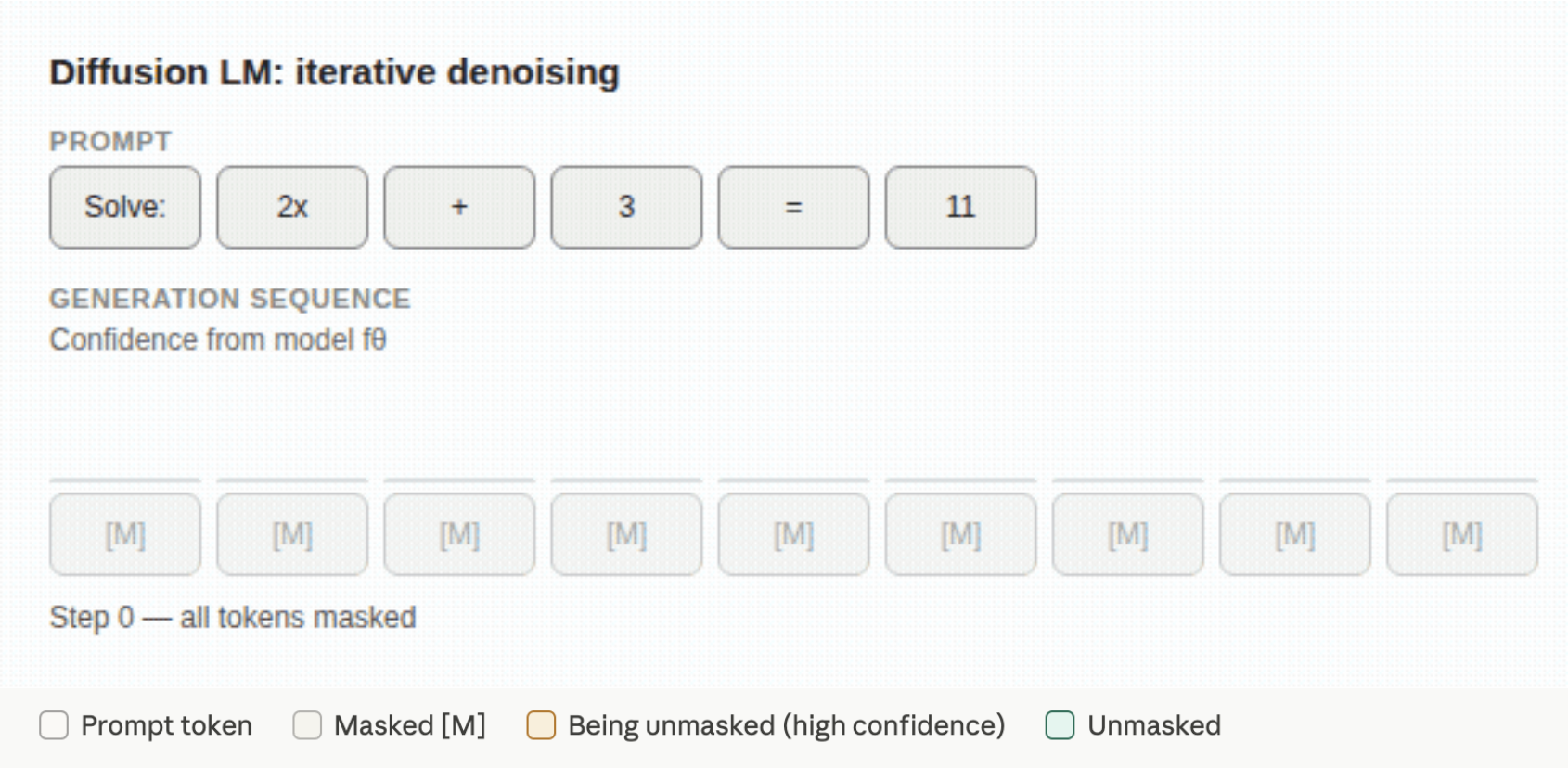
Fatemeh Doudi

PhD student, Computer Engineering

Texas A&M university

Reinforcement Learning for Diffusion LLMs with Entropy-Guided Step Selection and Stepwise Advantages

