AI based denoise for scientific images

(Image to Image translation, e.g., inpaint, denoise, segmentation etc.)

Zhengchun Liu

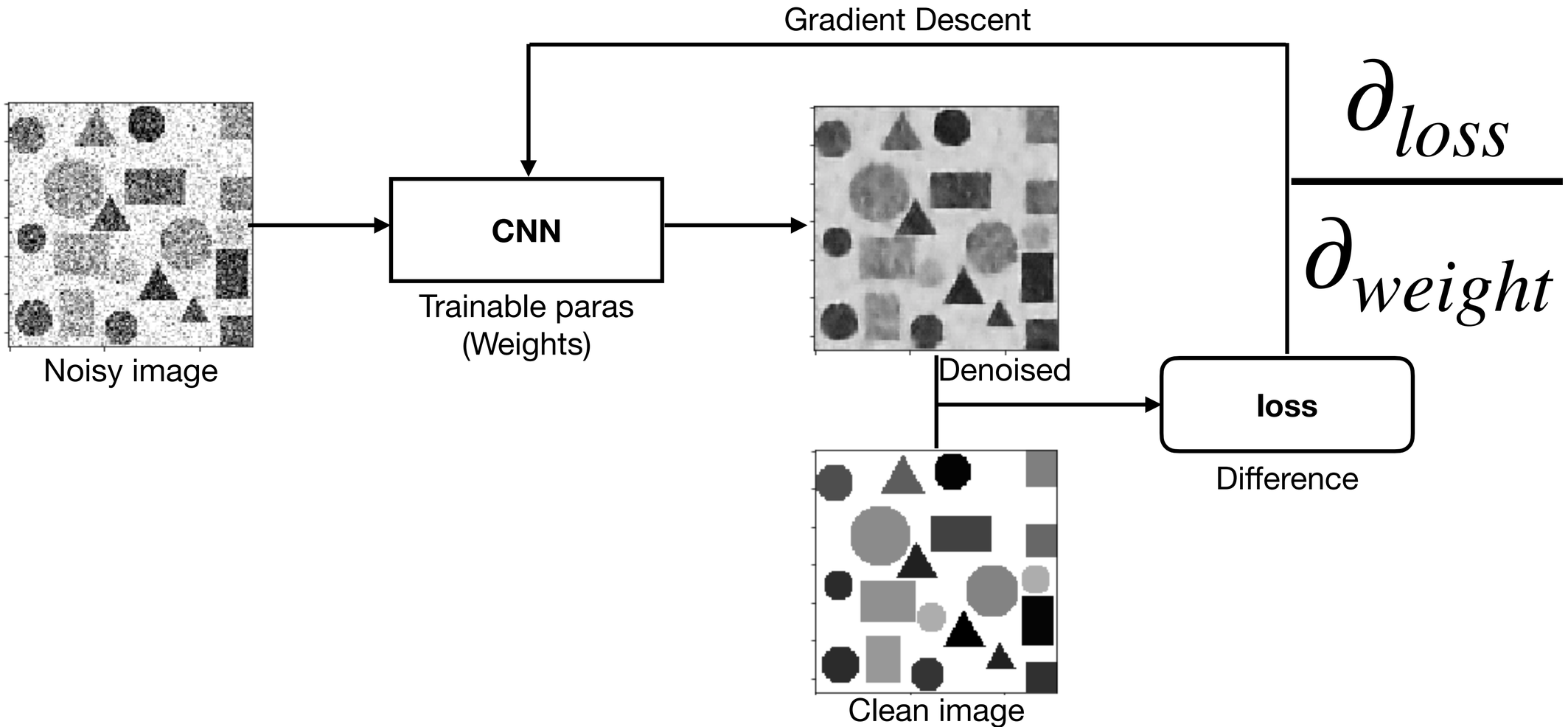
Assistant Computer Scientist at Data Science and Learning Division

Agenda

- Using CNN for denoising
- TomoGAN for denoising (e.g., artifact removal)
- [Advanced] Self-supervised learning to train denoise models without clean images
- [Advanced] Noise2Noise: Learning Image Restoration without Clean Data
- [Advanced] Segmentation

