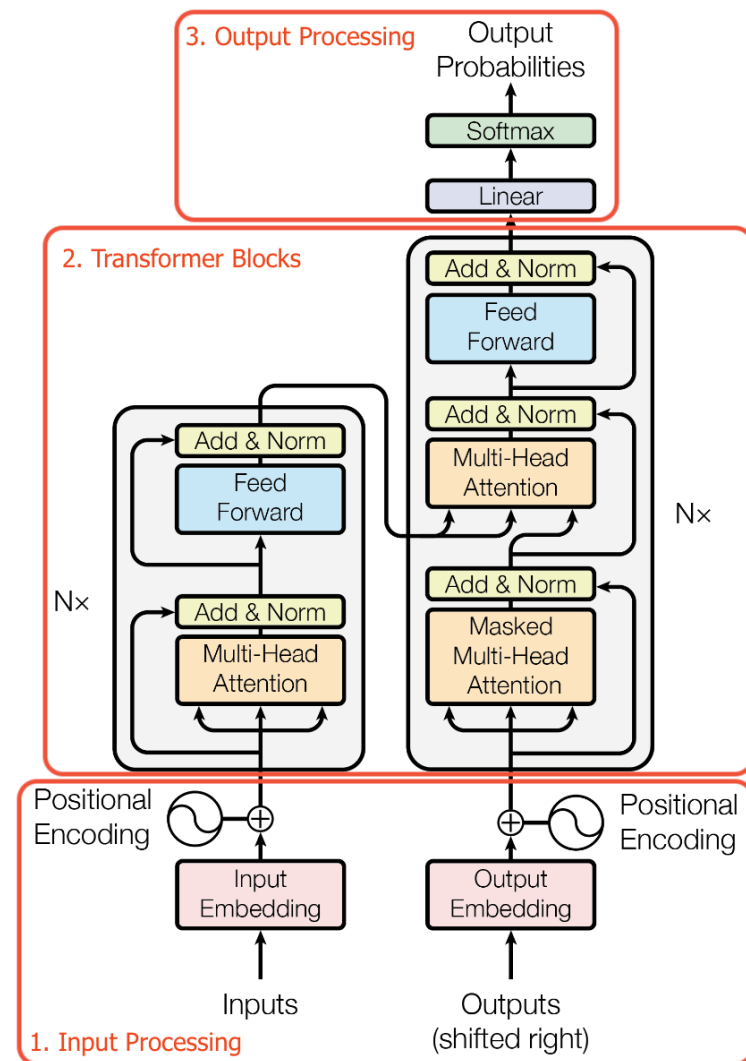
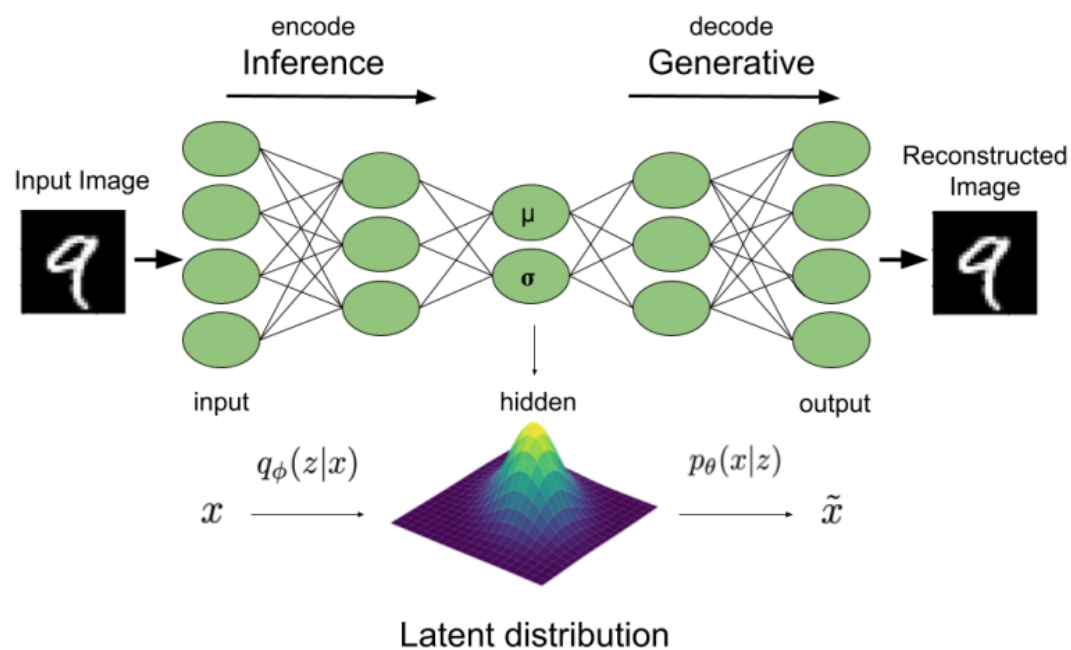