What is Diffusion LLMs:



What is Diffusion LLMs:

- Most LLMs (GPT, LLaMA, DeepSeek) generate text left to right, one token at a time.
- Diffusion LMs do something fundamentally different:
    a. Start with a fully masked sequence [M] [M] [M] [M]
    b. Iteratively unmask tokens over T denoising steps
    c. At each step, the model sees the full (partially masked) context — bidirectional attention
    d. End result: the clean text x_0

Pros and Cons:

a. Parallel multi-token generation → higher throughput
b. Bidirectional context at every step → richer representations
c. But: breaks the causal structure that RL methods rely on

The Problem: why RL doesn't transfer to diffusion LMs

The setup: RL for AR models works because of "causal factorization". The likelihood of a sequence decomposes token by token:

$$\log \pi(x) = \sum \log \pi(x_i \mid x_{<i})$$

One forward pass gives you everything. Policy gradient is straightforward.

The problem with diffusion LMs:

- There's no causal structure.
- The model uses bidirectional attention!
- You can't factorize $\log \pi(x_0)$ token by token
- Sequence-level likelihood is intractable
- Naively porting GRPO → requires T separate forward passes (one per denoising step) → prohibitively expensive.

What existing work does: Approximate it by (i)surrogate likelihoods, (ii)mean-field approximations. They work, but they introduce bias and lose the step-level structure.

Our question: Can we derive an exact policy gradient that respects the denoising structure, without evaluating sequence likelihood at all?

Our Solution

- **The key idea:** Instead of approximating the likelihood, we ask a different question — what's the right MDP formulation for diffusion generation?

Step 1 - Reframe as an MDP:

- **State at step t:** the partially masked sequence x_t
- **Action:** which tokens to unmask $\rightarrow x_{t-1}$
- **Reward**: only at the end $\rightarrow r(x_0, q)$
- This gives us an exact policy gradient that decomposes over denoising steps

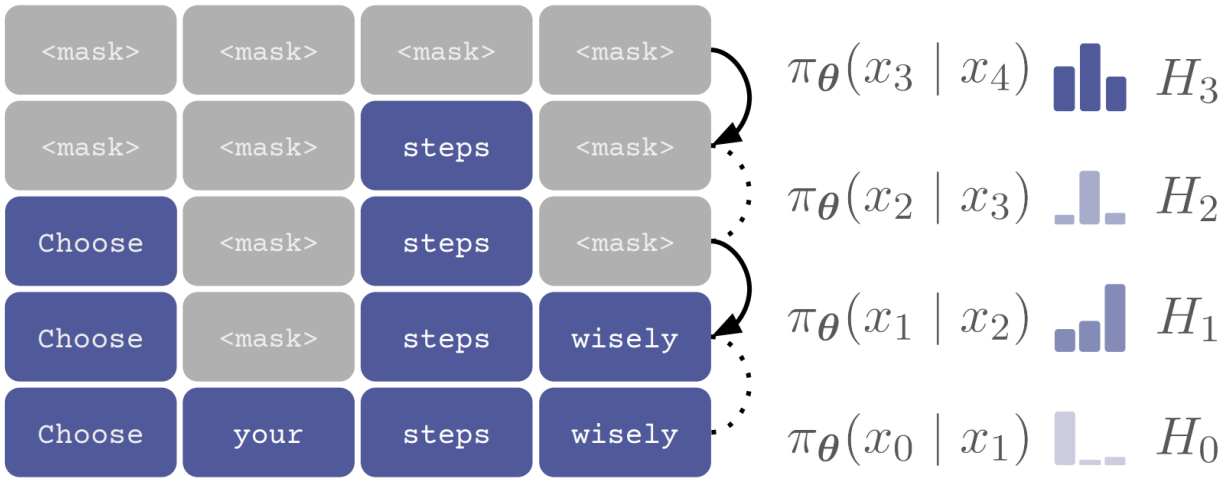
Step 2 - EGSPO: You can't afford T forward passes. So which steps matter most?