Fully Convolution Neural Network for denoising

Supervised learning

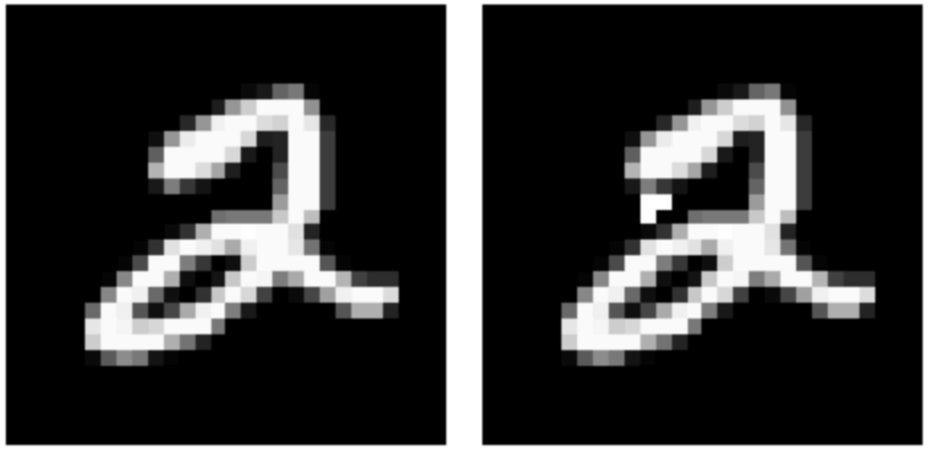


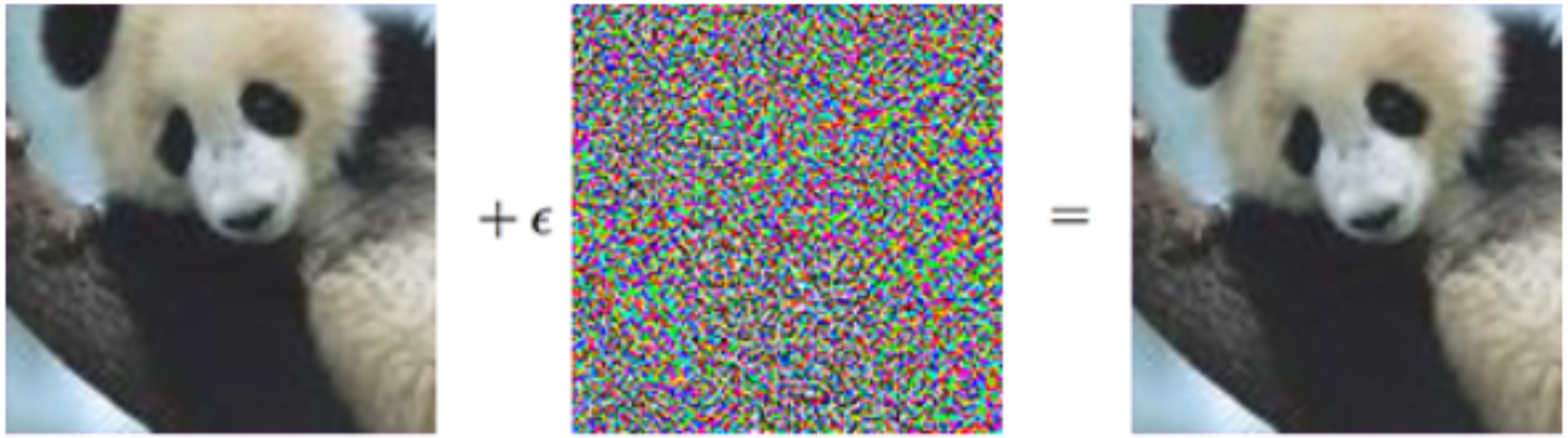
Switch to hands on

What is Generative Adversarial Network(GAN)

Intuition behind GAN

Original goal of GAN: Take as input training samples from some distribution and learn a model that represents that distribution.





"panda"

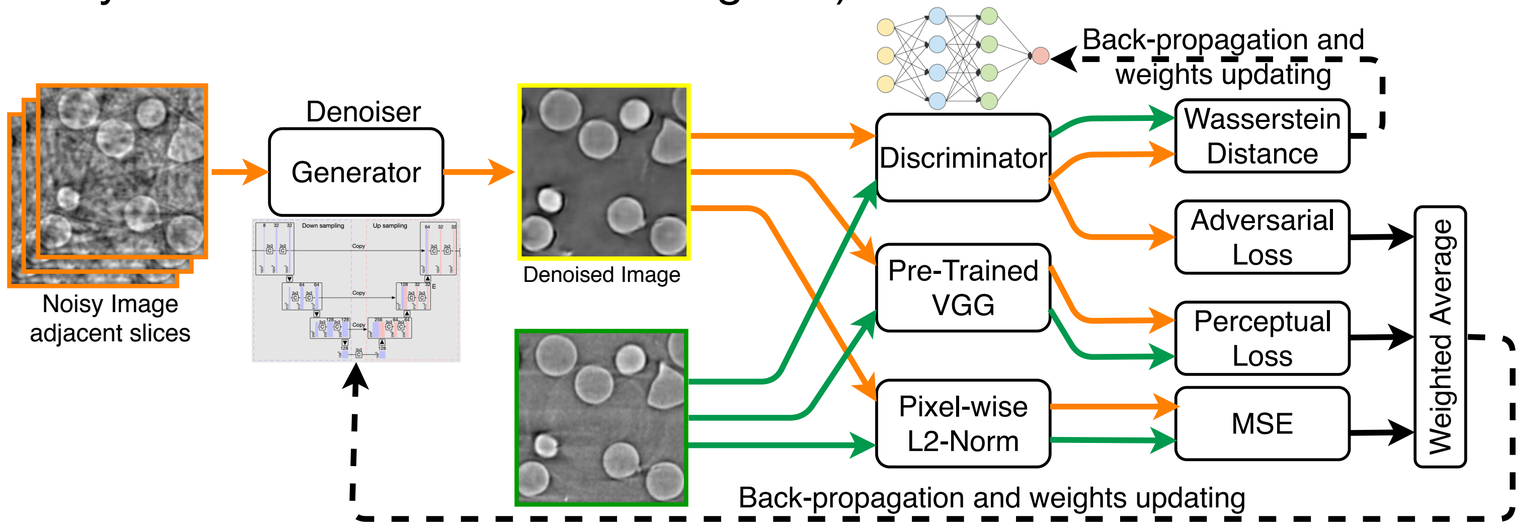
57.7% confidence

"gibbon"

