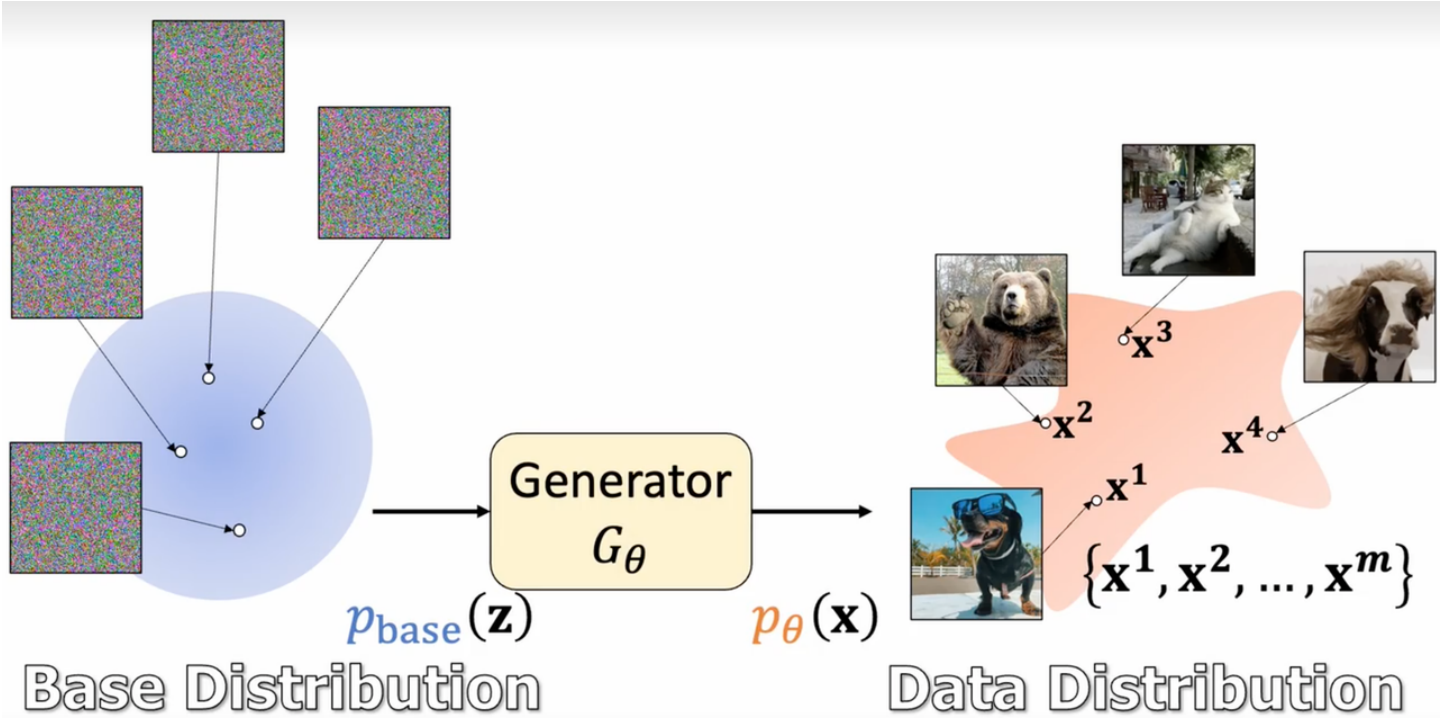
JiT PKUML组会

Back to Basics: Let Denoising Generative Models Denoise

Tianhong Li Kaiming He

MIT

Flow Matching



生成模型：从一个已知分布中采样，通过generator，变换为目标分

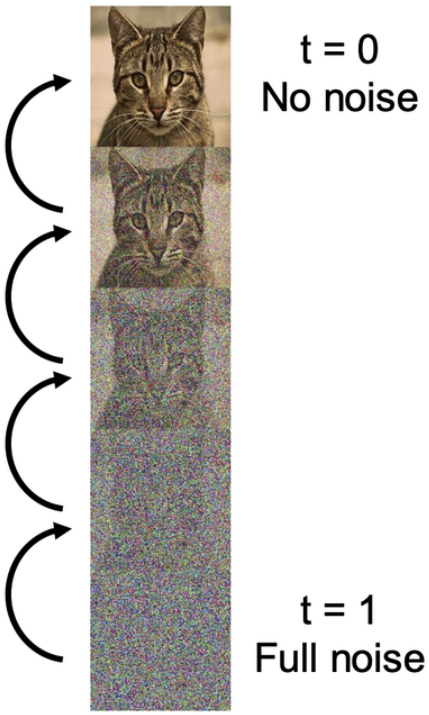
Diffusion Models: Intuition

Pick a **noise distribution** $z \sim p_{noise}$ (Usually unit Gaussian)

Consider data x corrupted under varying **noise levels** t to give noisy data x_t

Train a neural network to **remove a little bit of noise**: $f_\theta(x_t, t)$

At inference time, sample $x_1 \sim p_{noise}$ and apply f_θ many times in sequence to generate a noiseless sample x_0



Diffusion Models: Rectified Flow

Suppose we have a simple p_{noise} (e.g. Gaussian) and samples from p_{data}

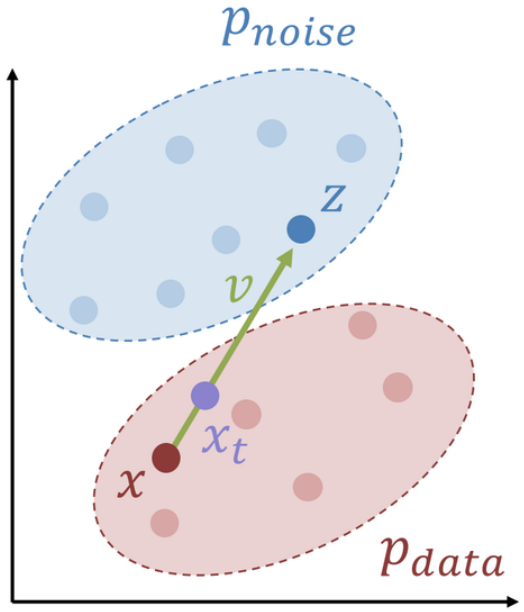
On each training iteration, sample:

$z \sim p_{noise} \quad x \sim p_{data} \quad t \sim Uniform[0, 1]$

Set $x_t = (1-t)x + tz$, $v = z - x$

