adversarial loss alone.

Experiments

Datasets

- **Three foam simulation datasets, each with 1024 slices**
- **Two shale samples imaged at both APS and SLS, totals four datasets and each with 2048 slices.**

Label	projection	reconstruction	Facility	Sample	Scan	Axis
tomo_00001	(1501, 1792, 2048)	(1792, 2048, 2048)	APS	B1	hornby	1024
tomo_00002	(1501, 1792, 2048)	(1792, 2048, 2048)	APS	N1	blakely	1029
tomo_00003	(1441, 2048, 2048)	(2048, 2048, 2048)	SLS	B1	hornby	1011
tomo_00004	(1441, 2048, 2048)	(2048, 2048, 2048)	SLS	N1	blakely	1048

Low dose cases

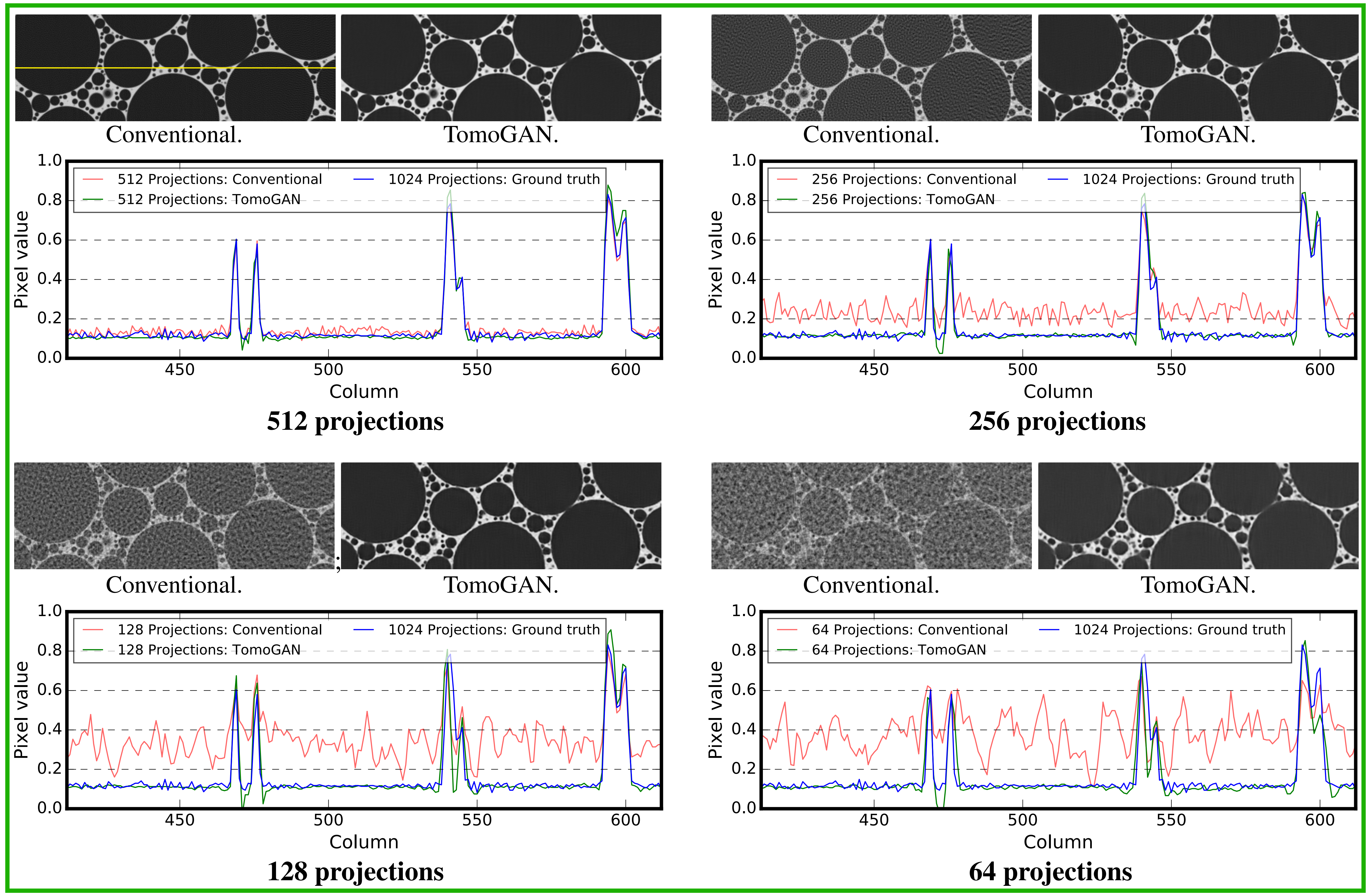
- **Sparse views**

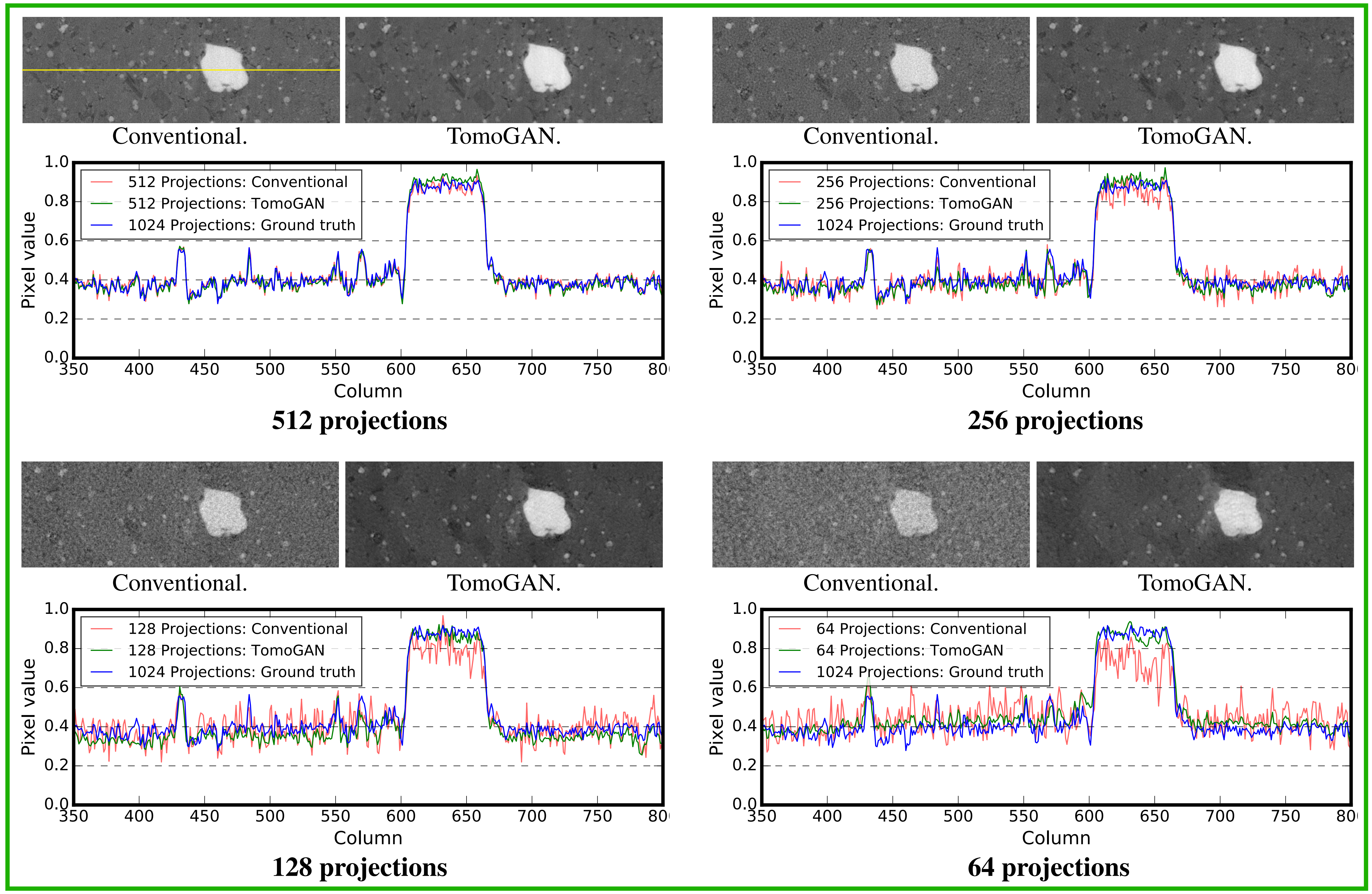
  Subsample the original, (i.e., normal dose) projections to 1/2, 1/4, 1/8 and 1/16 for experiments and model evaluation.

- **Short exposure time.**

  For simulation datasets, we simulate x-ray projections with different photon intensities to simulate different exposure times
  For experimental shale datasets, we used added noise using a Poisson distribution to simulate different exposure times.