99.3% confidence

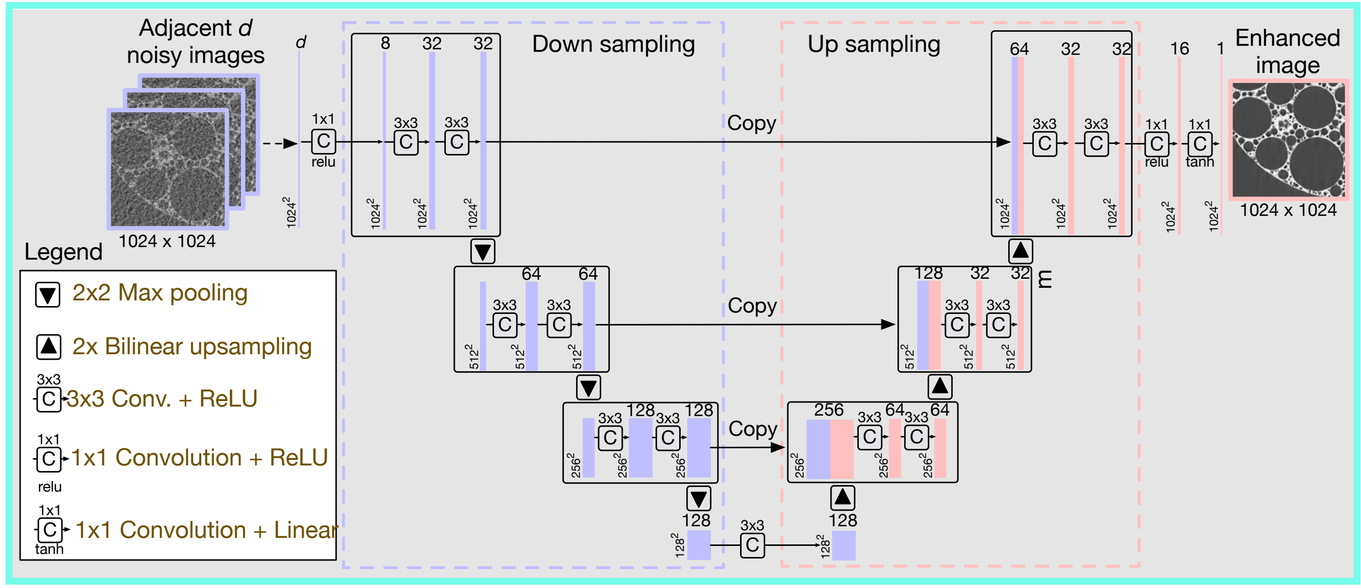
Method

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

$$L\left(\theta_D\right) = \frac{1}{m}\sum_{i=1}^{m}\left[D\left(G\left(I_{LD}^{i}\right)\right) - D\left(I_{ND}^{i}\right)\right]$$

Generator

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

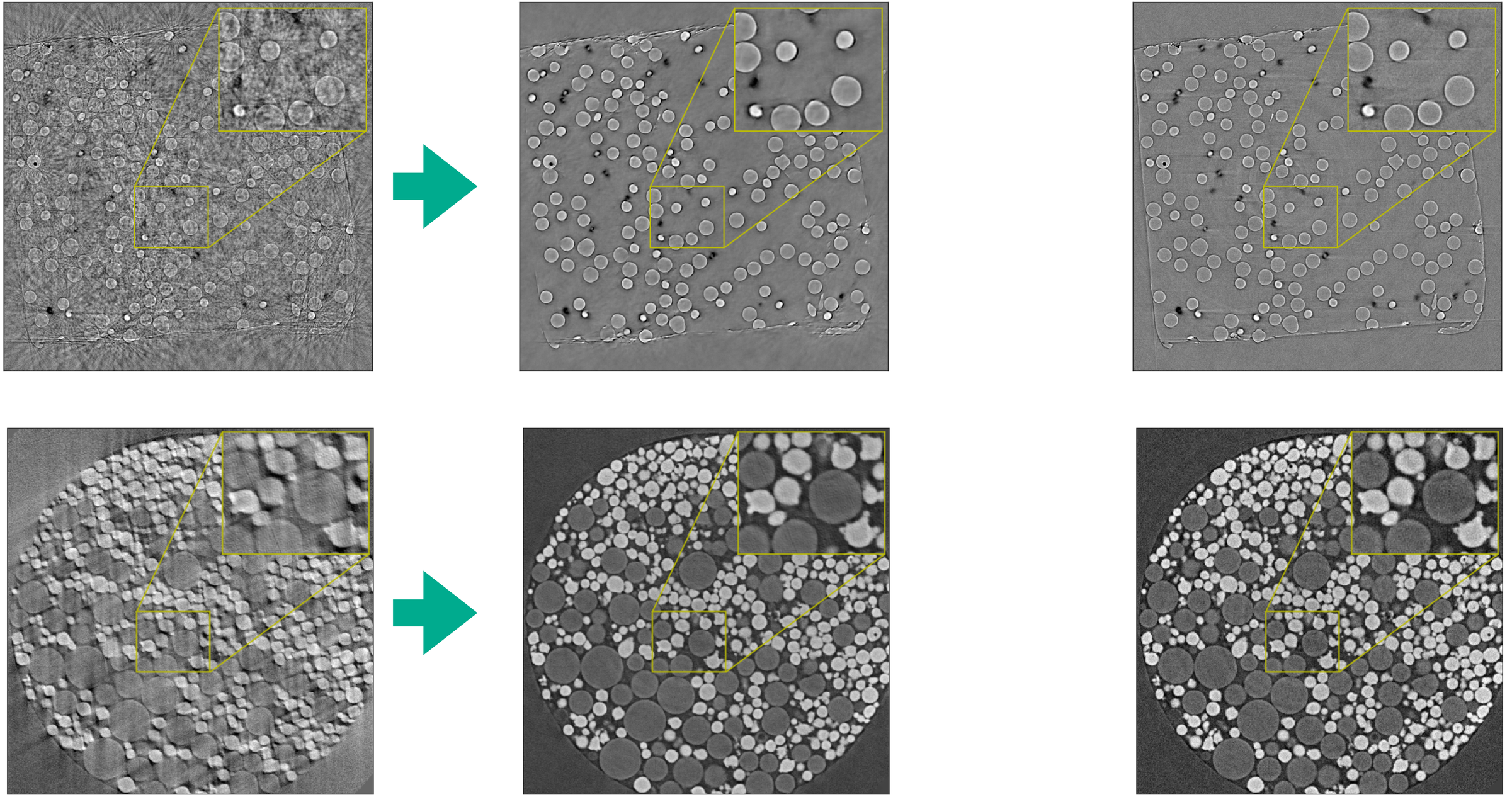
$$\ell^{G} = \lambda_g \ell_{adv} + \lambda_p \ell_{mse} + \lambda_v \ell_{vgg}$$