Train a neural network to predict v:

$$L = \|f_\theta(x_t, t) - v\|_2^2$$

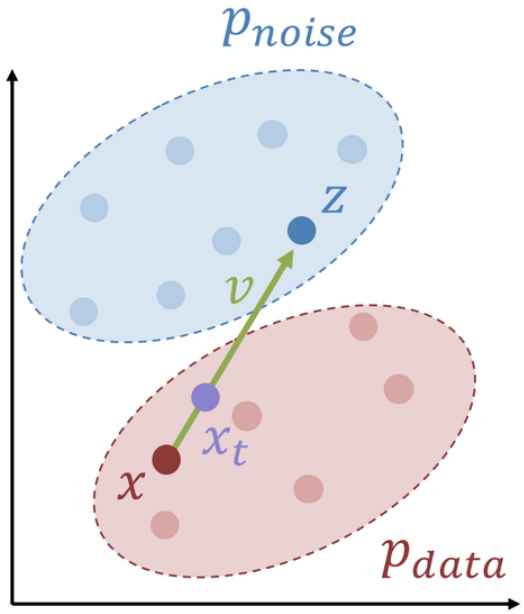


Liu et al, "Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow", 2022
Lipman et al, "Flow Matching for Generative Modeling", 2022

Diffusion Models: Rectified Flow

Core training loop is just a few lines of code!

```
for x in dataset:
    z = torch.randn_like(x)
    t = random.uniform(0, 1)
    xt = (1 - t) * x + t * z
    v = model(xt, t)
    loss = (z - x - v).square().sum()
```



Liu et al, "Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow", 2022
Lipman et al, "Flow Matching for Generative Modeling", 2022

3.1. Background: Diffusion and Flows

Diffusion models can be formulated from the perspective of ODEs [5, 60, 37, 58]. We begin our formulation with the flow-based paradigm [37, 38, 1], *i.e.*, in the $\boldsymbol{v}$-space, as a simpler starting point, and then discuss other spaces.

