Carnegie Mellon University

Lecture 6: The Design Space of Diffusion Models & Solvers for Fast Sampling

Yutong (Kelly) He

10-799 Diffusion & Flow Matching, Jan 27th, 2026

Modal aws

Housekeeping Announcements

- Homework 2 is out! https://kellyyutonghe.github.io/10799S26/homework/
    - Due date: 2/3 Tue, Late Due date: 2/5 Thur
    - Training models takes time! Start early!
    - Maybe change in due date? Vote in Discord!
- Quiz 3 next class!

Now that we have learned the basics

Is DDPM perfect now? What can we improve?

Us defending our DDPM model from HW1 be like…

Video from World Economic Forum Annual Meeting Davos 2026

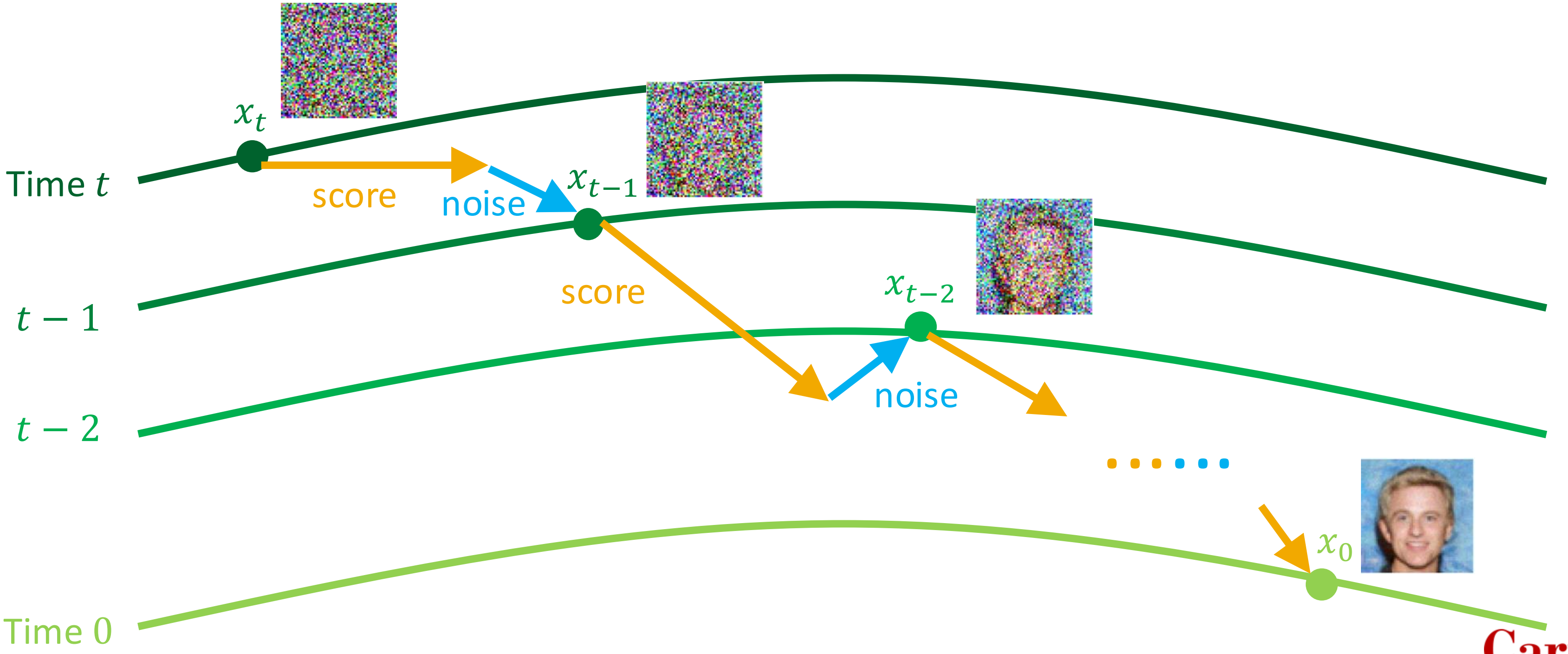
Vanilla DDPM is slooooooooow

- Take 1000 steps to generate an image
- The image quality severely degrades if you use fewer steps
- Scaling to higher resolution/larger model size becomes nightmare

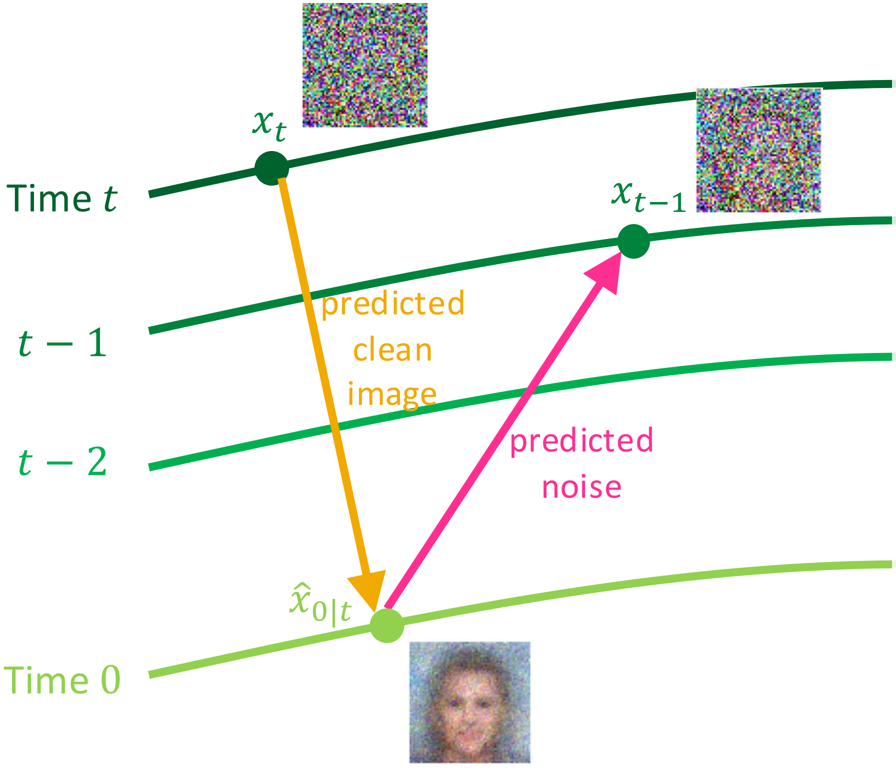
=>

- Any application that needs real time generation ❌
- Video is going to be a big pain □
- GPU cost 💸

Currently, the DDPM sampling is like



But there is another way!



We know $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$, and our model predicts $\epsilon_\theta(x_t, t) \approx \epsilon$, then

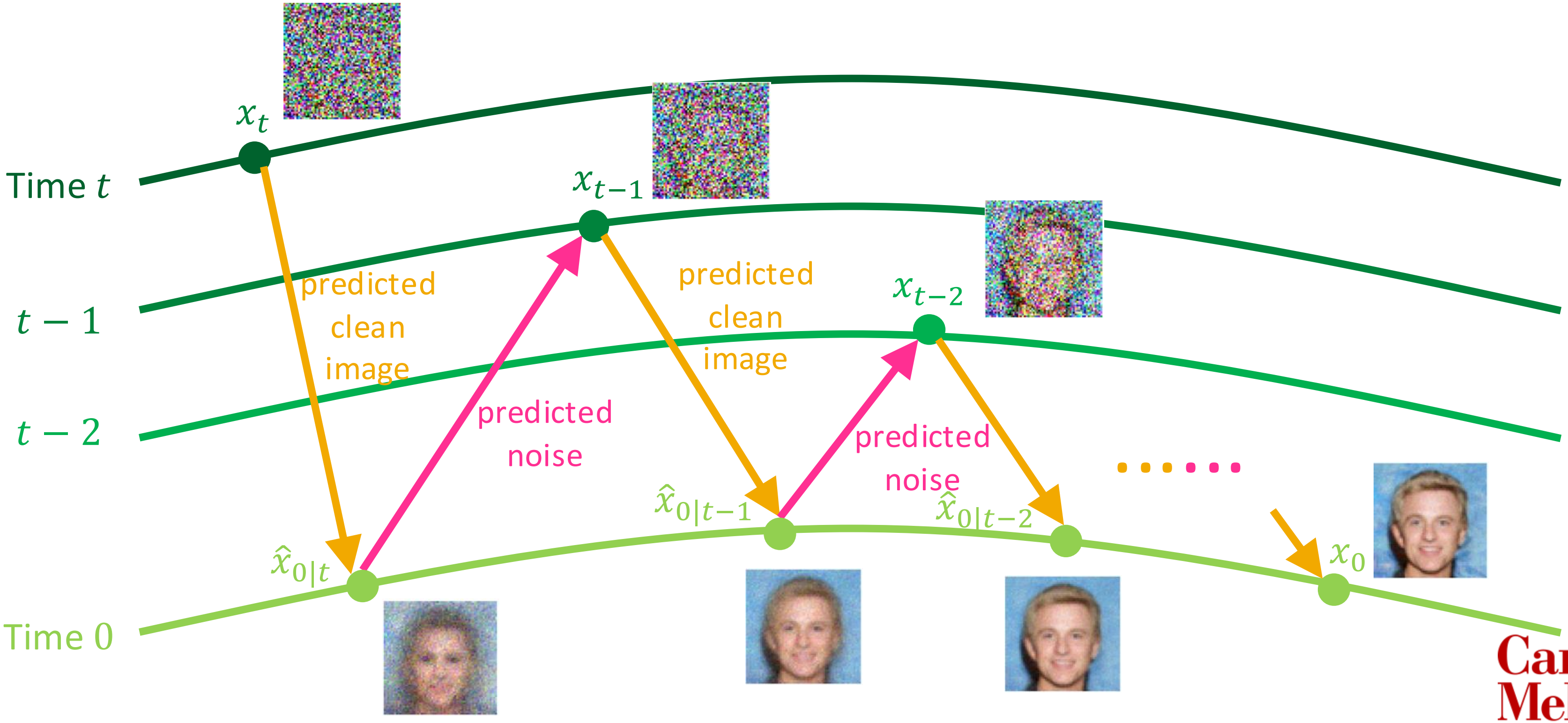
$$\hat{x}_{0|t} \approx \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(x_t, t)\right)$$