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

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

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

TomoGAN: noise and artifacts removal



Best practice (unnecessary in most cases)

- Unlike RGB camera images that only holds 0-255, experiment data may have outliers. Remove outliers (e.g., bad pixel) from your data as they may skew the MSE.
- Hyper-parameters, weights of losses: -ladv, -lmse, -lperc ideally all have similar scale after weighting
- Input patch size, 256 - 512

Switch to demo on training with a given dataset

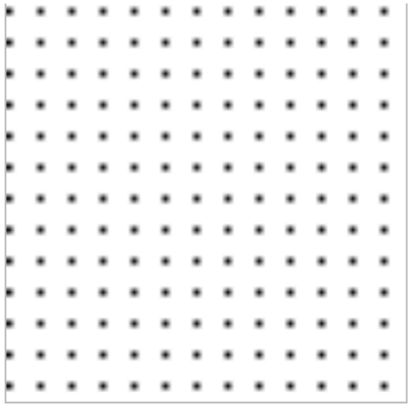
AI based denoise for scientific images

What if there is no clear image (ground truth) to train the model?

noise is element-wise statistically independent and mean-zero(not artifact),
and
the images exhibit some spatial correlation.

Noise2Noise

Noisy measurement

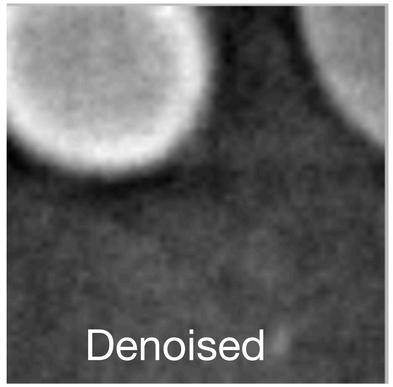


mask (W=1, B=0)

.X

CNN

Trainable paras (Weights)



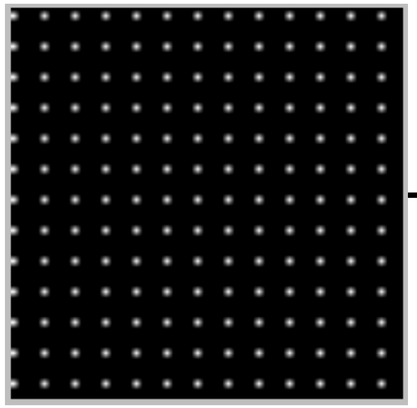
Denoised

Gradient Descent

$$\frac{\partial_{loss}}{\partial_{weight}}$$

$$\sum_{i,j} \left((N-D)M\right)^2$$

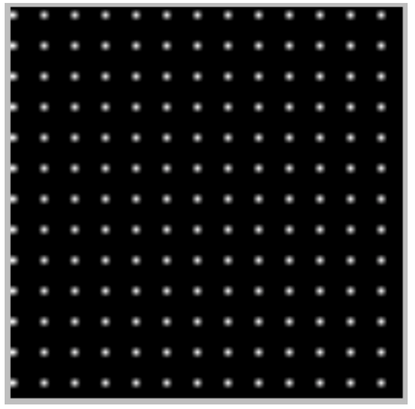
MSE Loss



Flipped mask (W=1, B=0)