Denoising Diffusion Probabilistic Models

Presenter: Binxu Li & Haoyi Duan

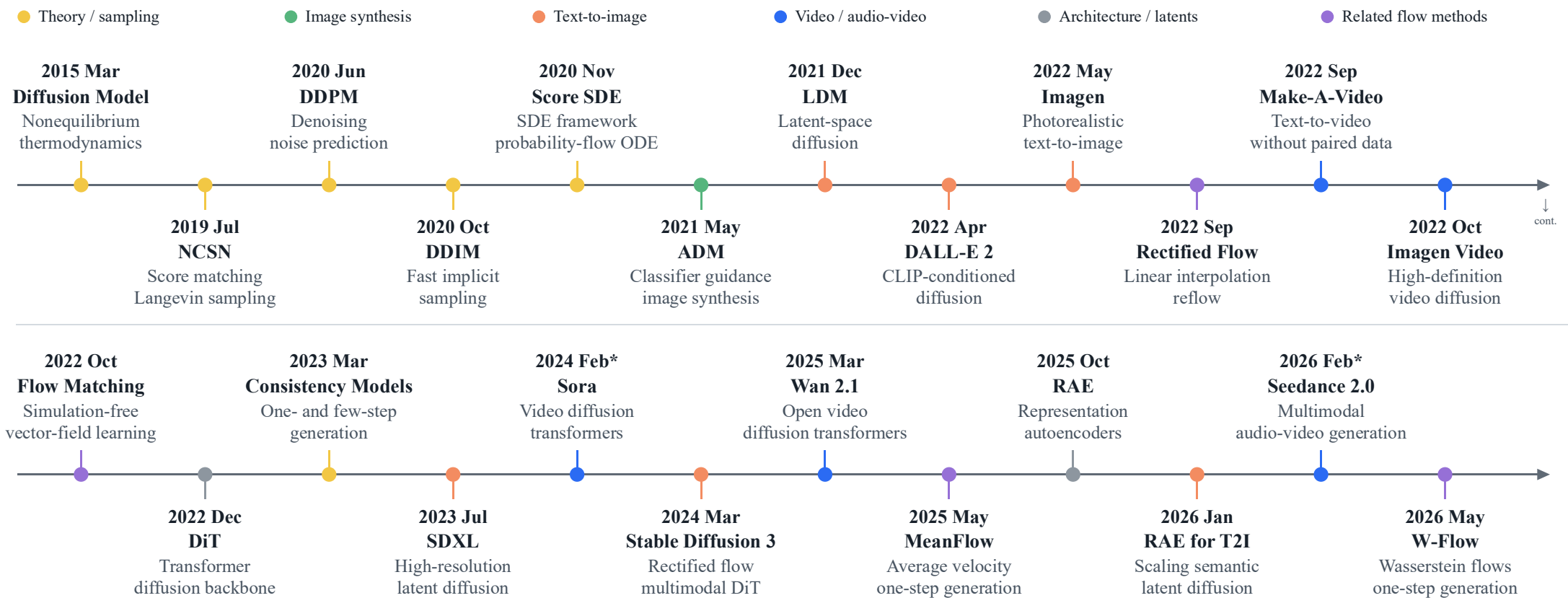
Recap on VAE & Transformer



Diffusion Model Timeline

Diffusion Models: Selected Milestones (2015–2026)

Including related flow-based generative models

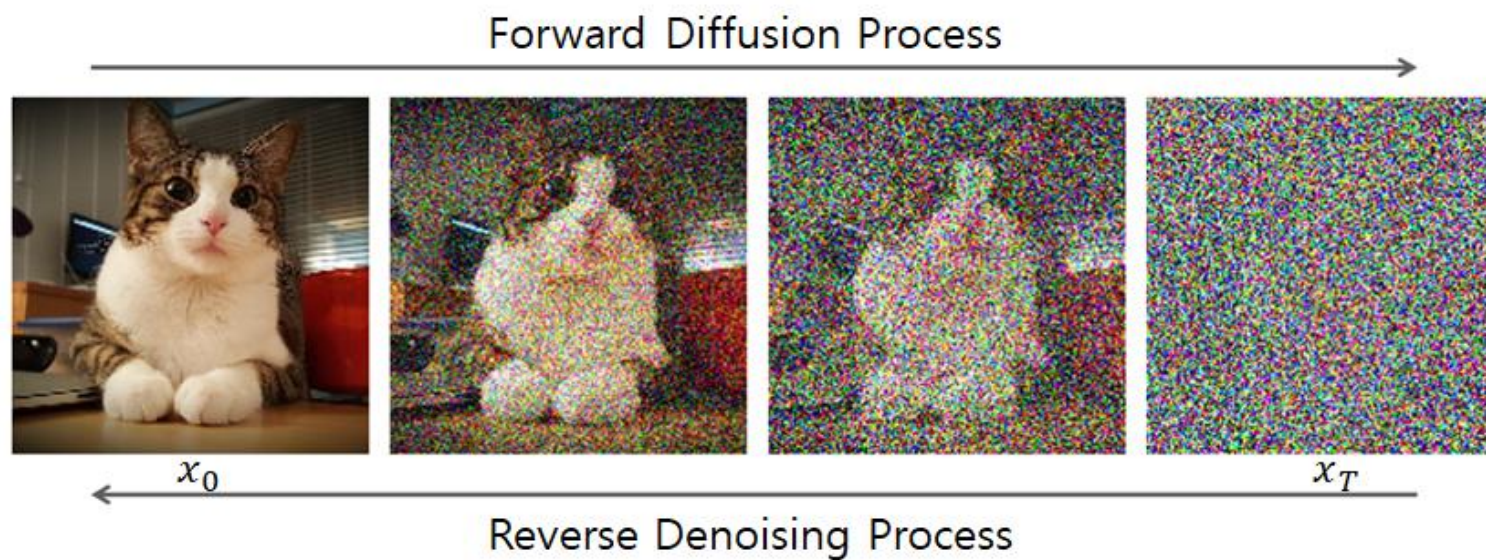


Dates: first arXiv submissions; * official reports / announcements. Spacing is schematic; selected works only.
Each milestone links to its primary source. Sources checked 30 Sep 2026.

Denoising Diffusion Probabilistic Models (DDPM)

$$x_0 \sim q(x_0)$$

Real data
distribution



Denoising Diffusion Probabilistic Models (DDPM)

Forward Process:

$q(x_t|x_{t-1})$ is known, it is the forward transition we defined in the forward process

$$q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) := \prod_{t=1}^T q(\mathbf{x}_t \mid \mathbf{x}_{t-1}), \quad q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I})$$
$$\alpha_t = 1 - \beta_t \quad q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t} \mathbf{x}_{t-1}, (1 - \alpha_t) \mathbf{I}).$$

Reverse Process:

$q(x_{t-1}|x_t)$ is unknown, so we trained a model to approximate it, it learns the reverse of noise generation

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t), \quad p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(\mu_\theta(x_t, t), \sigma_t^2 I).$$

here if we know x_0 , the reverse process is Gaussian, if we don't know, reverse can be also modeling using Gaussian when each step is small enough based on Bayes rule.

