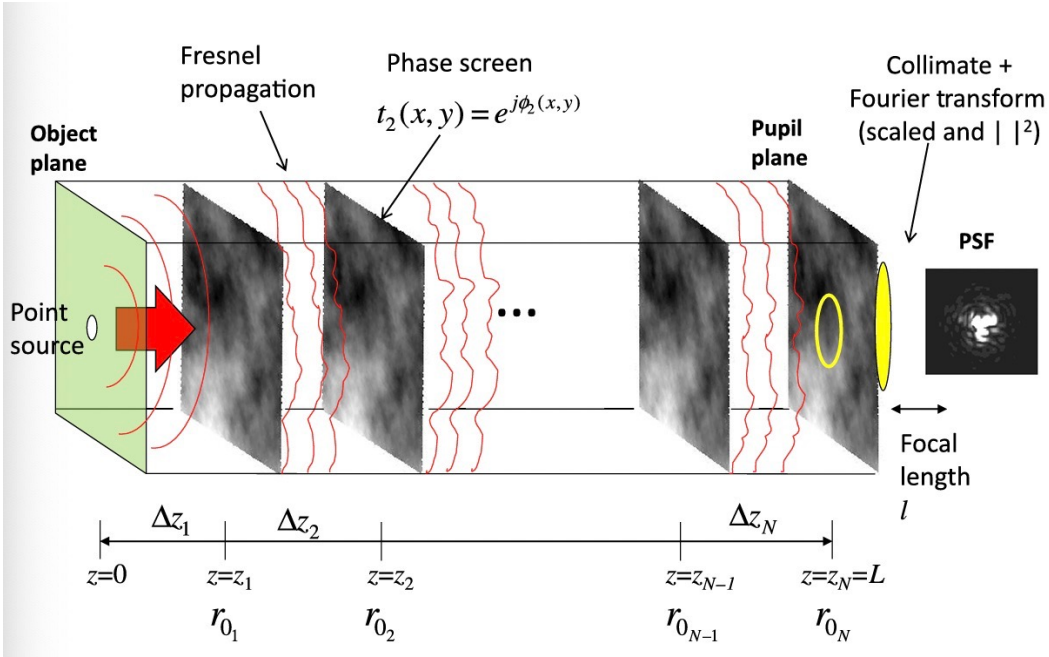
Anisoplanatic turbulence effect

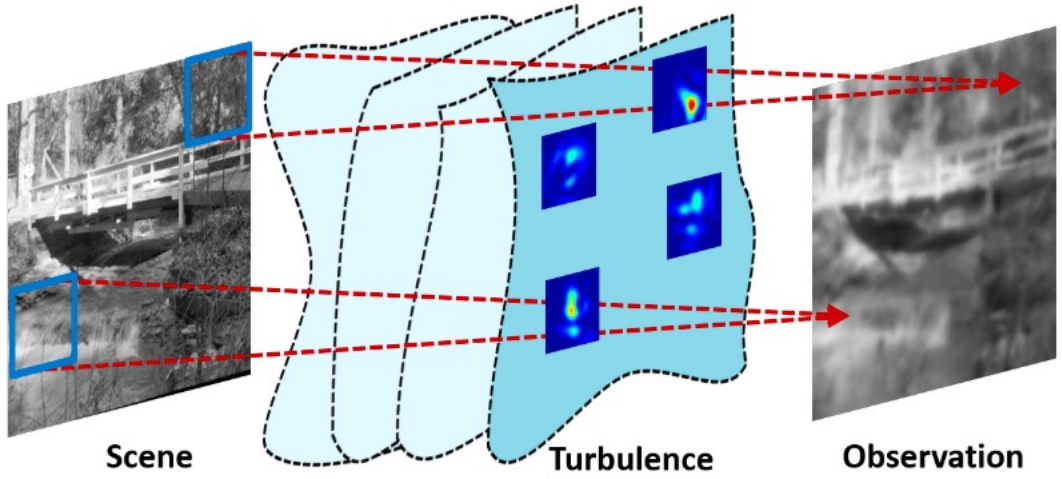
Atmospheric turbulence causes long-range imaging systems to capture blurry and distorted measurements. The wave front of a point source should be isoplanatic, or ”flat” in ideal condition, but since the medium is not uniform and static, the wave will be anisoplanatic.
The perturbation introduced by atmosphere could be seen as a spatial-temporal random field. When it is not strong and we have enough exposure time to take pictures, the turbulence is not obvious, but if given long-range, short exposure or hot air, the effect could be strong enough.

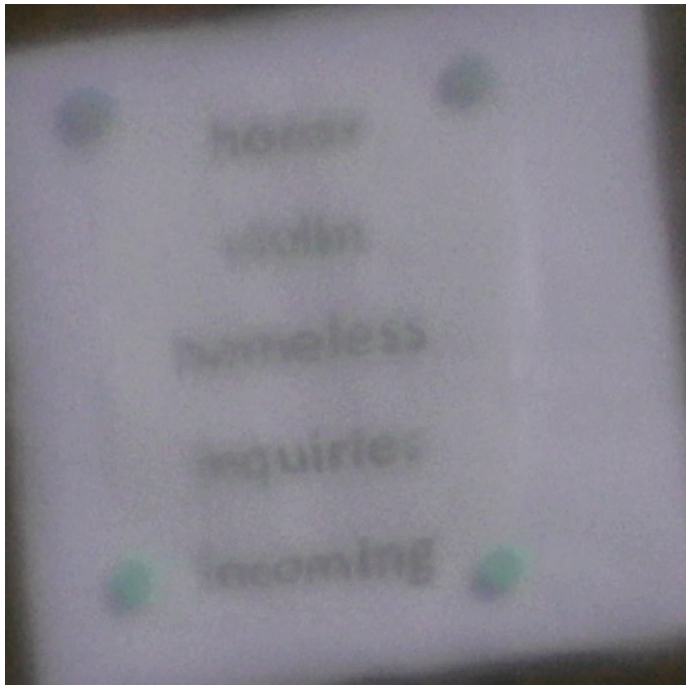


Anisoplanatic turbulence effect

We can model blurry effect on images using convolution between the blurry kernel with the clear image, the kernel is a discretized point spread function (PSF). But unlike defocus blur or motion blur, the anisoplanatic turbulence effect is modeled by spatial variant PSFs

$$\boldsymbol{y}_i(u,v) = \boldsymbol{h}_i(u,v) * \boldsymbol{x} + \boldsymbol{\epsilon}_i$$





