New Pre-training Paradigms from a Inference-First Perspective

Jiaming Song

Luma AI

Video Models’ 1 year birthday

Native Multi-Modal Generation

The current "hype"

Prompt: <img> While chameleons are masters of camouflage in the natural world, let's see how they might fare in a man-made setting.
1. Analyze the colors, patterns, and textures in this image and identify the most challenging spot for a chameleon to blend in.
2. Then generate an image of a chameleon with the optimal pattern to camouflage itself.
3. Then, rate and explain how successful each attempt is on a scale from 1-10.

Response: The image is of a stop sign in a forested area, with some sunlight filtering through the trees. The sign has a red octagonal shape with a white border and the word "STOP" written in white letters. In the background, there are green leaves and trees.

Challenging Spot for a Chameleon: The stop sign would be the most challenging spot for a chameleon to blend in due to its bright red color and bold white letters, which would stand out in the forested environment.

Optimal Pattern for Camouflage: To camouflage itself on the stop sign, a chameleon would need to have a pattern that would break up its outline and make it less conspicuous. A pattern with shades of brown and red would help the chameleon blend in more effectively. Additionally, a mottled or dappled pattern would help to scatter light and further reduce the chameleon's visibility.

Rating and Explanation: A rating of 2/10 would be given for the chameleon's ability to blend in on the stop sign. While the chameleon's green and brown pattern helps to break up its outline, the bright red color of the stop sign will still make it stand out in the forested environment.

From
- Vision Language Models (Text + Image in, Text out)
- Diffusion Models (Text + Image in, Image out)

To:
- Interleaved Models (Text + Image in, Text + Image out)

A lot of papers in the past year

Chameleon: Mixed-Modal Early-Fusion Foundation Mo

Chame

SHOW-O: ONE SINGLE TRANSFORMER TO UNIFY MU

Jinh
Kevi

Transfusion: Predict the Next Token and Diffuse Images with One Multi-Modal Model

Unified Multimodal Discrete Diffusion

C
Mich

Alexander Swerdlow* Mihir Prabhudesai* Siddharth Gandhi

Deepak Pathak Katerina Fragkiadaki

Text: Discrete AR
Image: Discrete AR

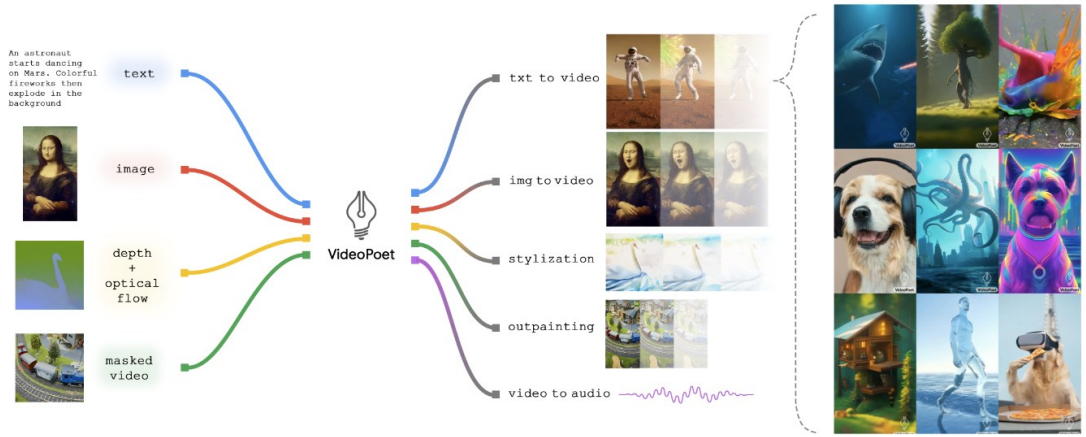
Text: Discrete AR
Image: Discrete Diffusion

Text: Discrete AR
Image: Continuous Diffusion

Text: Discrete Diffusion
Image: Discrete Diffusion

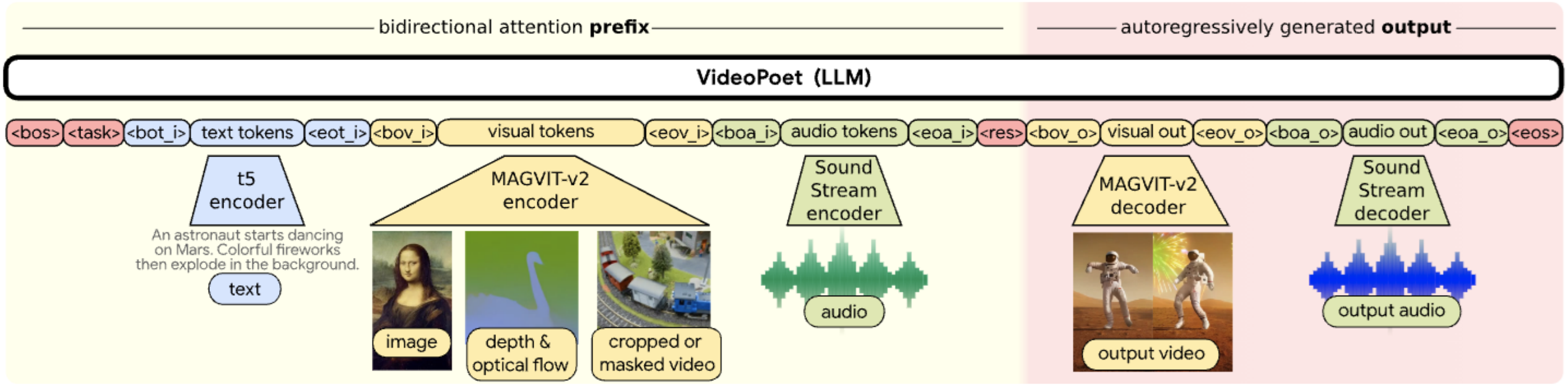
All based on combinations of Discrete AR / Discrete Diffusion / Continuous Diffusion etc..

Why not stick to next-token prediction?



VideoPoet: A Large Language Model for Zero-Shot Video Generation

Dan Kondratyuk [*1] Lijun Yu [*12] Xiuye Gu [*1] José Lezama [*1] Jonathan Huang [*1] Grant Schindler [1]
Rachel Hornung [1] Vighnesh Birodkar [1] Jimmy Yan [1] Ming-Chang Chiu [1] Krishna Somandepalli [1]
Hassan Akbari [1] Yair Alon [1] Yong Cheng [1] Josh Dillon [1] Agrim Gupta [1] Meera Hahn [1] Anja Hauth [1]
David Hendon [1] Alonso Martinez [1] David Minnen [1] Mikhail Sirotenko [1] Kihyuk Sohn [1] Xuan Yang [1]
Hartwig Adam [1] Ming-Hsuan Yang [1] Irfan Essa [1] Huisheng Wang [1] David A. Ross [1] Bryan Seybold [*1]
Lu Jiang [*12]



Discrete tokens have a quality issue

Original

Reconstructed

It looks quite different up close!

*You cannot use this to know who they are even for understanding purposes

Emu3: Next-Token Prediction is All You Need

Discrete tokens have a quality issue

Discrete tokens have much worse reconstruction than continuous ones

Configurations	VisionTokenizer
Pretrained Weights	SBER-MoVQGAN-270M[5]
Codebook Size	32768
Latent Size	4
Compression	4 × 8 × 8

Table 1: **Emu3** vision tokenizer configurations.

Video Resolution	LPIPS↓	PSNR↑	SSIM↑
128 × 128	0.099	21.71	0.630
256 × 256	0.109	21.59	0.622
512 × 512	0.112	22.69	0.690
720 × 720	0.110	24.30	0.771

Table 2: Video compression metrics.

Discrete

Emu3: Next-Token Prediction is All You Need

https://arxiv.org/abs/2409.18869

Table 1: Ablation with different variants of 3D VAE. The baseline is SDXL(Podell et al., 2023) 2D VAE. Flickering calculates the L1 difference between each pair of adjacent frames to evaluate the degree of flickering in the video. We use variant B for pretraining.

Variants	Baseline	A	B	C	D	E
Compression	8×8×1	8×8×4	8×8×4	8×8×4	8×8×8	16×16×8
Latent channel	4	8	16	32	32	128
Flickering↓	93.2	87.6	86.3	87.7	87.8	87.3
PSNR↑	28.4	27.2	28.7	30.5	29	27.9

Continuous

CogVideoX: Text-to-Video Diffusion Models with An Expert Transformer

https://arxiv.org/abs/2408.06072

Fundamental flaw of discrete tokens

Discrete tokens have to compress a lot more for the same sequence length

Configurations	VisionTokenizer
Pretrained Weights	SBER-MoVQGAN-270M[5]
Codebook Size	32768
Latent Size	4
Compression	4 × 8 × 8

Table 1: **Emu3** vision tokenizer configurations.