Denoising Diffusion Probabilistic Models (DDPM)

Training: Minimize variational lower bound of the model $p_\theta(x_0)$

$$\mathbb{E} [-\log p_\theta(\mathbf{x}_0)] \leq \mathbb{E}_q \left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \right]$$

It can be decomposed to step-wise losses (for each step t)

! Note: we
know x_0 while
training

$$\mathbb{E}_q \left[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T | \mathbf{x}_0) \| p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t))}_{L_{t-1}} - \underbrace{\log p_\theta(\mathbf{x}_0 | \mathbf{x}_1)}_{L_0} \right]$$

Here, the true reverse step $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$ can be computed as a **closed form** of β_t

$$q(x_{t-1} | x_t, x_0) = \frac{q(x_t | x_{t-1}, x_0) q(x_{t-1} | x_0)}{q(x_t | x_0)}.$$

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}),$$

$$q(x_t | x_{t-1}, x_0) = q(x_t | x_{t-1}),$$

where $\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t} \mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t$ and $\tilde{\beta}_t := \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t$ $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$

Denoising Diffusion Probabilistic Models (DDPM)

where $\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t}\mathbf{x}_t$ and $\tilde{\beta}_t := \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t}\beta_t$

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2 \right] + C$$

$$\begin{aligned} L_{t-1} - C &= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{1}{2\sigma_t^2} \left\| \tilde{\boldsymbol{\mu}}_t \left(\mathbf{x}_t(\mathbf{x}_0, \epsilon), \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t(\mathbf{x}_0, \epsilon) - \sqrt{1 - \bar{\alpha}_t}\epsilon) \right) - \boldsymbol{\mu}_\theta(\mathbf{x}_t(\mathbf{x}_0, \epsilon), t) \right\|^2 \right] \\ &= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{1}{2\sigma_t^2} \left\| \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t(\mathbf{x}_0, \epsilon) - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}}\epsilon \right) - \boldsymbol{\mu}_\theta(\mathbf{x}_t(\mathbf{x}_0, \epsilon), t) \right\|^2 \right] \end{aligned}$$

By parameterizing the mean

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \tilde{\boldsymbol{\mu}}_t \left(\mathbf{x}_t, \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t}\epsilon_\theta(\mathbf{x}_t)) \right) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t) \right)$$

Loss can be simplified as:

$$\mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1 - \bar{\alpha}_t)} \|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t)\|^2 \right]$$

Denoising Diffusion Probabilistic Models (DDPM)

$$L_T = D_{\text{KL}}(q(x_T | x_0) \| p(x_T)). \quad L_0 = -\log p_\theta(x_0 | x_1). \quad L_1 + \dots + L_{T-1} = \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1 - \bar{\alpha}_t)} \|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t)\|^2 \right]$$

DDPM authors found it performs very good if using the reverse transition loss and also simplify it to

$$L_{\text{simple}} = \mathbb{E}_{t, x_0, \epsilon} \left[\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t)\|^2 \right]$$

Denoising Diffusion Probabilistic Models (DDPM)

Algorithm 1 Training

```
1: repeat  
2:    $\mathbf{x}_0 \sim q(\mathbf{x}_0)$   
3:    $t \sim \text{Uniform}(\{1, \dots, T\})$   
4:    $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$   
5:   Take gradient descent step on  
        $\nabla_{\theta} \|\epsilon - \epsilon_{\theta}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t)\|^2$   
6: until converged
```

Algorithm 2 Sampling

```
1:  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$   
2: for  $t = T, \dots, 1$  do  
3:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  if  $t > 1$ , else  $\mathbf{z} = \mathbf{0}$   
4:    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$   
5: end for  
6: return  $\mathbf{x}_0$ 
```

Deterministic Sampling: Denoising Diffusion Implicit Model (DDIM)

$$q(x_t | x_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)I),$$

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I). \quad x_{t-1} \sim \mathcal{N}(\sqrt{\bar{\alpha}_{t-1}}x_0, (1 - \bar{\alpha}_{t-1})I).$$

$$x_{t-1} = \underbrace{\sqrt{\bar{\alpha}_{t-1}}x_0}_{\text{Signal}} + \underbrace{A_t\epsilon}_{\text{Current noise direction}} + \underbrace{\sigma_t z}_{\text{Random noise}}$$

$$\text{Var}(A_t\epsilon + \sigma_t z) = A_t^2 + \sigma_t^2, \quad A_t^2 + \sigma_t^2 = 1 - \bar{\alpha}_{t-1}.$$

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\hat{x}_0 + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2}\epsilon_\theta(x_t, t) + \sigma_t z.$$

When $\sigma_t = 0$, $x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\hat{x}_0 + \sqrt{1 - \bar{\alpha}_{t-1}}\epsilon_\theta(x_t, t).$

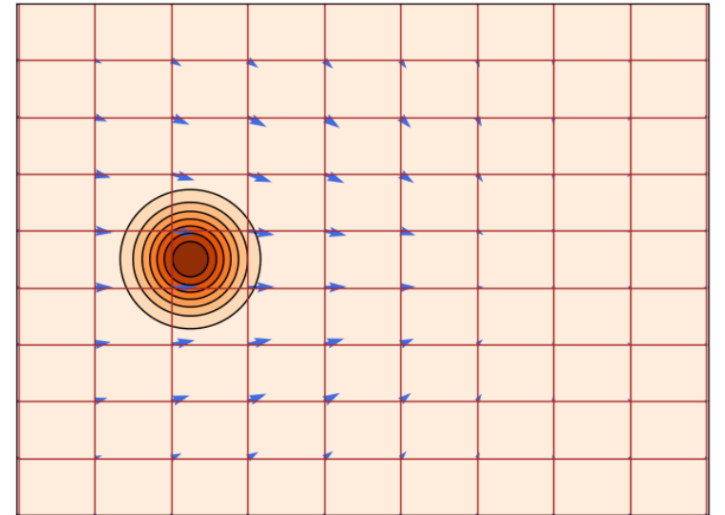
$$x_{t-1} = \sqrt{\frac{\bar{\alpha}_{t-1}}{\bar{\alpha}_t}}x_t + \left(\sqrt{1 - \bar{\alpha}_{t-1}} - \sqrt{\frac{\bar{\alpha}_{t-1}}{\bar{\alpha}_t}}\sqrt{1 - \bar{\alpha}_t} \right) \epsilon_\theta(x_t, t). \quad \text{It becomes a deterministic ODE (DDIM)}$$

$$\text{When } \sigma_t^2 = \tilde{\beta}_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t}\beta_t, \quad x_{t-1} = \frac{1}{\sqrt{\bar{\alpha}_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) + \sqrt{\tilde{\beta}_t} z \quad \text{It becomes DDPM}$$