And

$$x_{t-1} \approx \sqrt{\bar{\alpha}_{t-1}}\hat{x}_{0|t} + \sqrt{1-\bar{\alpha}_{t-1}}\epsilon_\theta(x_t, t)$$

Image from Harry Potter

So we can also do sampling like this

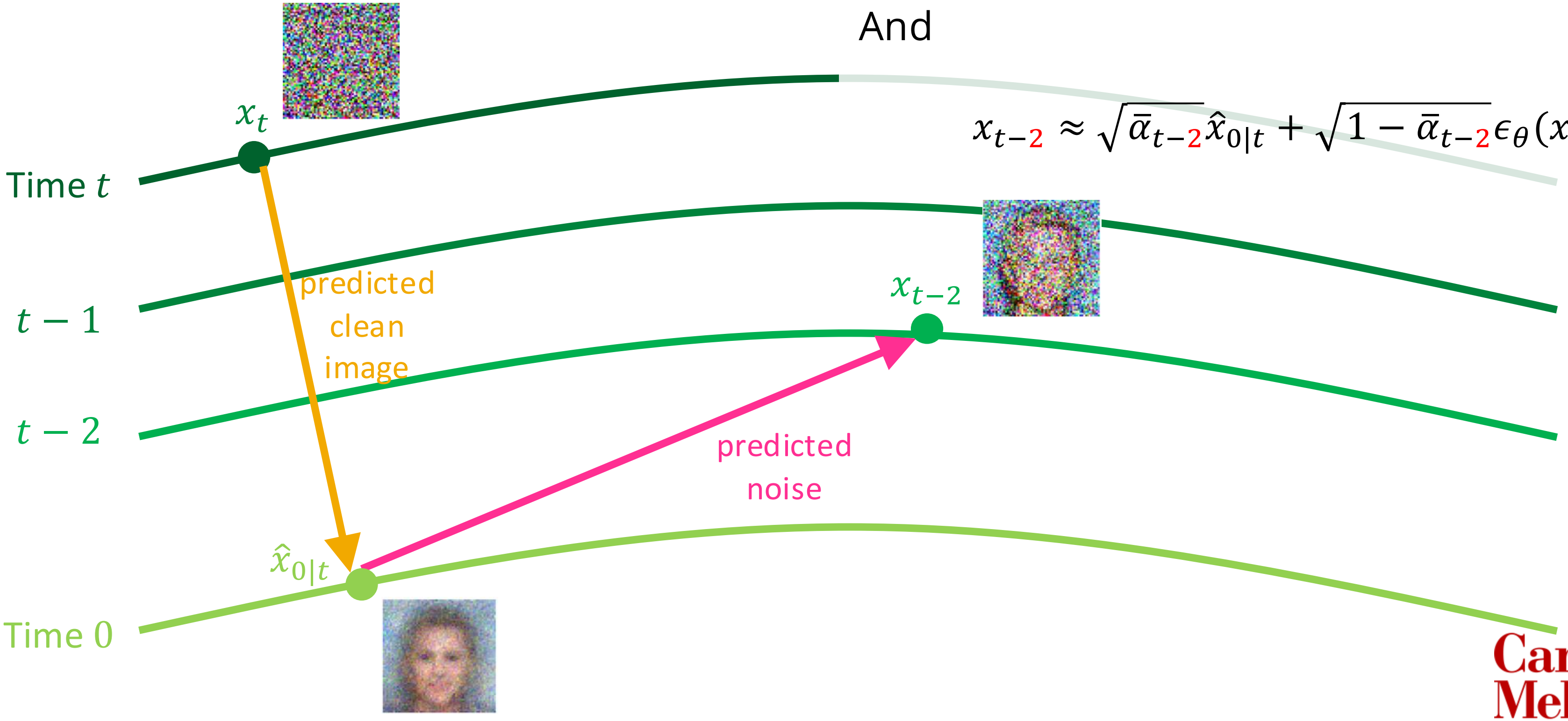


Now what if we skip t-1

$$\hat{x}_{0|t} \approx \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(x_t,t)\right)$$

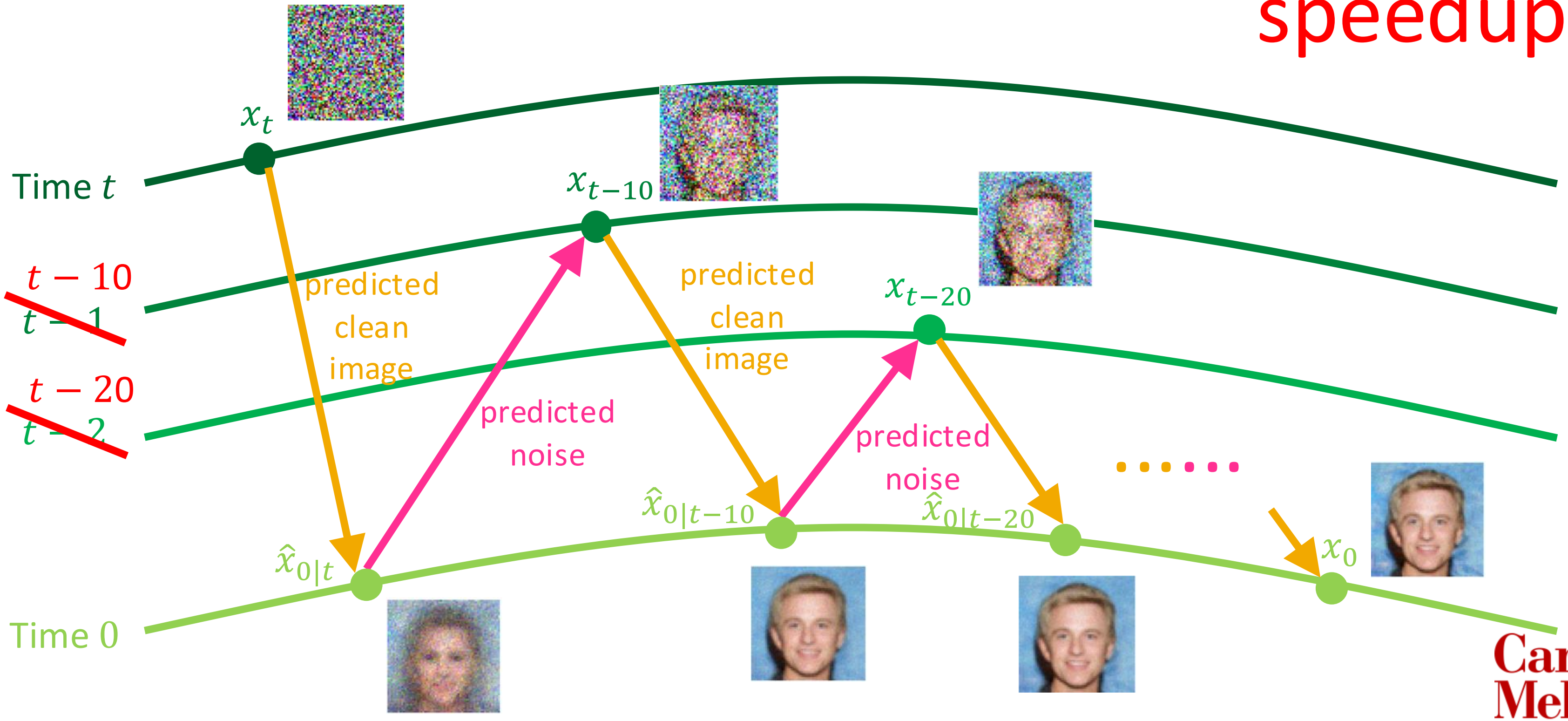
And

$$x_{t-2} \approx \sqrt{\bar{\alpha}_{t-2}}\hat{x}_{0|t} + \sqrt{1-\bar{\alpha}_{t-2}}\epsilon_\theta(x_t,t)$$