8 bit color

Bit compression = (4 * 8 * 8) * 3 * 8 / 15 = 409.6

Sequence Compression

Channels

log2(32768)

Table 1: Ablation with different variants of 3D VAE. The baseline is SDXL(Podell et al., 2023) 2D VAE. Flickering calculates the L1 difference between each pair of adjacent frames to evaluate the degree of flickering in the video. We use variant B for pretraining.

Variants	Baseline	A	B	C	D	E
Compression	8×8×1	8×8×4	8×8×4	8×8×4	8×8×8	16×16×8
Latent channel	4	8	16	32	32	128
Flickering↓	93.2	87.6	86.3	87.7	87.8	87.3
PSNR↑	28.4	27.2	28.7	30.5	29	27.9

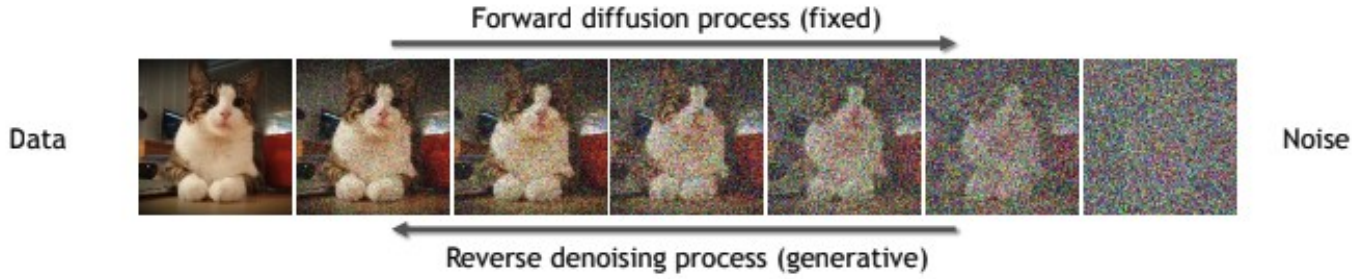
Bit compression
= (4 * 8 * 8) * 3 * 8 / (16 * 8) = 48

bfloat16

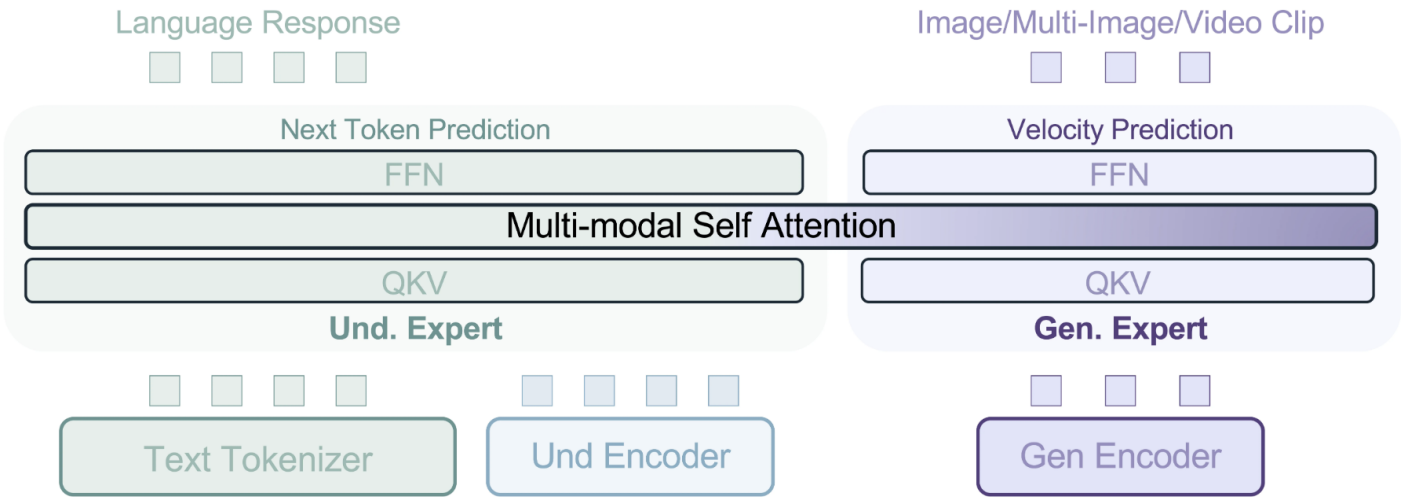
Latent channels

Continuous tokens has much higher quality in the same sequence length!

Continuous tokens have a speed issue



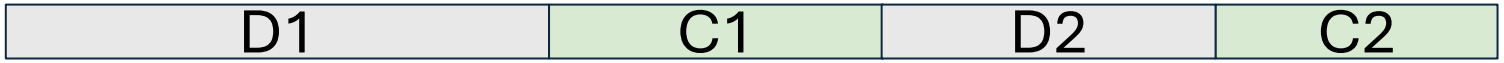
Diffusion requires many timesteps to converge



BAGEL: MoT with discrete + continuous tokens

Continuous tokens have a speed issue

While the sequence looks like this



Discrete tokens

Continuous signal.
Can be image / video / sound / actions etc...

The compute on the hardware is really like this!

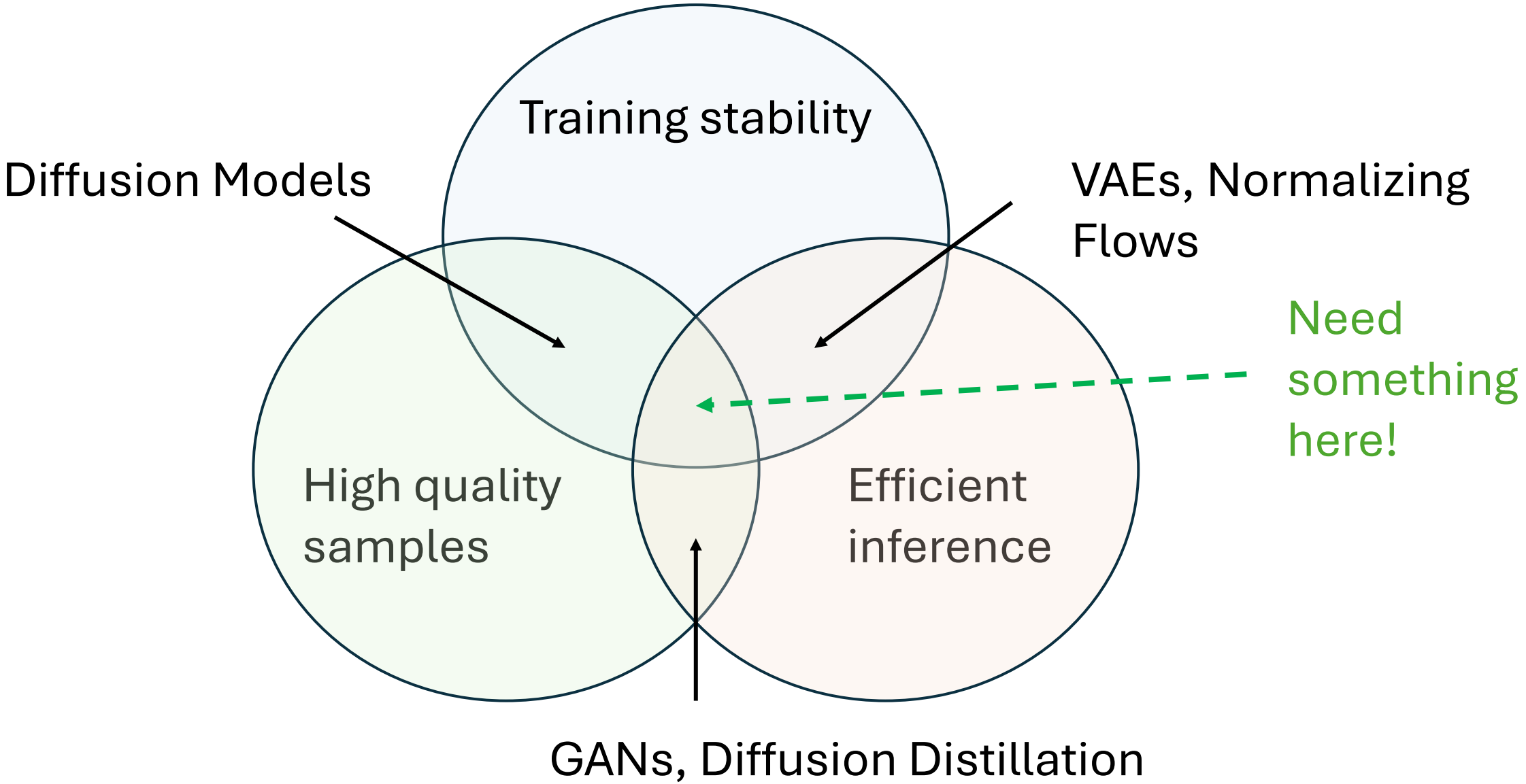
Discrete tokens only requires 1 pass of the transformer

Continuous tokens requires many passes of the transformer.