Flow Matching

A generative modeling framework that learns a velocity (vector) field to transport samples from a source distribution p_0 (normally Gaussian Noise) to a target distribution p_1

$$\frac{dx_t}{dt} = v_\theta(x_t, t), \quad t : 0 \rightarrow 1.$$



Rectified Flow Matching

We define a **straight-line** interpolation between noise x_0 and data x_1 :

$$x_t = (1 - t)x_0 + tx_1,$$

The target velocity for each sample pair is therefore:

$$u_t = x_1 - x_0.$$

The training objective is simply:

$$\mathcal{L} = \mathbb{E}[\|v_\theta(x_t, t) - (x_1 - x_0)\|^2].$$

Training:

```
def loss(self, x0, y):
    t = torch.randint(0, self.T, (x0.shape[0],), device=x0.device)
    eps = torch.randn_like(x0)
    ab = self.abar[t][:, None, None, None]
    xt = ab.sqrt() * x0 + (1 - ab).sqrt() * eps
    return F.mse_loss(self.model(xt, t.float(), y), eps)
```

```
def loss(self, x1, y):
    x0 = torch.randn_like(x1)
    t = torch.rand(x1.shape[0], device=x1.device)
    tb = t[:, None, None, None]
    xt = (1 - tb) * x0 + tb * x1
    return F.mse_loss(self.model(xt, t * (self.T - 1), y), x1 - x0)
```

Sampling:

```
@torch.no_grad()
def sample(self, y, steps, algo="ddim", w=0.0, shape=(1, 32, 32)):
    eta = 1.0 if algo == "ddpm" else 0.0
    n, device = y.shape[0], y.device
    x = torch.randn(n, *shape, device=device)
    ts = torch.linspace(self.T - 1, 0, steps).long().to(device)
    for i in range(steps):
        t = ts[i]
        eps = self._eps(x, t.repeat(n).float(), y, w)
        ab = self.abar[t]
        # no per-step clamp: clamping x0 each step biases the fine-step (high NFE) trajectory
        x0 = (x - (1 - ab).sqrt() * eps) / ab.sqrt()
        if i == steps - 1:
            return x0.clamp(-1, 1)
        abp = self.abar[ts[i + 1]]
        sigma = eta * ((1 - abp) / (1 - ab)).sqrt() * (1 - ab / abp).sqrt()
        x = abp.sqrt() * x0 + (1 - abp - sigma**2).clamp(min=0).sqrt() * eps + sigma * torch.randn_like(x)
    return x
```

```
@torch.no_grad()
def sample(self, y, steps, algo="euler", w=0.0, shape=(1, 32, 32)):
    n, device = y.shape[0], y.device
    x = torch.randn(n, *shape, device=device)
    dt = 1.0 / steps
    for i in range(steps):
        t = torch.full((n,), i * dt, device=device) * (self.T - 1)
        v = self._v(x, t, y, w)
        x = x + v * dt
    return x
```