300 meters distance

1300 mm focal length camera

1ms exposure time

Statistic model

We have a group of observations $(y_1,\ y_2, \dots ,\ y_n)$ from a real, clean scene x, neglecting the noise term, then the forward model is

$$\mathrm{p}(y_i|x,\theta_i) = H(\theta_i) * x$$

Where θ is a spatial Gaussian random noise and H could map it to a random field of blurry kernels (neglecting temporal correlation here), inferencing $\mathrm{p}(\hat{x}|\{y_i\})$ is now an inverse problem that could be solved by GAN.

Luckily, I have a physical-based, accurate, differentiable forward simulator developed by our group. The generator in the GAN should try to generate a reconstruction $\hat{x} = R(\{y_j\})$, then using the $\hat{x}$ and deterministic simulation, it generates another set of samples $\{\hat{y}_i\}$. It contains two modules:

1. A reconstruction network which infers the clean image from input frames: $\hat{x} = R(\{y_j\})$
2. A forward simulator models $\mathrm{q}(\hat{y}_i|\hat{x},\theta_i) = H(\theta_i) * \hat{x}$

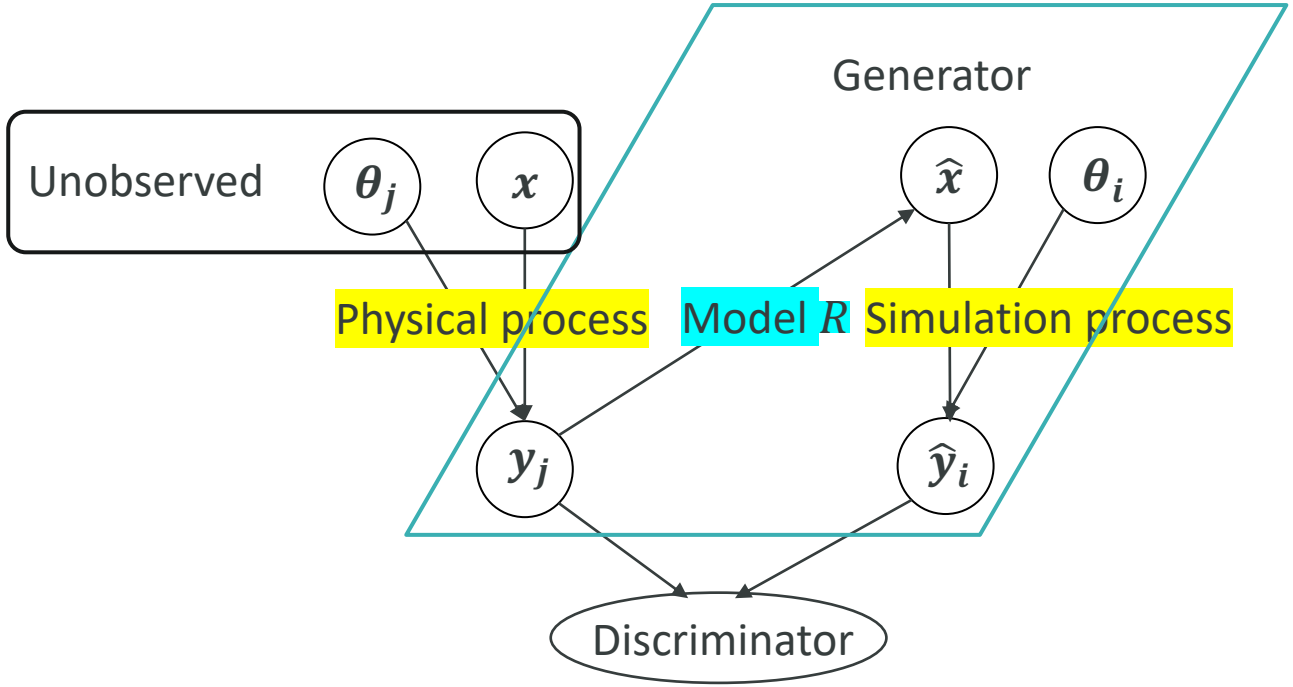
The discriminator should distinguish $q(\hat{y}|\hat{x},\theta)$ from $\mathrm{p}(y|x,\theta)$

Statistic model

The generator contains two modules:

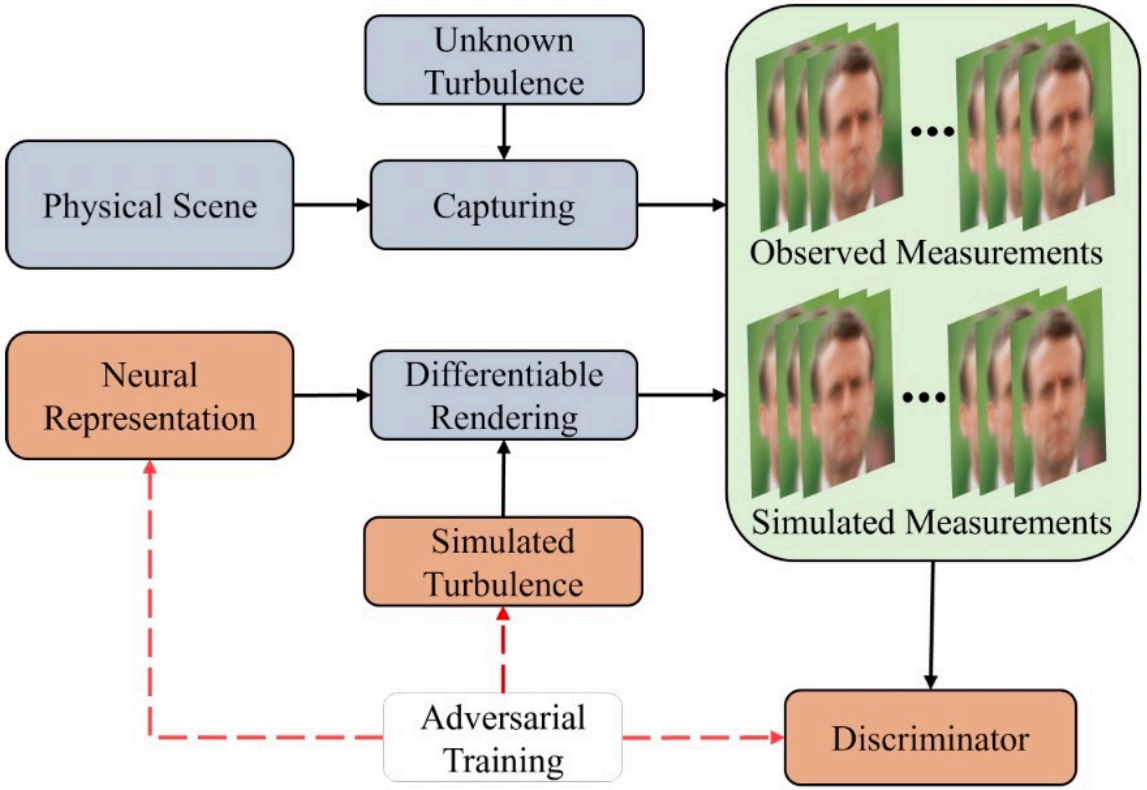
1. A reconstruction network **R** which infers the clean image from input frames: $\hat{x} = R(\{y_j\})$
2. A forward simulator models $\mathrm{q}(\hat{y}_i|\hat{x}, \theta_i) = H(\theta_i) * \hat{x}$

The discriminator should distinguish $q(\hat{y}|\hat{x}, \theta)$ from $\mathrm{p}(y|x, \theta)$, where two θ are different instances.



Method setting

The scheme in the paper is:



But here to reduce complexity, this paper and I just used simulated images as the real data

The neural representation is a model to generate $\hat{y} = R(\{x_i\})$ by a simple 3D Unet

The paper didn't mention their network, but they Used a Unet to extract an initial latent vector z and set

$$\hat{x} = R(\hat{z})$$

$\{y_i\}$ is a set of **100 simulated images**, in the paper they used 20~200k images to train.

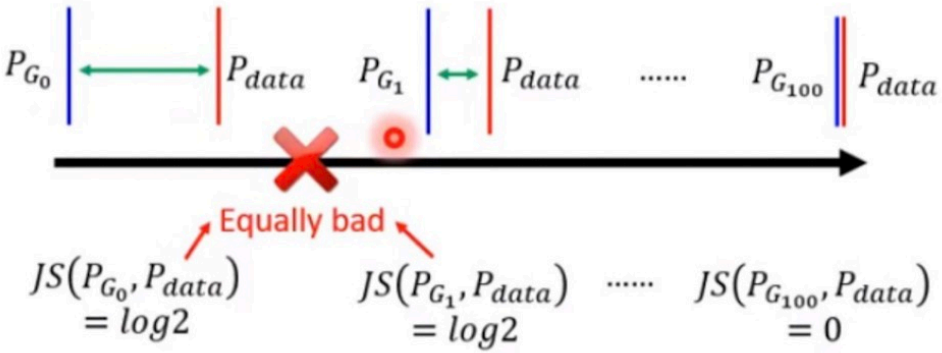
The discriminator I used has 20 layers of convnet to learn a 9x4x4 feature map for every image

Wasserstein GAN and gradient penalty

JS-divergence makes GAN training hard: if two distributions has no overlapping, the gradient of JS-divergence is always 0

$$\text{JSD}(P \parallel Q) = \frac{1}{2}D(P \parallel M) + \frac{1}{2}D(Q \parallel M)$$

where $M = \frac{1}{2}(P + Q)$

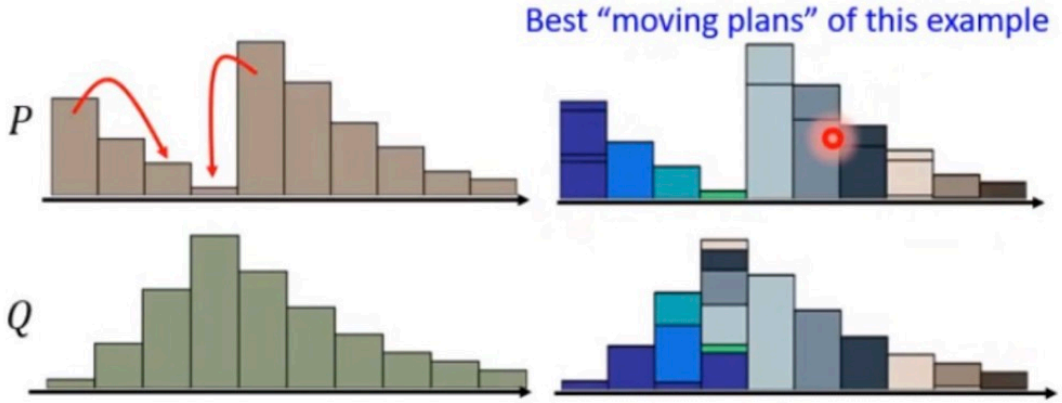


Wasserstein GAN: change JS-divergence to Wasserstein distance

- The *Earth-Mover* (EM) distance or Wasserstein-1

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\|x - y\| \right] , \quad (1)$$