The algorithms are dominated by AR and diffusion…

But none are perfect!

The trilemma of continuous generative models



The algorithms are dominated by AR and diffusion…

But none are perfect!

Is there anything that would break the ceiling of the two?

The answer is Yes!

Outline

1. Two axes of inference scaling – sequence and refinement.
2. “Inference-first perspective” for algorithms that scale.
3. Why is DDIM (and by extension, diffusion) “sub-optimal”.
4. New algorithms and insights from inference-first perspective.

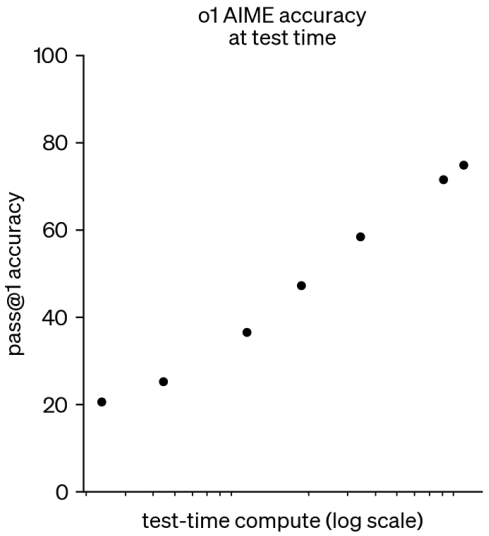
How can we scale at inference-time?

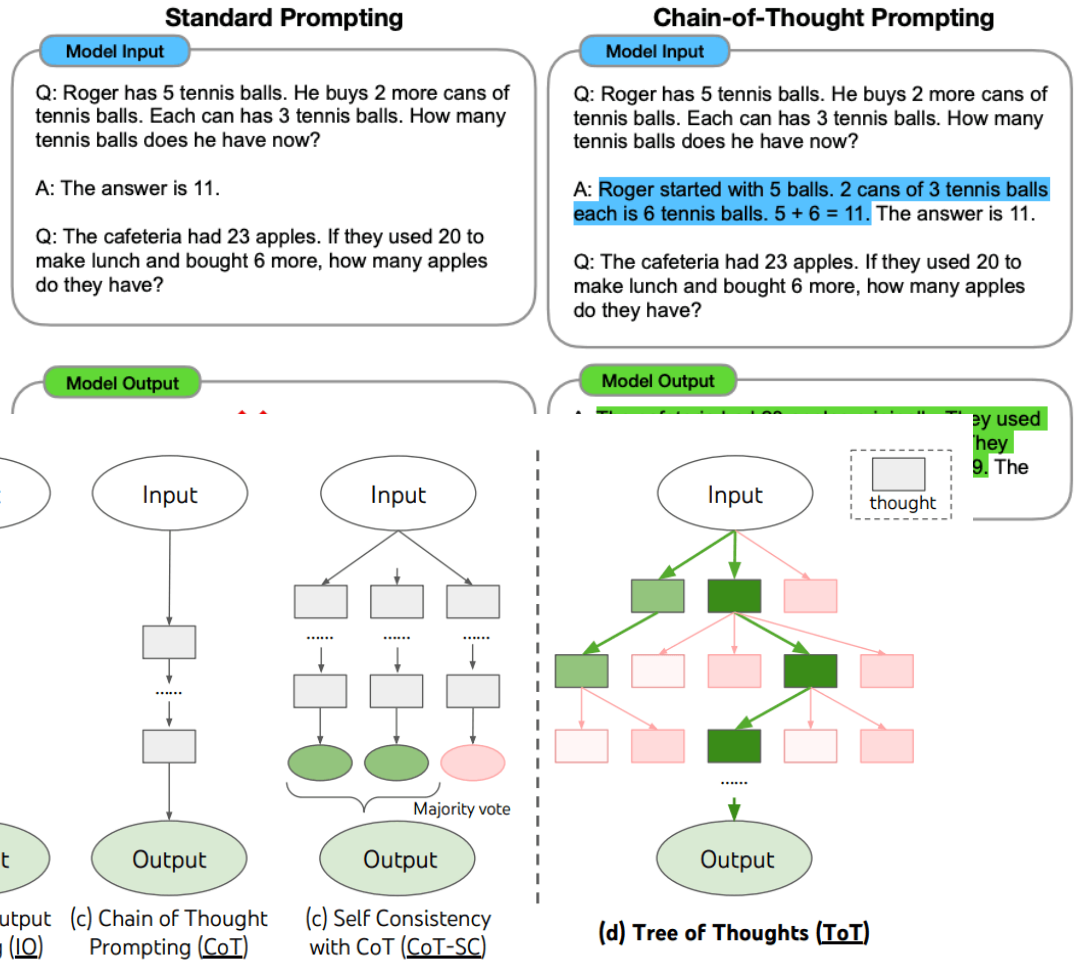
- Increase the number of tokens
- Don’t increase the number of tokens

Inference-Time Scaling in Sequence Length

Increases the number of tokens

- LLM Chain-of-Thought (CoT)
- CoT with reasoning data
- RL (DeepSeek-R1)