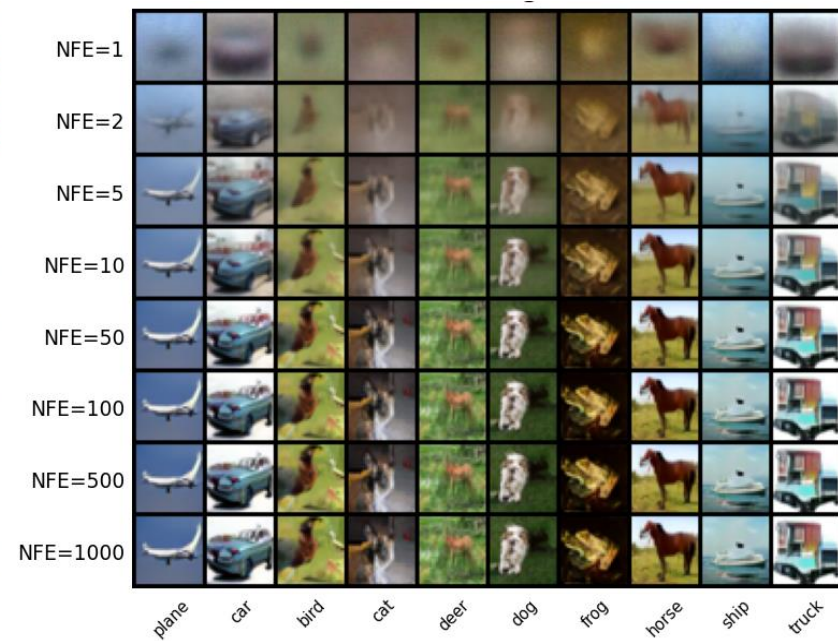
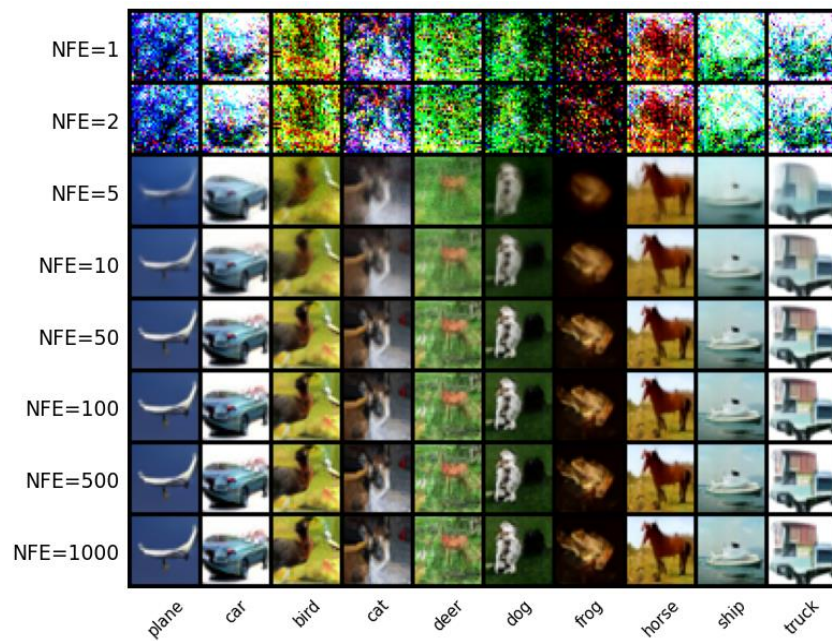
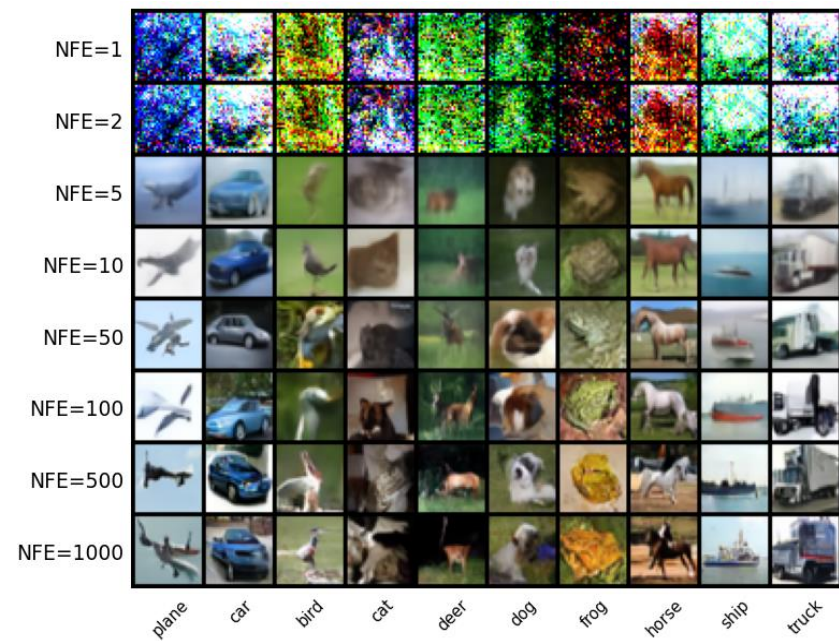
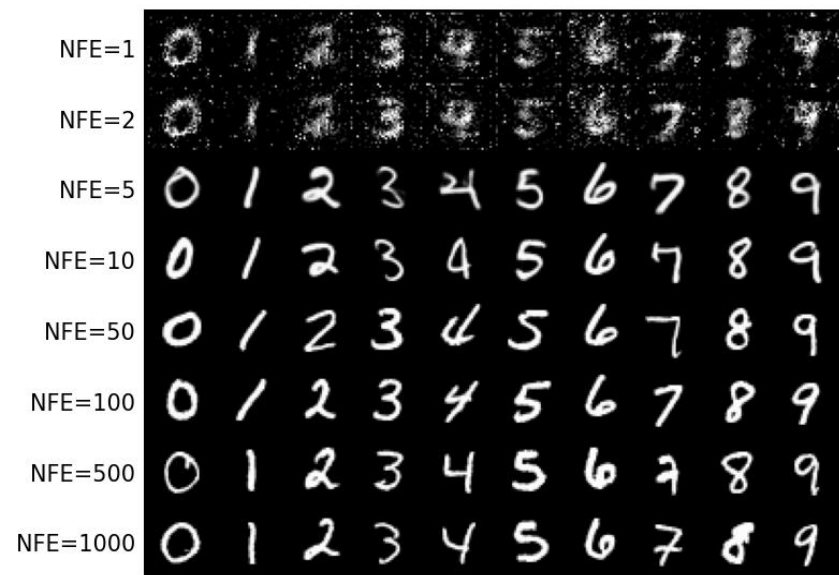
Experiment

Train and test DDPM, DDIM and Flow Matching Model on MNIST and CIFAR

DDPM

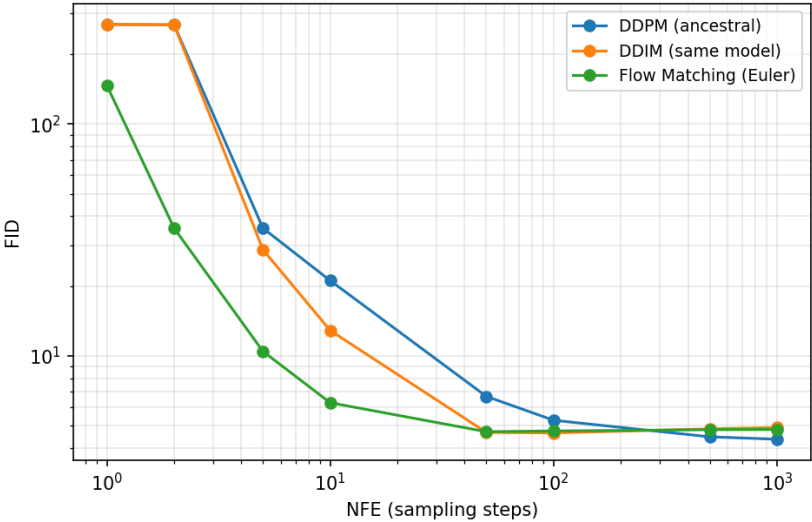
DDIM

Flow Matching



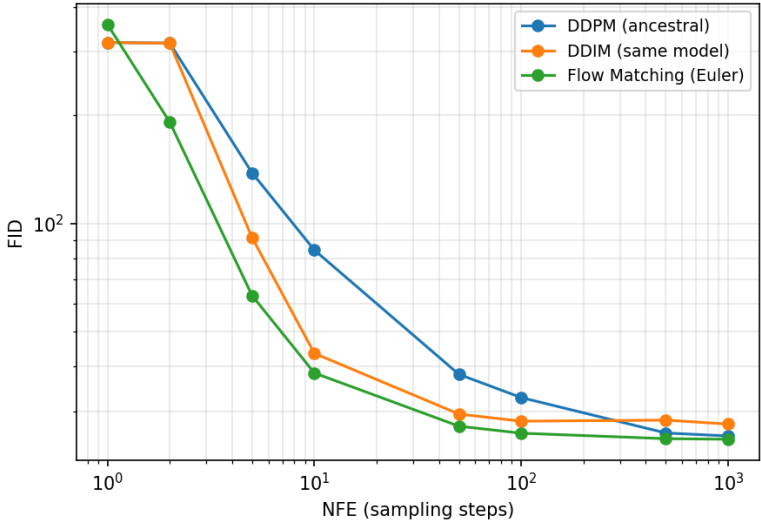
Result

Inception FID vs NFE (2000 samples/config, MNIST test ref)