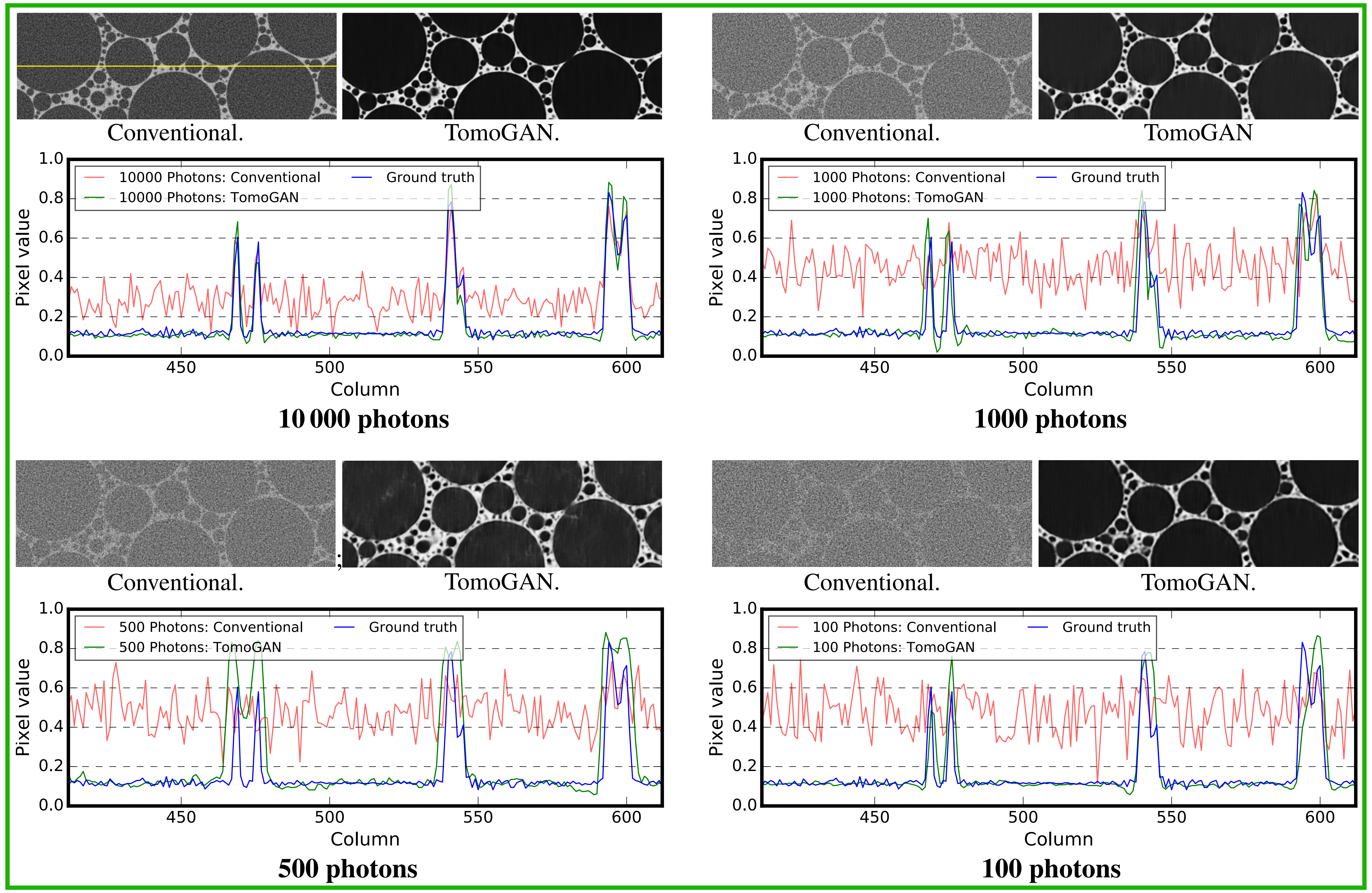
Results - Sparse views





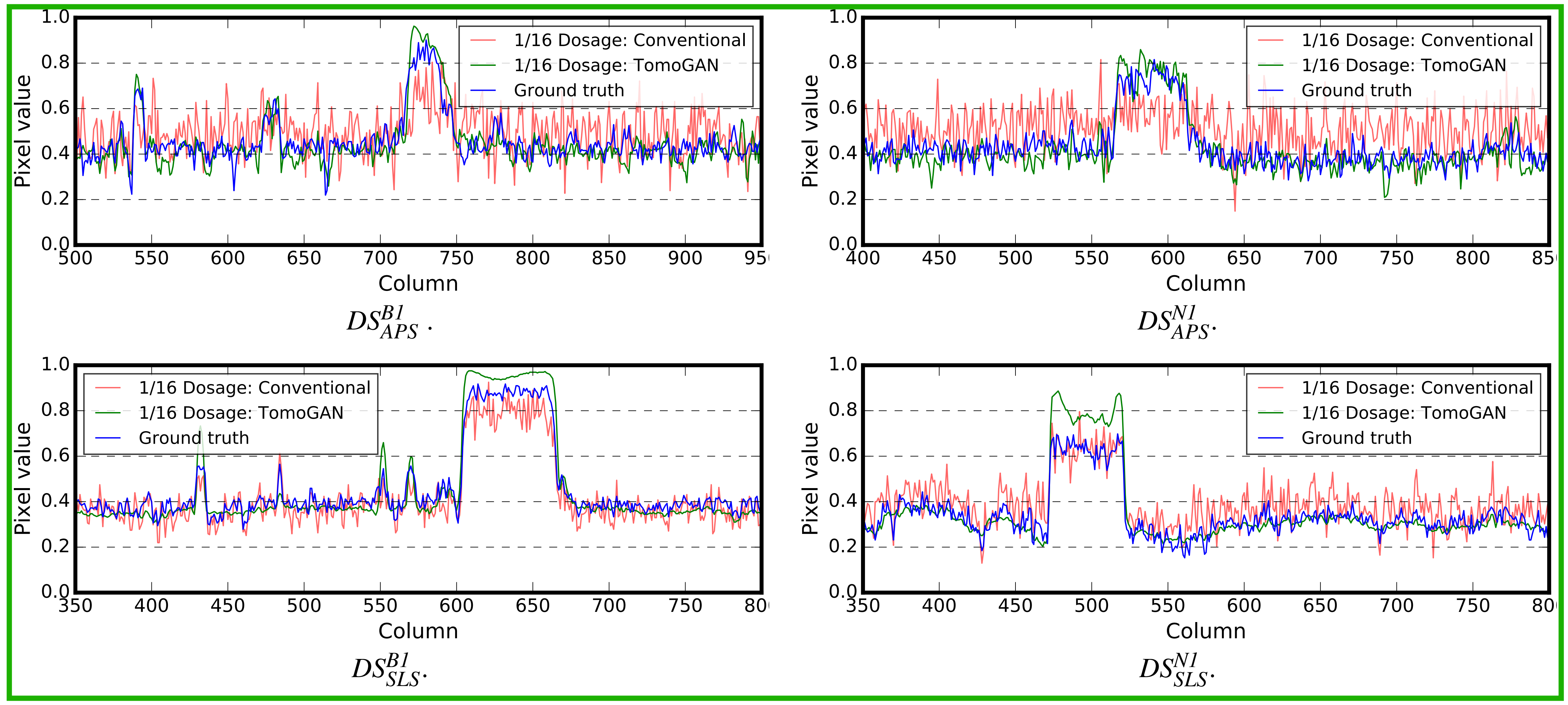
Conventional vs. TomoGAN-enhanced reconstructions of simulated (left) data and shale (right), subsampled to (512, 256, 128, 64) projections. In each group of three elements, the two images show conventional and TomoGAN reconstructions, while the plot shows conventional, TomoGAN, and ground truth values for the 200 pixels on the horizontal line in the top left image.

Results - Short exposure time



Conventional vs. TomoGAN-enhanced reconstructions of simulated data with intensity limited to 10000, 1000, 500, 100 photons per pixel.

Pixel values of an arbitrarily chosen feature in each of the four experimental datasets, with projections generated by using 1/16 of the normal exposure time. Feature shapes are different for each dataset.

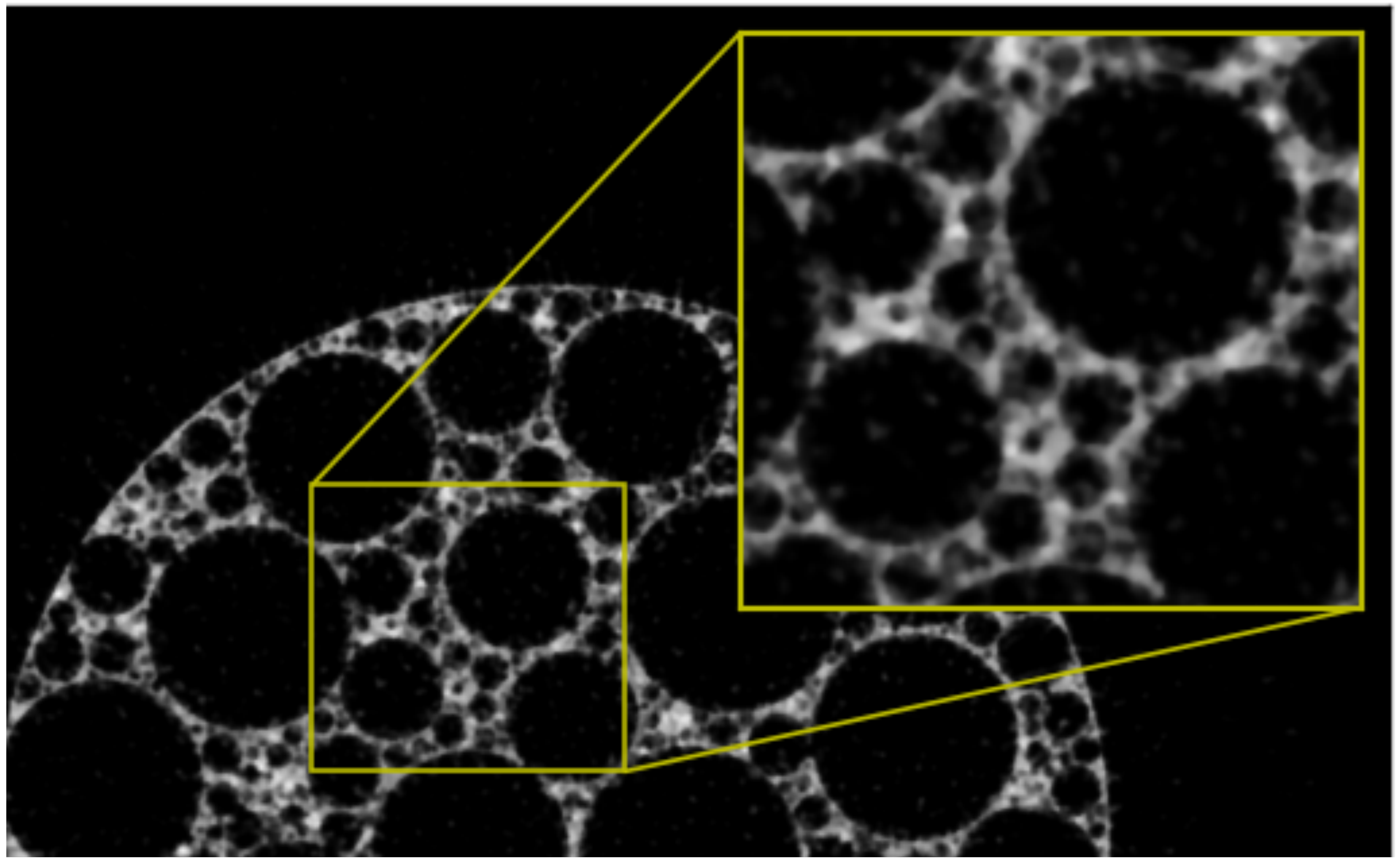


Computational superiority

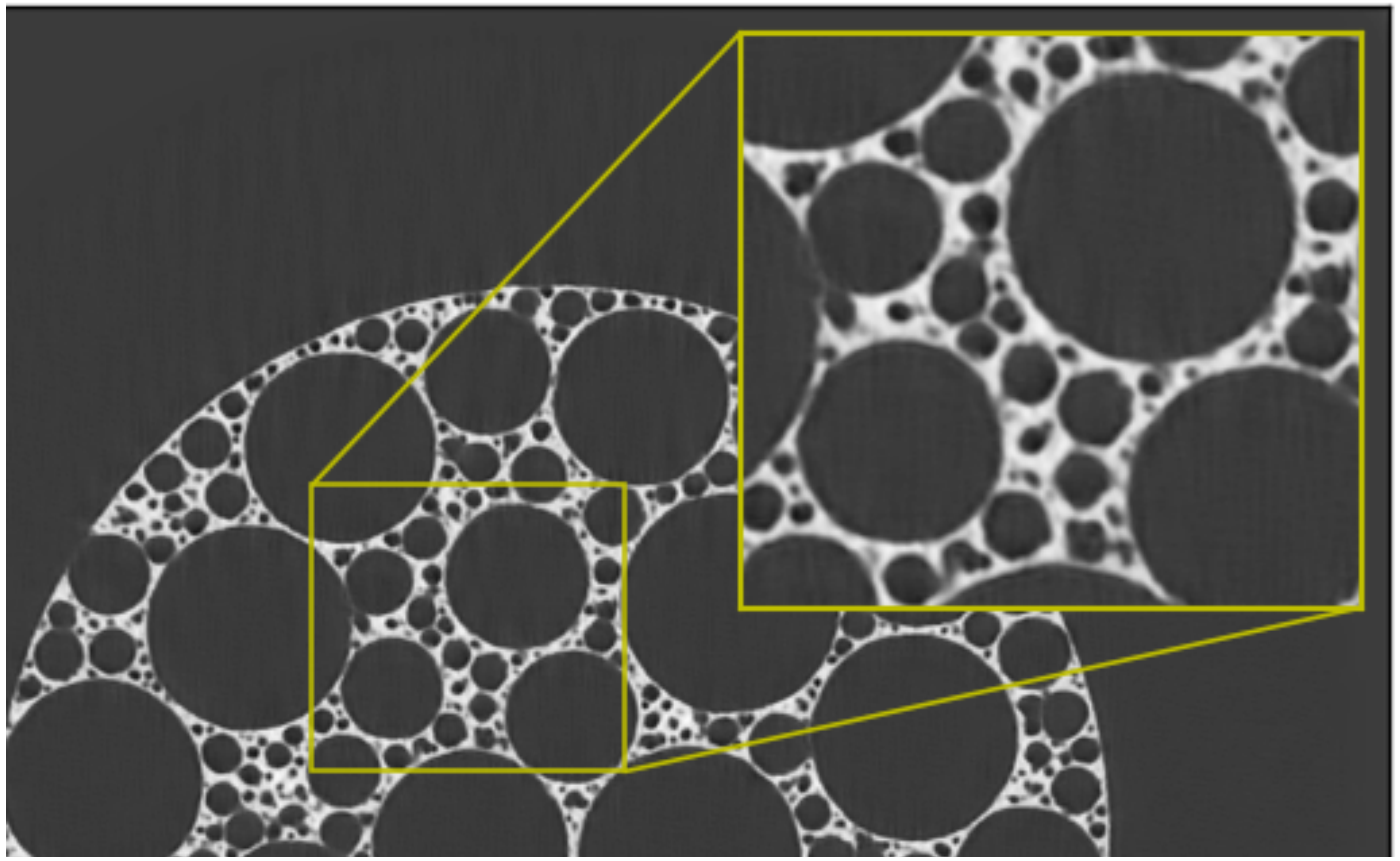
The filtered back projection (FBP) algorithm takes 40 ms to reconstruct one image (using TomoPy) and TomoGAN takes 30 ms to enhance the reconstruction, totals **70 ms** per image.