Consider a data distribution $\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})$ and a noise distribution $\boldsymbol{\epsilon} \sim p_{\text{noise}}(\boldsymbol{\epsilon})$ (*e.g.*, $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$). During training, a noisy sample $\boldsymbol{z}_t$ is an interpolation: $\boldsymbol{z}_t = a_t\,\boldsymbol{x} + b_t\,\boldsymbol{\epsilon}$, where a_t and b_t are pre-defined noise schedules at time $t \in [0, 1]$. In this paper, we use a linear schedule [37, 38, 1][1]: $a_t = t$ and $b_t = 1 - t$. This gives:

$$\boldsymbol{z}_t = t\,\boldsymbol{x} + (1 - t)\,\boldsymbol{\epsilon}, \tag{1}$$

which leads to $\boldsymbol{z}_t \sim p_{\text{data}}$ when t=1. We use the logit-normal distribution over t [15], *i.e.*, $\text{logit}(t) \sim \mathcal{N}(\mu, \sigma^2)$.

A flow velocity $\boldsymbol{v}$ is defined as the time-derivative of $\boldsymbol{z}$, that is, $\boldsymbol{v}_t = \boldsymbol{z}'_t = a'_t\boldsymbol{x} + b'_t\boldsymbol{\epsilon}$. Given Eq. (1), we have:

$$\boldsymbol{v}_t = \boldsymbol{x} - \boldsymbol{\epsilon}. \tag{2}$$

$$\boldsymbol{v} = \boldsymbol{x} - \boldsymbol{\epsilon}. \tag{2}$$

The flow-based methods [37, 38, 1] minimize a loss function defined as:

$$\mathcal{L} = \mathbb{E}_{t,\boldsymbol{x},\boldsymbol{\epsilon}} \left\| \boldsymbol{v}_\theta(\boldsymbol{z}_t, t) - \boldsymbol{v} \right\|^2, \tag{3}$$

where $\boldsymbol{v}_\theta$ is a function parameterized by θ. While $\boldsymbol{v}_\theta$ is often the *direct* output of a network $\boldsymbol{v}_\theta = \texttt{net}_\theta(\boldsymbol{z}_t, t)$ [37, 38, 1], it can also be a transform of it, as we will elaborate.

Given the function $\boldsymbol{v}_\theta$, sampling is done by solving an ordinary differential equation (ODE) for $\boldsymbol{z}$ [37, 38, 1]:

$$d\boldsymbol{z}_t / dt = \boldsymbol{v}_\theta(\boldsymbol{z}_t, t), \tag{4}$$

starting from $\boldsymbol{z}_0 \sim p_{\text{noise}}$ and ending at $t = 1$. In practice, this ODE can be approximately solved using numerical solvers. By default, we use a 50-step Heun.

JiT!