where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$. Intuitively, $\gamma(x, y)$ indicates how much "mass" must be transported from x to y in order to transform the distributions $\mathbb{P}_r$ into the distribution $\mathbb{P}_g$. The EM distance then is the "cost" of the optimal transport plan.

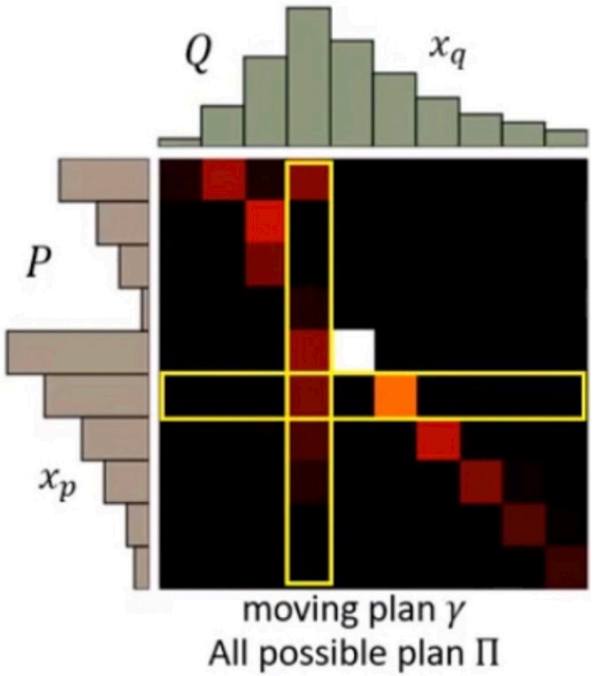


Wasserstein GAN and gradient penalty

Optimize:

$$W[p_{data}(x), p_g(x)] = \inf_{\gamma \in \Pi[p_{data}(x), p_g(x)]} \int \int \gamma(x_{data}, x_g) d(x_{data}, x_g) dx_{data} dx_g$$

Given: $\int \gamma(x_{data}, x_g) dx_{data} = p(x_g)$ $\int \gamma(x_{data}, x_g) dx_g = p(x_{data})$



$$\Gamma = \begin{bmatrix} \gamma(x_{data1}, x_{g1}) \\ \gamma(x_{data1}, x_{g2}) \\ \cdots \\ \gamma(x_{data2}, x_{g1}) \\ \gamma(x_{data2}, x_{g2}) \\ \cdots \\ \gamma(x_{data3}, x_{g1}) \\ \gamma(x_{data3}, x_{g2}) \\ \cdots \\ \cdots \end{bmatrix} \quad D = \begin{bmatrix} d(x_{data1}, x_{g1}) \\ d(x_{data1}, x_{g2}) \\ \cdots \\ d(x_{data2}, x_{g1}) \\ d(x_{data2}, x_{g2}) \\ \cdots \\ d(x_{data3}, x_{g1}) \\ d(x_{data3}, x_{g2}) \\ \cdots \\ \cdots \end{bmatrix}$$

Discretize our goal:

$$\min_{\Gamma} \{ < \Gamma, D > | \ A\Gamma = b, \Gamma \geq 0 \}$$

$$A = \left[\begin{array}{ccc|ccc|c|ccc|c} 1 & 1 & \cdots & 0 & 0 & \cdots & \cdots & 0 & 0 & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & 1 & \cdots & \cdots & 0 & 0 & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 1 & 1 & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 0 & \cdots & 1 & 0 & \cdots & \cdots & 1 & 0 & \cdots & \cdots \\ 0 & 1 & \cdots & 0 & 1 & \cdots & \cdots & 0 & 1 & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 0 & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{array}\right]$$

$$b = \begin{bmatrix} p_{data}(x_{data1}) \\ p_{data}(x_{data2}) \\ \cdots \\ p_g(x_{g1}) \\ p_g(x_{g1}) \\ \cdots \\ \cdots \end{bmatrix}$$

Wasserstein GAN and gradient penalty

Discretize our goal:

$$\min_{\Gamma}\{< \Gamma, D >|\ A\Gamma = b, \Gamma \geq 0\}$$

This is a linear programming problem, by Farkas' lemma, there exists a strong duality:

$$\min_{\Gamma}\{< \Gamma, D >|\ A\Gamma = b, \Gamma \geq 0\} = \max_{F}\{< b, F >|\ A^{\mathrm{T}}F \leq D\}$$

Where F is a Lagrangian multiplier

$$F = \begin{bmatrix} f_1(x_{data1}) \\ f_1(x_{data2}) \\ \cdots \\ f_2(x_{g1}) \\ f_2(x_{g1}) \\ \cdots \\ \cdots \end{bmatrix}$$