In contrast, the SIRT based solution (using TomoPy) takes **550 ms** (400 iterations), i.e., 8x faster. Times are measured using one Tesla V100 graphic card.

Moreover, iterative reconstruction does not provide better image quality than does our method.



SIRT + total variation postprocess.

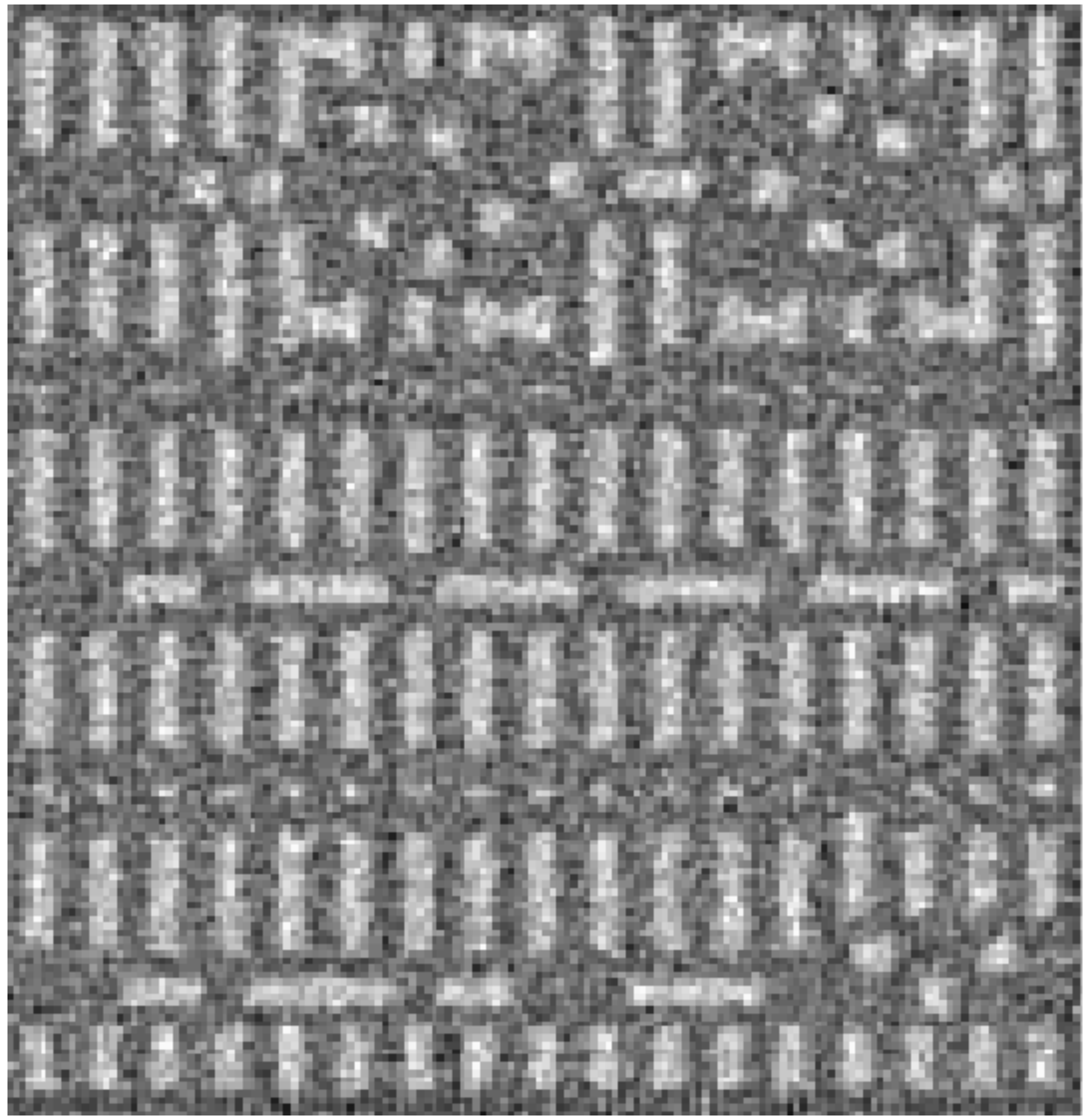


Filtered back projection + TomoGAN post-process.

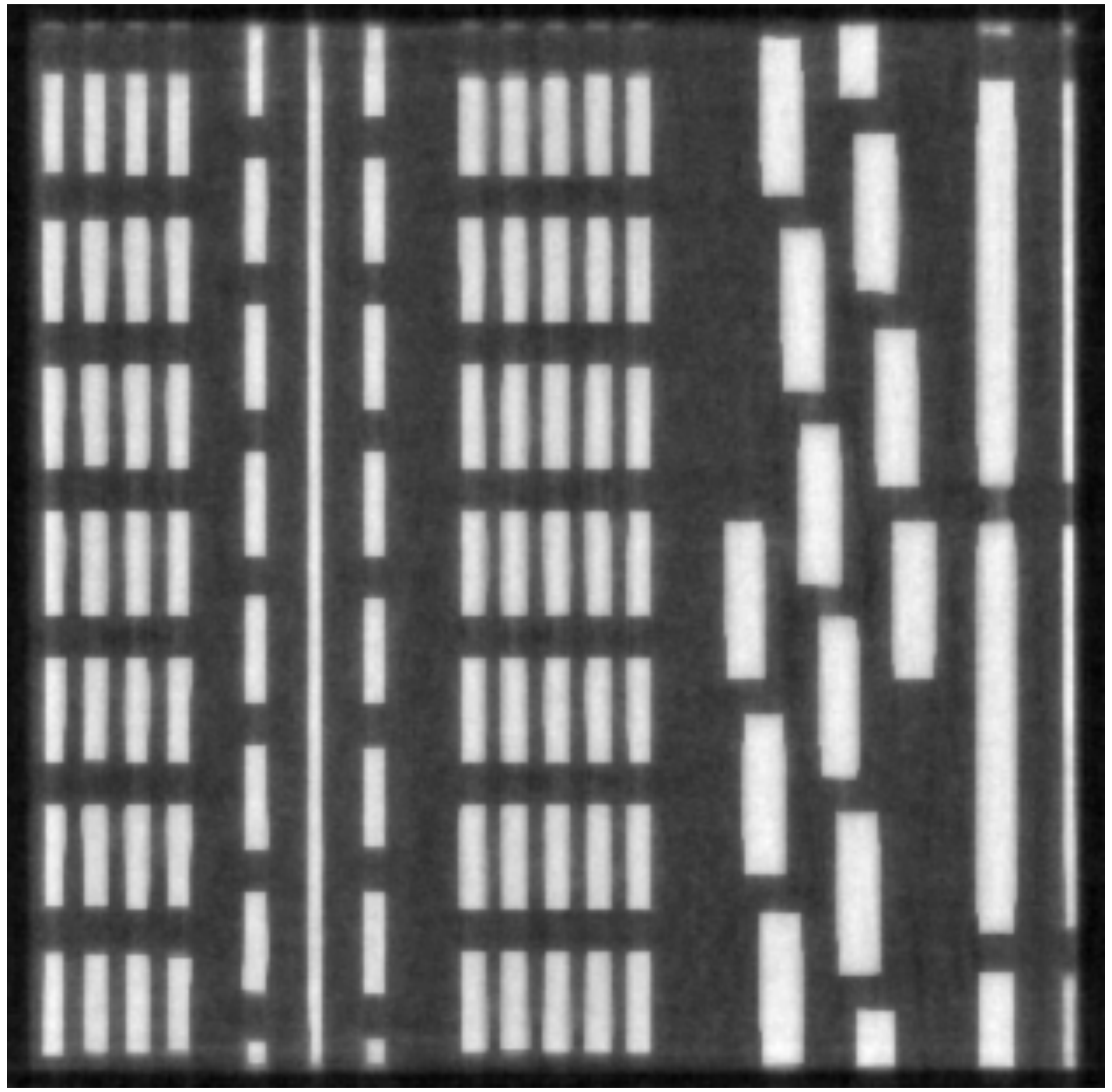
TomoGAN - Extended use case 1 - Prior in ADMM

It has been applied to the joint ptycho-tomography problem for reconstructing the complex refractive index of a 3D object.

- There is a ptychography process to reconstruct projections needed for tomography.
- Less datapoint results in noisier ptychography reconstruction and worse tomography images.
- TomoGAN here was used to enhance tomography images.



