LDCT Denoising With REDCNN & WGAN-GP

Hanbie Ryu
Dept. of A.I.
2024 Medisys Intern

Abstract

- LDCT (Low-dose CT) images require denoising systems to enhance image quality
- Used REDCNN (Residual Encoder-Decoder CNN) for primary model architecture
- Experimented with WGAN-GP (Wasserstein GAN + Gradient Penalty) with simple CNN discriminator for REDCNN generator model training procedure
- Expanded image dataset with variable doses for generalized denoising across different noise levels

1

REDCNN

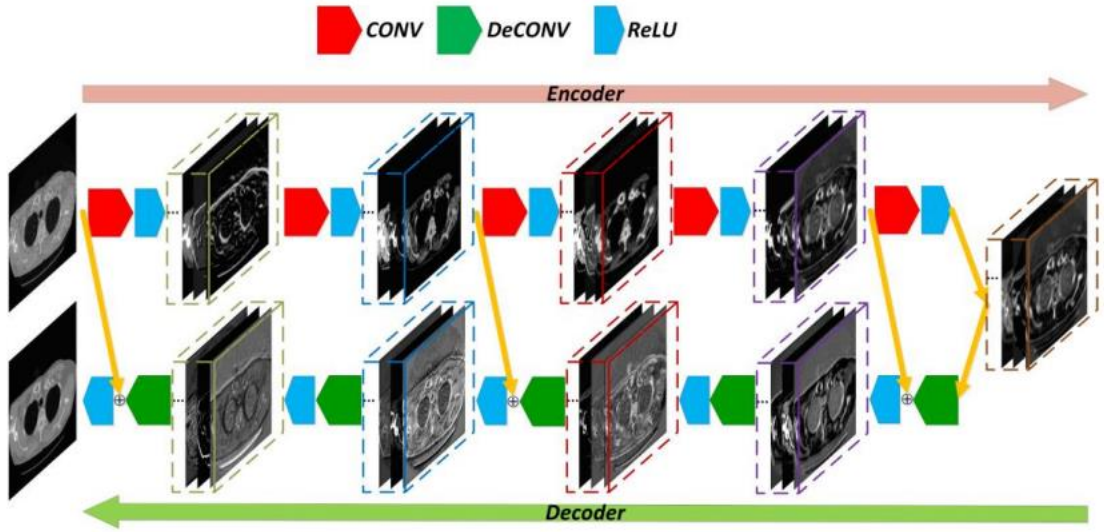


Fig.1. Overall architecture of our proposed RED-CNN network.

"Low-Dose CT with a Residual Encoder-Decoder Convolutional Neural Network (RED-CNN)"

Hu Chen, et al., 2017

YONSEI UNIVERSITY

Theory

- Residual connections for improved learning process
- Lack of max-pooling layers to minimize down-sampling
- Symmetrical convolutional & deconvolutional layers

Images from Hu Chen, et. al., 2017

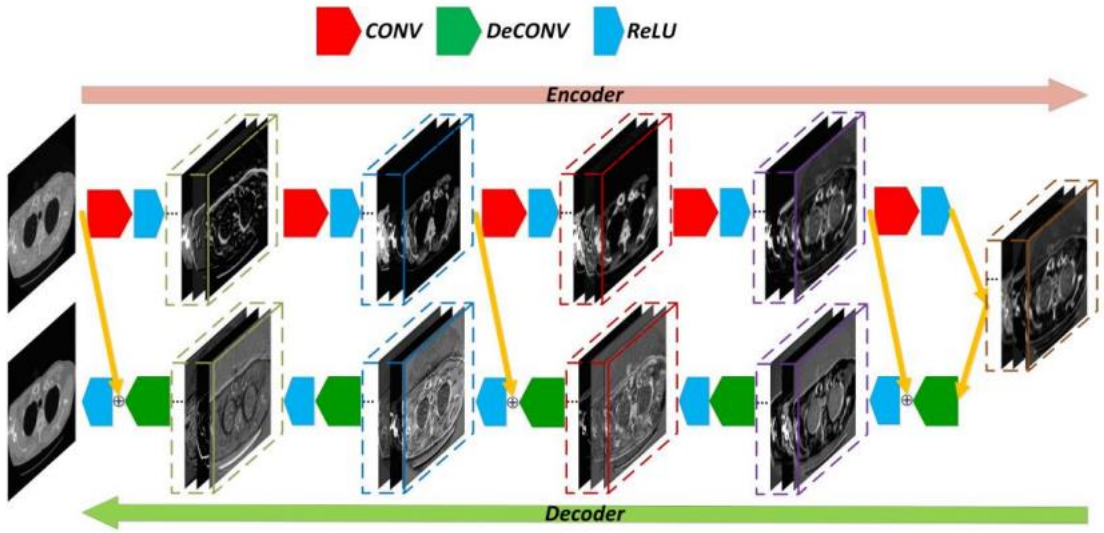


Fig.1. Overall architecture of our proposed RED-CNN network.