- The ones where the model is most uncertain (highest entropy).
- Train only on those K steps.
- Provably minimizes the approximation error.

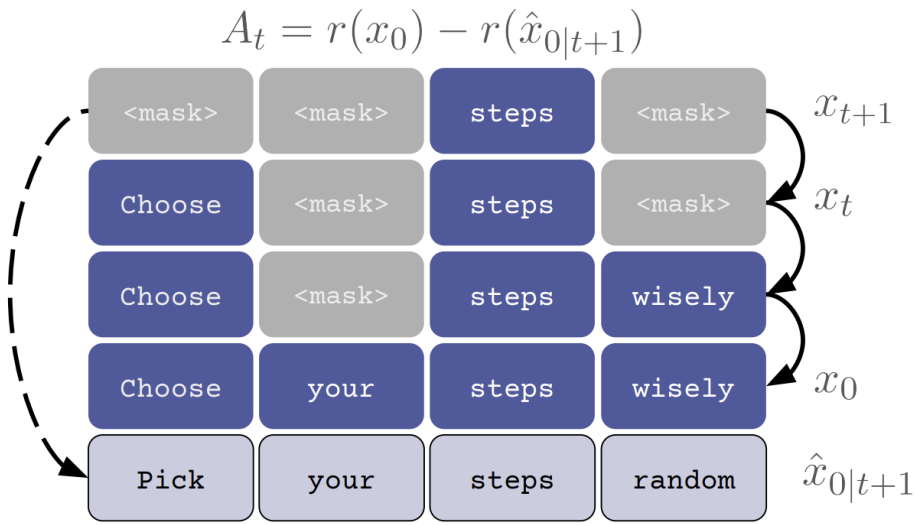
Step 3 - EGSPO-SA: How do you estimate the advantage at each step without a value network?

- Use the model itself.
- greedily complete from x_{t+1} in one shot.
- get a reward estimate.

Our Solution



(a) **Entropy-Guided Step Selection**



(b) **Stepwise Advantage Estimation**

Empirical Results

	Sudoku				Countdown				GSM8K				MATH500			
Model / Seq. Len.	128	256	512	Best	128	256	512	Best	128	256	512	Best	128	256	512	Best
LLaDA-8B-Instruct	11.7	6.7	5.5	11.7	20.7	19.5	16.0	20.7	68.7	76.7	78.2	78.2	26.0	32.4	36.2	36.2
d1	22.1	16.7	9.5	22.1	34.8	32.0	42.2	42.2	73.2	81.1	82.1	82.1	**33.8**	38.6	40.2	40.2
wd1	-	76.4	62.8	76.4	-	51.2	46.1	51.2	-	80.8	82.3	82.3	-	34.4	39.0	39.0
SPG	82.9†	94.0†	93.1†	94.0†	68.8	71.5	70.3	71.5	**78.5**	**86.1**	84.5	**86.1**	33.4	**40.0**	41.8	**41.8**
d2				91.9				56.6				85.0				41.6
EGSPO (Ours)	93.3	93.6	89.1	93.6	70.3	71.5	75.8	75.8	77.5	85.7	84.98	85.7	32.2	37.8	39.0	39.0
EGSPO-SA (Ours)	**93.7**	**94.3**	**93.4**	**94.3**	**78.5**	**77.3**	**76.5**	**78.5**	73.5	84.6	**85.03**	85.03	33.2	38.2	39.6	39.6

Table 1. Main results across reasoning benchmarks. For each method, we report accuracy at different generation lengths and the best performance across lengths. d2 reports a single best performance independent of generation length, following its original evaluation protocol. wd1 does not report the evaluation on length 128. For Sudoku, † denotes 3-shot evaluation, whereas ours is 0-shot.