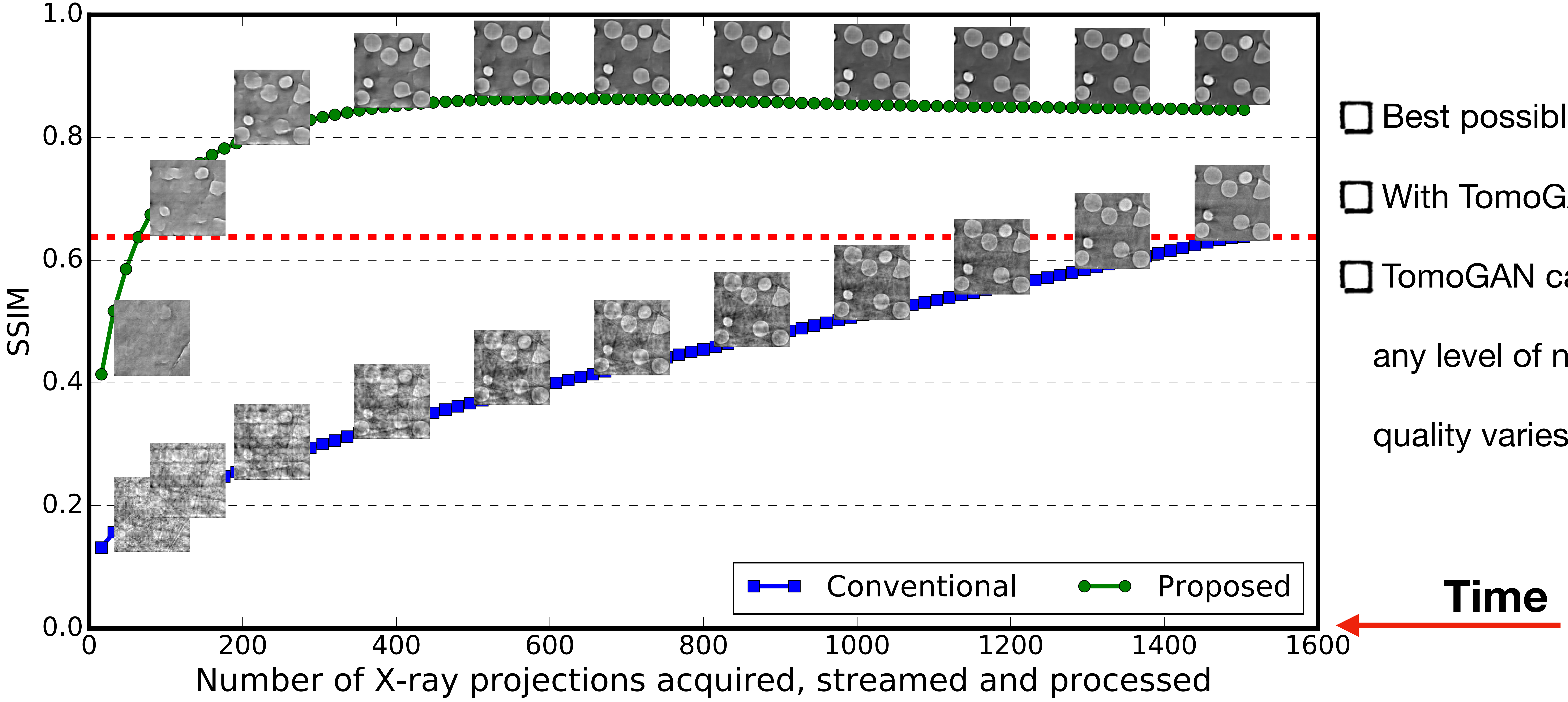
Original ADMM w/o using TomoGAN
One of images inTraining Dataset



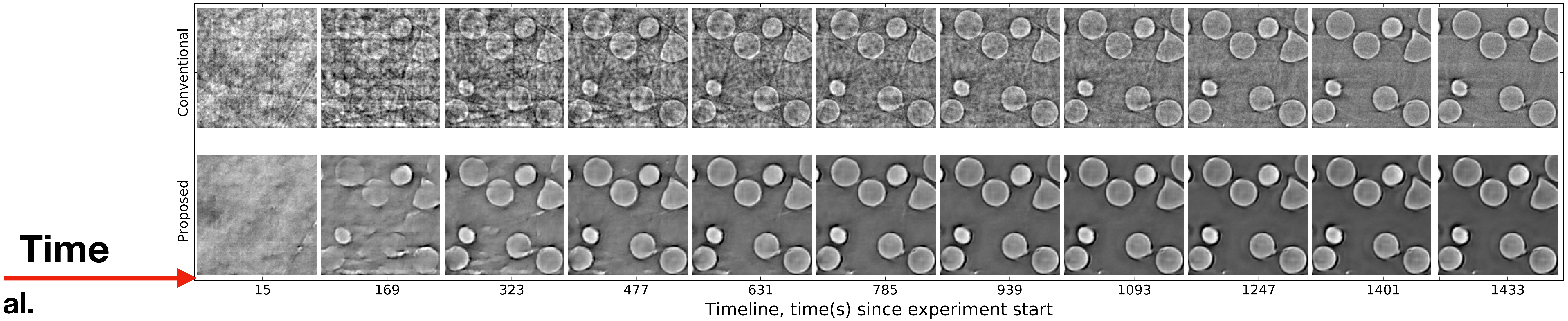
TomoGAN as learned prior in ADMM
One of images in Testing dataset

With **Selin Aslan et al.**

TomoGAN - Extended use case 2 - Streaming tomography

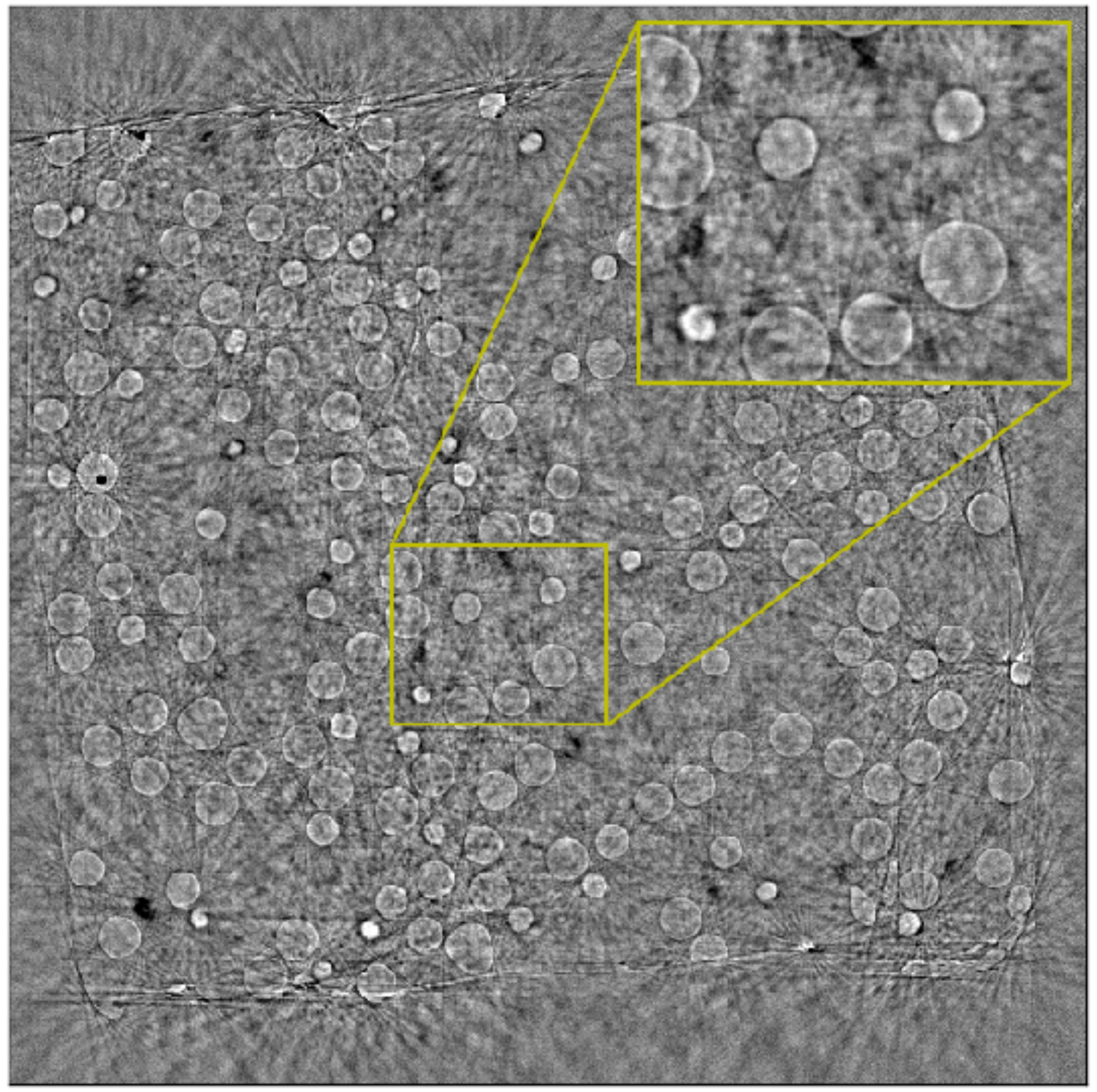


- Best possible as ground truth to train TomoGAN
- With TomoGAN, reconstruction converts faster
- TomoGAN captured features, thus it works for any level of noisy although denoised image quality varies.