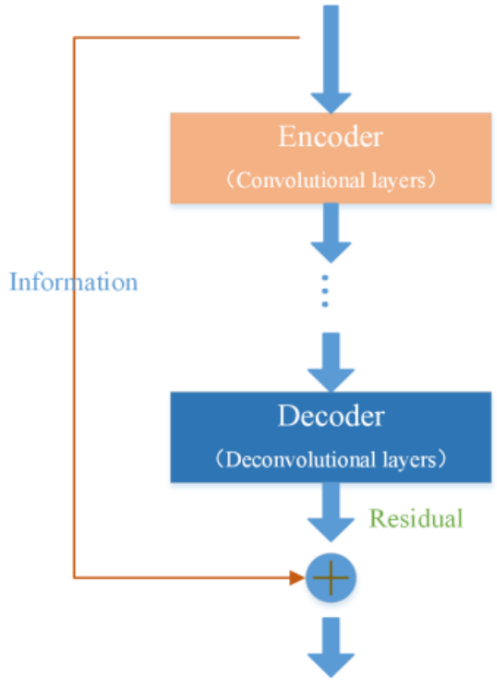


Fig.2. Shortcut in the residual compensation structure.

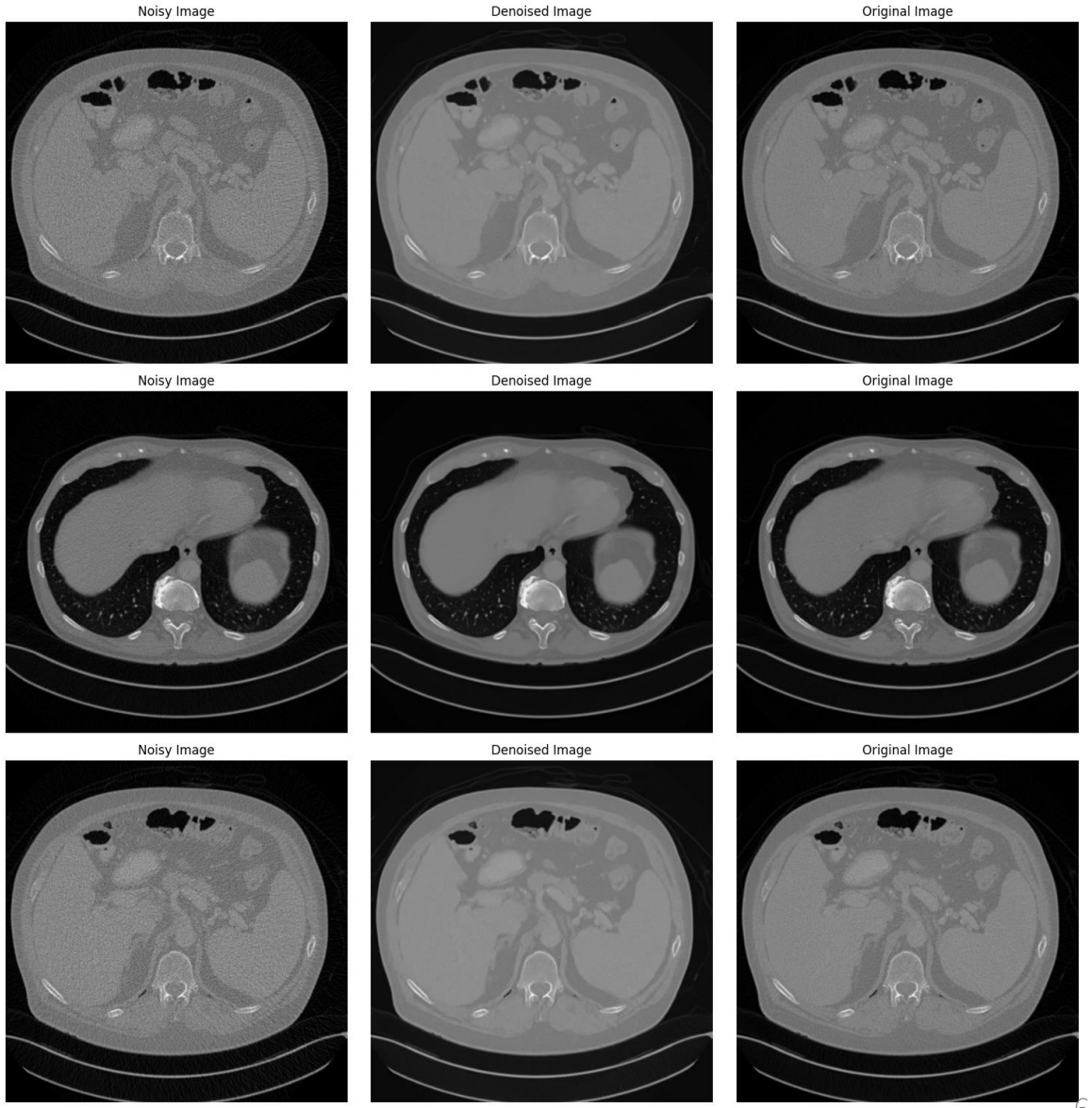
Simple NN training with REDCNN architecture