.X

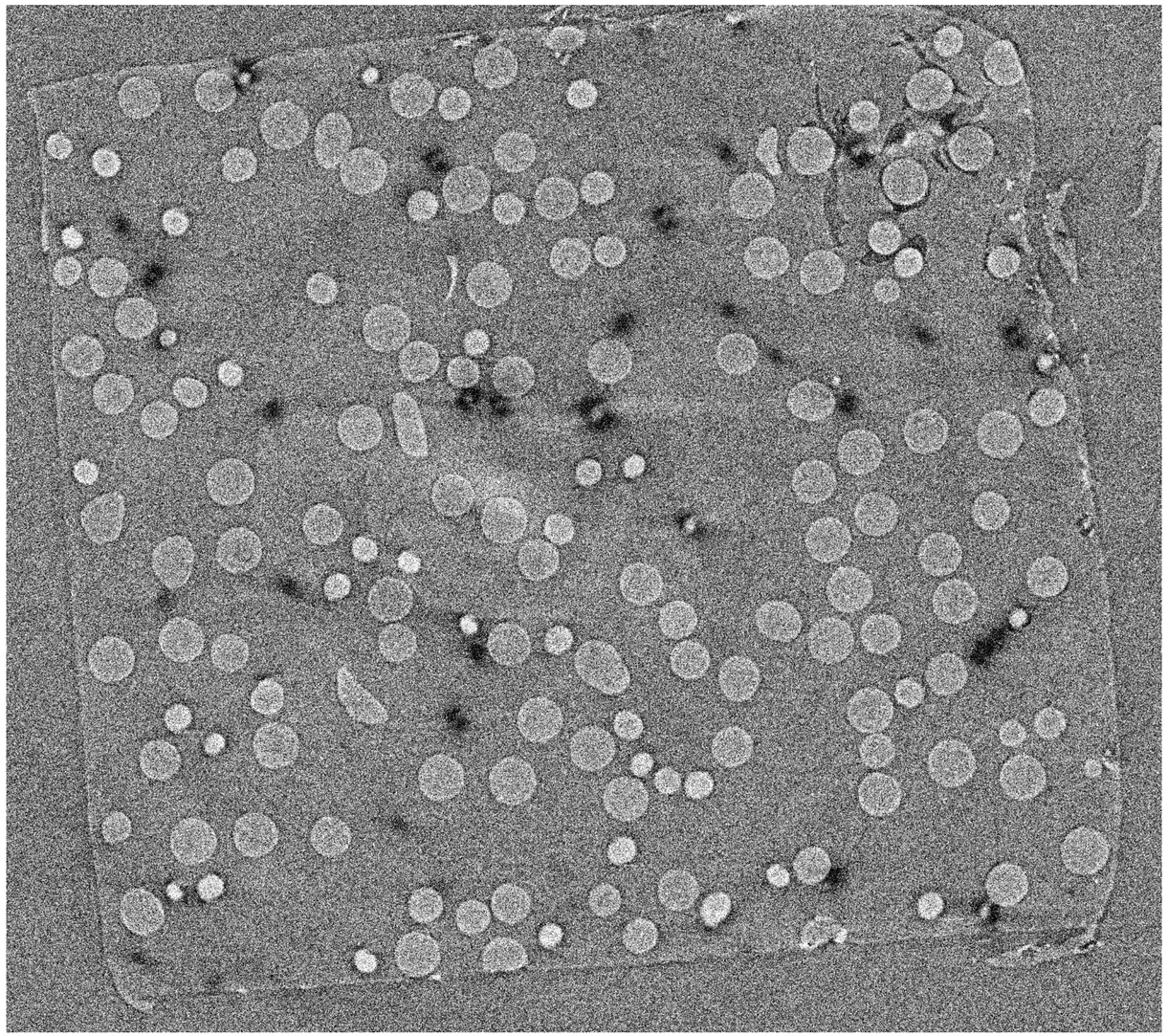
.X

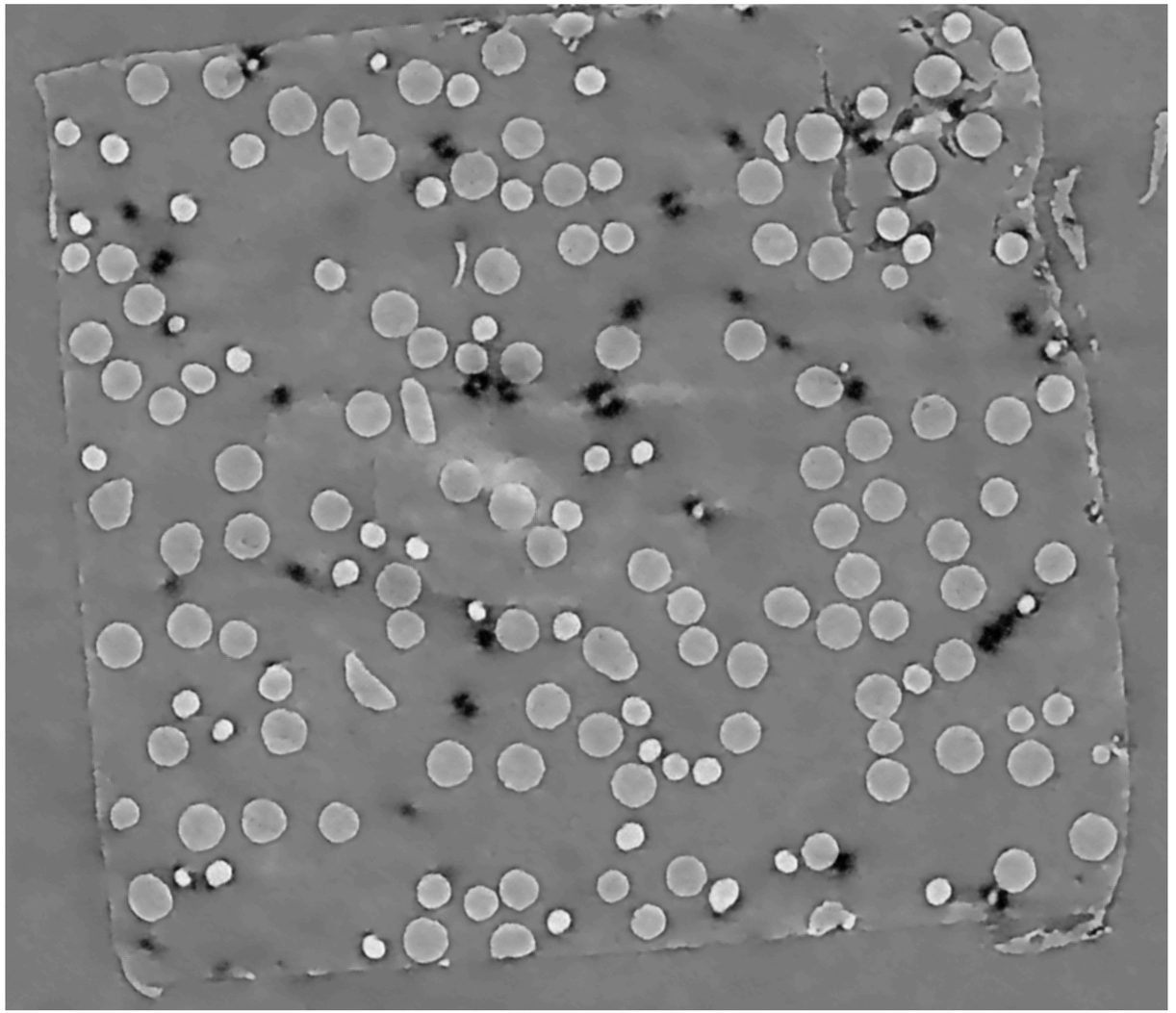


Noise2Self

Self-supervised learning.

noise is element-wise statistically independent and mean-zero, and that the clean images exhibit some spatial correlation.