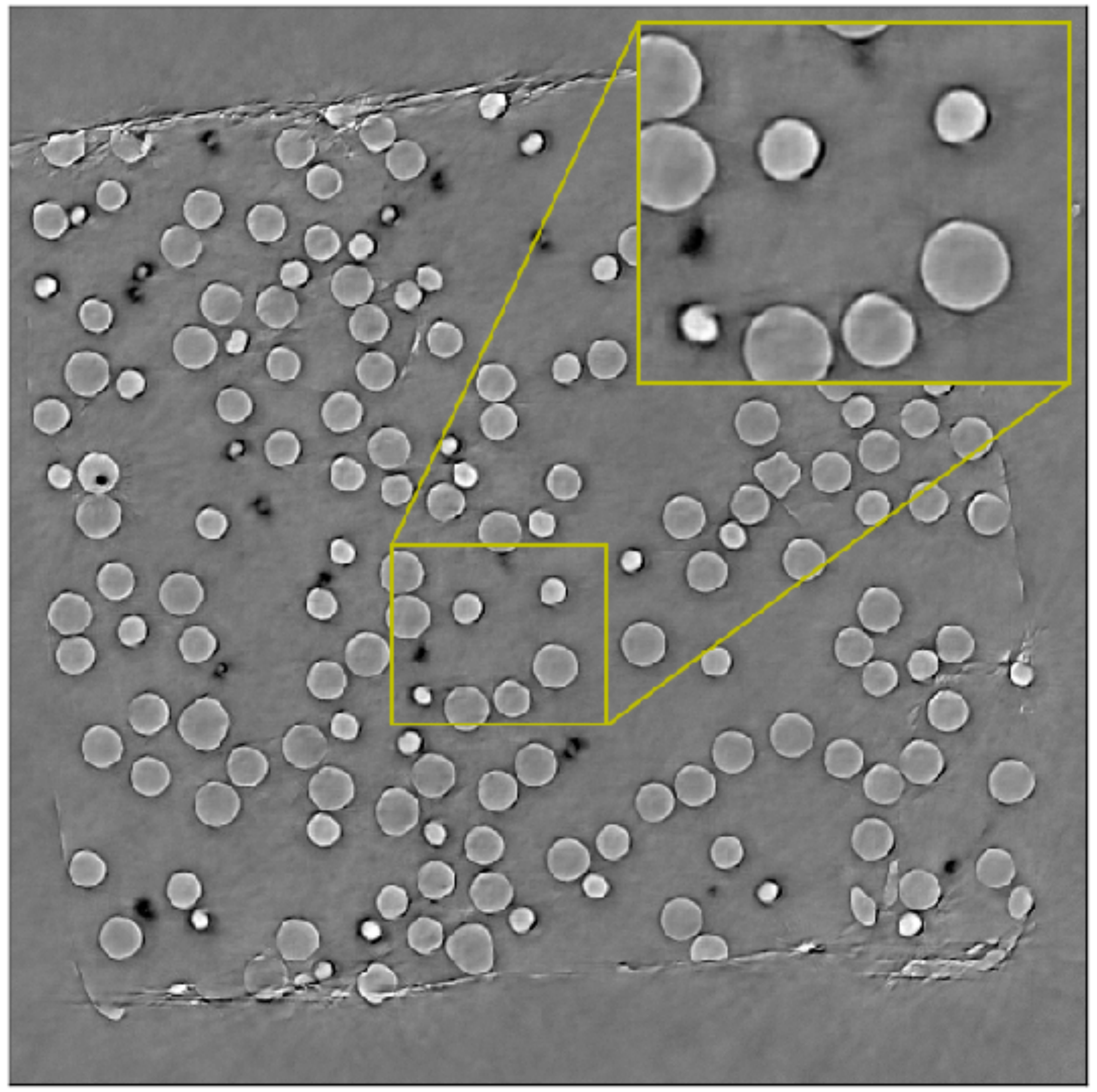


With **Tekin Bicer et al.**

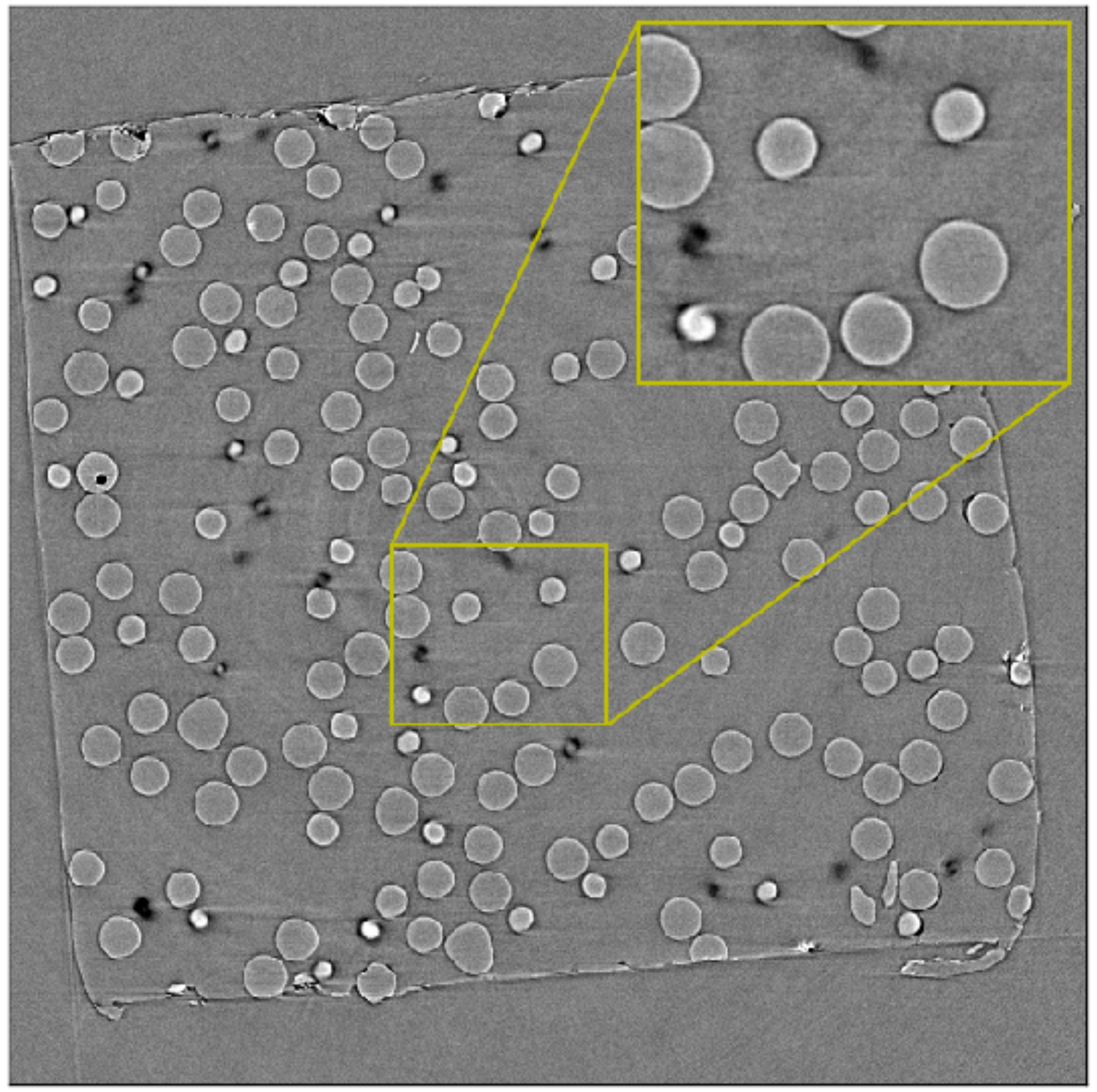
TomoGAN - Extend use case - Streaming tomography



with data up to 462s (480 projections), before enhancement;



with the same data, after enhancement;



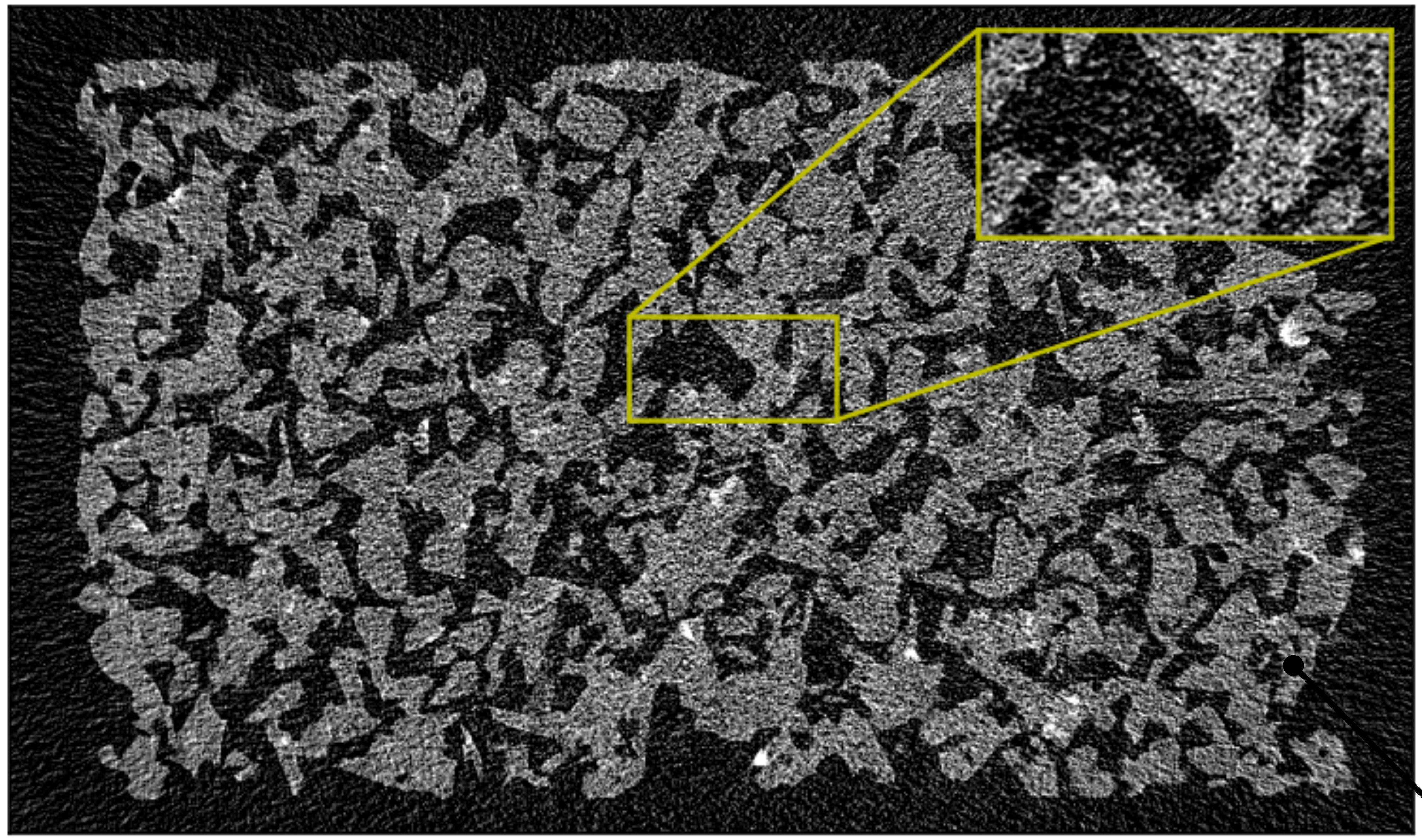
with data up to 1433s (1504 projections), before enhancement.

Three times faster turnaround time for domain scientists. A.K.A., three times increased throughput for the light source and computing facility.

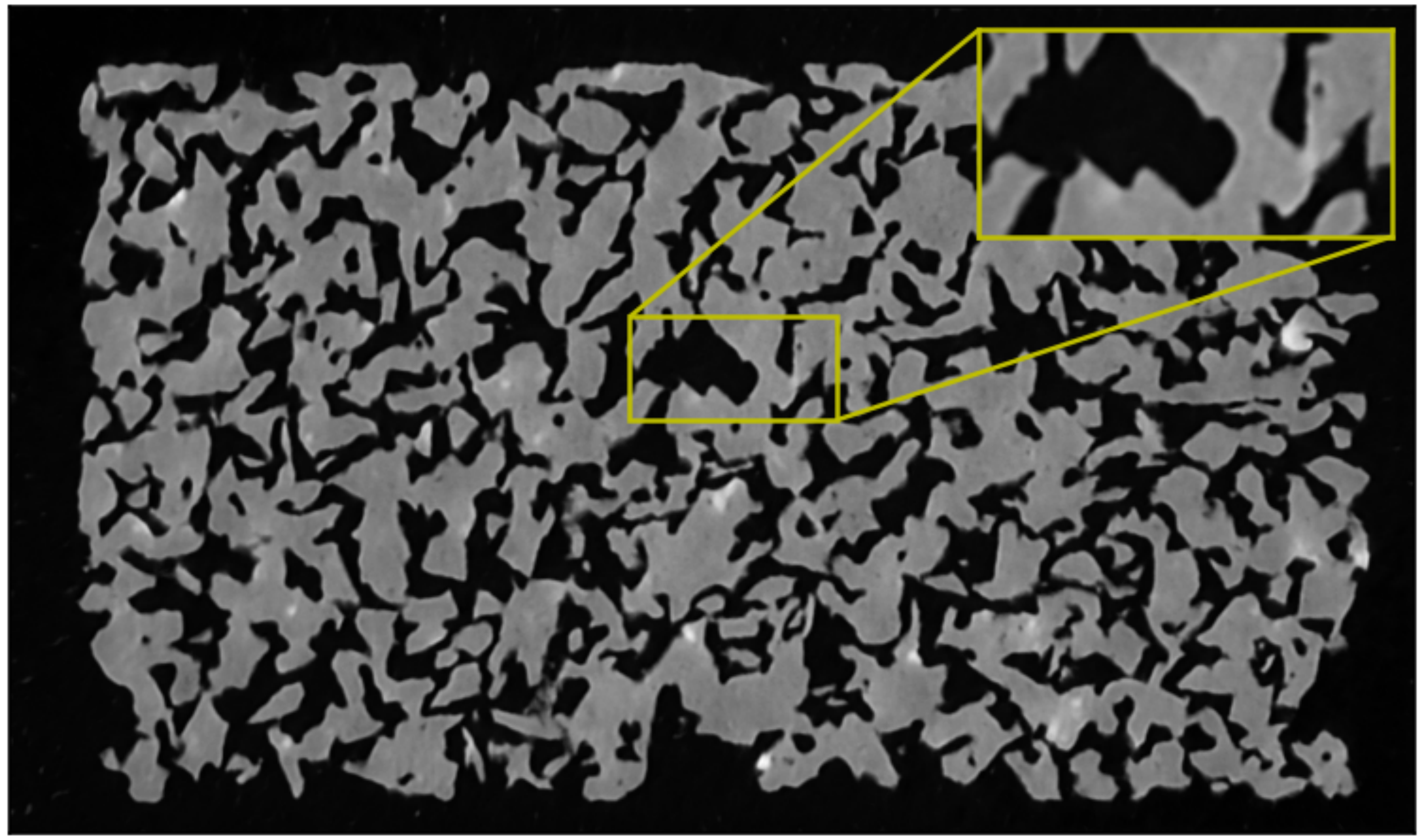
Important as enablers of experiment steering, where quick turnaround is required.