This says for all x:

$$f_1(x_{datai}) + f_2(x_{gj}) \leq d(x_{datai}, x_{gj})$$

If $x_{datai} = x_{gj} = x$:

$$f_1(x) + f_2(x) \leq d(x, x) = 0$$

$$f_2(x) \leq -f_1(x)$$

Wasserstein GAN and gradient penalty

Finally the distance becomes:

$$
\begin{aligned}
W[p_{data}(x), p_g(x)] &= \min_{\Gamma}\{< \Gamma, D > |\ A\Gamma = b, \Gamma \geq 0\} \\
&= \max_{F}\{< b, F > |\ A^{\mathrm{T}}F \leq D\} \\
&= \max_{f_1, f_2}\{\int [p_{data}(x)f_1(x) + p_g(x)f_2(x)]dx \mid A^{\mathrm{T}}F \leq D\} \\
&= \max_{f_1}\{\int [p_{data}(x)f_1(x) - p_g(x)f_1(x)]dx \mid f_1(x) - f_1(y) \leq | x - y |\}
\end{aligned}
$$

$f_1(x) - f_1(y) \leq | x - y |$ is the **1-Lipschitz constraint**, denoted by $\|f\|_L \leq 1$, then we get the final form:

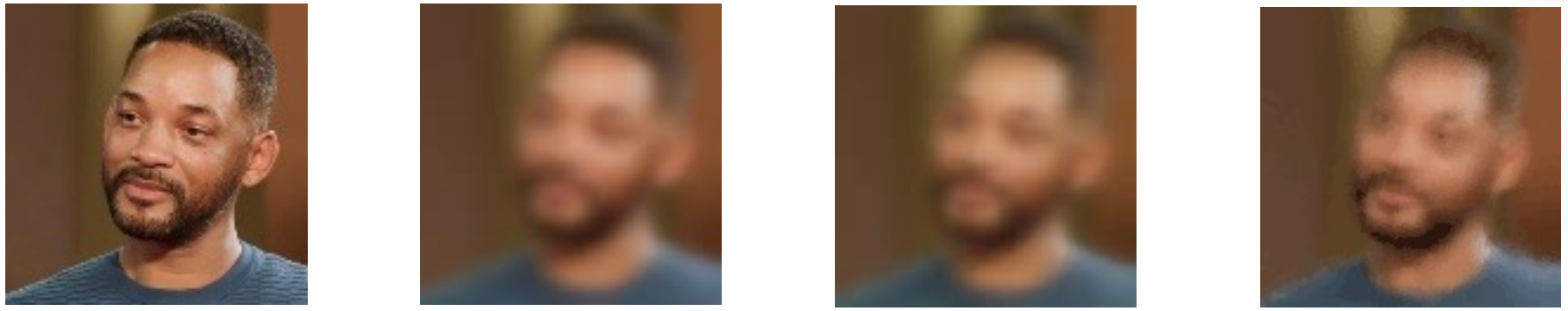
$$
W[p_{data}(x), p_g(x)] = \max_{f, \|f\|_L \leq 1} \mathbb{E}_{x \sim p_{data}}[f(x)] - \mathbb{E}x \sim p_g[f(x)]
$$

1-Lipschitz constraint says, the gradient of weights in the discriminator should be less than 1 (but we don't want it to be 0). Weight clipping can help but gradient penalty is better:

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

Experiments

WGAN-gp is good, but GAN is generally hard to train. This is my first time training a GAN, my goal is to recover the following left image:

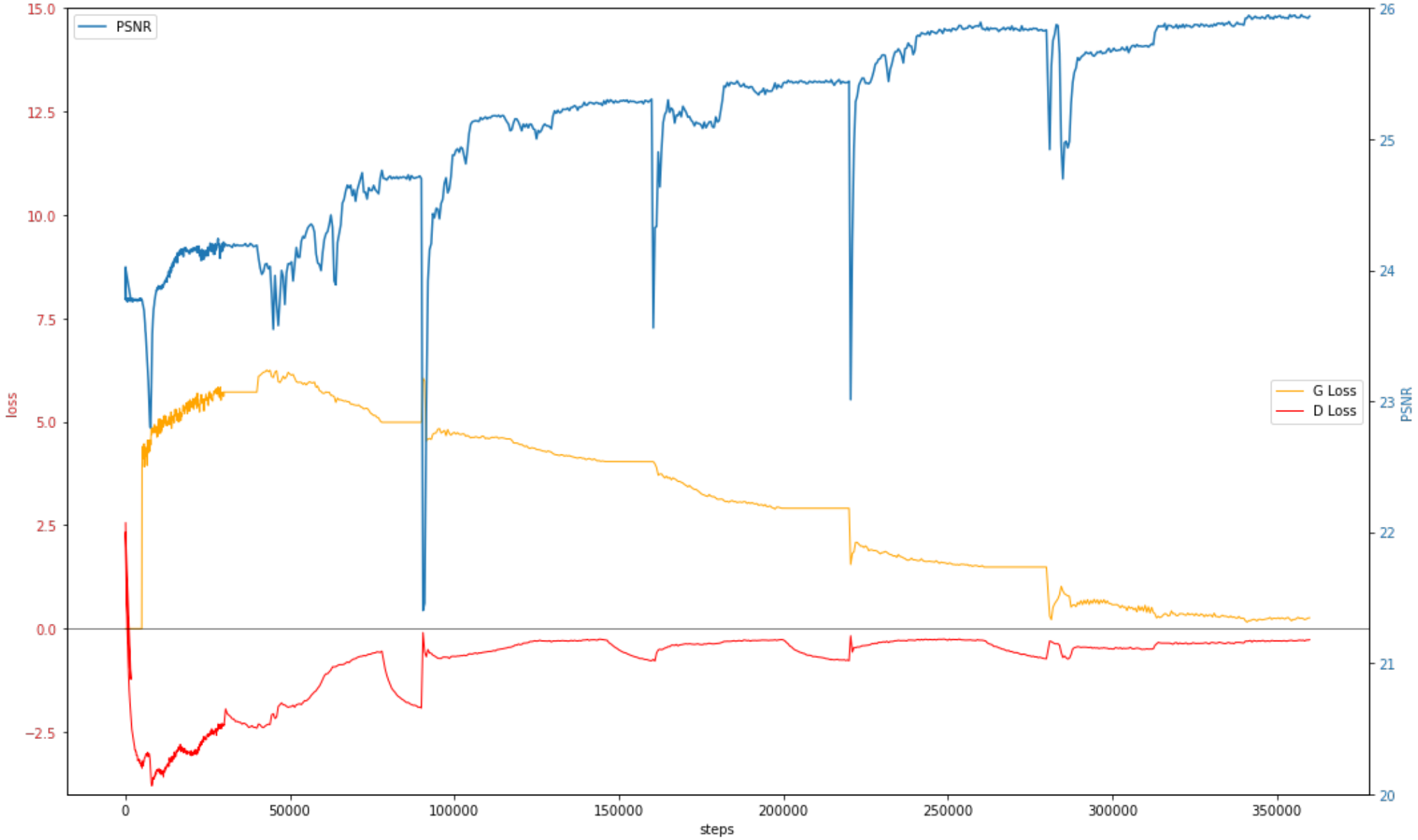


First initialize the UNet in the generator for 10k iterations by setting the target to the mean of all available images, which is shown on the second one

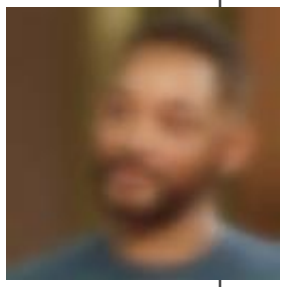
The final output after 10k iterations is the third one, and all inputs are distorted heavily as shown on the right

Then I initialized the discriminator for 5k iterations, updated it for every step but updated the generator for every 6 steps

Result and discussion



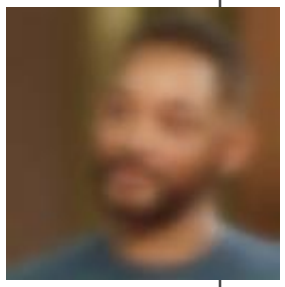
Result and discussion



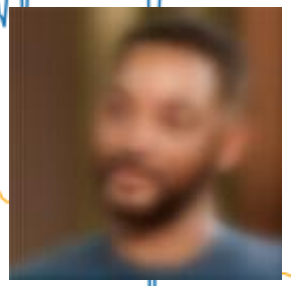
PSNR: 23.78

Initialize Discriminator

Result and discussion



PSNR: 23.78



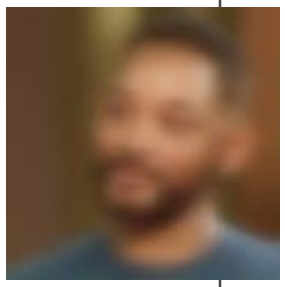
Stuck at 24.25 dB

Initialize Discriminator

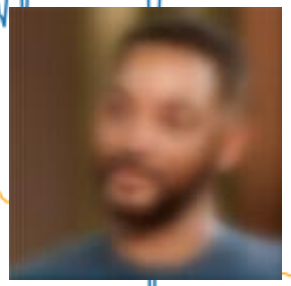
Discriminator loss increase

Result and discussion

Increased the $\{x_i\}$ set to 1000 images but didn't work



PSNR: 23.78

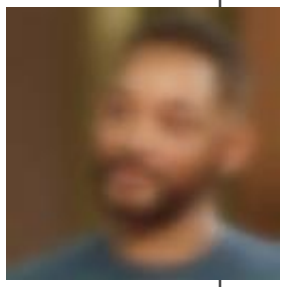


Stuck at 24.25 dB

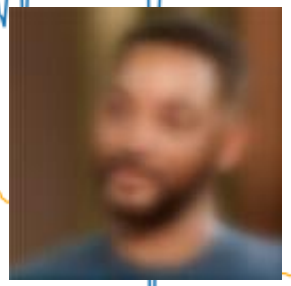
Discriminator loss increase

Result and discussion

The discriminator learns **n/255 patterns** but not continuous pixel intensity!
Add **Gaussian noise** $\mathcal{N}(0, 0.01)$ to both input and generated images before feeding them into discriminator!