当前的去噪扩散模型并不以传统意义上的‘去噪’方式运作，即它们并**不直接生成清晰图像**。相反，神经网络预测的是噪声或某种跟噪声相关的量（比如速度）。

似乎在数学上这些无太大所谓，因为这些变量是可以互相转化的，

	(a) $\boldsymbol{x}$-pred $\boldsymbol{x}_\theta := \texttt{net}_\theta(\boldsymbol{z}_t, t)$	(b) $\boldsymbol{\epsilon}$-pred $\boldsymbol{\epsilon}_\theta := \texttt{net}_\theta(\boldsymbol{z}_t, t)$	(c) $\boldsymbol{v}$-pred $\boldsymbol{v}_\theta := \texttt{net}_\theta(\boldsymbol{z}_t, t)$
(1) $\boldsymbol{x}$-loss: $\mathbb{E}\|\boldsymbol{x}_\theta - \boldsymbol{x}\|^2$	$\boldsymbol{x}_\theta$	$\boldsymbol{x}_\theta = (\boldsymbol{z}_t - (1-t)\boldsymbol{\epsilon}_\theta)/t$	$\boldsymbol{x}_\theta = (1-t)\boldsymbol{v}_\theta + \boldsymbol{z}_t$
(2) $\boldsymbol{\epsilon}$-loss: $\mathbb{E}\|\boldsymbol{\epsilon}_\theta - \boldsymbol{\epsilon}\|^2$	$\boldsymbol{\epsilon}_\theta = (\boldsymbol{z}_t - t\boldsymbol{x}_\theta)/(1-t)$	$\boldsymbol{\epsilon}_\theta$	$\boldsymbol{\epsilon}_\theta = \boldsymbol{z}_t - t\boldsymbol{v}_\theta$