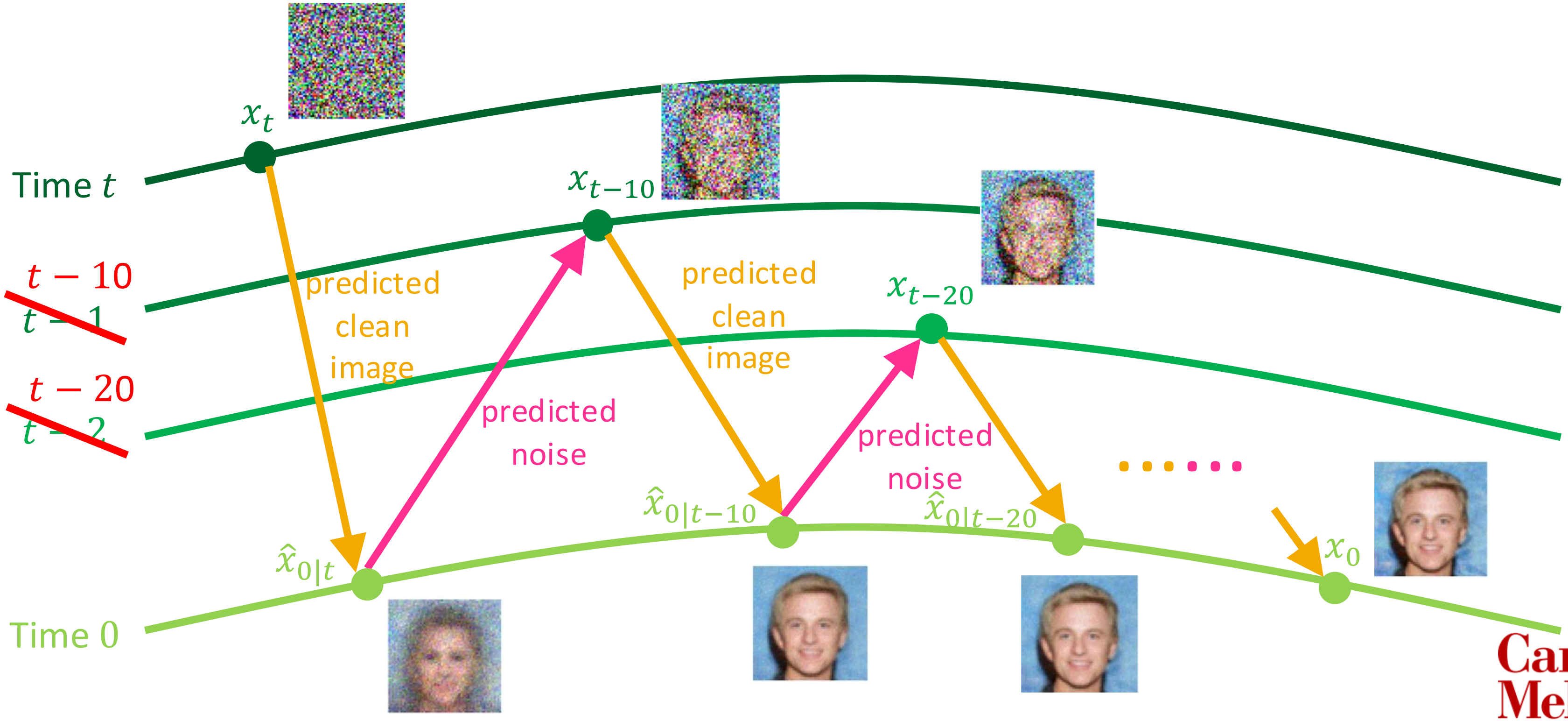


We can skip more than 1 step!

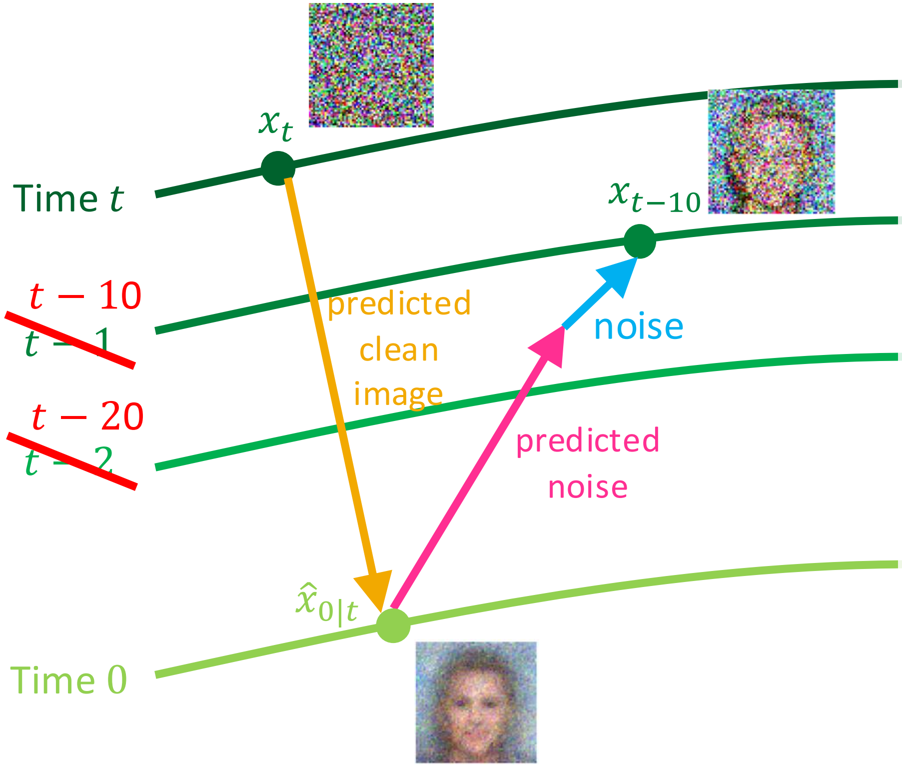
10X speedup!



Notice how right now everything is deterministic



But we can add the stochasticity back!



We know $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$, and our model predicts $\epsilon_\theta(x_t, t) \approx \epsilon$, then

$$\hat{x}_{0|t} \approx \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(x_t, t))$$

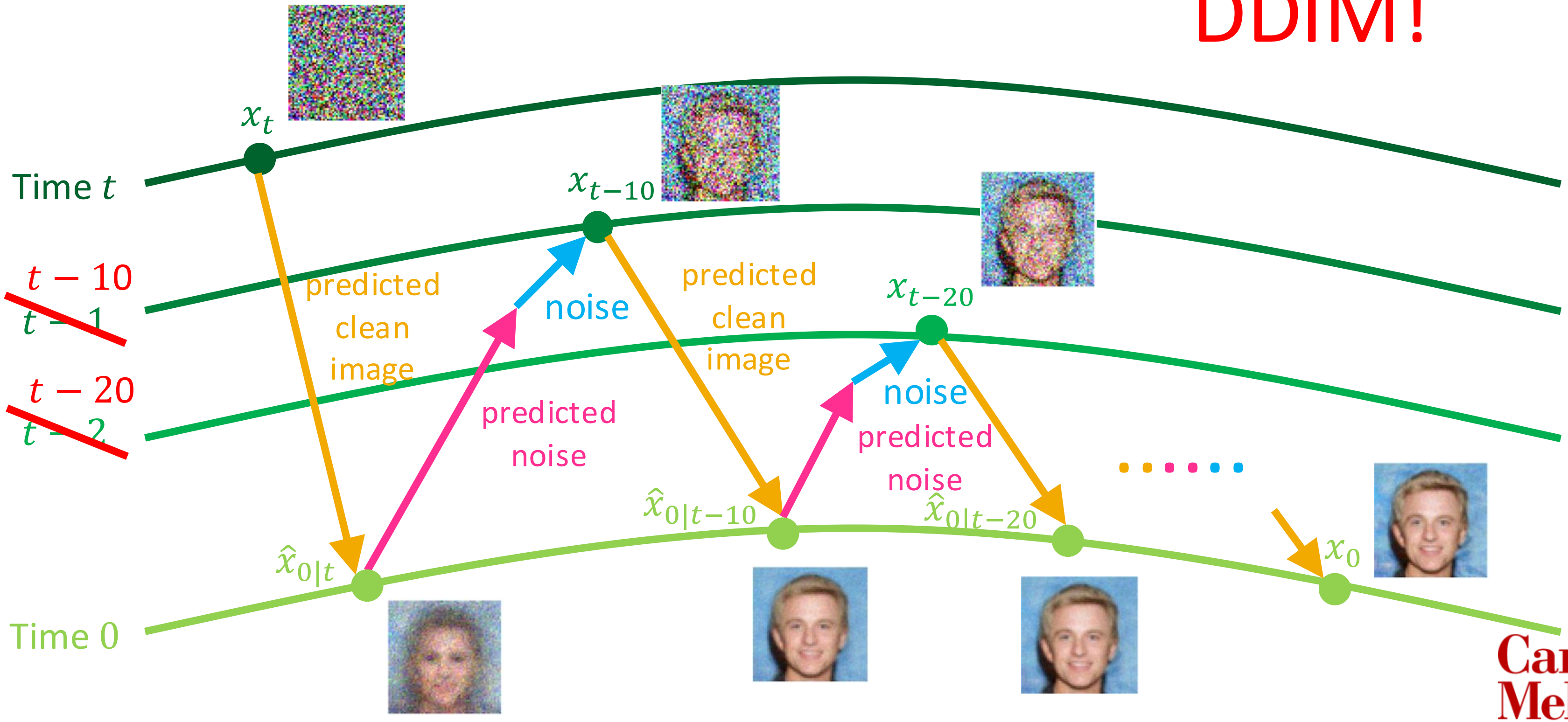
And

$$x_{t-1} \approx \sqrt{\bar{\alpha}_{t-1}}\hat{x}_{0|t} + \sqrt{1-\bar{\alpha}_{t-1} - \sigma_t^2}\epsilon_\theta(x_t, t) + \sigma_t z$$