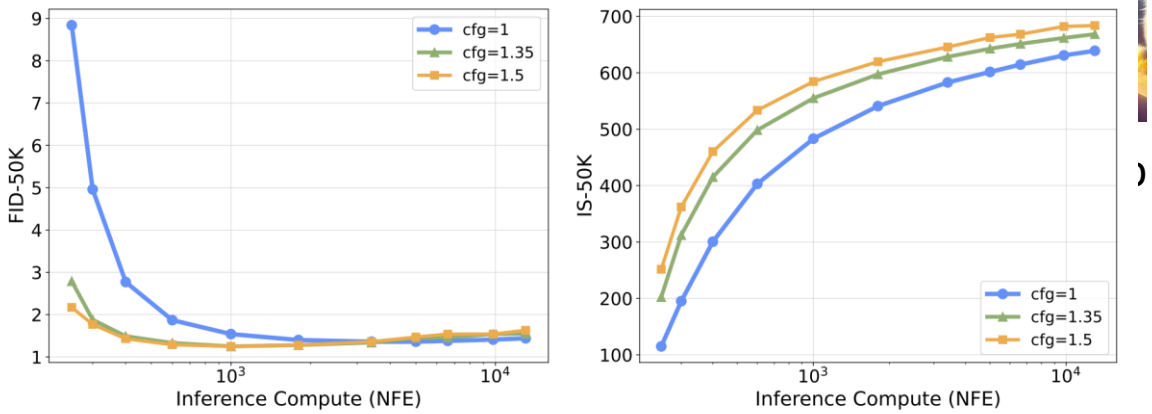


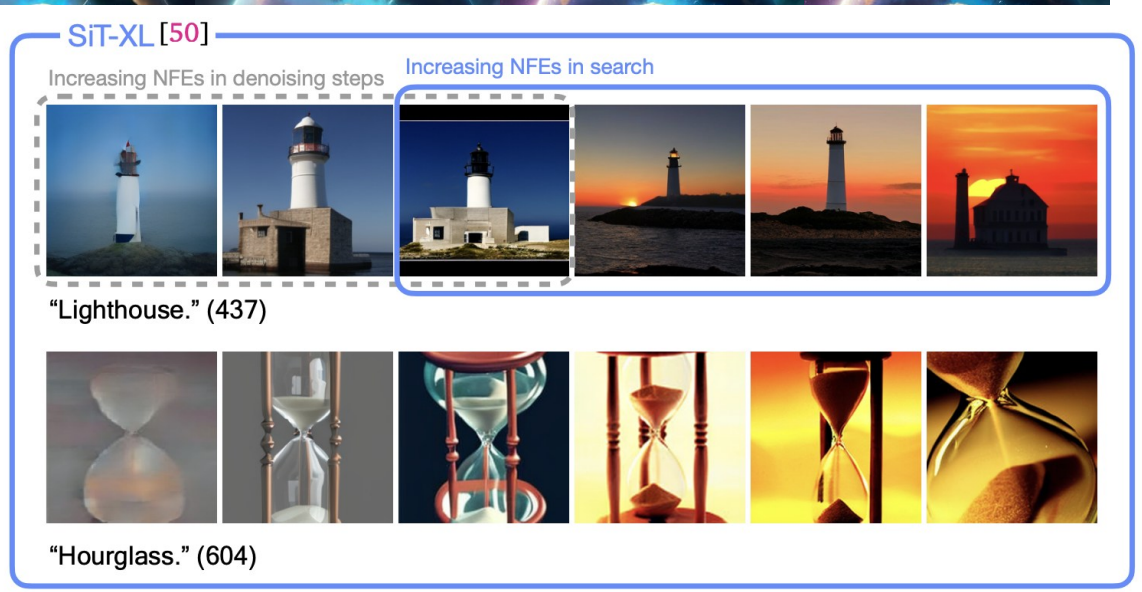


Inference-time Scaling in Refinement Steps

Does not increase the number of tokens

- Diffusion models / Flow Matching



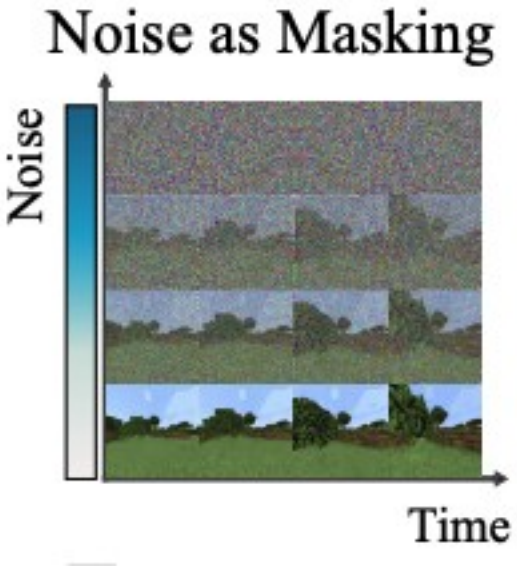


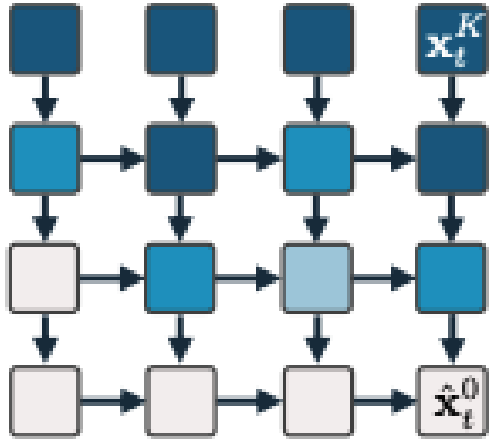
Categorizing existing algorithms

- Does not scale in either
  - VAE, GAN, Normalizing Flows
- Scale in sequence length, but not in refinement steps
  - GPT, PixelCNN, VAR, MaskGiT
- Scale in refinement steps, not in sequence length
  - Diffusion models, energy-based models, consistency models
- Scale in both (cont’d)

A lot of algorithms that scale in both axes

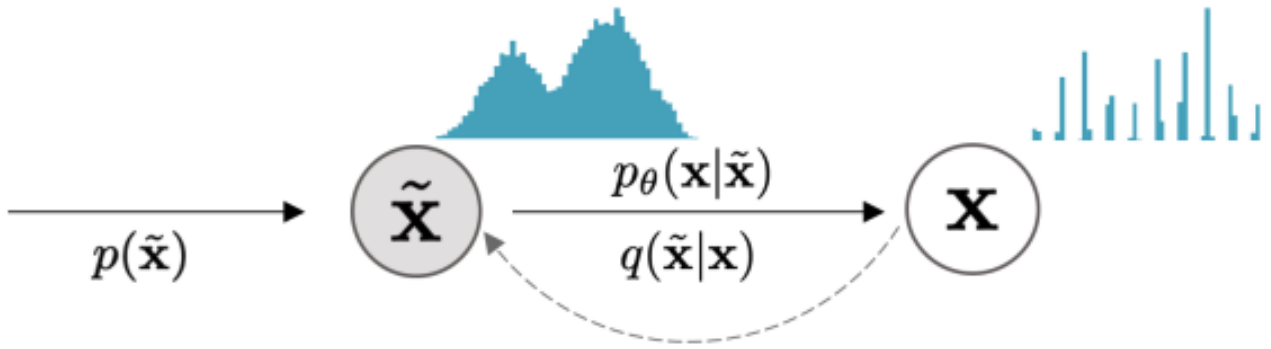
- Sequence length in outer loop, refinement steps in inner loop.
  - Basically, how most “Autogressive + Diffusion” are done
  - MAR
  - Diffusion Forcing
  - Discrete LLMs





A lot of algorithms that scale in both axes

- Scale refinement in outer loop, sequence length in inner loop.
  - Autoregressive distribution smoothing

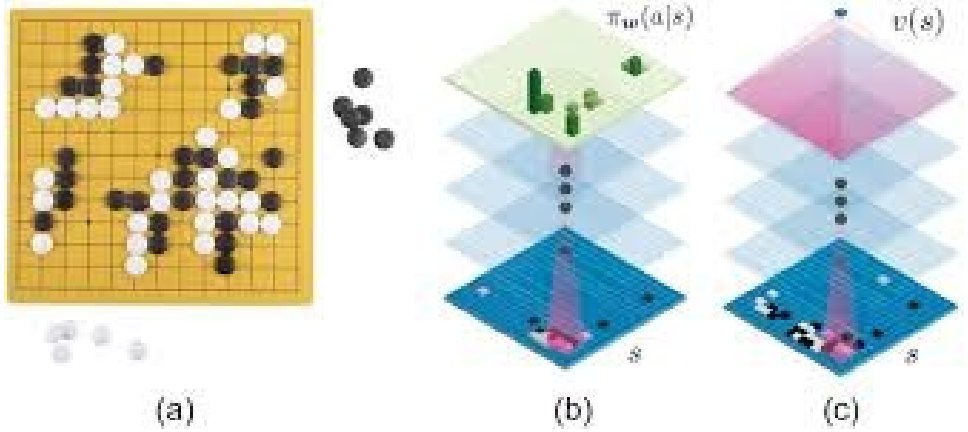


Scaling efficiency in inference algorithm

Of course, just being able to scale up is not enough!

We also have to scale efficiently!

Infinite monkeys “can” type Shakespeare



AlphaGo enabled by how to search more efficiently

Three positions

1. The right inference algorithm should scale in both axes.

2. Assuming that the model has enough capacity (under universal approximation theorem), it should use as few steps as possible.

3. Analyze the inference algorithm before the training algorithm!

(Applies to continuous and discrete cases, but will focus on continuous today)

Application to Continuous Diffusion

1. The right inference algorithm should scale in both axes. (✓)

    - Diffusion does scale in the refinement axis.

2. Assuming that the model has enough capacity (under universal approximation theorem), it should use as few steps as possible. (*X*)

    - DDIM requires multiple steps even when model have infinite capacity!

Application to Continuous Diffusion

What do we want from the ”right” inference algorithm?

There exists a solution to the model such that both holds:

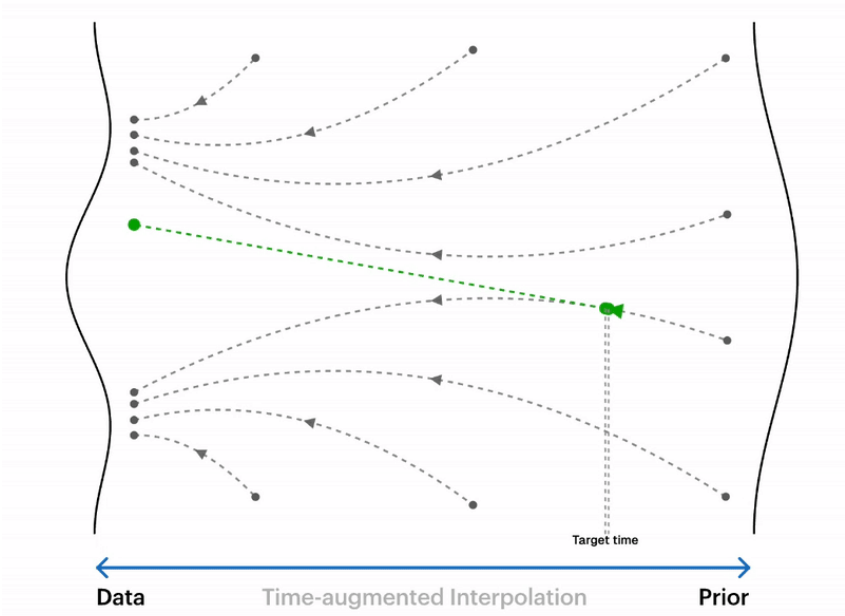
1. The inference algorithm generates the right distribution in N steps (scale correctly)
2. The inference algorithm generates the right distribution in 1 step (scale efficiently)

Unfortunately, DDIM is NOT the “right” inference algorithm!