(3) $\boldsymbol{v}$-loss: $\mathbb{E}\|\boldsymbol{v}_\theta - \boldsymbol{v}\|^2$	$\boldsymbol{v}_\theta = (\boldsymbol{x}_\theta - \boldsymbol{z}_t)/(1-t)$	$\boldsymbol{v}_\theta = (\boldsymbol{z}_t - \boldsymbol{\epsilon}_\theta)/t$	$\boldsymbol{v}_\theta$

Table 1. **All possible combinations** of defining the loss and network prediction in $\boldsymbol{x}$, $\boldsymbol{v}$, or $\boldsymbol{\epsilon}$ spaces. The direct network outputs are highlighted in colors. For any off-diagonal entry where the network output space differs from the loss space, a transformation on the network output is applied.

大家现在使用预测噪声/速度只是因为远古时期的DDPM消融：

Table 2: Unconditional CIFAR10 reverse process parameterization and training objective ablation. Blank entries were unstable to train and generated poor samples with out-of-range scores.

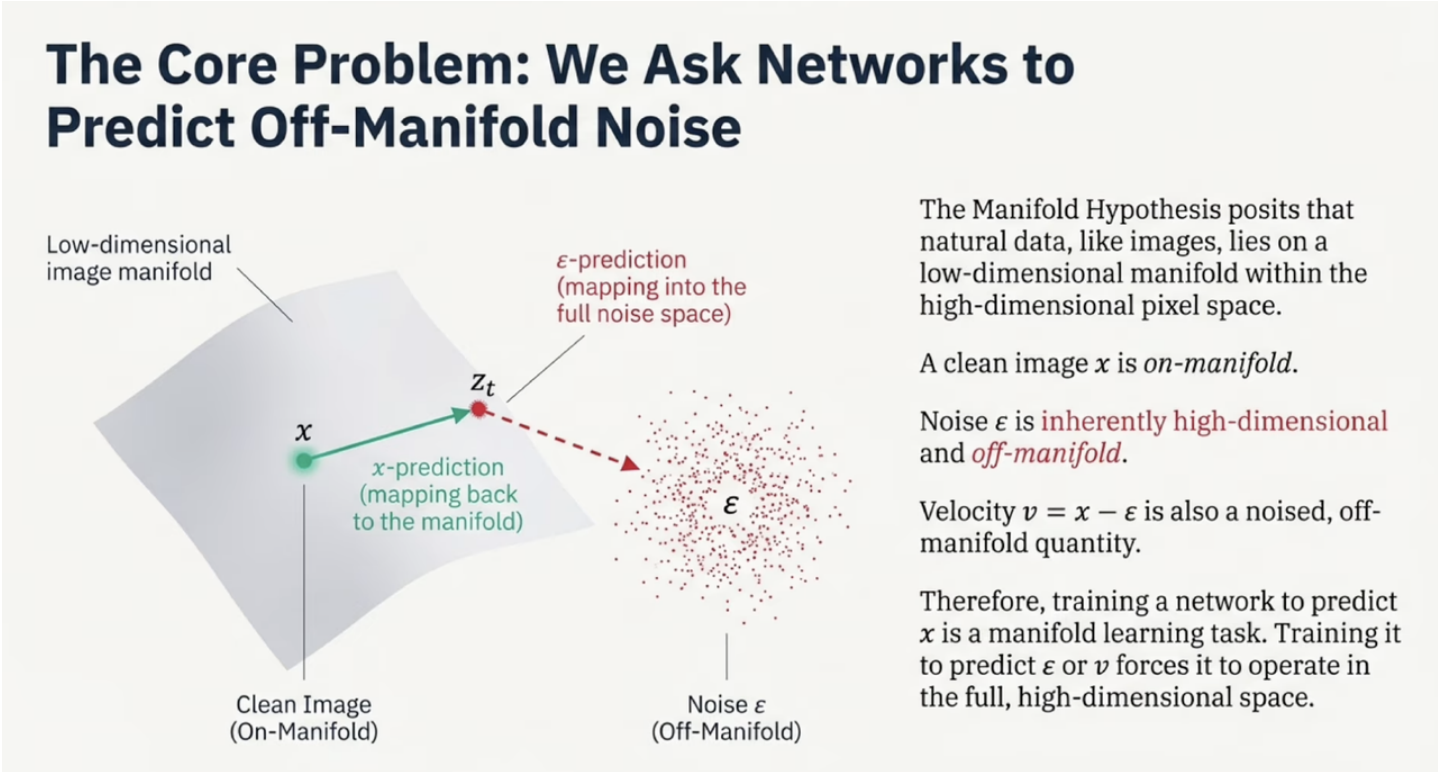
Objective	IS	FID
$\tilde{\mu}$ prediction (baseline)		
L, learned diagonal Σ	7.28±0.10	23.69
L, fixed isotropic Σ	8.06±0.09	13.22
$\|\|\tilde{\mu} - \tilde{\mu}_\theta\|\|^2$	–	–
ϵ prediction (ours)		
L, learned diagonal Σ	–	–
L, fixed isotropic Σ	7.67±0.13	13.51
$\|\|\tilde{\epsilon} - \epsilon_\theta\|\|^2$ (L_{simple})	**9.46±0.11**	**3.17**