for constant σ_t and $z \sim N(0, I)$

Putting everything together

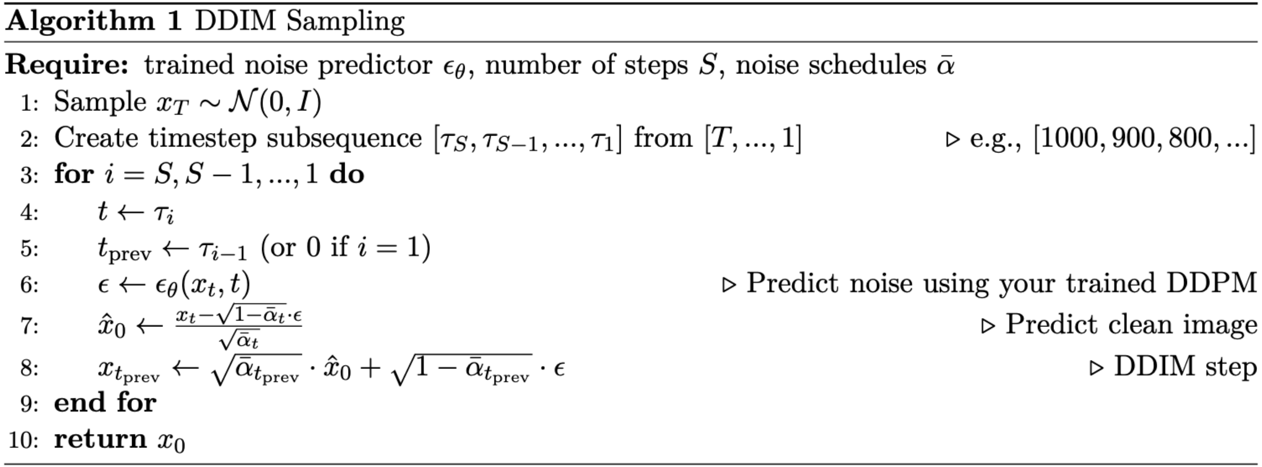
Now you have DDIM!



DDIM: The OG diffusion fast sampling algorithm

$$\boldsymbol{x}_{t-1} = \sqrt{\alpha_{t-1}} \underbrace{\left(\frac{\boldsymbol{x}_t - \sqrt{1-\alpha_t}\,\epsilon_\theta^{(t)}(\boldsymbol{x}_t)}{\sqrt{\alpha_t}}\right)}_{\text{“ predicted } \boldsymbol{x}_0\text{”}} + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\epsilon_\theta^{(t)}(\boldsymbol{x}_t)}_{\text{“direction pointing to } \boldsymbol{x}_t\text{”}} + \underbrace{\sigma_t\epsilon_t}_{\text{random noise}}$$

The pseudocode of the deterministic version of DDIM (also in HW2)



Algorithm 1 DDIM Sampling

```
Require: trained noise predictor $\epsilon_\theta$, number of steps $S$, noise schedules $\bar{\alpha}$
Sample $x_T \sim \mathcal{N}(0, I)$
Create timestep subsequence $[\tau_S, \tau_{S-1}, ..., \tau_1]$ from $[T, ..., 1]$        ▷ e.g., $[1000, 900, 800, ...]$
for $i = S, S-1, ..., 1$ do
    $t \leftarrow \tau_i$
    $t_{\text{prev}} \leftarrow \tau_{i-1}$ (or 0 if $i = 1$)
    $\epsilon \leftarrow \epsilon_\theta(x_t, t)$        ▷ Predict noise using your trained DDPM
    $\hat{x}_0 \leftarrow \frac{x_t - \sqrt{1-\bar{\alpha}_t}\cdot\epsilon}{\sqrt{\bar{\alpha}_t}}$        ▷ Predict clean image
    $x_{t_{\text{prev}}} \leftarrow \sqrt{\bar{\alpha}_{t_{\text{prev}}}}\cdot\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t_{\text{prev}}}}\cdot\epsilon$        ▷ DDIM step
end for
return $x_0$
```

Jiaming Song, Chenlin Meng, Stefano Ermon. “Denoising Diffusion Implicit Models”. ICLR 2021. https://arxiv.org/pdf/2010.02502

Any other ways to sample?

Remember how everything can be ODE now

Cond-OT flow matching:

$$p_0 = N(0, I), p_1 = \delta(x_1)$$

$$x_t = tx_1 + (1-t)x_0, \qquad x_0 \sim p_0$$

$$p_t(x_t|x_1) = N(tx_1, (1-t)^2 I)$$

$$\frac{dx_t}{dt} = u(x_t|x_1) = x_1 - x_0$$

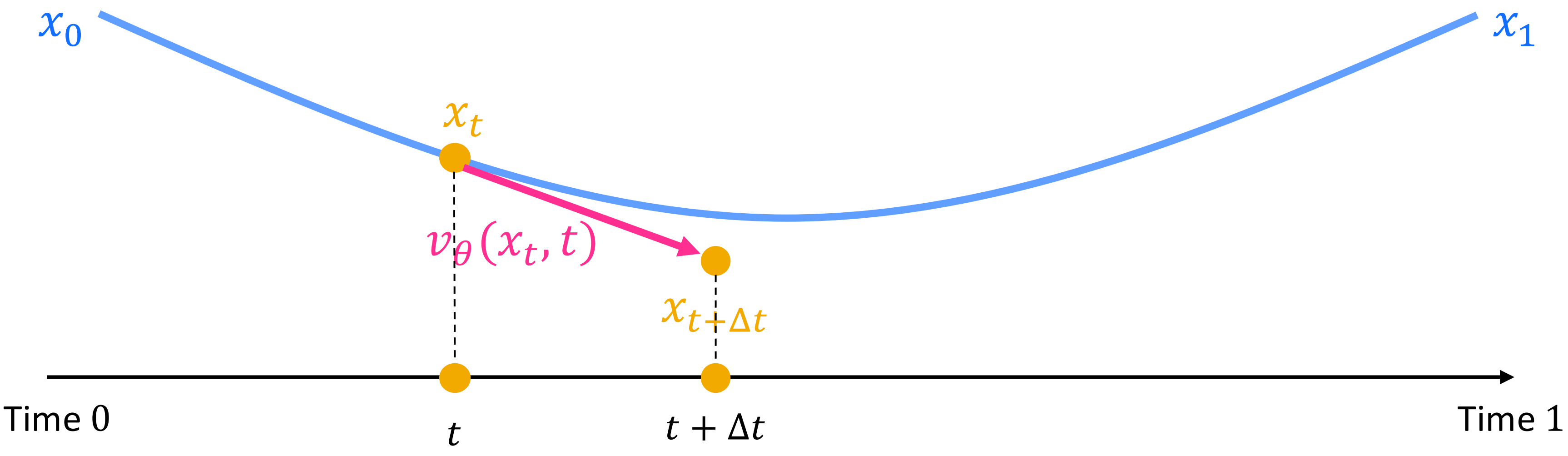
Probability flow ODE:

$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right]\mathrm{d}t,$$

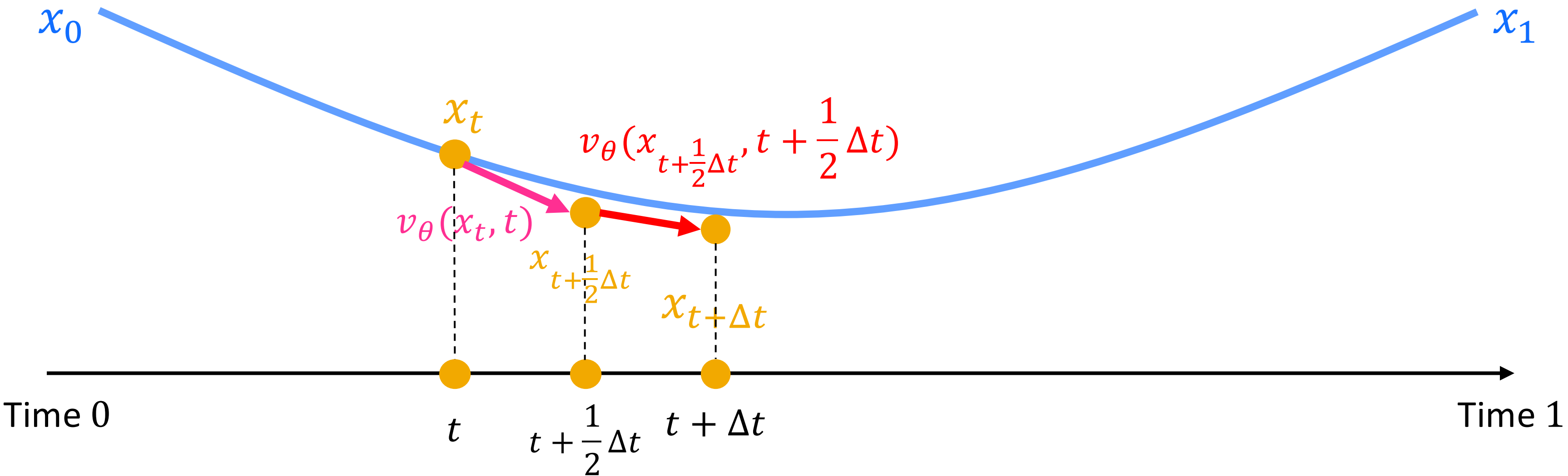
We can use different solvers to solve the ODEs!

Carnegie Mellon University

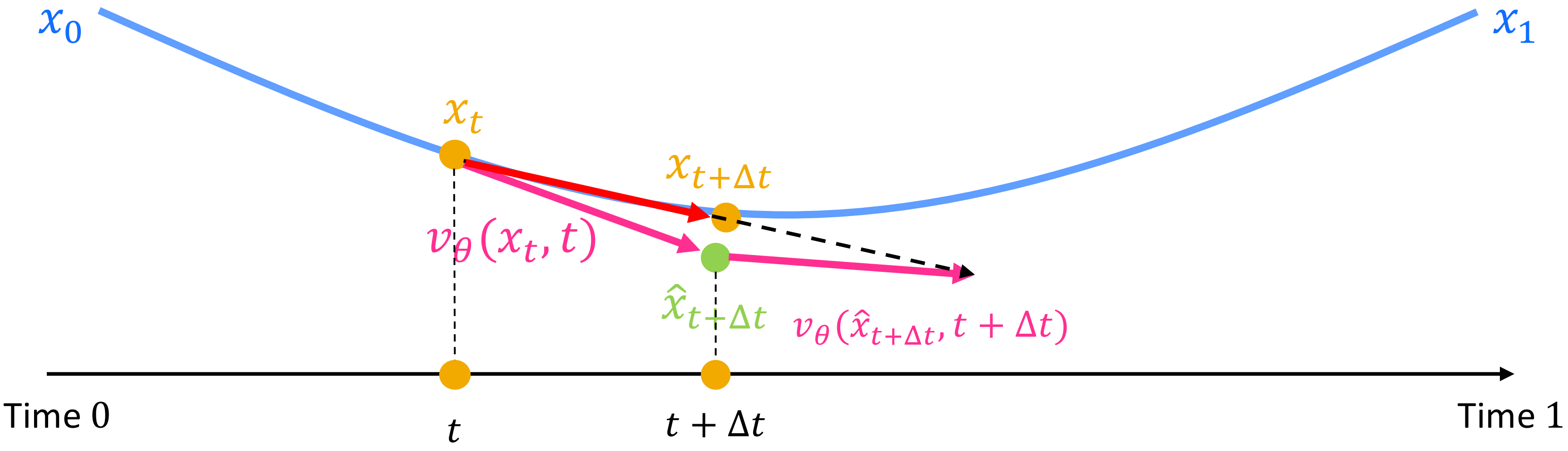
Classic ODE solver 1 – Euler solver



Classic ODE solver 2 – Midpoint solver



Classic ODE solver 3 – 2nd order Heun solver



Diffusion specific solver – DPM solver

The diffusion ODE is

$$\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = \underbrace{f(t)\boldsymbol{x}_t}_{\text{linear}} - \frac{1}{2}\underbrace{g^2(t)\nabla_{\boldsymbol{x}}\log q_t(\boldsymbol{x}_t)}_{\text{model learned}},$$

$$f(t) = \mathrm{d}\log\alpha_t/\mathrm{d}t$$

$$g^2(t) = \frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t} - 2\frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\sigma_t^2 = 2\sigma_t^2\left(\frac{\mathrm{d}\log\sigma_t}{\mathrm{d}t} - \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\right) = -2\sigma_t^2\frac{\mathrm{d}\lambda_t}{\mathrm{d}t}.$$

$$\Downarrow$$

$$\boldsymbol{x}_t = e^{\int_s^t f(\tau)\mathrm{d}\tau}\boldsymbol{x}_s + \int_s^t \left(e^{\int_\tau^t f(r)\mathrm{d}r}\frac{g^2(\tau)}{2\sigma_\tau}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_\tau, \tau)\right)\mathrm{d}\tau.$$

Lu et al. “DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps”. NeurIPS 2022. https://arxiv.org/pdf/2206.00927

Diffusion specific solver – DPM solver

From here

$$\boldsymbol{x}_{t_{i-1}\to t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \alpha_{t_i}\int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda}\hat{\boldsymbol{\epsilon}}_\theta(\hat{\boldsymbol{x}}_\lambda, \lambda)\mathrm{d}\lambda.$$

We can channel Taylor expansion

$$\hat{\boldsymbol{\epsilon}}_\theta(\hat{\boldsymbol{x}}_\lambda, \lambda) = \sum_{n=0}^{k-1}\frac{(\lambda-\lambda_{t_{i-1}})^n}{n!}\hat{\boldsymbol{\epsilon}}_\theta^{(n)}(\hat{\boldsymbol{x}}_{\lambda_{t_{i-1}}}, \lambda_{t_{i-1}}) + \mathcal{O}((\lambda-\lambda_{t_{i-1}})^k),$$

$$\Downarrow$$

$$\boldsymbol{x}_{t_{i-1}\to t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \alpha_{t_i}\sum_{n=0}^{k-1}\hat{\boldsymbol{\epsilon}}_\theta^{(n)}(\hat{\boldsymbol{x}}_{\lambda_{t_{i-1}}}, \lambda_{t_{i-1}})\underbrace{\int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda}\frac{(\lambda-\lambda_{t_{i-1}})^n}{n!}}_{\text{Can be calculated analytically}}\mathrm{d}\lambda + \mathcal{O}(h_i^{k+1}),$$

Lu et al. "DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps". NeurIPS 2022. https://arxiv.org/pdf/2206.00927

Diffusion specific solver – DPM solver

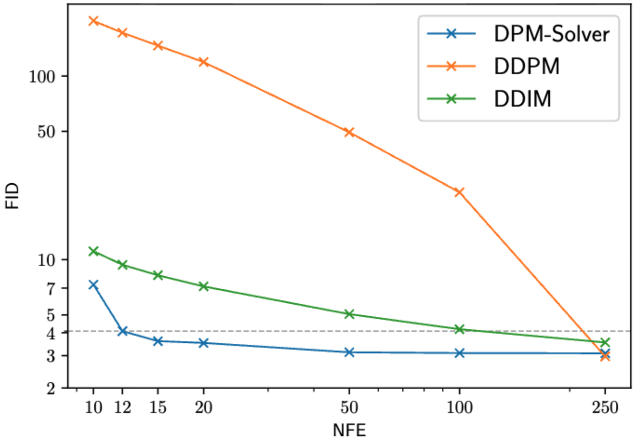
$$\boldsymbol{x}_{t_{i-1}\to t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \alpha_{t_i}\sum_{n=0}^{k-1} \underbrace{\hat{\boldsymbol{\epsilon}}_\theta^{(n)}(\hat{\boldsymbol{x}}_{\lambda_{t_{i-1}}}, \lambda_{t_{i-1}})}_{\text{Can be estimated}} \underbrace{\int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda}\frac{(\lambda-\lambda_{t_{i-1}})^n}{n!}\mathrm{d}\lambda}_{\text{Can be calculated analytically}} + \underbrace{\mathcal{O}(h_i^{k+1})}_{\text{Ignored}},$$