DDIM and the Inference Capacity Issue

DDIM uses the Euler sampler under Flow Matching schedule

$$\mathbf{x}_s = \mathbf{x}_t + \hat{\mathbf{u}} \cdot (s - t)$$
$$\hat{\mathbf{u}} = \hat{\mathbf{u}}(\hat{\mathbf{x}}_t; t)$$



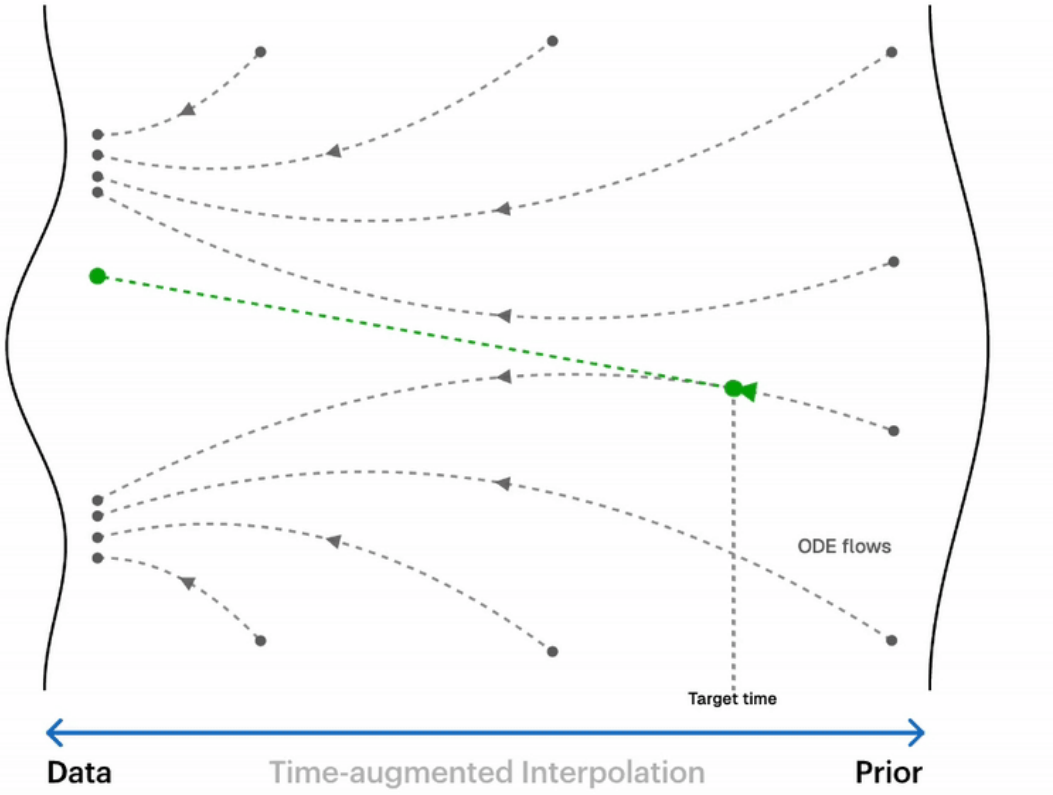
- Linear w.r.t. s
- Right distribution in 1 step -> Wrong distribution in N steps

The Fix

- Inject s into the network

$$\mathbf{x}_s = \mathbf{x}_t + \hat{\mathbf{u}} \cdot (s - t)$$
$$\hat{\mathbf{u}} = \hat{\mathbf{u}}(\mathbf{x}_t; t, s)$$

- Covers complex solutions
  - ODE integration

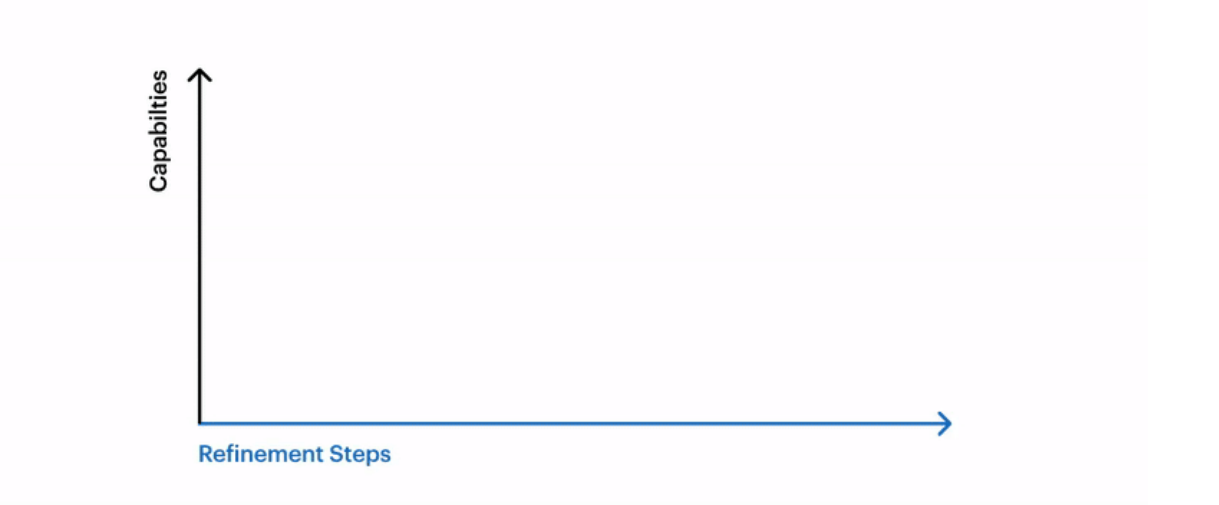


Diffusion Models and Flow Matching

- NOT optimal in utilizing network capacity.

$$\mathrm{d}\mathbf{x}_t = \mathbf{u}_t \mathrm{d}t$$

- Learns ODE, requiring MANY steps for accurate simulation
- Ideal case: **optimal** use of model capacity / **efficient** inference-time scaling



Application to Continuous Diffusion

DDIM is NOT the “right” inference algorithm because model only takes a single timestep!

We can fix it by asking the model to take 2 timesteps!

- Something new in the literature, known as “flow maps”

Analyze inference before training

Once th
many

CONSI
PROBA

Dongjun K
Sony AI
Tokyo, Japa
dongjour

Inductive Moment Matching

Linqi Zhou [1] Stefano Ermon [2] Jiaming Song [1]

odeling

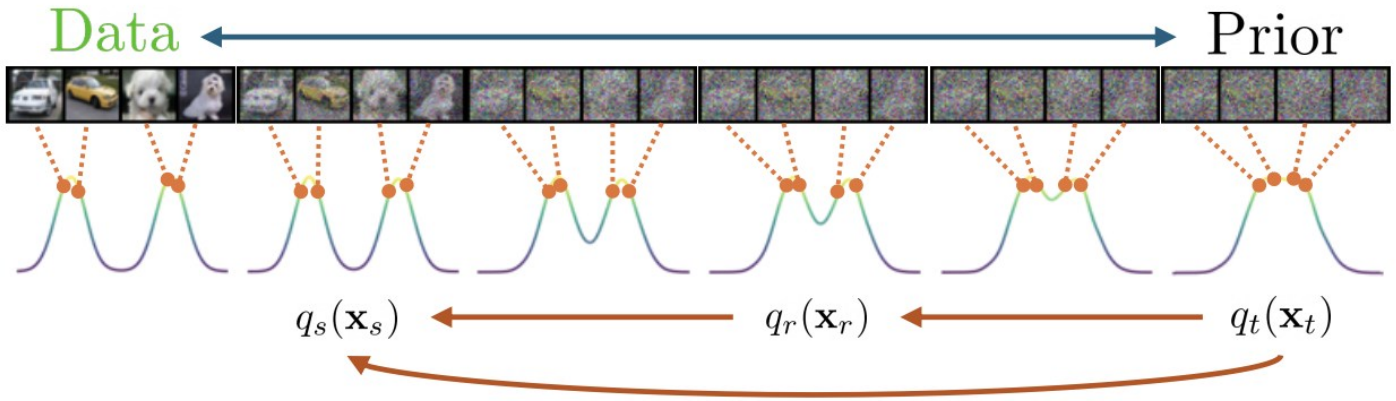
Kaiming He[2]

[1]CMU [2]MIT

Inductive Moment Matching

- Not dependent on denoising score matching / flow matching
- Not dependent on score-based stochastic differential equations
- Solution does not have to be connected to the probabilistic ODE!

Intuition: ”consistency” in distributions



For timesteps s < r < t, the two distributions should be close:

- Sample from x_t, one step prediction from x_t to x_s.
- Sample from x_r, one step prediction from x_r to x_s.