However：

Tianhong&Kaiming: 干净图像与含噪量（包括噪声本身）的作用并非等同。

原始数据（干净图像）往往处于低维子流形

假设存在一个低维流形嵌入高维观测空间中的场景。在此高维空间中预测噪声需要高容量：网络需保留关于噪声的全部信息。相比之下，有限容量网络仍能预测纯净数据，因其仅需保留低维信息并滤除噪



https://www.youtube.com/watch?v=u5yKZzTTEHo

- **低维流形：** 自然图像实际上存在于超高维像素空间中一个非常低维的流形面上。

- **高维噪声：** 当我们向图像添加高斯噪声时，数据点会被抛出这个流形，进入广阔且无结构的空旷高维空间。
- **预测难度的差异：**
  - **预测噪声：** 噪声是高维且无结构的。为了完美预测它，神经网络需要巨大的容量来表示这些高频信息。
  - **预测图像：** 网络只需要学习流形的结构，并将噪声数据投影回流形面上。（低频信息更利于神经网络预测）

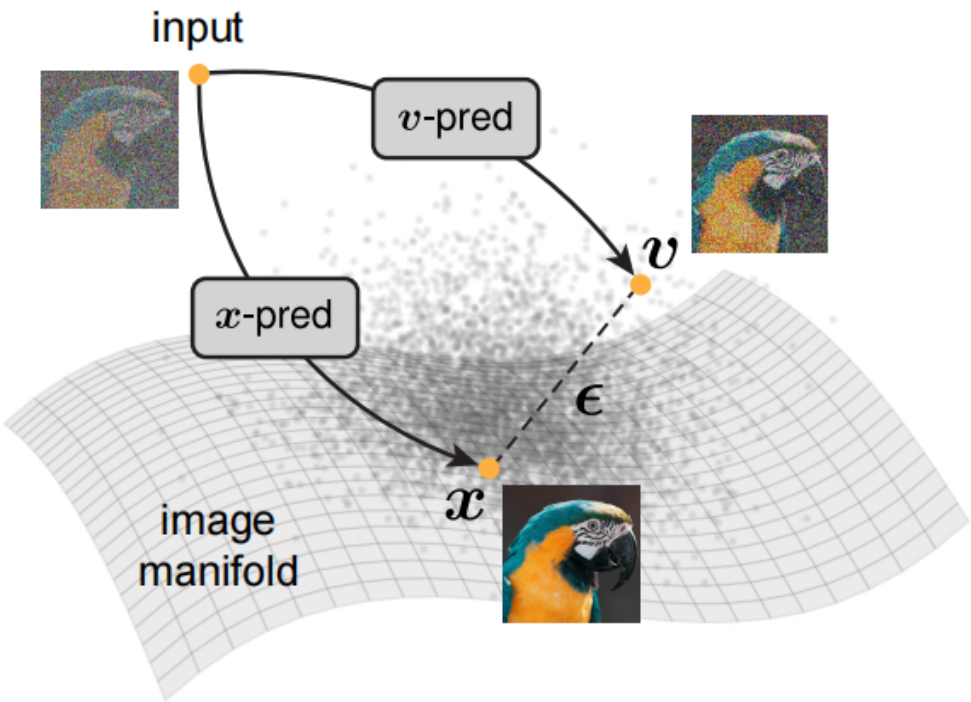
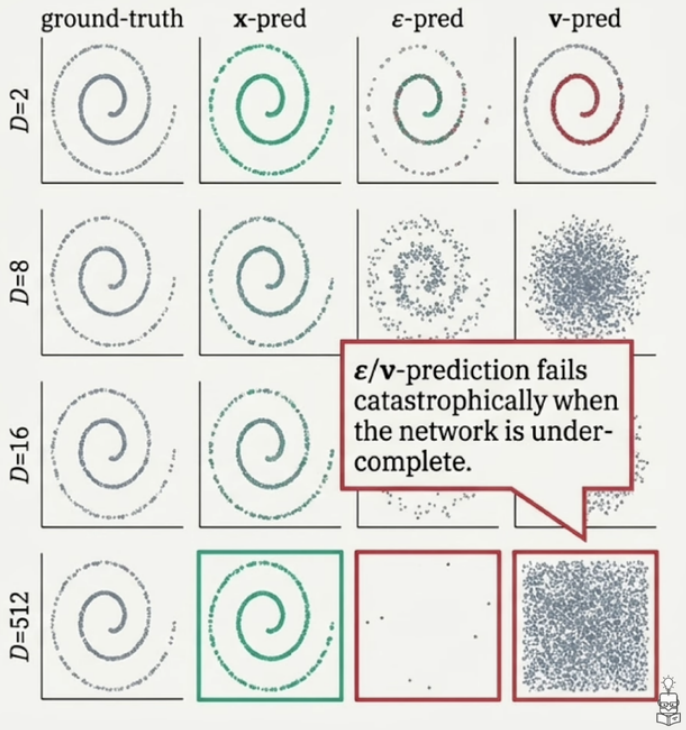


Figure 1. **The Manifold Assumption** [4] hypothesizes that natural images lie on a low-dimensional manifold within the high-dimensional pixel space. While a clean image $\boldsymbol{x}$ can be modeled as on-manifold, the noise $\boldsymbol{\epsilon}$ or flow velocity $\boldsymbol{v}$ (*e.g.*, $\boldsymbol{v} = \boldsymbol{x} - \boldsymbol{\epsilon}$) is inherently off-manifold. Training a neural network to predict a clean image (*i.e.*, $\boldsymbol{x}$-prediction) is *fundamentally different* from training it to predict noise or a noised quantity (*i.e.*, $\boldsymbol{\epsilon}/\boldsymbol{v}$-prediction).

Toy EXP:

A Toy Experiment Shows ε/v-Prediction Fails as Dimensionality Grows

- **Setup:** A 2D spiral data manifold is randomly projected into a high-dimensional D-space. A simple 5-layer MLP (256 hidden units) is trained to generate the data, without knowing the projection.
- **Observation:**
  - At low dimension (D=2), all prediction methods work.
  - As dimension increases to D=512, the network is under-complete. **x-prediction** still successfully recovers the 2D manifold structure.
  - ε-prediction and **v**-prediction fail catastrophically, as the network lacks the capacity to model high-dimensional noise.