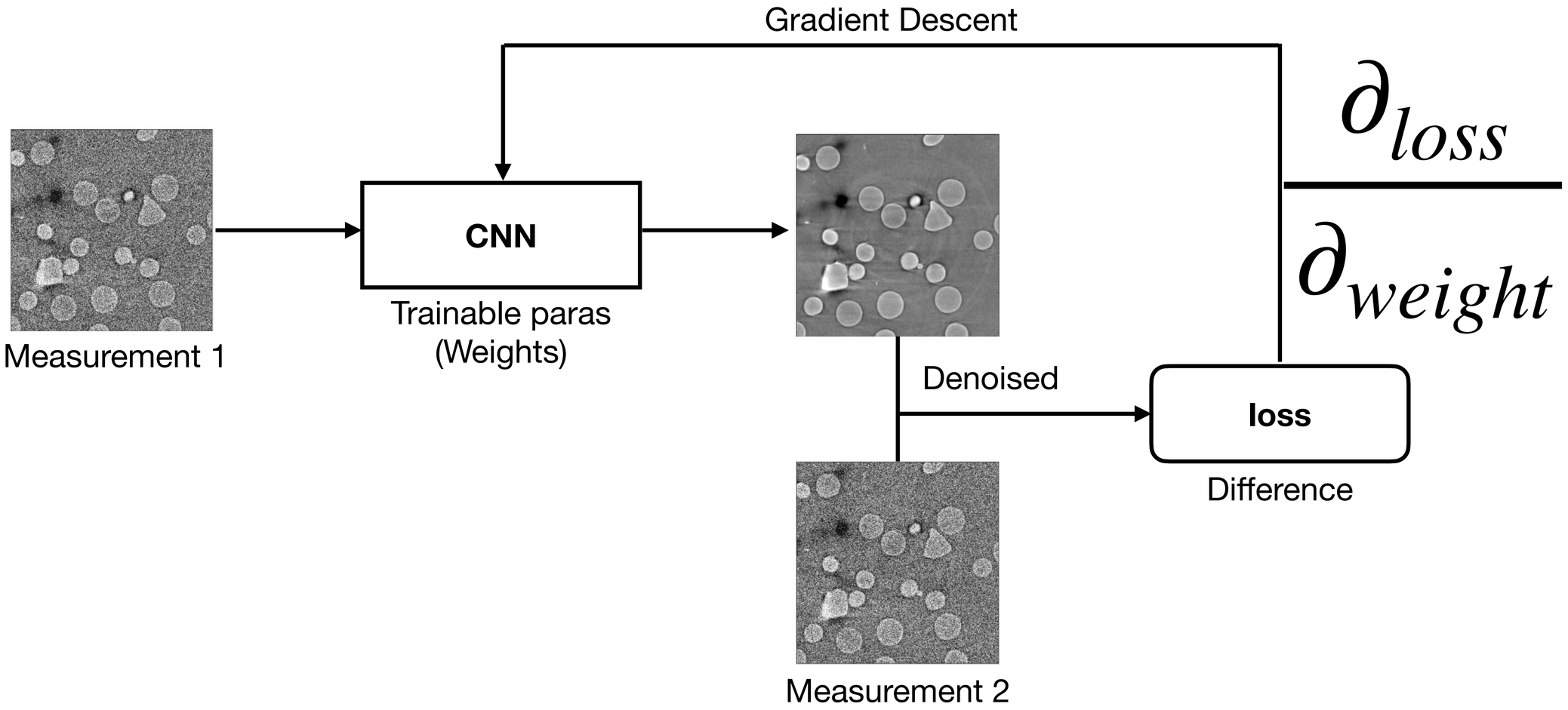


Reproduction of: Batson et al. arXiv:1901.11365

no clean images but
one can measure **independent** instances of the noise several times

Noise2Noise

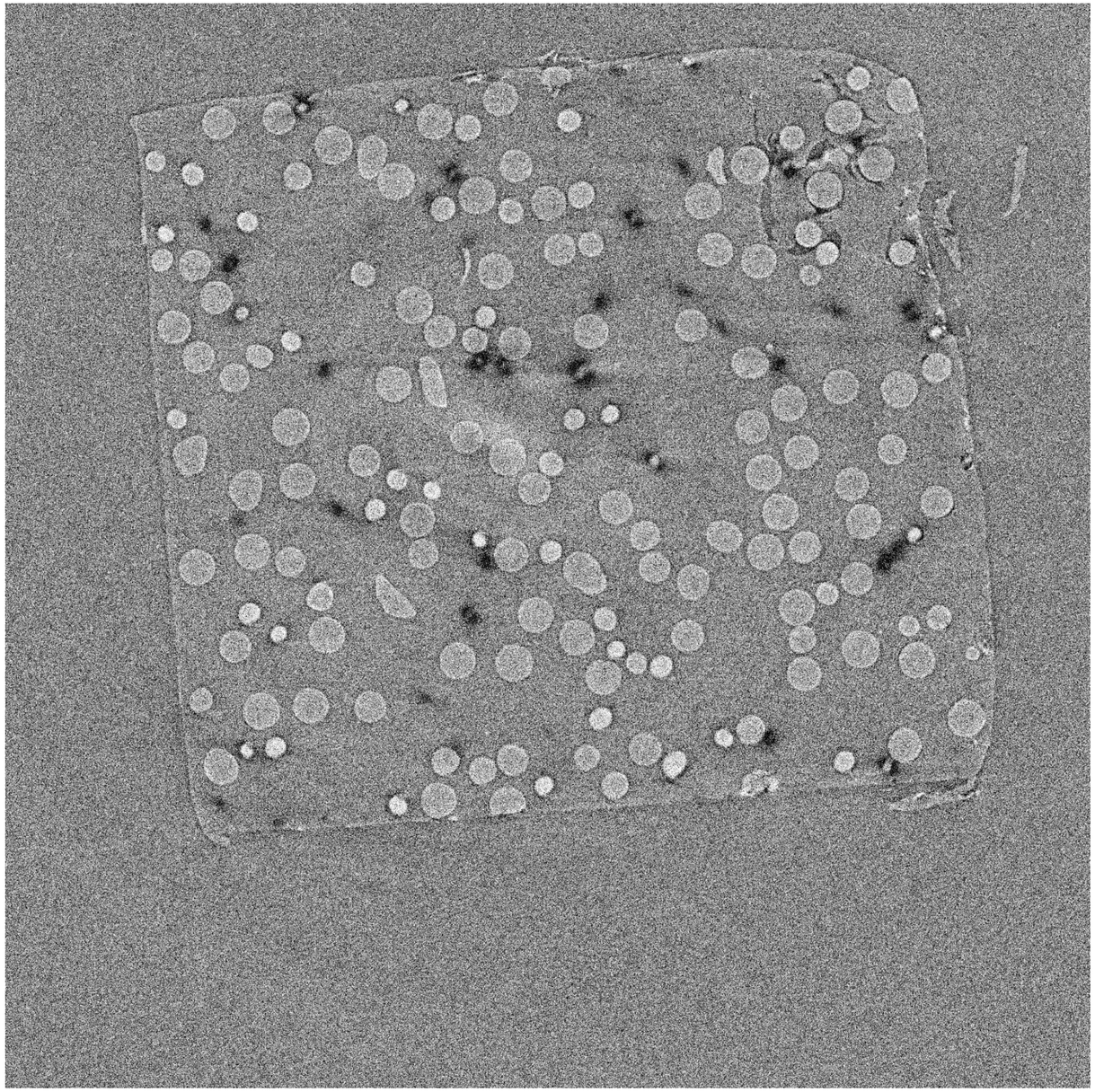
no clean images are needed, but one can measure **independent** instances of the noise for each image.