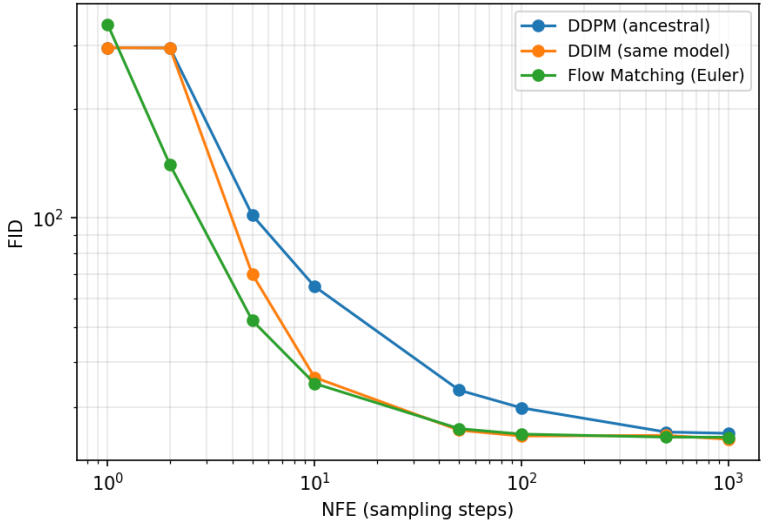


MNIST

CIFAR-10 FID vs NFE (w=0.0, 2000 samples)



CIFAR-10 FID vs NFE (w=1.5, 2000 samples)



CIFAR

Diffusion Meets Flow Matching: Two Sides of the Same Coin

Diffusion models

$$z_t = \alpha_t x + \sigma_t \epsilon, \quad \epsilon \sim \mathcal{N}(0, I).$$

Flow matching

$$z_t = (1 - t)x + t\epsilon$$

This corresponds to the diffusion forward process if the noise is Gaussian and we use the schedule:

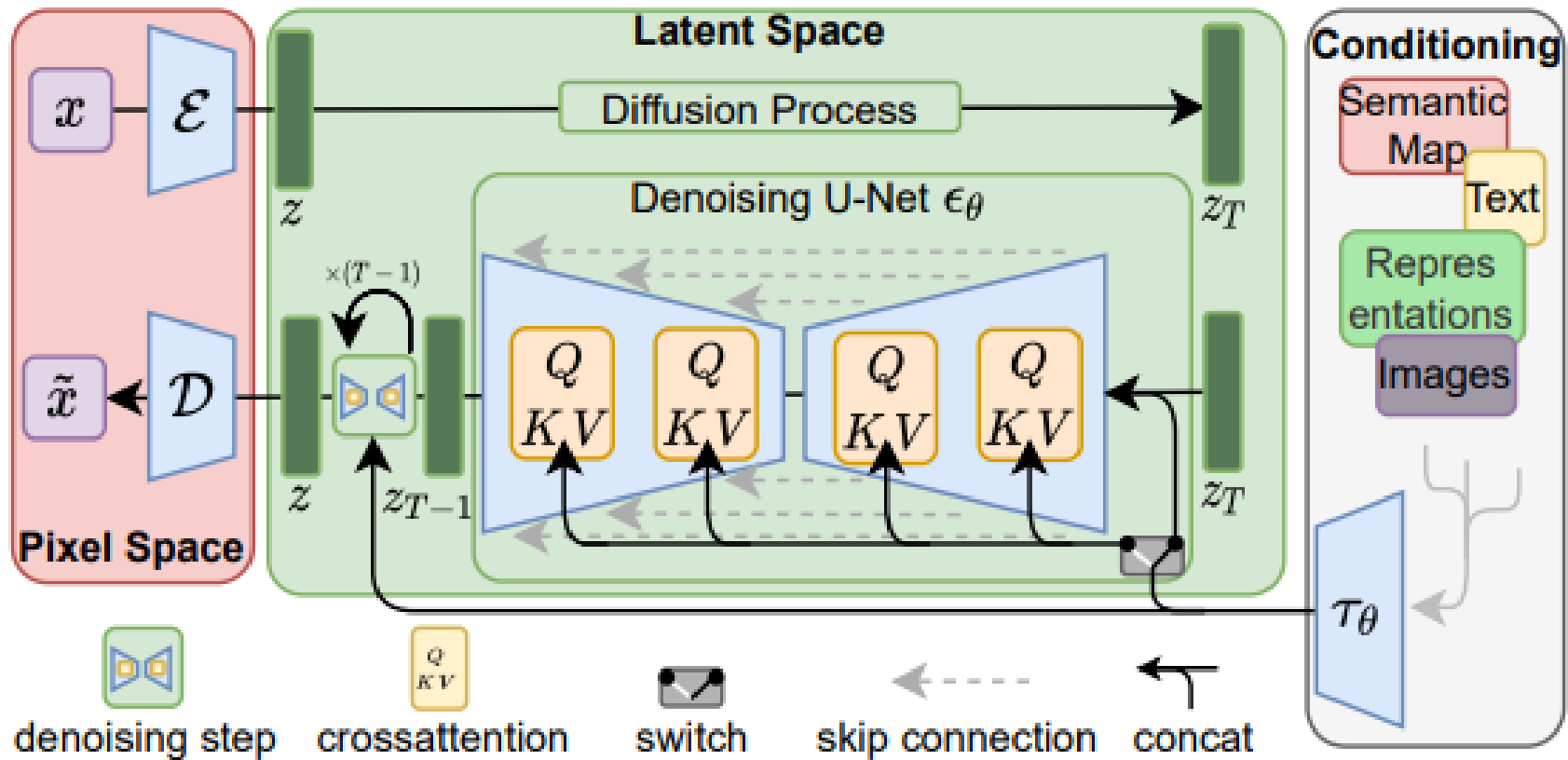
$$\alpha_t = 1 - t, \sigma_t = t$$

Diffusion Meets Flow Matching: Two Sides of the Same Coin

$$\tilde{\mathbf{z}}_s = \tilde{\mathbf{z}}_t + \text{Network output} \cdot (\eta_s - \eta_t)$$