- 32 epochs*
- MSE loss
- Adam optimizer
- L2 regularization
- LR decay with patience=4, factor=0.5
- Early stopping with patience=6

* Further experiments revealed the network could have used less epochs (16), but early stopping has been implemented here so not a big issue with regularization in this case

REDCNN model analysis

- Rotation of random degrees for augmentation found to be ineffective, unless of 90-degree units
- Followed hyperparameters and model structure from the IEEE 2017 article, except for the patching stride of 16 (small changes seemed to have no particular significance)
- Excellent in terms of pure noise reduction



Limitations

- Built to minimize RMSE loss from target NDCT image

=> Results in over-smoothing (blurry outputs)

=> Possibility of blurring out crucial details in CT image

=> Recognized the need to have a regularising procedure to have the model output images similar to the NDCT images

2

WGAN-GP

$$L = \underbrace{\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} [D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} [D(\boldsymbol{x})]}_{\text{Original critic loss}} + \underbrace{\lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}} \left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2 \right]}_{\text{Our gradient penalty}}.$$

GAN — Wasserstein GAN & WGAN-GP | by Jonathan Hui | Medium

Theory

- GAN (Generative Adversarial Networks)
  - A training process that consists of a generator and a discriminator, where:
  - The generator takes the LDCT image as input (in this application)
  - The generator outputs a fake image
  - The discriminator scores the output image in range [0, 1] => classification
  - The discriminator and generator's loss is calculated from the discriminator's classification output (the generator also takes the regular MSE loss from the NDCT image, weighted between 0 ~ 1)

  ⇒The resulting images should look more like the NDCT images in theory

Theory

- WGAN-GP (Wasserstein GAN + Gradient penalty)
  - The discriminator outputs a ‘realness’ score with no fixed range, instead of a classification score between [0, 1]

  ⇒ Improves gradient feedback to the generator
  - Gradient penalty (GP) is applied instead of weight clipping (WGAN-GP) [1]

  ⇒ Improves optimization while maintaining Lipschitz constraint [2]

$$[1]: \lambda \cdot E_{\hat{x} \sim P_{\hat{x}}}[(\| \nabla_{\hat{x}} \ f(\hat{x}) \|_2 \ - 1)^2]$$

$$[2]: | f(x_1) - f(x_2) | \leq K \cdot \| x_1 - x_2 \|$$

```
g_loss = disc_weight * -torch.mean(fake_validity) + mse_weight * MSE(gen_images, images)

disc_weight = 0.95
mse_weight = 0.05
```