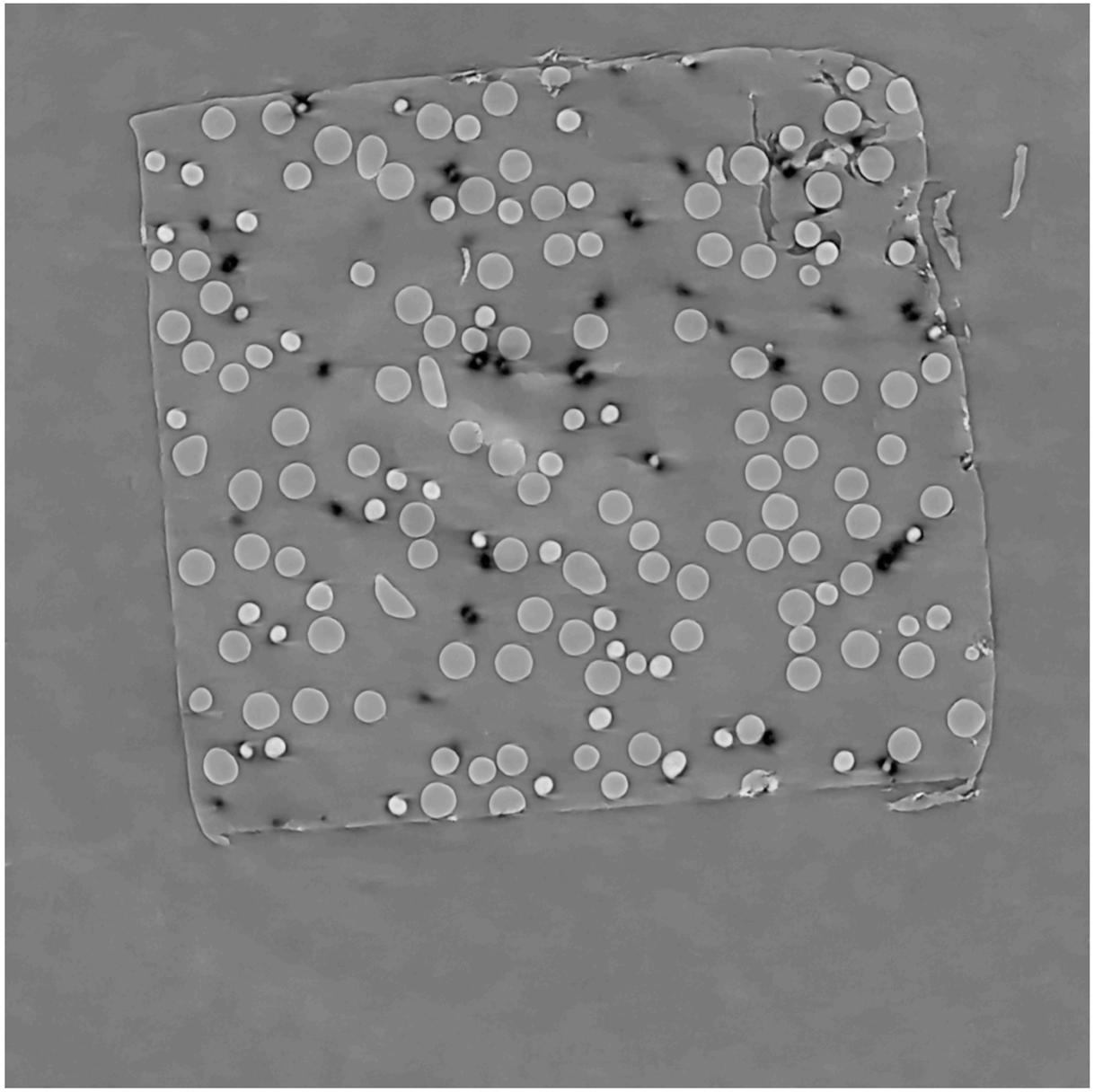


Reproduction of: Lehtinen1 et al. arXiv:1803.04189

Noise2Noise

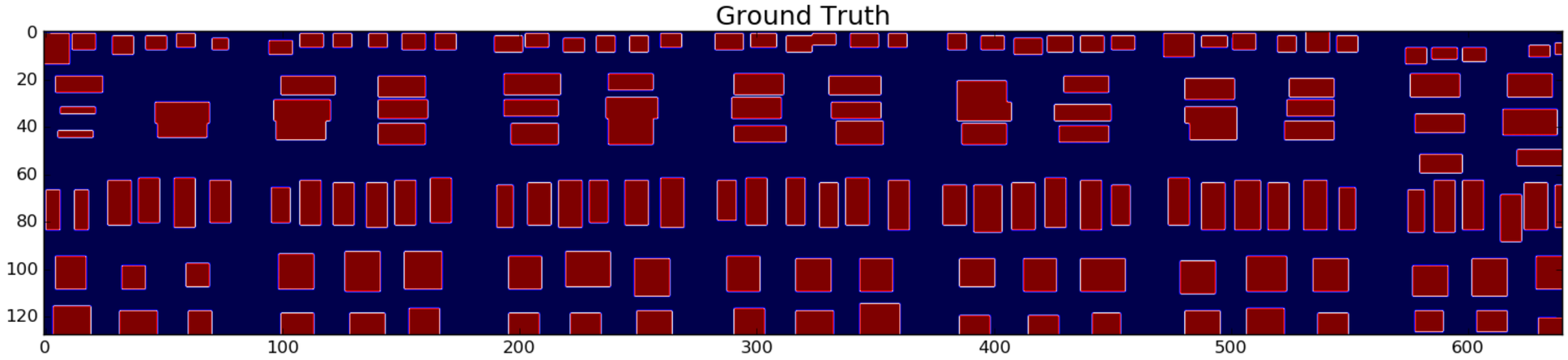
no clean images are needed, but one can measure independent instances of the noise for each image.