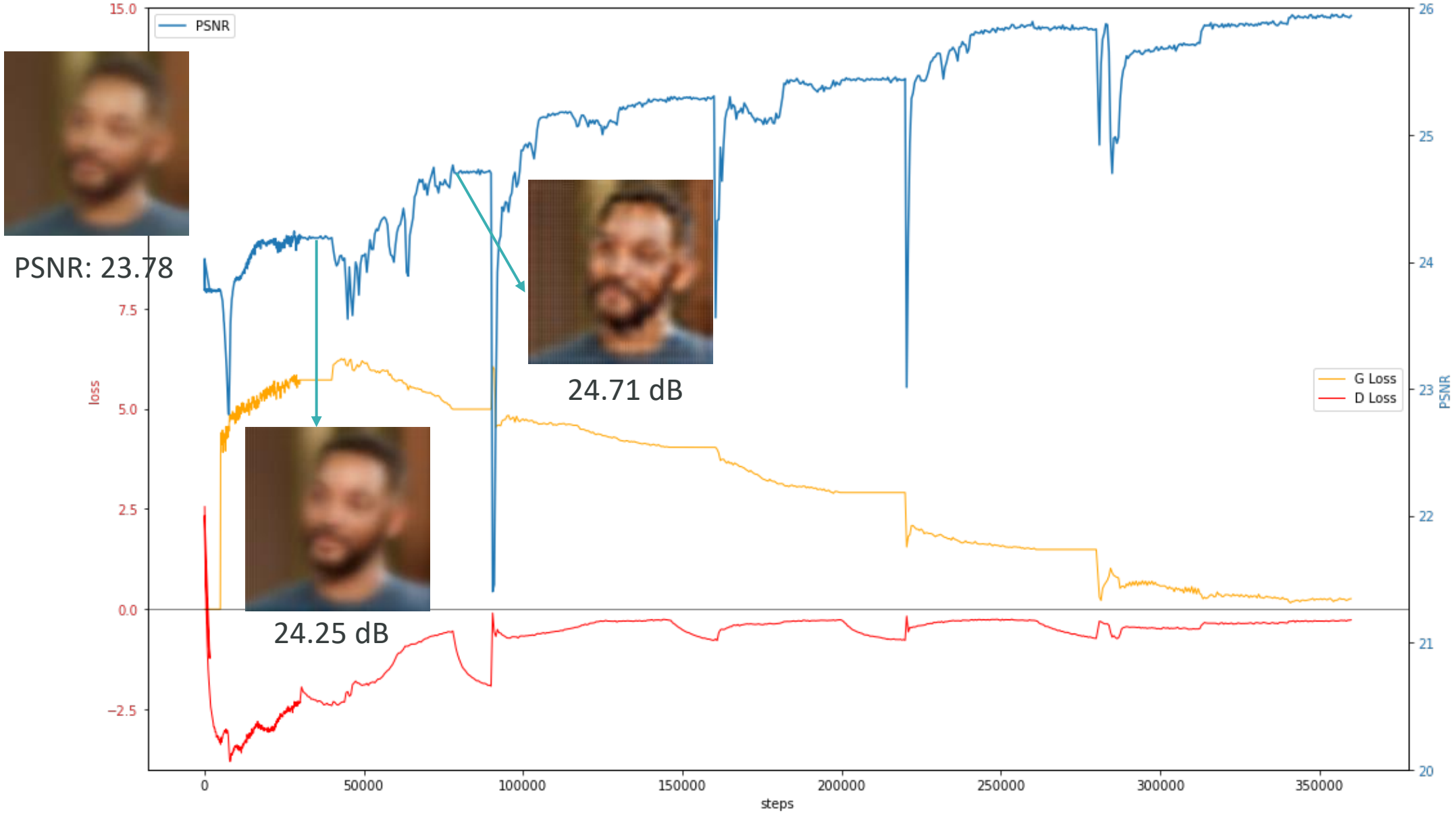
PSNR: 23.78



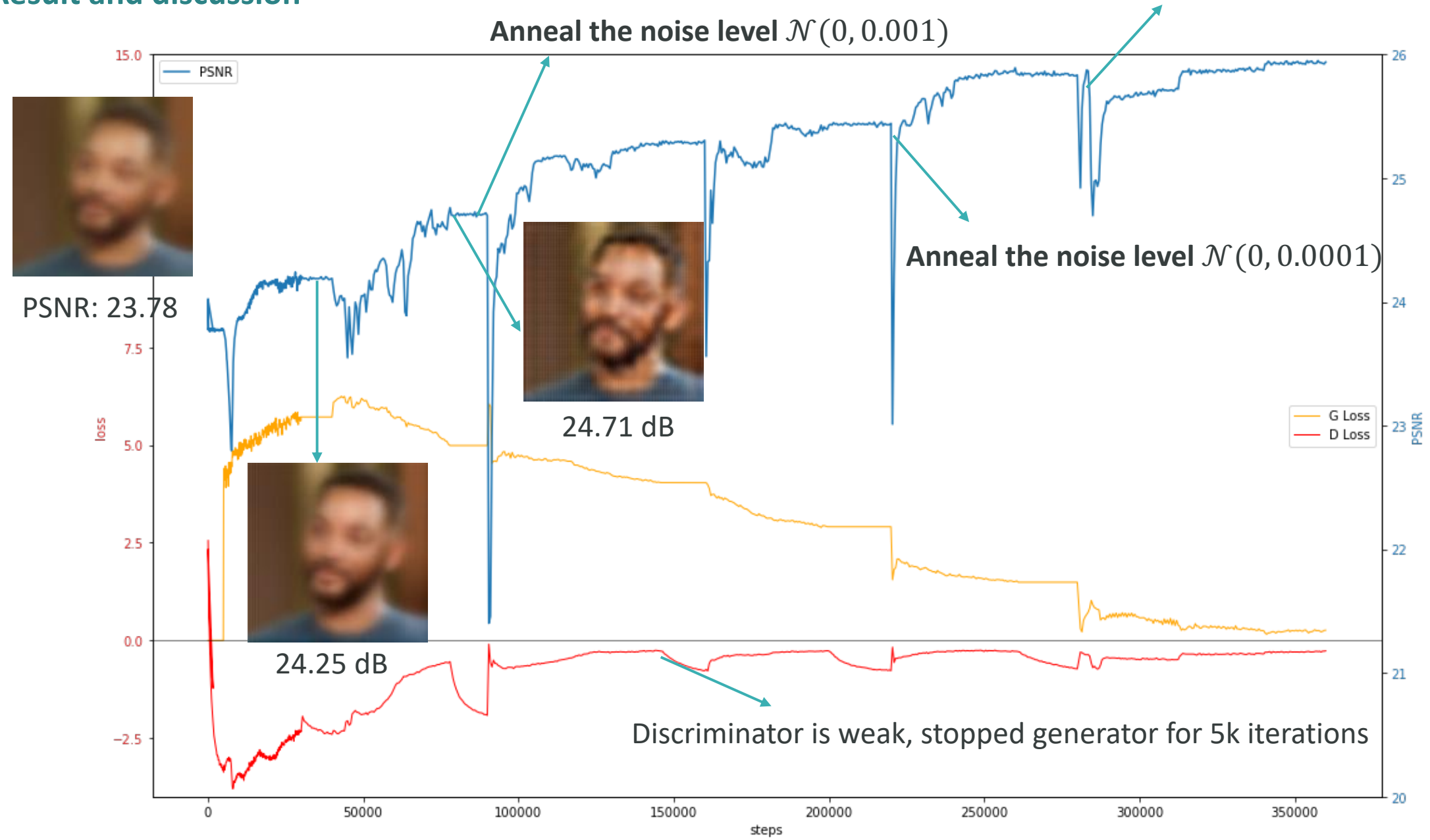
Stuck at 24.25 dB

Discriminator loss increase

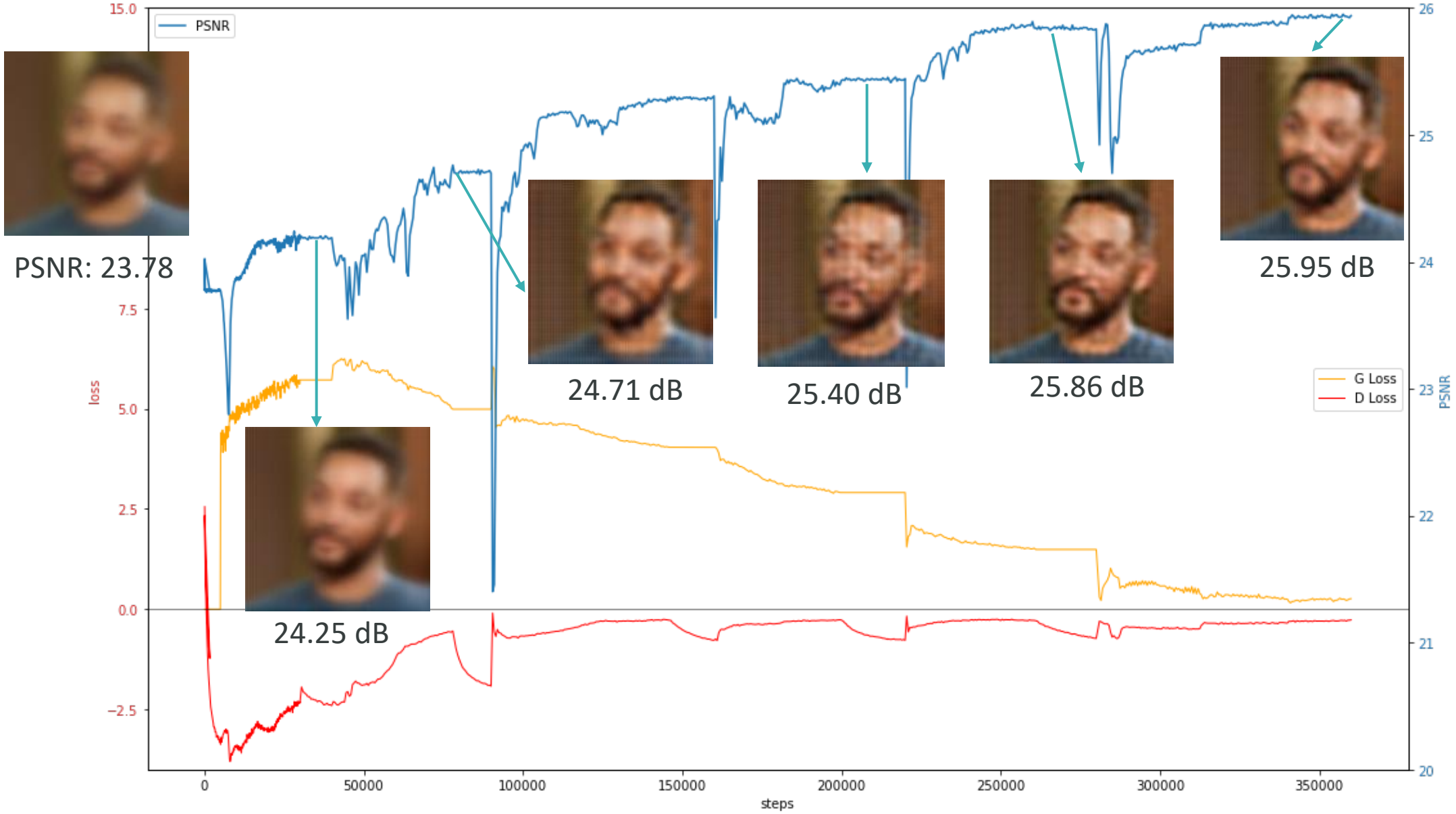
Result and discussion



Result and discussion



Result and discussion



Result and discussion

We can clearly find WGAN-gp can help to stabilize the training. After I froze the generator and just trained discriminator, the model could continue improving, the gradient didn't vanish. This property make the GAN more controllable.

I didn't restore from a latent code z to train generator because:

1. Unet is able to dynamically encode and decode at different level of latent map
2. I want to expand this work to restore other images, I don't want to extract different latent code z explicitly, I want the latent code z to depend on the input y when training the generator

A major flaw in this work is we used a simulator to generate real observed data, however it is still not the real data. Parameters in the simulator could also be updated by the generative perspective, it's like the maximization step in EM step, where "E" could be the GAN training. This could be the next research goal.s