Discriminator model

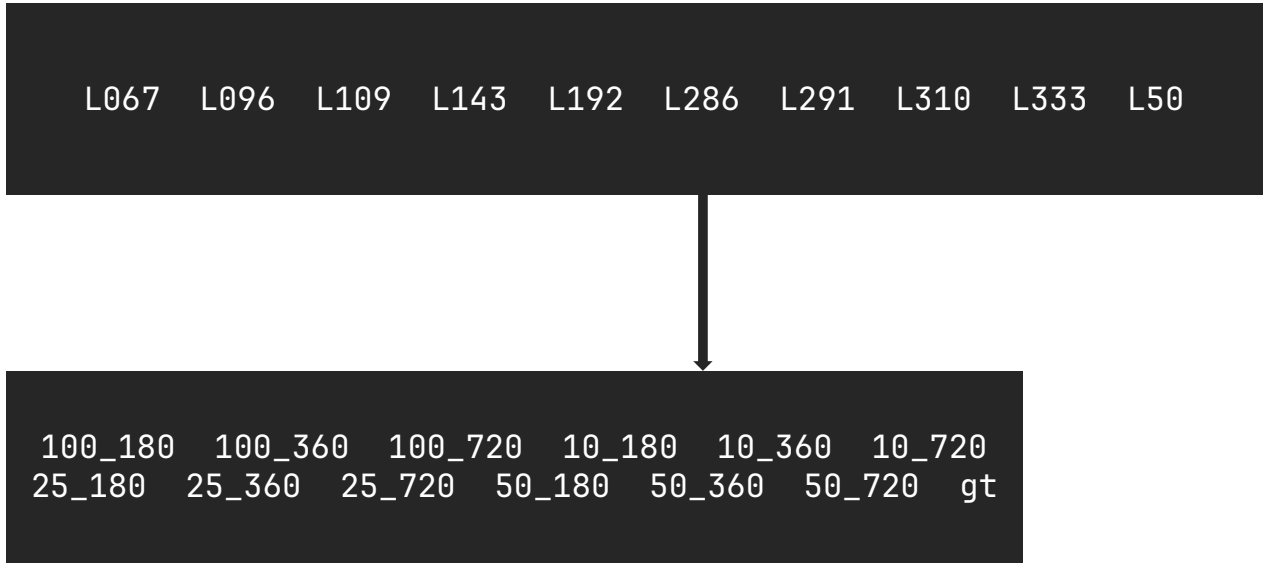
- Simple CNN architecture with down-sampling convolution layers
- Used LeakyReLU for better gradient flow

```
self.model = nn.Sequential(
            nn.Conv2d(1, 64, kernel_size=3, stride=1, padding=1),
            nn.LeakyReLU(0.2),
            nn.Conv2d(64, 128, kernel_size=5, stride=3, padding=1),
            nn.LeakyReLU(0.2),
            nn.Conv2d(128, 128, kernel_size=3, stride=1, padding=1),
            nn.LeakyReLU(0.2),
            nn.Conv2d(128, 128, kernel_size=5, stride=3, padding=1),
            nn.LeakyReLU(0.2),
            nn.Conv2d(128, 1, kernel_size=3, stride=1, padding=0)
        )
```

Variable dose dataset

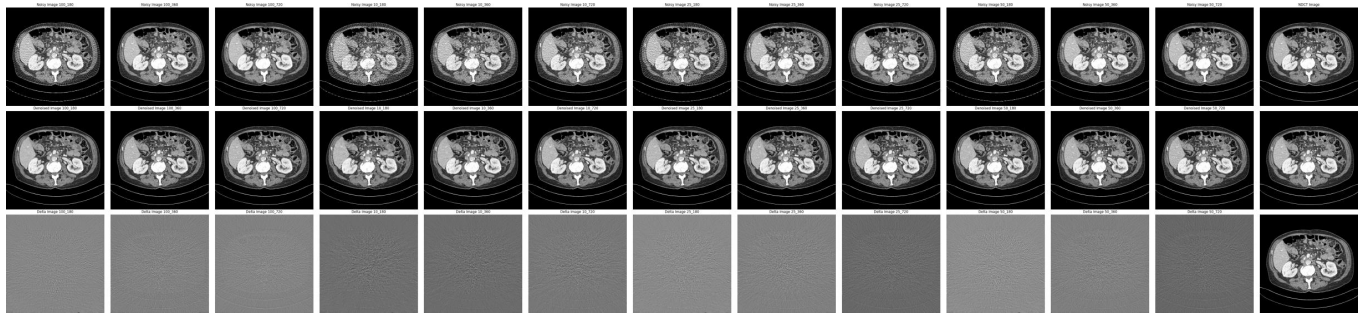
- Trained on dataset of variable noise levels