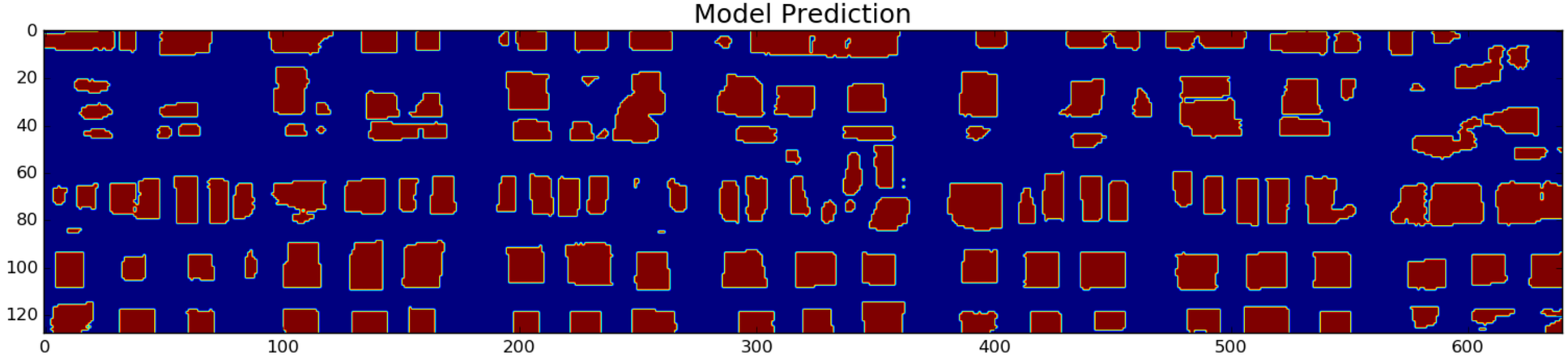


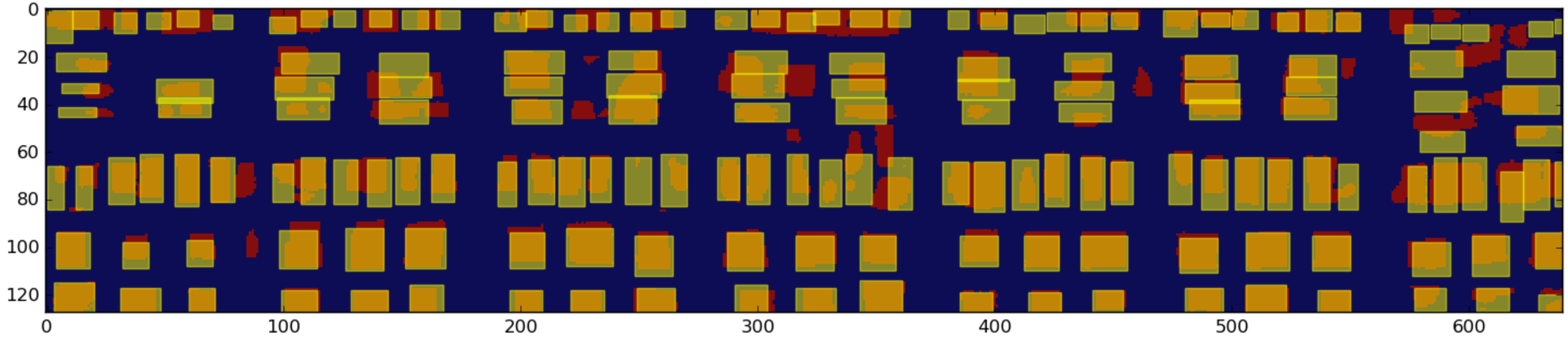


Reproduction of: Lehtinen1 et al. arXiv:1803.04189

Also works for Segmentation, only need to change loss function







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