With **Tekin Bicer et al.**

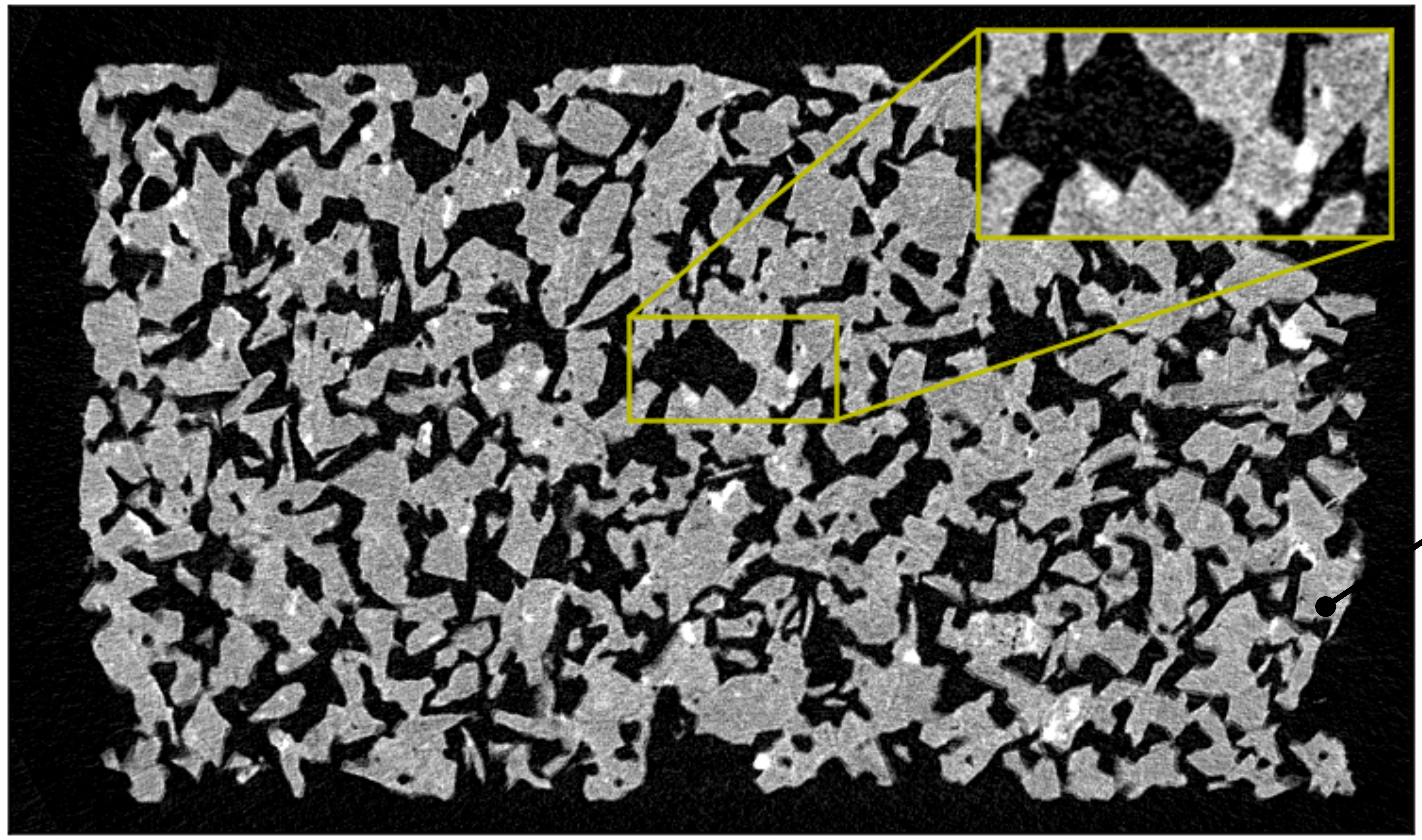
TomoGAN - Extended use case 3 - 3M with alignment issue



180°, large step size, no frame avg. (45 minutes)



TomoGAN enhanced (Model Output)



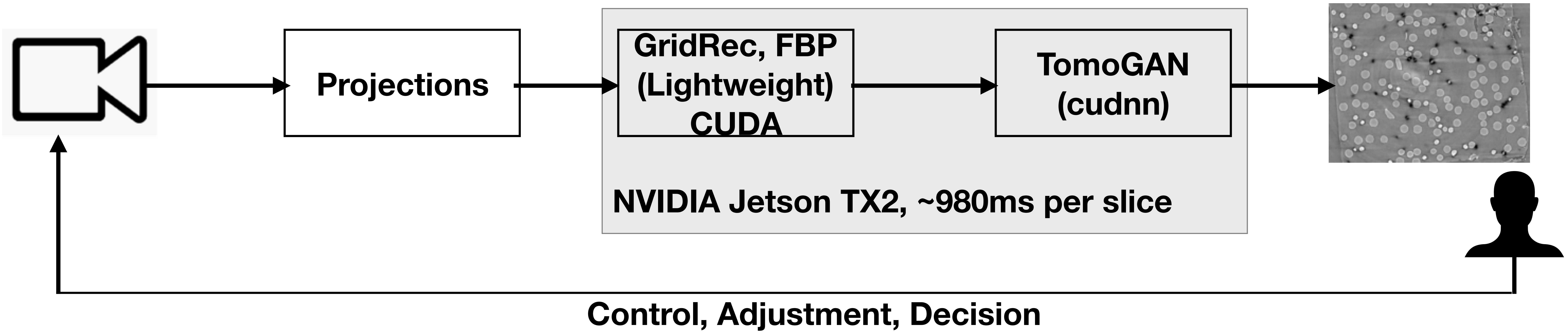
Best attempt (4 hours)

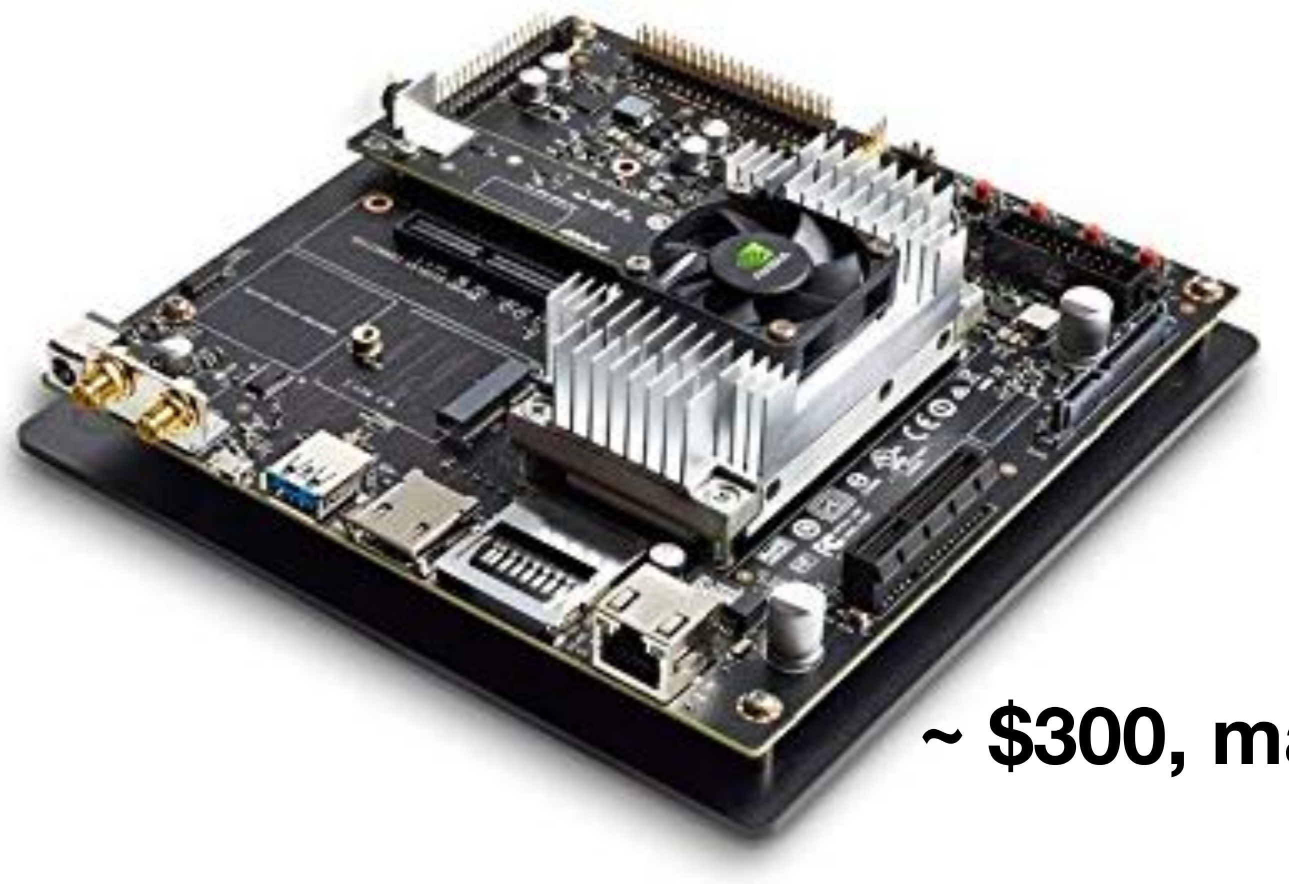
- (X,y) pair comes from two experiments;
- Impossible to perfectly aligned, like rotated a bit;
- Not a big problem for scientists but a big problem to ℓ_{mse}
- Tune the weight of ℓ_{mse}, ℓ_{vgg} and ℓ_{adv} works.