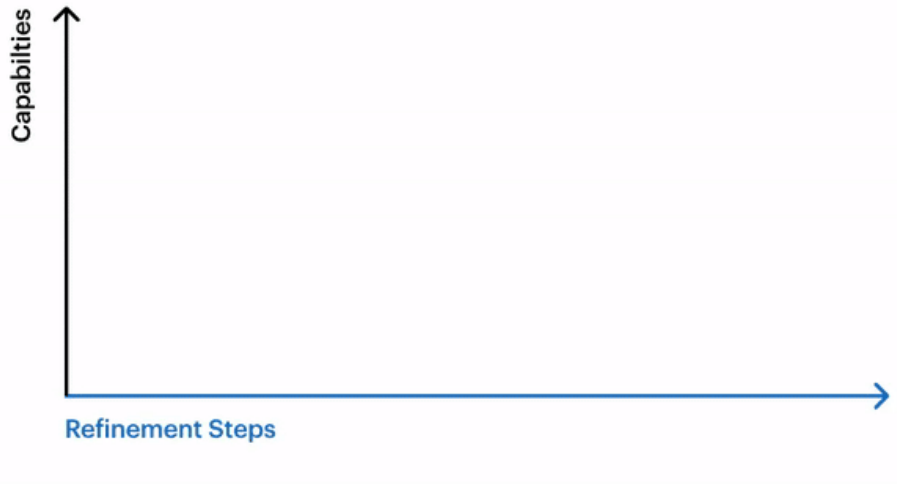
Intuition: ”consistency” in distributions

We can simply use Maximum Mean Discrepancy (MMD):

1. Like a GAN, MMD has a “discriminator”
2. Unlike a GAN,
    1. MMD uses a special family of discriminators called RKHS.
    2. No need to “optimize” the discriminator, so **training is stable**!

Advantages of IMM

1. Single stage training, single objective function
2. Generalizes consistency models (when comparing distributions with 1 sample)
3. Quite stable to train
4. Reaches SOTA few step generation



Stable Training

- Consistency model is a 1-particle special case
- Stable training as long as >4 particles

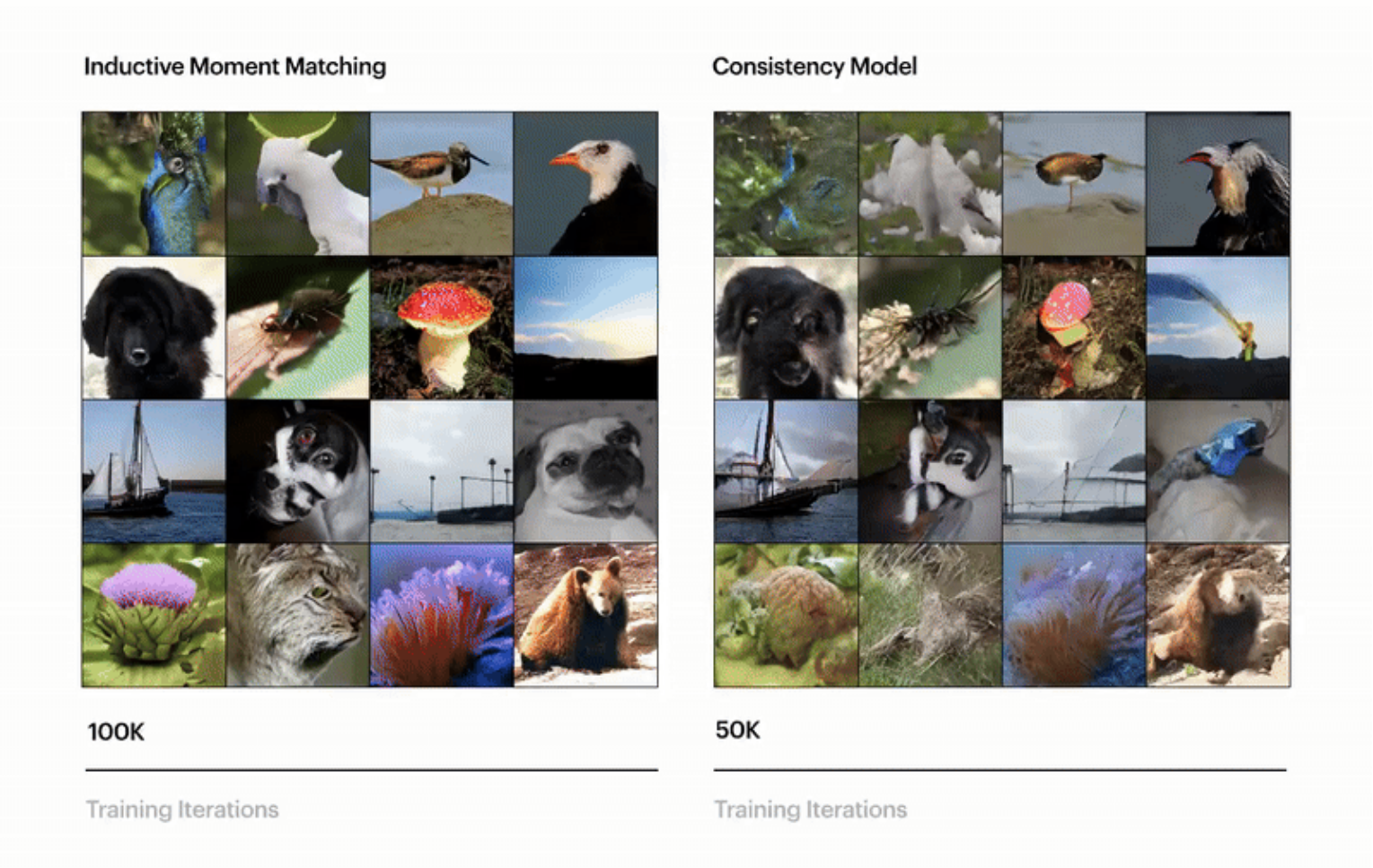


Image Generation

- Better than DiT/SiT
- Outperform VAR-d20 (600M param)

- ImageNet-256x256 16-step FID: **1.90**
  - Outperform VAR-d30 (2B param)
- CIFAR-10 2-step FID: **1.98**

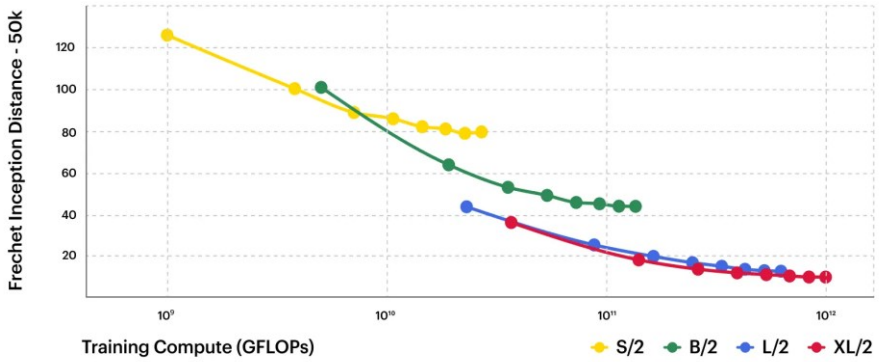
Family	Method	FID (↓)	Steps (↓)
Diffusion & Flow	DDPM (Ho et al., 2020)	3.17	1000
	DDPM++ (Song et al., 2020b)	3.16	1000
	NCSN++ (Song et al., 2020b)	2.38	1000
	DPM-Solver (Lu et al., 2022)	4.70	10
	iDDPM (Nichol & Dhariwal, 2021)	2.90	4000
	EDM (Karras et al., 2022)	2.05	35
	Flow Matching (Lipman et al., 2022)	6.35	142
	Rectified Flow (Liu et al., 2022)	2.58	127
Few-Step via Distillation	PD (Salimans & Ho, 2022)	4.51	2
	2-Rectified Flow (Salimans & Ho, 2022)	4.85	1
	DFNO (Zheng et al., 2023)	3.78	1
	KD (Luhman & Luhman, 2021)	9.36	1
	TRACT (Berthelot et al., 2023)	3.32	2
	Diff-Instruct (Luo et al., 2024a)	5.57	1
	PID (LPIPS) (Tee et al., 2024)	3.92	1
	DMD (Yin et al., 2024)	3.77	1
	CD (LPIPS) (Song et al., 2023)	2.93	2
	CTM (w/ GAN) (Kim et al., 2023)	**1.87**	2
	SiD (Zhou et al., 2024)	1.92	1
	SiM (Luo et al., 2024b)	2.06	1
	sCD (Lu & Song, 2024)	2.52	2
Few-Step from Scratch	iCT (Song & Dhariwal, 2023)	2.83	1
		2.46	2
	ECT (Geng et al., 2024)	3.60	1
		2.11	2
	sCT (Lu & Song, 2024)	2.97	1
		2.06	2
	IMM (ours)	3.20	1
		1.98	2

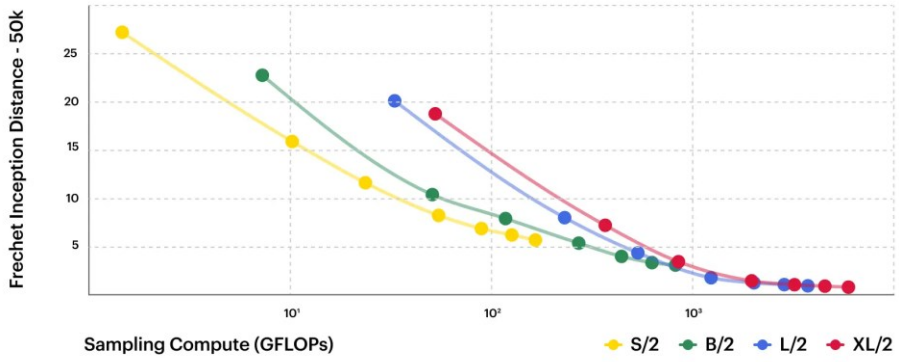
Table 1. CIFAR-10 results trained without label conditions.