Empirical Results

HumanEval	128	256	512	Best
LLaDA-8B-Instruct	27.4	35.3	37.8	37.8
d1	31.1	32.9	37.8	37.8
EGSPO (Ours)	32.3	40.2	39.6	40.2
EGSPO-SA (Ours)	**32.3**	**41.5**	**44.5**	**44.5**
MBPP	128	256	512	Best
LLaDA-8B-Instruct	36.2	41.2	40.4	41.2
d1	40.5	44.7	42.8	44.7
EGSPO (Ours)	49.9	**50.6**	**50.3**	50.6
EGSPO-SA (Ours)	**51.1**	48.7	49.2	**51.1**

Table 2. Results on coding benchmarks. We report accuracy at different generation lengths and the best performance across lengths.

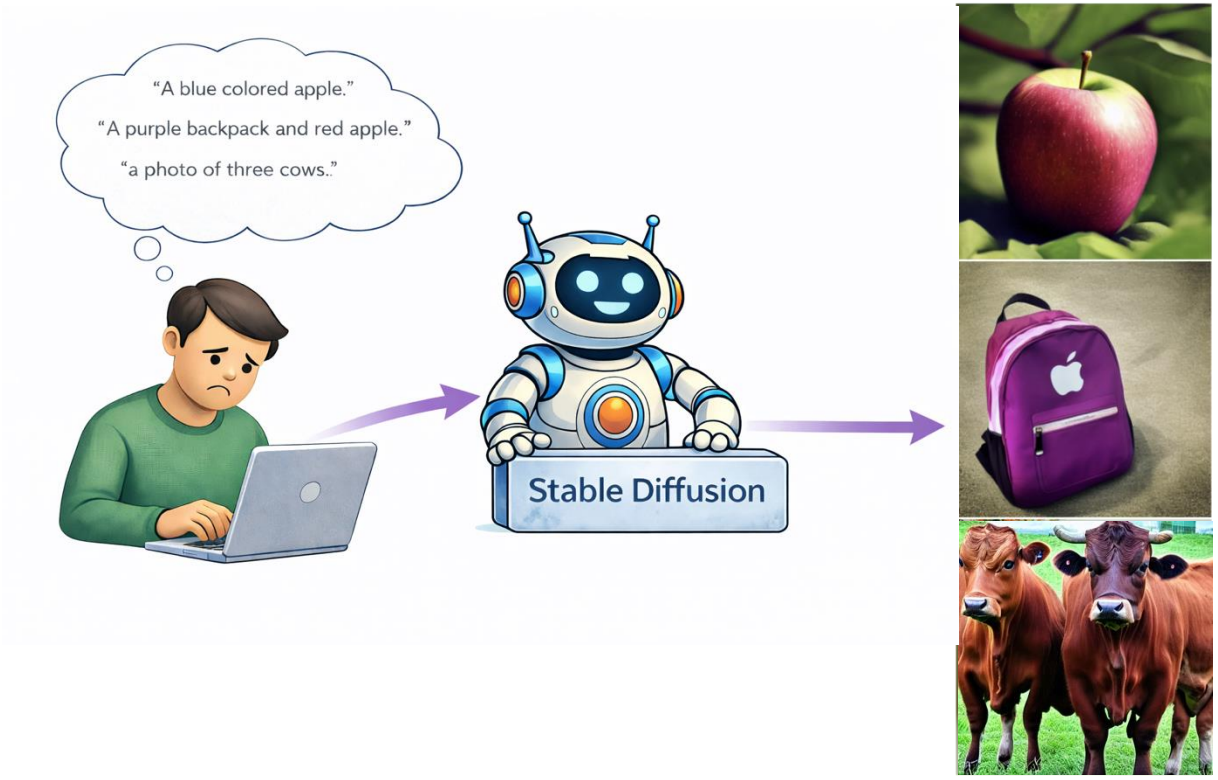
Diffusion Blend: Inference-Time Multi-Preference Alignment for Diffusion Models

Diffusion Models Are Powerful, But Not Aligned!