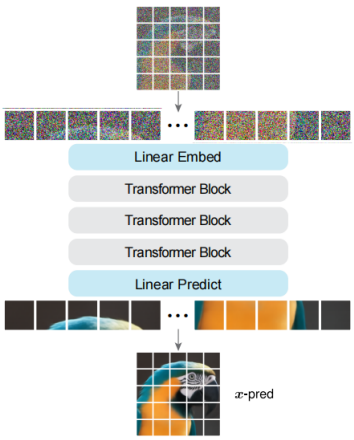


Figure 3. **The "Just image Transformer" (JiT) architecture**: simply a plain ViT [13] on patches of pixels for x-prediction.

架构

	x-pred	ϵ-pred	v-pred
x-loss	10.14	379.21	107.55
ϵ-loss	10.45	394.58	126.88
v-loss	8.62	372.38	96.53

(a) ImageNet 256×256, **JiT-B/16**

	x-pred	ϵ-pred	v-pred
x-loss	5.76	6.20	6.12
ϵ-loss	3.56	4.02	3.76
v-loss	3.55	3.63	3.46

(b) ImageNet 64×64, **JiT-B/4**

Table 2. **Results of all combinations** of loss space and network space (see Tab. 1), evaluated by FID-50K on ImageNet: **(a)** JiT-B/16 at 256 resolution, 768-d per patch; **(b)** JiT-B/4 at 64 resolution, 48-d per patch. We annotate catastrophic failures in red and reasonable results by green. Settings: 200 epochs, with CFG [22].

低维效果相差不大 高维只有predict x才能work

t-shift (μ)		x-pred	ϵ-pred	v-pred
(lower noise)	0.0	14.44	464.25	120.03
	–0.4	9.79	372.91	109.93
	–0.8	**8.62**	372.36	96.53
(higher noise)	–1.2	8.99	355.25	106.85

Table 3. **Noise-level shift** (JiT-B/16, ImageNet 256×256, FID-50K). We shift the noise level by adjusting μ in the logit-normal t-sampler [15]. An appropriate noise level is useful, but is not sufficient for addressing the catastrophic failure in ϵ-/v-prediction. Settings (the same as Tab. 2): 200 epochs, with CFG.

增加噪声有利于高分辨率生成，但不足以修正predict v

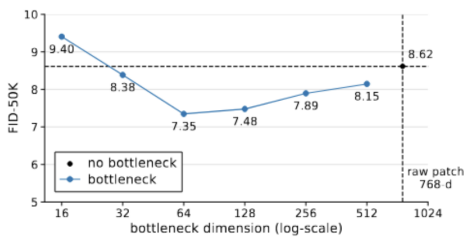


Figure 4. **Bottleneck linear embedding.** Results are for **JiT-B/16** on ImageNet 256×256. A raw patch is 768-dim (16×16×3) and is embedded by two sequential linear layers with an intermediate bottleneck dimension d' ($d' < 768$). Here, ***bottleneck embedding is generally beneficial***, and our x-prediction model can work decently even with aggressive bottlenecks as small as 32 or 16. Settings (the same as Tab. 3): 200 epochs, with CFG.

加入瓶颈层甚至有利于学习！

Algorithm 1 Training step

```
# net(z, t): JiT network
# x: training batch

t = sample_t()
e = randn_like(x)

z = t * x + (1 - t) * e
v = (x - z) / (1 - t)

x_pred = net(z, t)
v_pred = (x_pred - z) / (1 - t)

loss = l2_loss(v - v_pred)
```

Algorithm 2 Sampling step (Euler)

```
# z: current samples at t

x_pred = net(z, t)
v_pred = (x_pred - z) / (1 - t)

z_next = z + (t_next - t) * v_pred
```

Figure 5. **Qualitative Results.** Selected examples on ImageNet 512×512 using JiT-H/32. More uncurated results are in appendix.