Finally we get for k=1

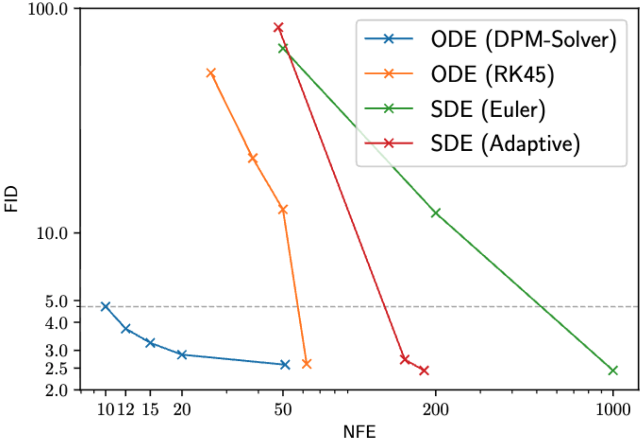
$$\tilde{\boldsymbol{x}}_{t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{t_i}(e^{h_i} - 1)\boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1})$$

Lu et al. "DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps". NeurIPS 2022. https://arxiv.org/pdf/2206.00927

Solver comparison



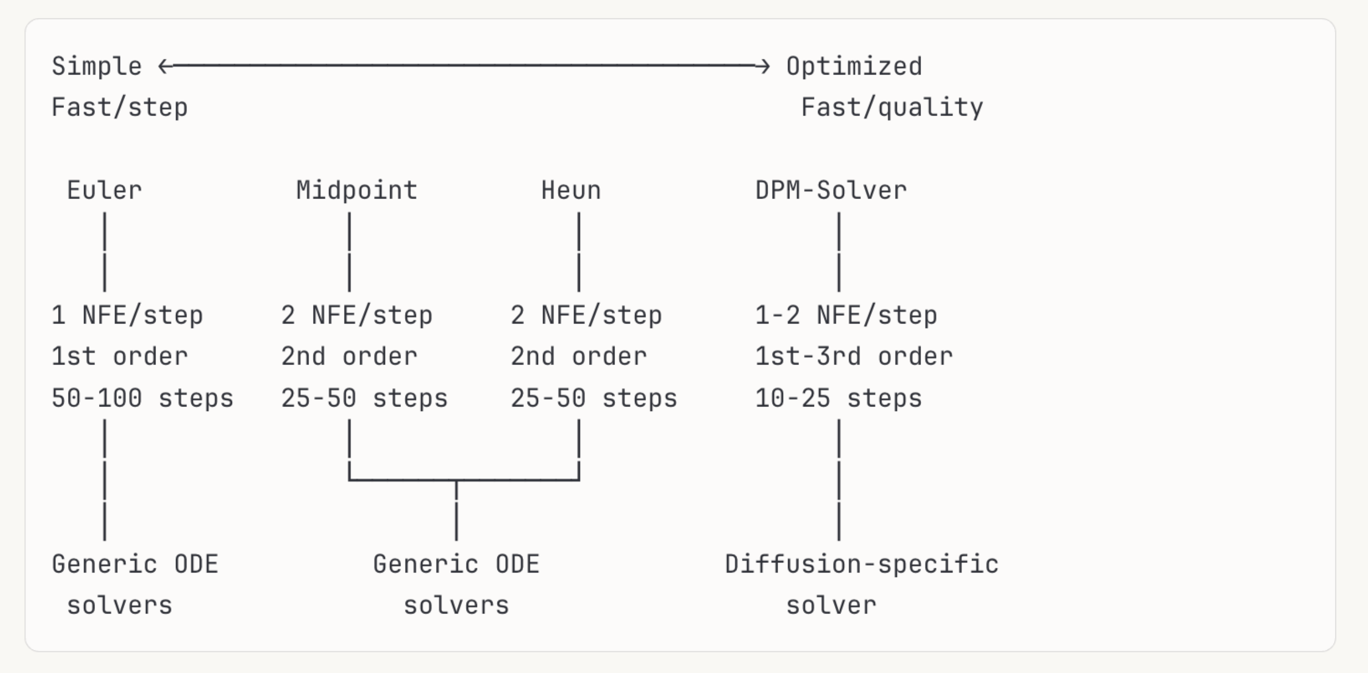
(e) ImageNet 128x128 (discrete)



(a) CIFAR-10 (continuous)

Lu et al. "DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps". NeurIPS 2022. https://arxiv.org/pdf/2206.00927

Solver comparison (from Claude)



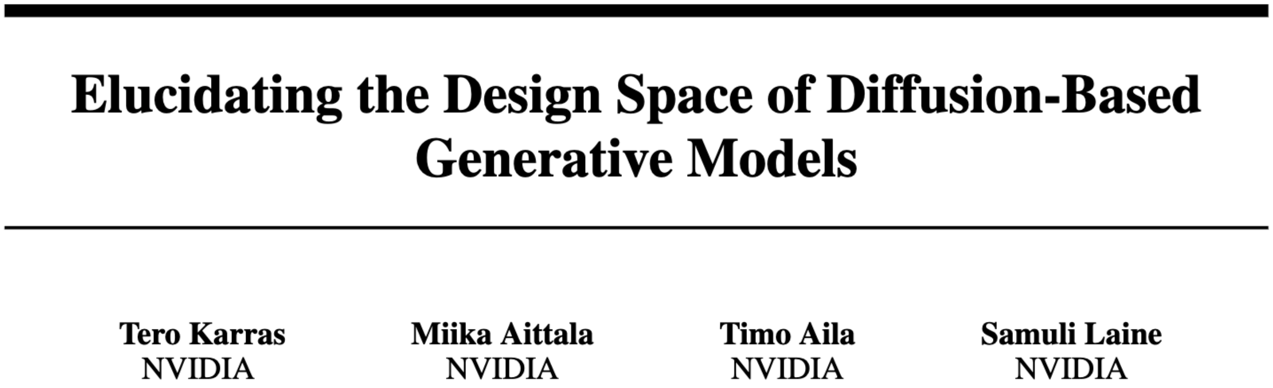
```
Simple ←────────────────────────────────────────→ Optimized
Fast/step                                           Fast/quality


 Euler           Midpoint        Heun          DPM-Solver
   │                │               │                │
   │                │               │                │
1 NFE/step      2 NFE/step      2 NFE/step     1-2 NFE/step
1st order       2nd order       2nd order      1st-3rd order
50-100 steps    25-50 steps     25-50 steps    10-25 steps
   │                │               │                │
   │                └───────┬───────┘                │
   │                        │                        │
Generic ODE            Generic ODE          Diffusion-specific
 solvers                 solvers                 solver
```

Besides sampling, any other ways to improve DDPM?

Actually there is a paper that studied them all

EDM: What are the actual independent design choices of diffusion models and let’s disentangle them as study them one by one



Elucidating the Design Space of Diffusion-Based Generative Models

Tero Karras
NVIDIA

Miika Aittala
NVIDIA

Timo Aila
NVIDIA

Samuli Laine
NVIDIA

The design space of diffusion models

The design space of diffusion models

Training
- Prefixed noise schedule
- Training noise sampling schedule
- Loss weighting w.r.t. time

Model
- Reparameterization
- Input/Output scaling
- How to do time conditioning

Sampling
- Solver ✓
- Sampling time noise schedule
- Number of time steps ✓

SNR = Signal/Noise

Signal-to-noise ratio (SNR)

$$\text{SNR} = \frac{\alpha_t^2}{\sigma_t^2}$$

Data x_0
$t = 0$

$x_{0.25T}$
$t = 0.25T$

$x_{0.5T}$
$t = 0.5T$

$x_{0.75T}$
$t = 0.75T$

Noise x_T
$t = T$

No noise,
all signal

easy

Some noise,
some signal

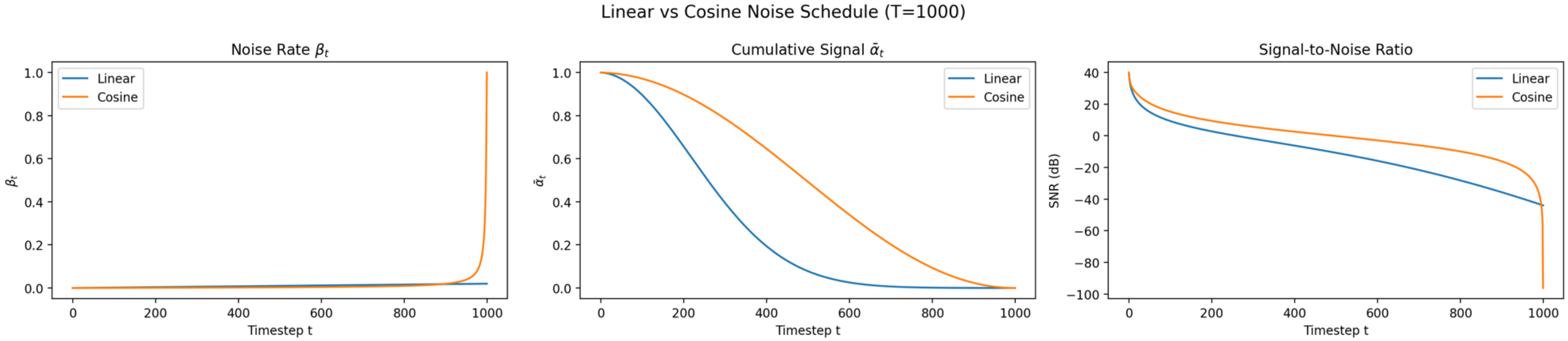
Hard!!!