RL fine-tuning bridges the gap: optimize the model toward a reward signal!

Pre-trained on massive data — not optimized for downstream objectives.
Generated images can fail on:

- Prompt fidelity
- Aesthetic quality
- Human preference



Diffusion Models Are Powerful, But Not Aligned!

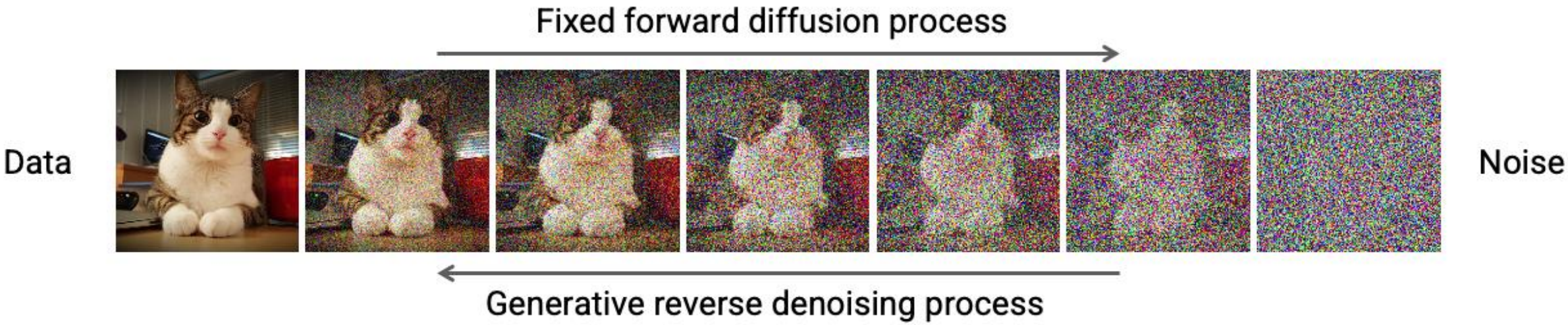
RL fine-tuning bridges the gap: optimize the model toward a reward signal!

Pre-trained on massive data — not optimized for downstream objectives.
Generated images can fail on:

- Prompt fidelity
- Aesthetic quality
- Human preference

RL fine-tuning bridges the gap: optimize the model toward a reward signal!

Diffusion Model in a glance



Forward Process

Goal: Gradually destroying structure with noise.

Define the forward SDE:

$$dx_t = -\frac{1}{2}\beta(t)x_t\,dt + \sqrt{\beta(t)}\,dW_t$$

- x_0: real data.
- $x_T \sim \mathcal{N}(0, I)$: pure noise.
- $\beta(t)$: noise schedule.

Closed form: $x_t|x_0 \sim \mathcal{N}(\mu(t)x_0, \sigma^2(t)I)$

Key insight: No learning here. This is a fixed, deterministic corruption process.

Denoising (Reverse) Process

Goal: Running time backwards -> from noise to image.

The reverse SDE (Anderson 1982):

$$dx_t = \underbrace{\left(-\frac{1}{2}\,\beta(t)\,x_t \;-\beta(t)\,\nabla x_t \log p_t(x_t)\right)}_{\boldsymbol{f^{pre}(x_t,t)}} dt + \sqrt{\beta(t)}\,dW_t$$

- Running from $t = T$ to $t = 0$ $\rightarrow$ generates from p_0.
- $\nabla_x \log p_t(x_t)$: the score function.
  - We can't compute it analytically.
  - Train a neural network $s_{\theta(x_t,t)} \approx \nabla_x \log p_t(x_t)$

RL Alignment of Diffusion Models

RL concept	Diffusion equivalent
State s_t	Noisy image x_t
Action a_t	Next denoised image x_{t-1}
Policy $\pi(a \mid s)$	Denoising transition $p_\theta(x_{t-1} \mid x_t)$
Trajectory τ	Full chain $x_T \to \cdots \to x_0$
Reward r	Score on final image $r(x_0)$
Reference π_{ref}	Pretrained model p_0^{pre}