With **Myles Brostrom et al.**

TomoGAN - Tomography at Edge

- [ ] **Both Tomography and DL are computation intensive but both GPU typically helps a lot;**
- [x] **A GPU friendly tomography for a rough (noisy) results plus DL based enhancement;**
- [ ] **Fusion of analytical (human knowledge) and deep learning (data driven).**





~ $300, maximum 15 watts

With **Viktor Nikitin et al.**

Make it usable

Hack and Play

open source implementation, better to have a GPU for training

```
Git clone git@github.com:ramsesproject/TomoGAN.git

python ./train.py -ld noise-img.hdf5 -nd clean-img.hdf5

python ./infer.py -ld ld-prod.hdf5
```

X as a Service

DLHub

Data and Learning Hub for Science

B. Blaiszik. arXiv:1811.11213

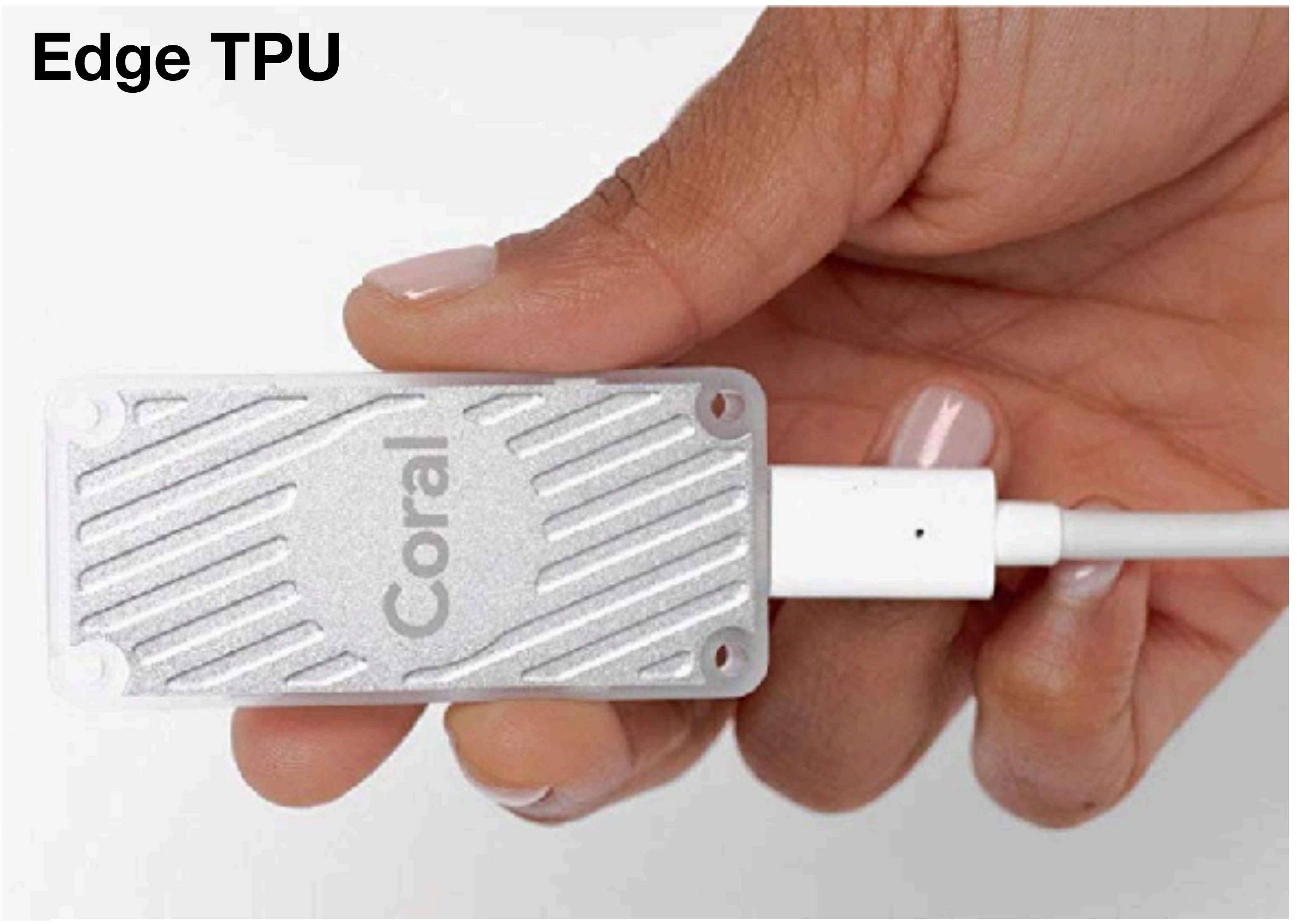
```
from dlhub_sdk.client import DLHubClient
dlhub = DLHubClient()

model = dlhub.get_id_by_name("tomoGAN")
data  = h5py.File("tomo_ld.hdf5", “r")["ld_img"]
pred  = dl.run(model, data)
```

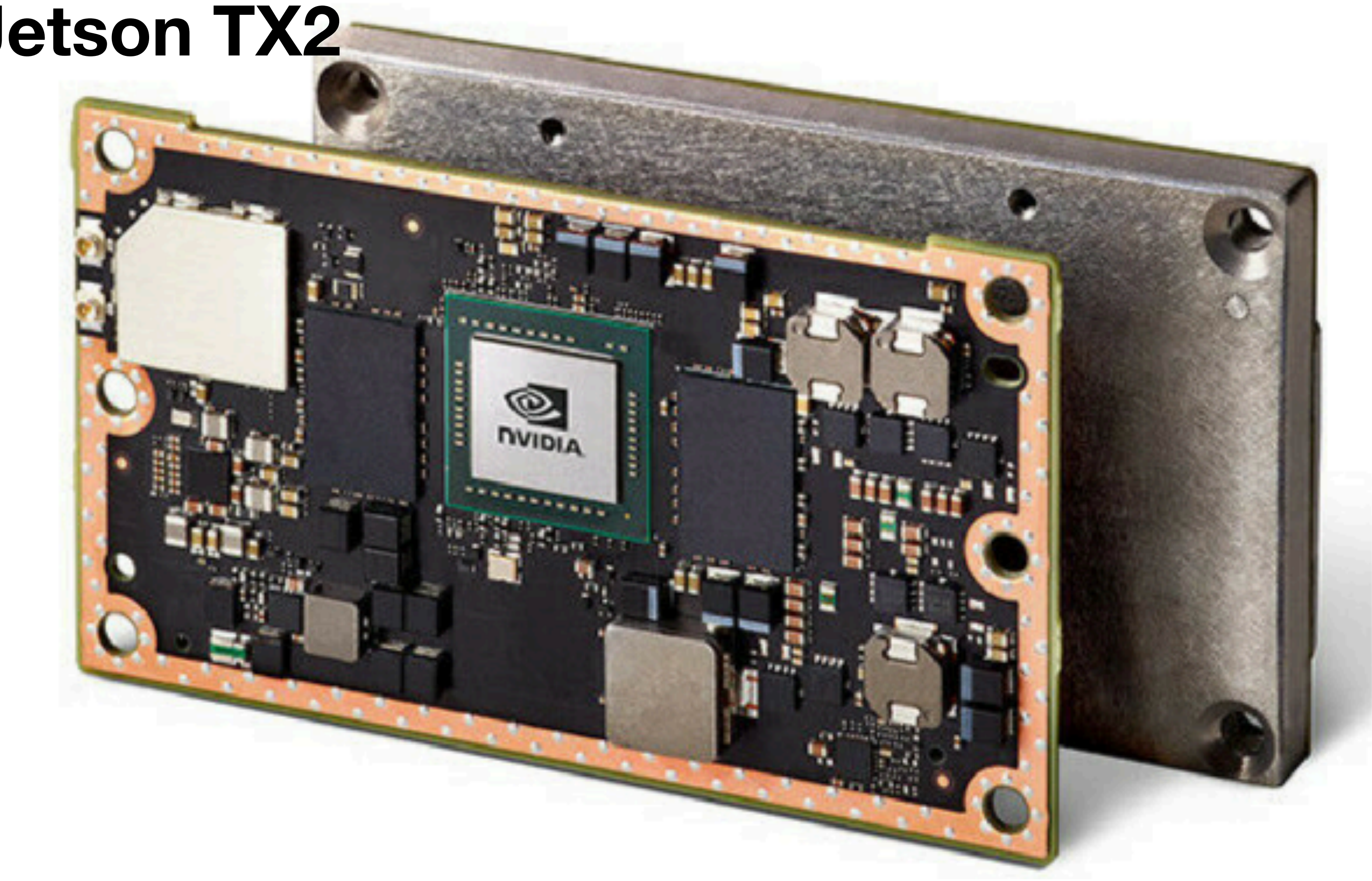
Plug and Play

Abeykoon et al.