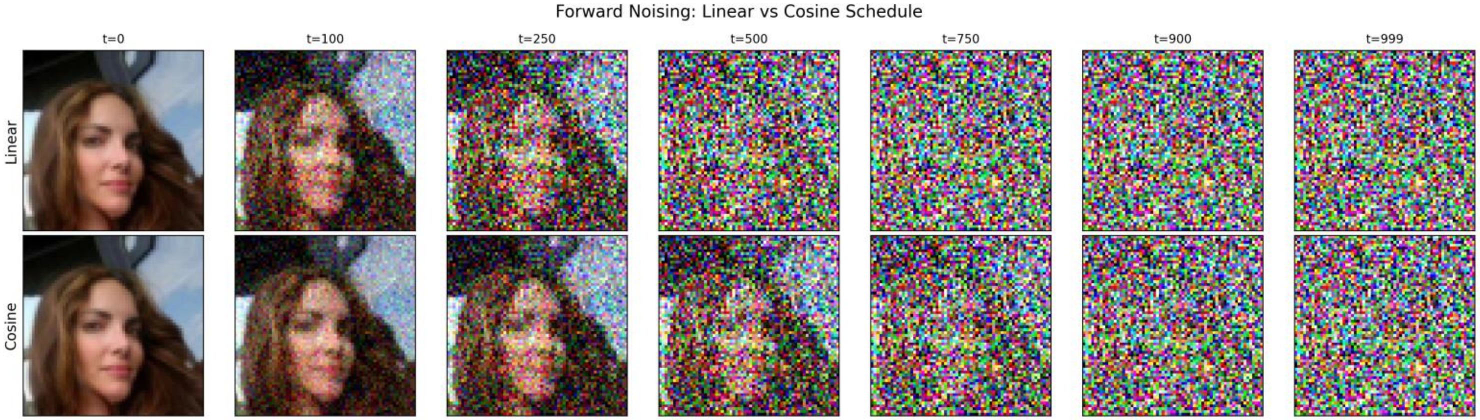
easy

All noise,
no signal

Cat stolen from Chieh-Hsin (Jesse) Lai

SNR Attempt 1: Linear scheduler -> Cosine scheduler

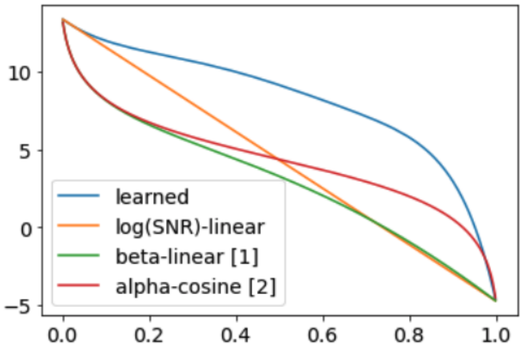




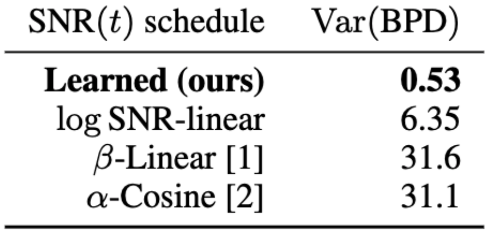
Carnegie Mellon University

Nichol & Dhariwal. “Improved Denoising Diffusion Probabilistic Models”. ICML 2021. https://arxiv.org/pdf/2206.00927

SNR Attempt 1.5: You can even learn your schedule



(a) log SNR vs time t



$\mathrm{SNR}(t)$ schedule	Var(BPD)
Learned (ours)	**0.53**
log SNR-linear	6.35
β-Linear [1]	31.6
α-Cosine [2]	31.1

(b) Variance of VLB estimate

Kingma et al. "Variational Diffusion Models". NeurIPS 2021. https://arxiv.org/pdf/2107.00630

SNR Attempt 2: At sampling time, you can also spend more time on the more difficult noise levels!

$$\sigma_{i<N} = \left({\sigma_{\max}}^{\frac{1}{\rho}} + \tfrac{i}{N-1}\left({\sigma_{\min}}^{\frac{1}{\rho}} - {\sigma_{\max}}^{\frac{1}{\rho}}\right)\right)^{\rho} \quad \text{and} \quad \sigma_N = 0.$$

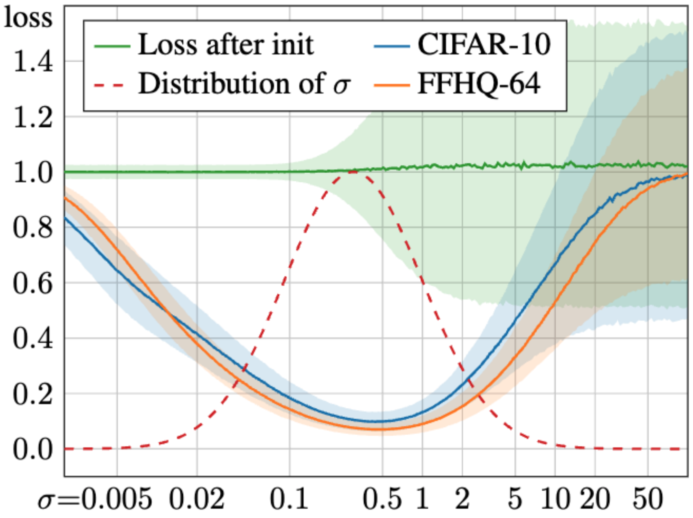
SNR Attempt 3: You should parameterize your model to take the actual noise level (σ or $\log \sigma$) instead of timesteps!

- Scheduler invariant parameterization
- Continuous => Easier to do make inference time changes (fewer/more steps)
- Better numerical scaling

Karras et al. "Elucidating the Design Space of Diffusion-Based Generative Models". NeurIPS 2022. https://arxiv.org/pdf/2206.00364

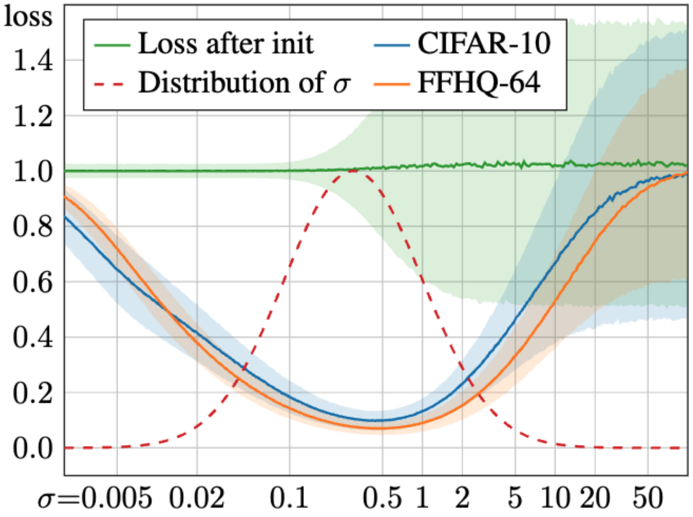
SNR Attempt 4: Train on harder time steps more

Just sampling the more difficult time steps to more frequently during training time!



Karras et al. "Elucidating the Design Space of Diffusion-Based Generative Models". NeurIPS 2022. https://arxiv.org/pdf/2206.00364

SNR Attempt 5: Weight different timestep differently according to their SNR



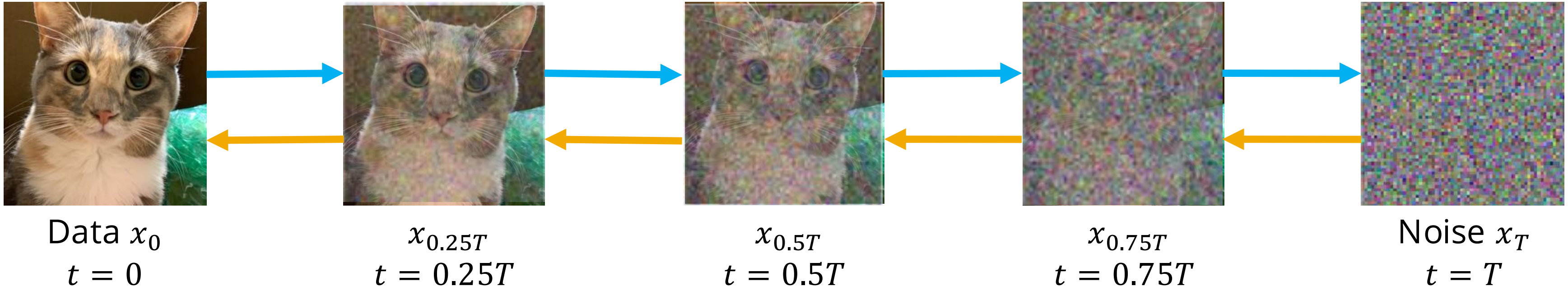
The gradient scale varies a lot depending on the SNR

⇓

Apply a weighting scalar to balance it out!