	JiT-B/16	JiT-L/16
Baseline (SwiGLU, RMSNorm)	7.48 (6.32)	-
+ RoPE, qk-norm	6.69 (5.44)	-
+ in-context class tokens	5.49 (**4.37**)	3.39 (**2.79**)

Table 4. **"Just Advanced" Transformers** with *general-purpose* designs. All are **JiT/16** for ImageNet 256×256, with bottleneck patch embedding (128-d, Fig. 4), evaluated by FID-50K. Settings: 200 epochs, with CFG (and with CFG interval [33] in brackets).

resolution	model	len	patch dim	hiddens	params	Gflops	FID
256×256	JiT-B/16	256	768	768	131	25	4.37
512×512	JiT-B/32	256	3072	768	133	26	4.64
1024×1024	JiT-B/**64**	256	12288	768	141	30	4.82

Table 5. **ImageNet 1024×1024 with JiT-B/64**. All entries have roughly the same number of parameters and compute. Settings: if not specified here, the same as Tab. 4 (all are with CFG interval).

256×256	200-ep	600-ep	**512×512**	200-ep	600-ep
JiT-B/**16**	4.37	3.66	JiT-B/**32**	4.64	4.02
JiT-L/**16**	2.79	2.36	JiT-L/**32**	3.06	2.53
JiT-H/**16**	2.29	1.86	JiT-H/**32**	2.51	1.94
JiT-G/**16**	2.15	**1.82**	JiT-G/**32**	2.11	**1.78**

Table 6. **Scalability on ImageNet 256×256 and 512×512**, eval-

ImgNet 256×256	pre-training: token	perc.	self-sup.	params	Gflops	FID↓	IS↑
Latent-space Diffusion							
DiT-XL/2 [46]	SD-VAE	VGG	-	675+49M	119	2.27	278.2
SiT-XL/2 [40]	SD-VAE	VGG	-	675+49M	119	2.06	277.5
REPA [74], SiT-XL/2	SD-VAE	VGG	DINOv2	675+49M	119	1.42	305.7
LightningDiT-XL/2 [73]	VA-VAE	VGG	DINOv2	675+49M	119	1.35	295.3
DDT-XL/2 [71]	SD-VAE	VGG	DINOv2	675+49M	119	1.26	310.6
RAE [78], DiT^{DH}-XL/2	RAE	VGG	DINOv2	839+415M	146	**1.13**	262.6
***Pixel**-space (non-diffusion)*							
JetFormer [65]	-	-	-	2.8B	-	6.64	-
FractalMAR-H [36]	-	-	-	848M	-	6.15	348.9
***Pixel**-space Diffusion*							
ADM-G [12]	-	-	-	559M	1120	7.72	172.7
RIN [28]	-	-	-	320M	334	3.95	216.0
SiD [25] , UViT/2	-	-	-	2B	555	2.44	256.3
VDM++, UViT/2	-	-	-	2B	555	2.12	267.7
SiD2 [26], UViT/2	-	-	-	N/A	137	1.73	-
SiD2 [26], UViT/1	-	-	-	N/A	653	**1.38**	-
PixelFlow [6], XL/4	-	-	-	677M	2909	1.98	282.1
PixNerd [70], XL/16	-	-	DINOv2	700M	134	2.15	297
JiT-B/16	-	-	-	131M	25	3.66	275.1
JiT-L/16	-	-	-	459M	88	2.36	298.5
JiT-H/16	-	-	-	953M	182	1.86	303.4
JiT-G/16	-	-	-	2B	383	1.82	292.6

Table 7. **Reference results on ImageNet 256×256.** FID [21] and IS [53] of 50K samples are evaluated. The "pre-training" columns list the external models required to obtain the results (note that the perceptual loss [77] uses a pre-trained VGG classifier [56]). The parameters include the generator and tokenizer decoder (used at inference-time), but exclude other pre-trained components. The Giga-flops are measured for a single forward pass (not counting the tokenizer) and are roughly proportional to the computational cost of an iteration during both training and inference (for the multi-scale method [6], we measure the finest level).

Table 10. **Comparisons with pre-conditioners** (FID-50K, ImageNet 256, JiT-B/16). The settings are the same as Tab. 2 (a).

FID-50K	**JiT-B/16**	**JiT-L/16**
$\boldsymbol{v}$-loss only	4.37	2.79
w/ cls loss	**4.14**	**2.50**

Table 11. **Exploration: additional classification loss**. We do ***not*** use this loss in any other experiments. Settings: ImageNet 256×256, 200-ep, with CFG interval.