Edge TPU

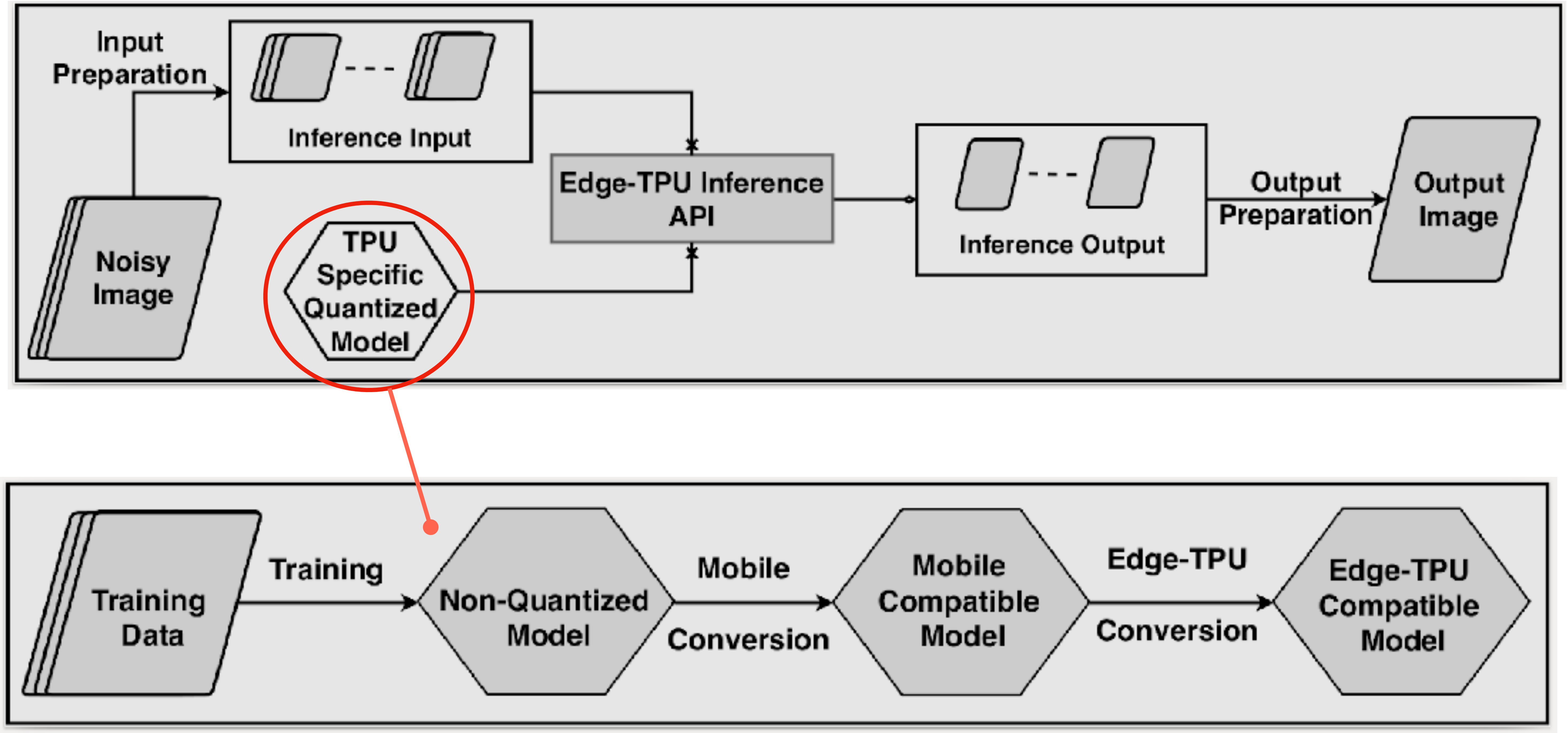


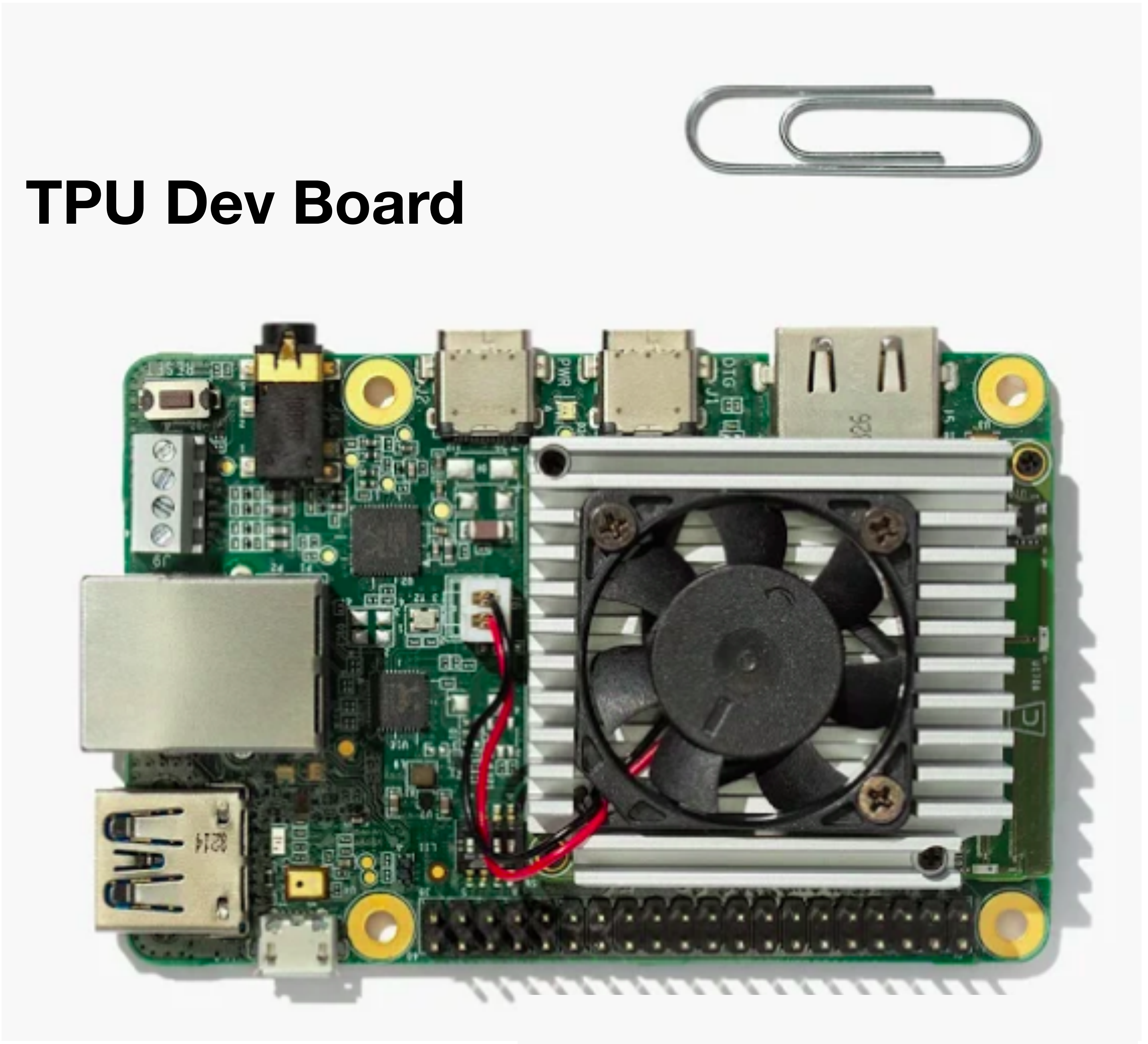
Jetson TX2

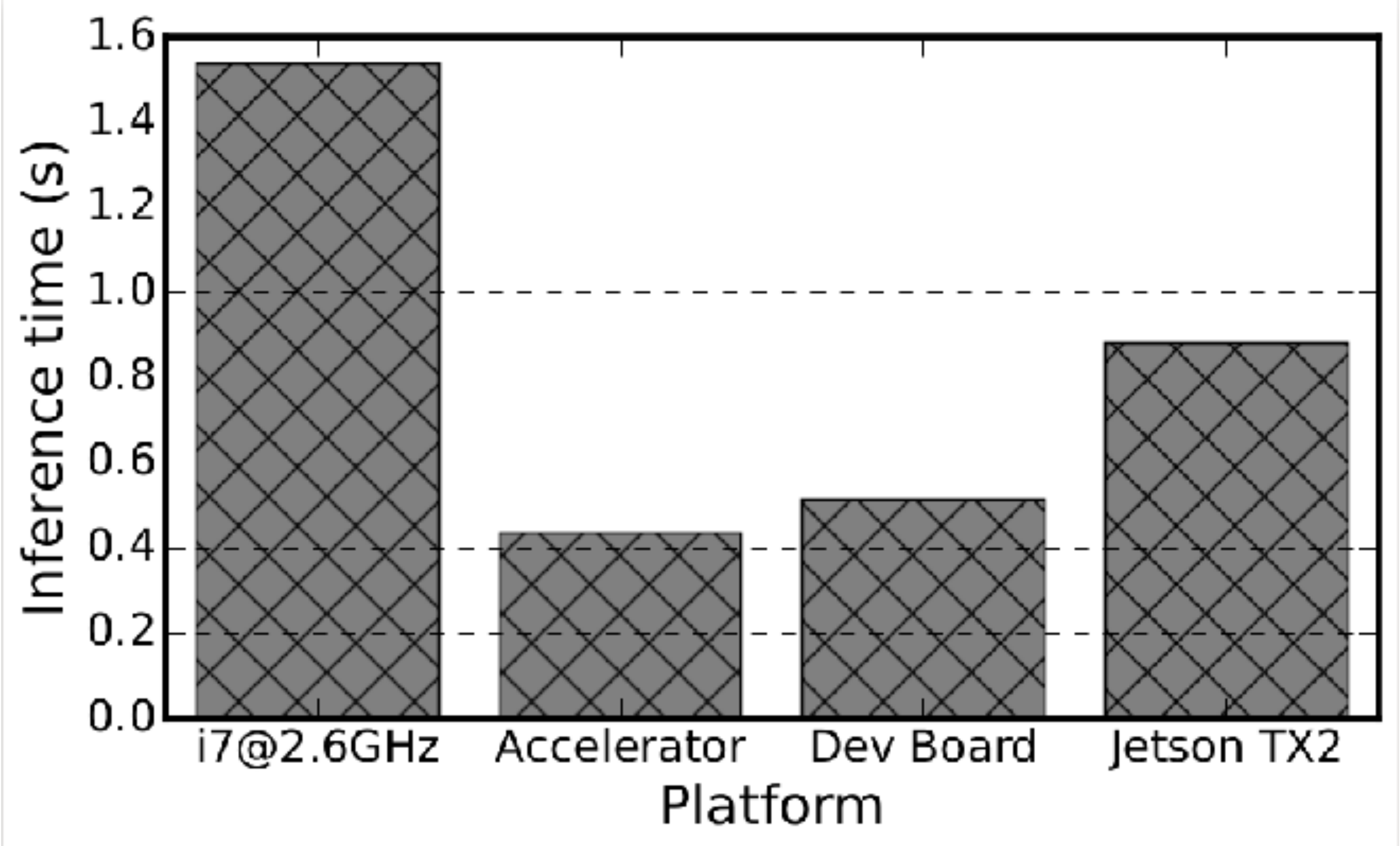


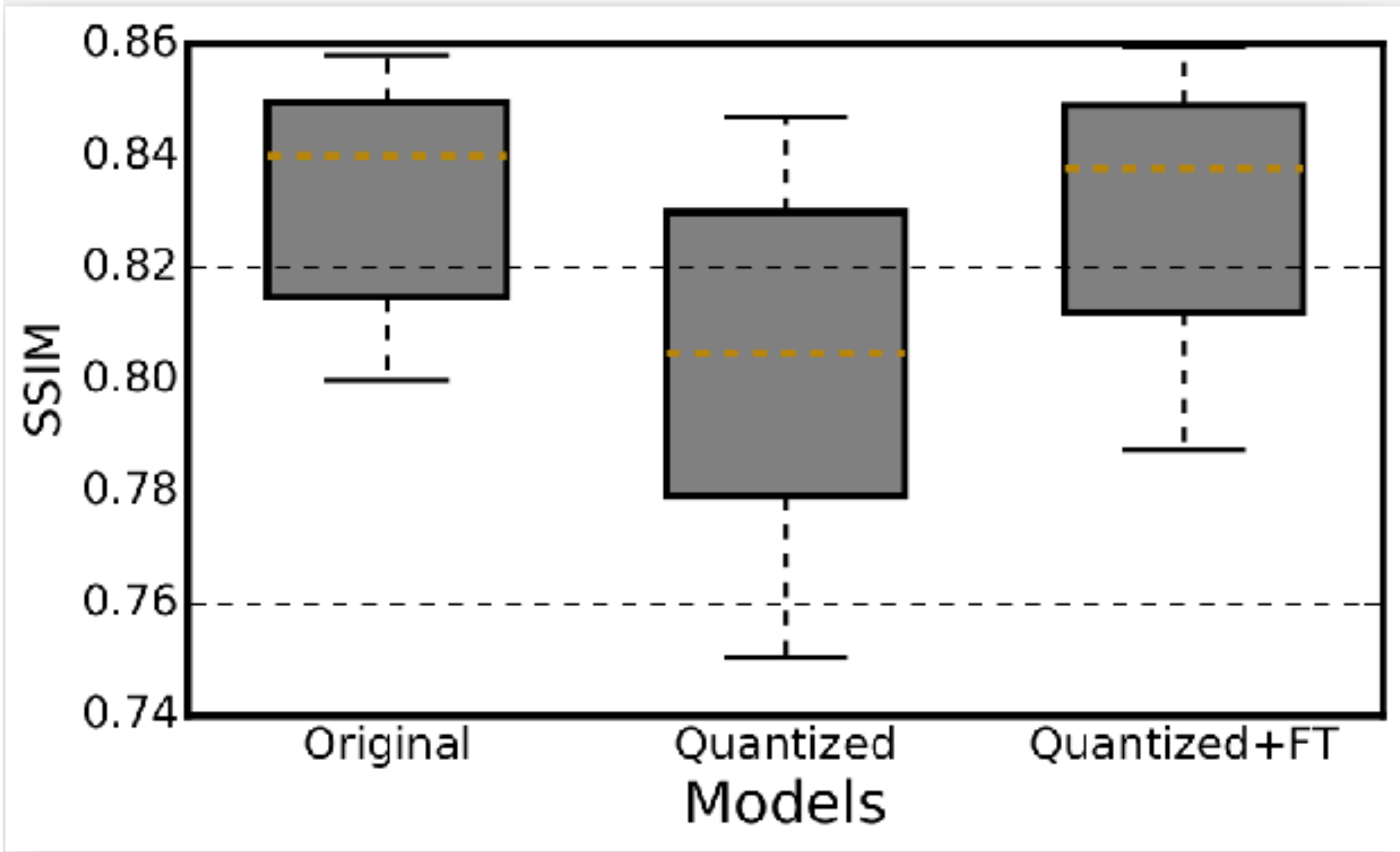
~700ms to denoise a 1k x 1k image

Make it usable - Continue Details









Thanks!

Want to try?

Open source at: https://github.com/ramsesproject/TomoGAN

python: Tensorflow and Keras based;
C++ : DNNL(MKL-DNN) based, good for CPU based e.g., KNL;
C++, CUDA: cuDNN and cuda based, good for NVIDIA GPU;
Pytorch: upon request