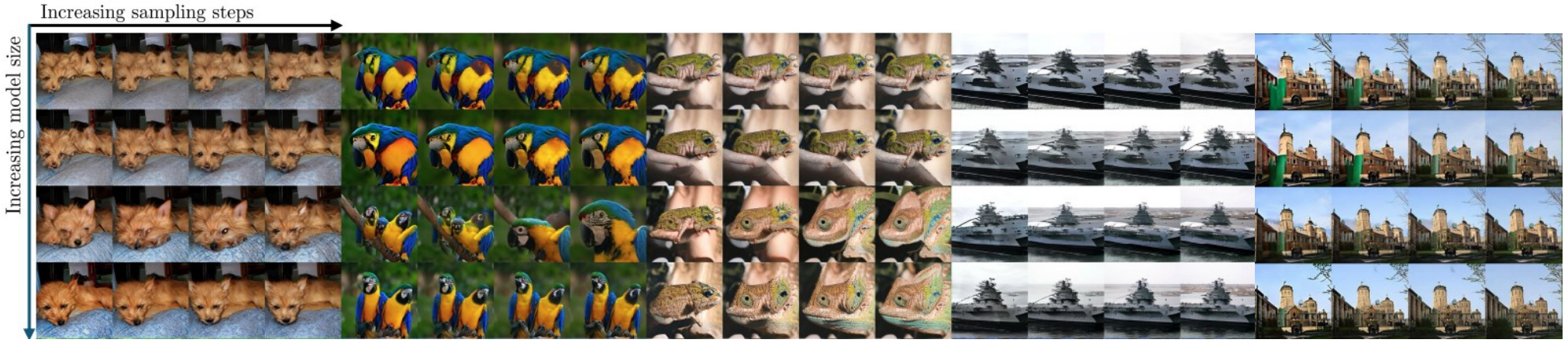
Family	Method	FID(↓)	Steps (↓)	#Params
GAN	BigGAN (Brock, 2018)	6.95	1	112M
	GigaGAN (Kang et al., 2023)	3.45	1	569M
	StyleGAN-XL (Karras et al., 2020)	2.30	1	166M
Masked & AR	VQGAN (Esser et al., 2021)	26.52	1024	227M
	MaskGIT (Chang et al., 2022)	6.18	8	227M
	MAR (Li et al., 2024)	1.98	100	400M
	VAR-*d*20 (Tian et al., 2024a)	2.57	10	600M
	VAR-*d*30 (Tian et al., 2024a)	**1.92**	10	2B
Diffusion & Flow	ADM (Dhariwal & Nichol, 2021)	10.94	250	554M
	CDM (Ho et al., 2022b)	4.88	8100	-
	SimDiff (Hoogeboom et al., 2023)	2.77	512	2B
	LDM-4-G (Rombach et al., 2022)	3.60	250	400M
	U-DiT-L (Tian et al., 2024b)	3.37	250	916M
	U-ViT-H (Bao et al., 2023)	2.29	50	501M
	DiT-XL/2 ($w = 1.0$) (Peebles & Xie, 2023)	9.62	250	675M
	DiT-XL/2 ($w = 1.25$) (Peebles & Xie, 2023)	3.22	250	675M
	DiT-XL/2 ($w = 1.5$) (Peebles & Xie, 2023)	2.27	250	675M
	SiT-XL/2 ($w = 1.0$) (Ma et al., 2024)	9.35	250	675M
	SiT-XL/2 ($w = 1.5$) (Ma et al., 2024)	2.15	250	675M
Few-Step from Scratch	iCT (Song et al., 2023)	34.24	1	675M
		20.3	2	675M
	Shortcut (Frans et al., 2024)	10.60	1	675M
		7.80	4	675M
		3.80	128	675M
	IMM (ours) (XL/2, $w = 1.25$)	7.77	1	675M
		5.33	2	675M
		3.66	4	675M
		2.77	8	675M
	IMM (ours) (XL/2, $w = 1.5$)	8.05	1	675M
		3.99	2	675M
		2.51	4	675M
		1.99	8	675M

Table 2. Class-conditional ImageNet-256×256 results.