Karras et al. "Elucidating the Design Space of Diffusion-Based Generative Models". NeurIPS 2022. https://arxiv.org/pdf/2206.00364

SNR Attempt 6: Reparameterization



Predict clean signal ($x_{0,\theta}(x_t, t)$):

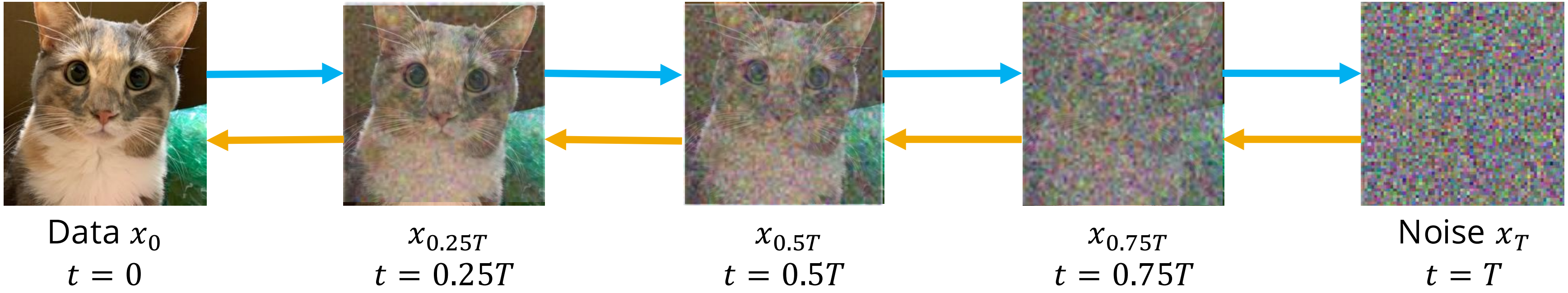
- Easy at low noise (high SNR)
- Hard at high noise (low SNR)

Predict noise ($\epsilon_\theta(x_t, t)$):

- Easy at high noise (low SNR)
- Hard at low noise (high SNR)

Something in between?

SNR Attempt 6: Reparameterization



Predict clean signal ($x_{0,\theta}(x_t, t)$):

- Easy at low noise (high SNR)
- Hard at high noise (low SNR)

Predict interpolation (a.k.a velocity $v_\theta(x_t, t)$):

- Good balance!
- $v = \alpha_t \epsilon - \sigma_t x_0$

Predict noise ($\epsilon_\theta(x_t, t)$):

- Easy at high noise (low SNR)
- Hard at low noise (high SNR)

Carnegie Mellon University

Salimans & Ho. "Progressive Distillation for Fast Sampling of Diffusion Models". ICLR 2022. https://arxiv.org/pdf/2202.00512

Another way to do network parameterization?

Remember how previously we want to do v-prediction as our target

$$v = \alpha_t \epsilon - \sigma_t x_0$$

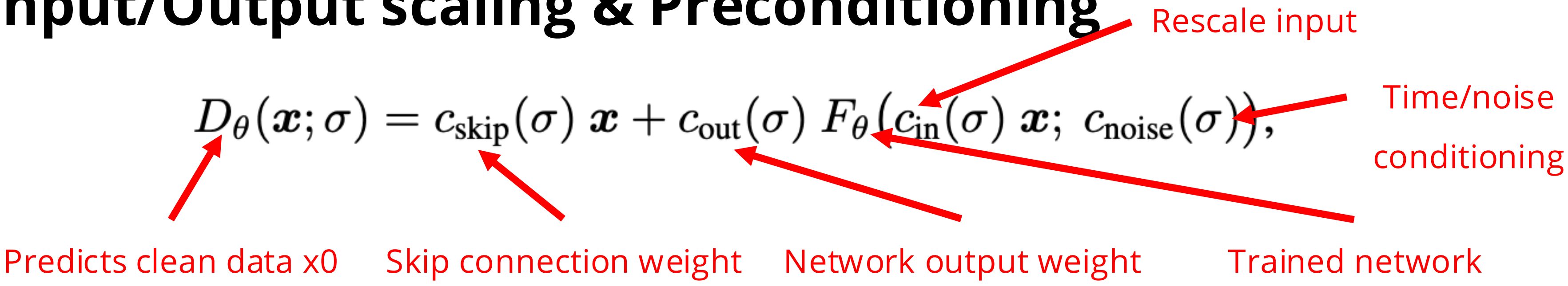
A mixture of noise and clean data

Now notice how x_t is also a mixture of noise and clean data

$\Rightarrow$ The network should reuse some information in x_t

$\Rightarrow$ Skip connection!

Input/Output scaling & Preconditioning



$$D_\theta(\boldsymbol{x};\sigma) = c_{\text{skip}}(\sigma)\,\boldsymbol{x} + c_{\text{out}}(\sigma)\,F_\theta\big(c_{\text{in}}(\sigma)\,\boldsymbol{x};\ c_{\text{noise}}(\sigma)\big),$$

Rescale input

Time/noise conditioning

Predicts clean data x0

Skip connection weight

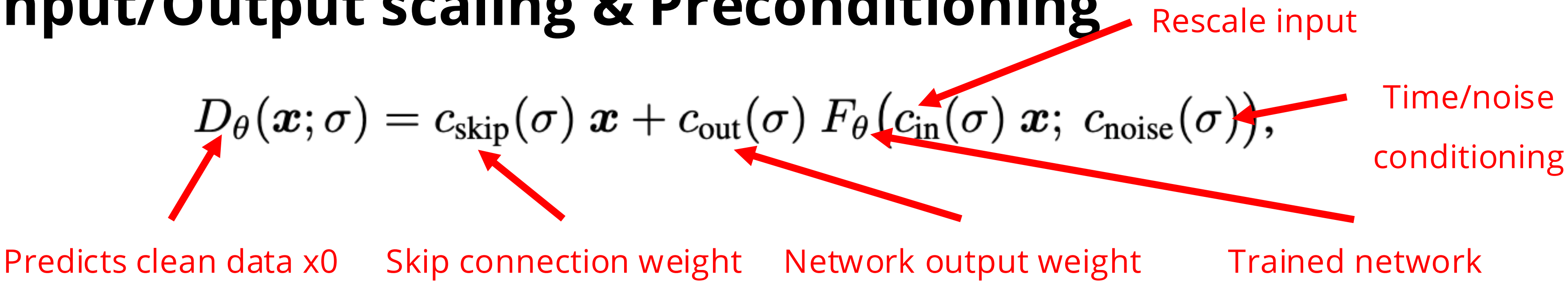
Network output weight

Trained network

- When $t \rightarrow 0$, $\sigma \rightarrow 0$ => input is mostly clean => can directly pass through more
- When $t \rightarrow 1$, $\sigma \rightarrow \infty$ => input is mostly noise => should ignore most of the input and rely on the network prediction more

Karras et al. "Elucidating the Design Space of Diffusion-Based Generative Models". NeurIPS 2022. https://arxiv.org/pdf/2206.00364

Input/Output scaling & Preconditioning



Rescale input

Time/noise conditioning

$$D_\theta(\boldsymbol{x};\sigma) = c_{\text{skip}}(\sigma)\,\boldsymbol{x} + c_{\text{out}}(\sigma)\,F_\theta\big(c_{\text{in}}(\sigma)\,\boldsymbol{x};\ c_{\text{noise}}(\sigma)\big),$$

Predicts clean data x0

Skip connection weight

Network output weight

Trained network

We also want:

- The network input to have unit variance
- Training target to have unit variance
- Reuse information in input as much as possible

=> $c_{\text{skip}}(\sigma) = \frac{\sigma_{\text{data}}^2}{\sigma^2+\sigma_{\text{data}}^2}$, $c_{\text{out}}(\sigma) = \frac{\sigma\sigma_{\text{data}}}{\sqrt{\sigma^2+\sigma_{\text{data}}^2}}$, $c_{\text{in}}(\sigma) = \frac{1}{\sqrt{\sigma^2+\sigma_{\text{data}}^2}}$, $c_{\text{noise}}(\sigma) = \frac{1}{4}\log\sigma$

Karras et al. “Elucidating the Design Space of Diffusion-Based Generative Models”. NeurIPS 2022. https://arxiv.org/pdf/2206.00364

The design space of diffusion models

The design space of diffusion models

Training

- Prefixed noise schedule ☑
- Training noise sampling schedule ☑
- Loss weighting w.r.t. time ☑

Model

- Reparameterization ☑
- Input/Output scaling ☑
- How to do time conditioning ☑

Sampling

- Solver ☑
- Sampling time noise schedule ☑
- Number of time steps ☑

Next class we will learn how to turn an unconditional diffusion model into a conditional one

Spoiler alert: You may or may not need to train for it!

Image from http://makeameme.org/meme/spoilers-are-coming