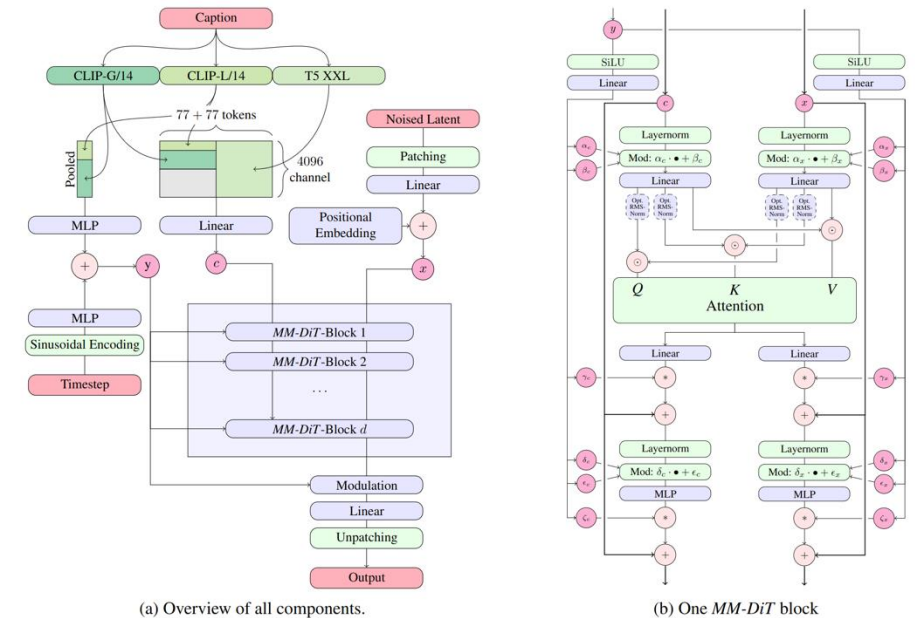
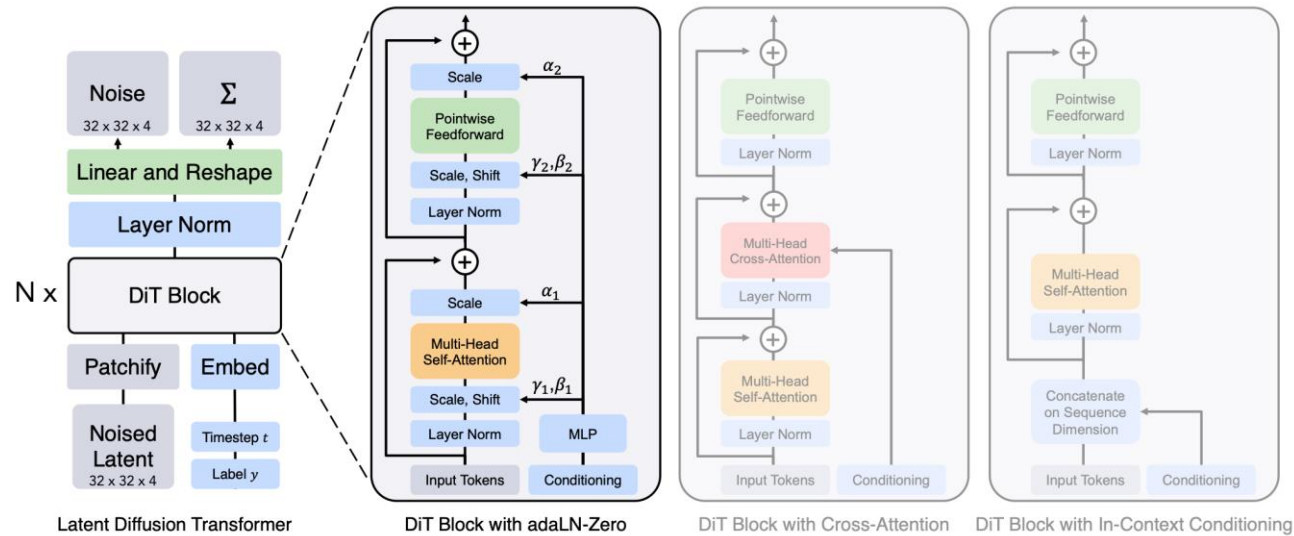
Network Output	Reparametrization
$\hat{\mathbf{x}}$ -prediction	$\tilde{\mathbf{z}}_t = \mathbf{z}_t / \sigma_t$ and $\eta_t = \alpha_t / \sigma_t$
$\hat{\mathbf{e}}$ -prediction	$\tilde{\mathbf{z}}_t = \mathbf{z}_t / \alpha_t$ and $\eta_t = \sigma_t / \alpha_t$
$\hat{\mathbf{u}}$ -flow matching vector field	$\tilde{\mathbf{z}}_t = \mathbf{z}_t / (\alpha_t + \sigma_t)$ and $\eta_t = \sigma_t / (\alpha_t + \sigma_t)$

Latent Diffusion Model: Using VAE to compress dimension



Stable Diffusion 1.0, XL, 1.5
Diffusion + Unet

Diffusion Transformer (DiT)



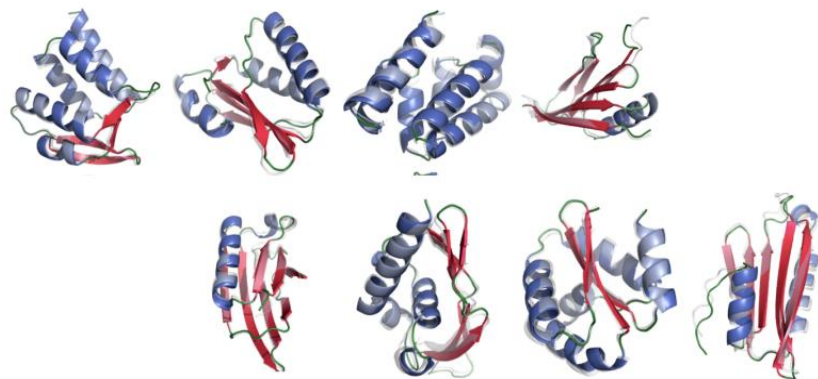
The Diffusion Transformer (DiT) architecture. Left: Conditional latent DiT models. The input latent is decomposed into patches and processed by several DiT blocks. Right: Details of our DiT blocks. We experiment with variants of standard transformer blocks that incorporate conditioning via adaptive layer norm, cross-attention and extra input tokens. Adaptive layer norm works best.