⇒Trained for generalized denoising



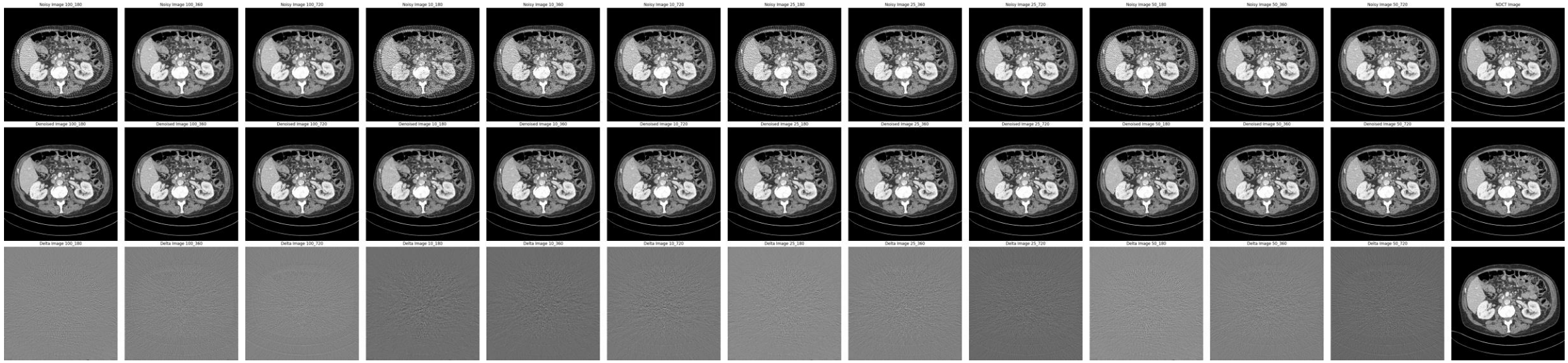
Dataset source: idk

3 Results



Effect of WGAN-GP training process

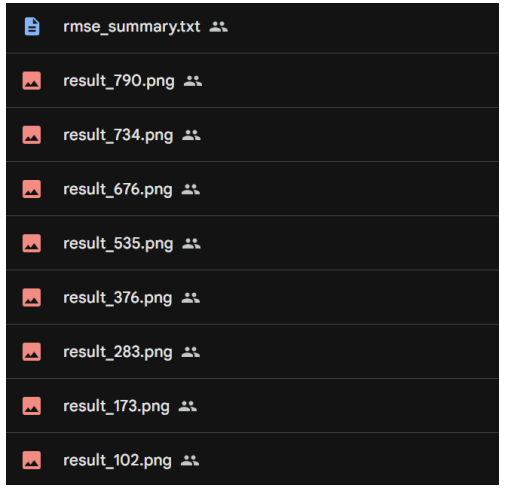
- Images generated by the WGAN-GP models appear to have noise-levels similar to the NDCT target images => retains image details, no over-smoothing
- Pure RMSE scores higher (lower loss) in regular REDCNN models with no GAN training
- Some artifacts (‘streaks’) remain in very noisy image inputs



Final RMSE scores

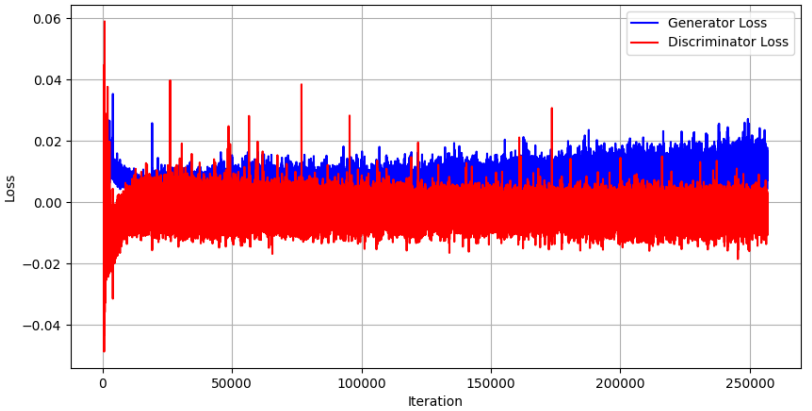
Dose	Average RMSE
100_180	24.04
100_360	20.18
100_720	17.67
10_180	30.61
10_360	25.66
10_720	23.25
25_180	26.77
25_360	22.86
25_720	20.95
50_180	25.18
50_360	21.34
50_720	19.44
mean:	23.1625

Denoised results for 8 randomly selected images
Drive Link here




```
[Config.yaml]
dataset:
  augment: true
  batch_size: 64
  patch_size: 64
  stride: 32
training:
  betas:
  - 0.9
  - 0.999
  critic_iterations: 5
  l2_regularization: 2e-5
  lambda_gp: 10.0
  learning_rate: 1e-4
  lr_decay_factor: 0.5
  lr_decay_patience: 4
  num_epochs: 16
  patience: 6
  sample_interval: 5
weights:
  disc_weight: 0.05
  mse_weight: 0.95
```

Generator and discriminator loss
(from iteration=500)



Moving forward

- Model variants
  - U-net, U-former etc
- GAN variants
  - ESRGAN
- Restormer?
- To be decided
- 좋은 자원을 제공해주시고, 지도해 주셔서 감사합니다. 앞으로도 최선을 다하겠습니다.