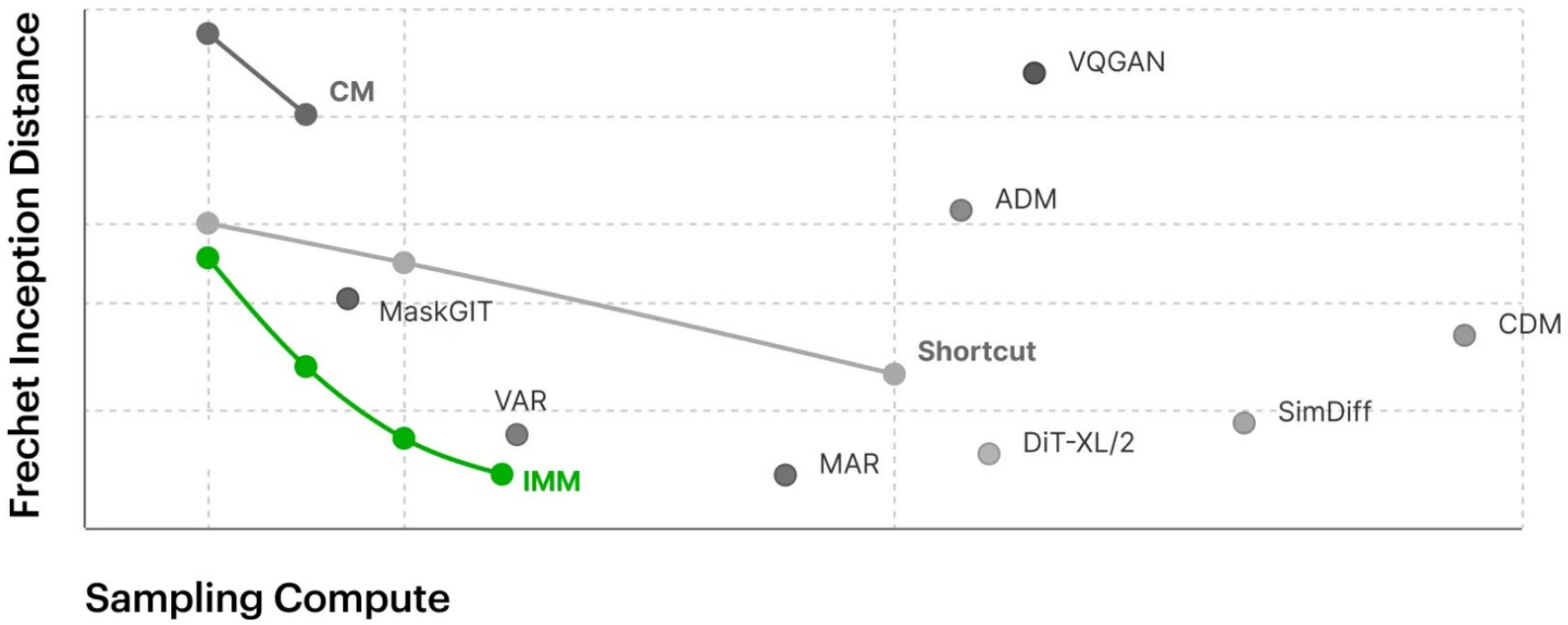
Scaling Property



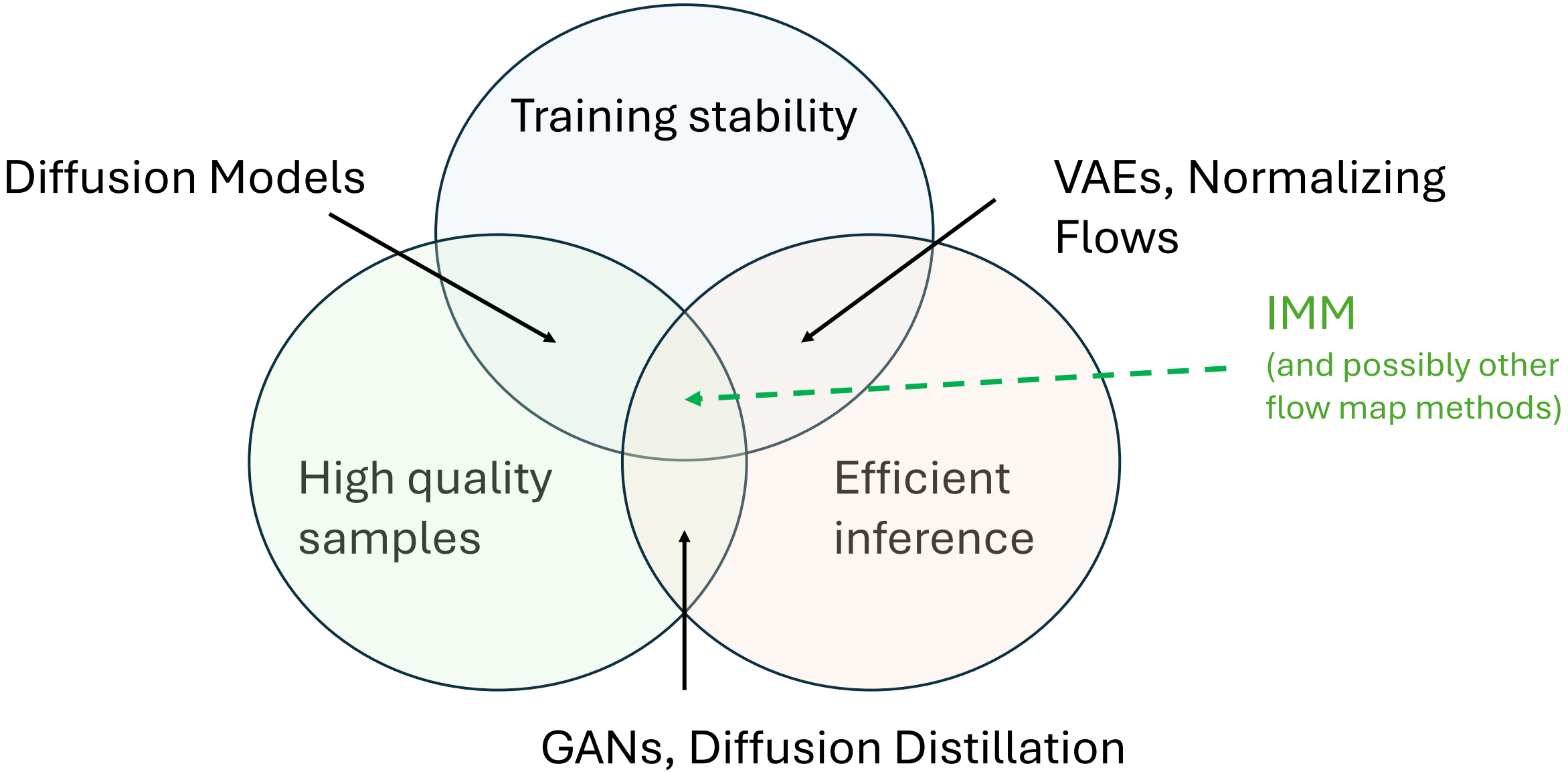




Advancing Efficiency / Quality Frontier

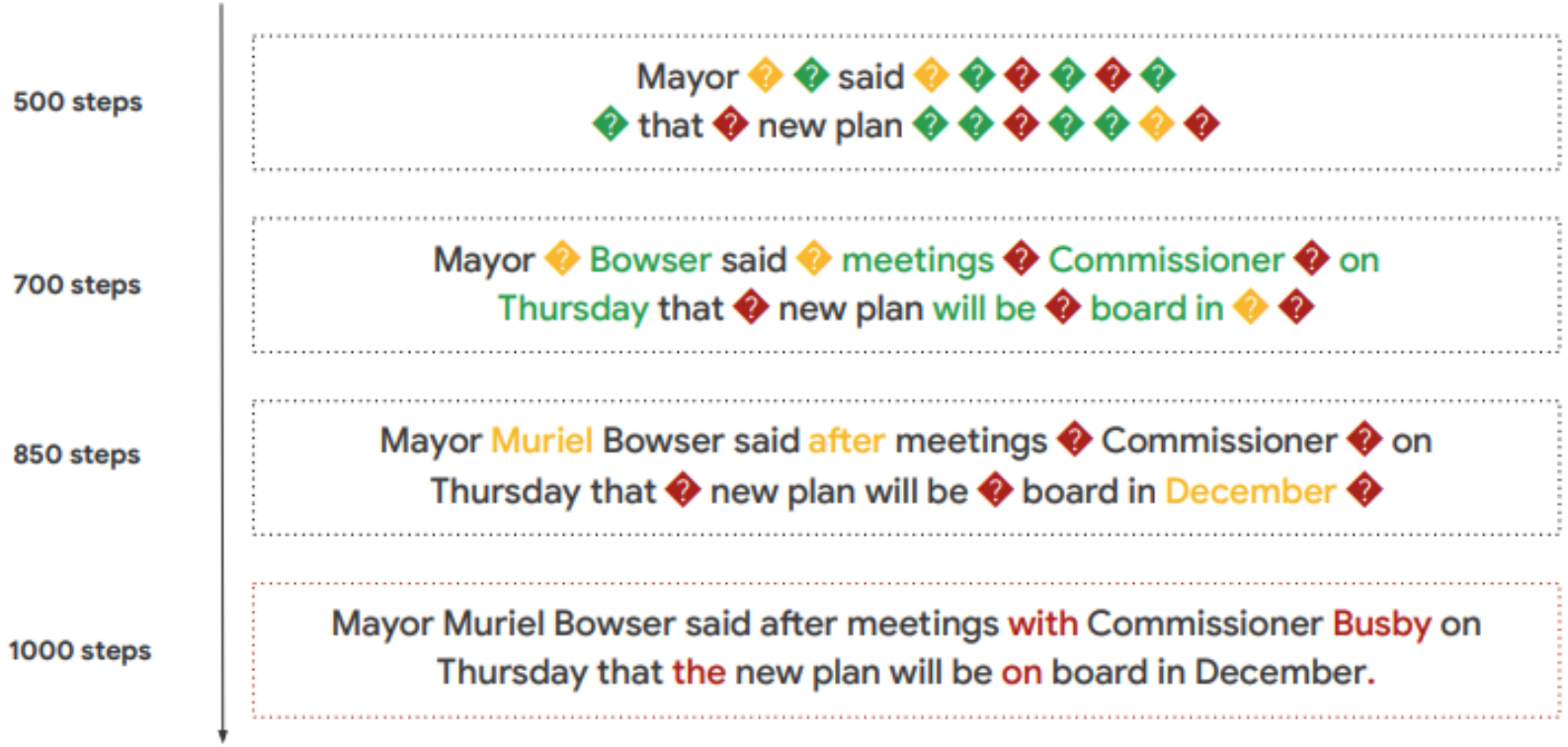


The trilemma of continuous generative models



Applications to Discrete Diffusion

Consider Masked Diffusion, a performant variant of discrete diffusion



Shi et al., Simplified and Generalized Masked Diffusion for Discrete Data

Applications to Discrete Diffusion

In mask diffusion, value changes only when input is [mask] token.

Algorithm 2 Unconditional and conditional generation (e.g., infilling) with MD4.

Input: Context sequence x^c of length N, with masks indicating the target areas for generation
Init: $\{t(i)\}_{i=0}^{T} \leftarrow \text{discretize}([0,1]), x_{t(T)} \leftarrow x^c$
for $i = T, T-1, \ldots, 1$ **do**
  $t \leftarrow t(i), s \leftarrow t(i-1)$
  for $n \in [N]$, if $x_t^{(n)} = m$, draw $x_s^{(n)} \sim \text{Cat}(\frac{\alpha_s - \alpha_t}{1-\alpha_t} \mu_\theta^{(n)}(x_t, t) + \frac{1-\alpha_s}{1-\alpha_t} e_m)$ else $x_s^{(n)} \leftarrow x_t^{(n)}$
return x_0.

Suppose seqlen = N, and we want to sample in L << N steps:

- Then there is one step that at least samples two tokens!

Shi et al., Simplified and Generalized Masked Diffusion for Discrete Data

Applications to Discrete Diffusion

Does the BERT-style model have “enough capacity”?

Suppose we try to predict:

The list of poker hands that consist of two English words are: [MASK] [MASK]

- Then the valid responses can be: “high card”, ”two pair”, etc…
- However, BERT samples each [MASK] independently, so it is also possible to generate ”high pair”, “two card” with the model!
- This is not an issue with AR models because words are generated one at a time.

Applications to Discrete Diffusion

From the inference-first perspective:

Masked discrete diffusion might have capacity issues when trying to sample in L << N steps when using the BERT-style model, regardless how it is trained!

Takeaway

Analyze the inference algorithm before the training algorithm!

- Continuous case: better alternatives to diffusion models
- Discrete case: limitations of the BERT-style diffusion LLM

Inductive Moment Matching:
https://github.com/lumalabs/imm

Inference first position paper:
https://arxiv.org/abs/2503.07154

Join us

https://lumalabs.ai/join