Stable Diffusion 3.0
Flow Matching + DiT

DiT at SCALE



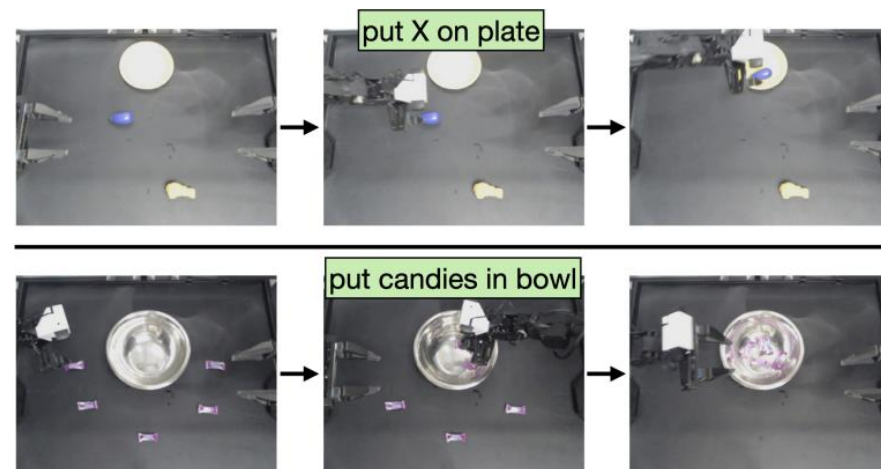
Text-2-Image
Qwen-Image



Protein Design
RFDiffusion3



Text-2-Video
Seedance



Robot Action Model
Cosmos Policy

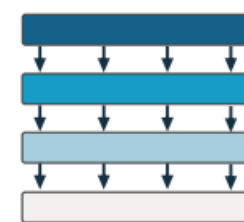
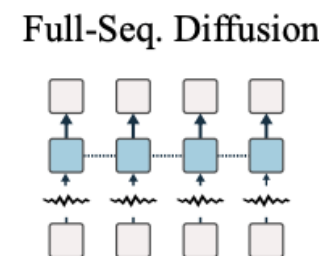
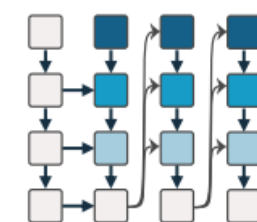
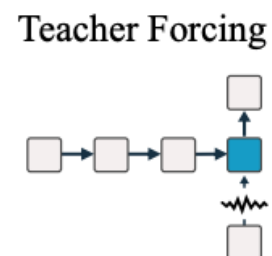
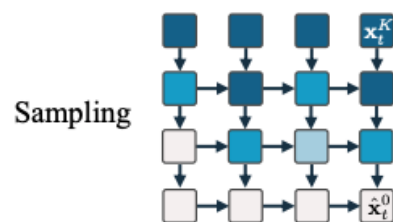
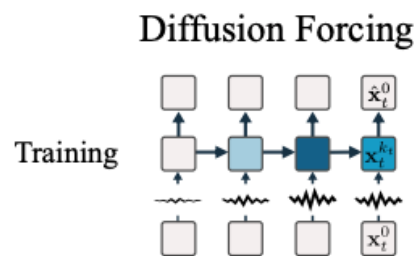
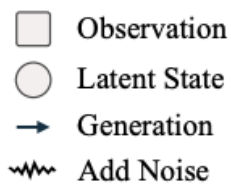
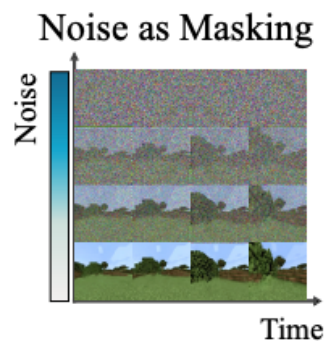
Related Research

1. Scaling across domains
 - Images / Videos / Proteins / Actions / ...
2. Faster generation
 - Many step -> Few step -> One step
3. Conditional control
 - CFG, ...
 - Uncond: $p(x)$, cond: $p(x|c)$

$$\epsilon_{cfg} = \epsilon_{uncond} + s(\epsilon_{cond} - \epsilon_{uncond})$$

Related Research

- 4. Sequential generation & long-horizon modeling
 - Diffusion Forcing, ...



Reference

1. Ho et al., *DDPM*, NeurIPS 2020
2. Song et al., *DDIM*, ICLR 2021
3. Lipman et al., *Flow Matching for Generative Modeling*, ICLR 2023
4. Rombach et al., *Latent Diffusion Models*, CVPR 2022
5. Peebles & Xie, *Diffusion Transformers (DiT)*, ICCV 2023
6. Esser et al., *Scaling Rectified Flow Transformers*, ICML 2024
7. Ingraham et al., *Programmable Protein Generation*, Nature 2023
8. Chen et al., *Diffusion Forcing*, NeurIPS 2024

Discussion

1. Why Gaussian noise specifically?
2. Does the diffusion model memorize the training images?
3. Is DDIM a different model from DDPM?
4. If DDIM is deterministic, where does diversity come from?
5. What exactly does Flow Matching learn?
6. Is Flow Matching always faster than diffusion?
7. Which is better: Diffusion or Flow Matching?
8. What is the main drawback of DiT?