RL alignment objective:

- terminal reward (alignment, aesthetics, preference)
- **After RL fine-tuning, the new drift will be:** $\boldsymbol{f^{(r,\alpha)}(x_t, t)}$

Limitations of RL Alignment

RL alignment objective:

$$\max_{p_0} \mathbb{E}\left[\boldsymbol{r}(x_0)\right] - \boldsymbol{\alpha}.\,\mathrm{KL}(p_0 \,\|\, p_0^{pre})$$

1. Fixed Reward Function

a. One reward → one fine-tuned model.
b. New objective → retrain

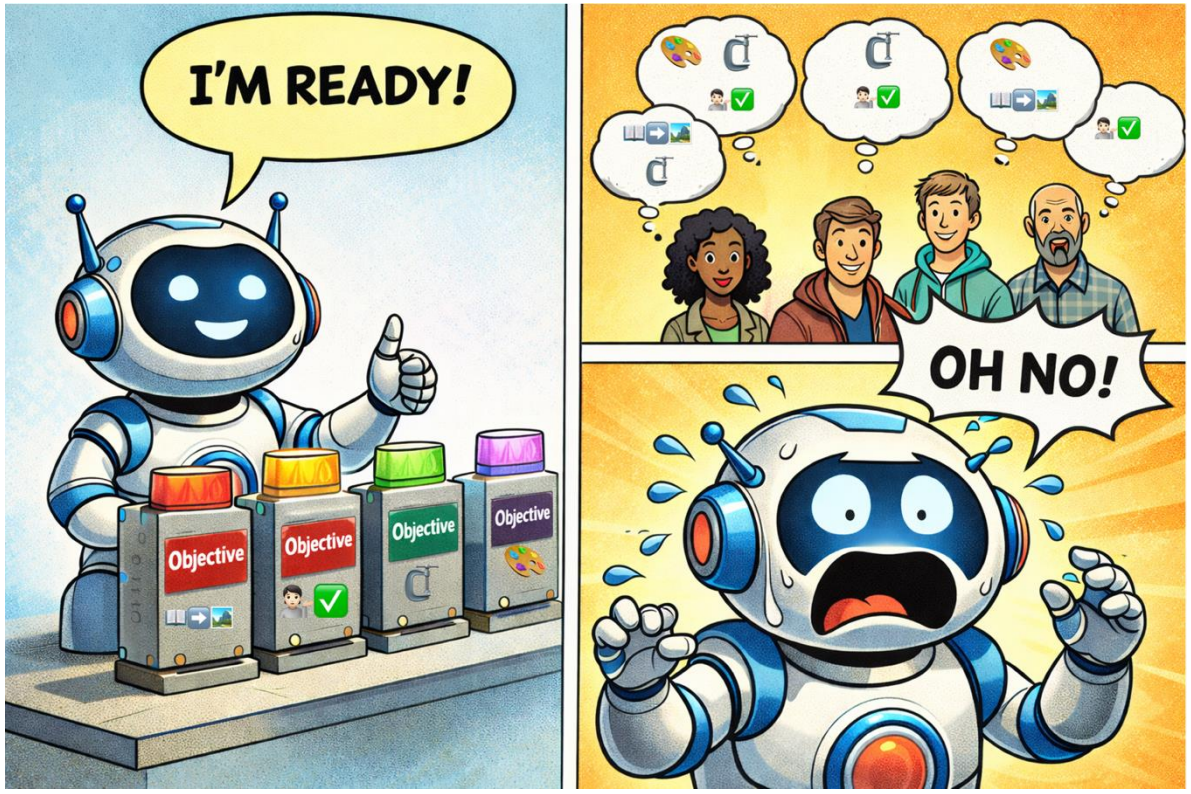
2. Fixed Trade-off Parameter α

a. Controls deviation from base model
b. Requires tuning via grid search

The Multi-Preference Problem

One Fine-Tuned Model Is Never Enough!

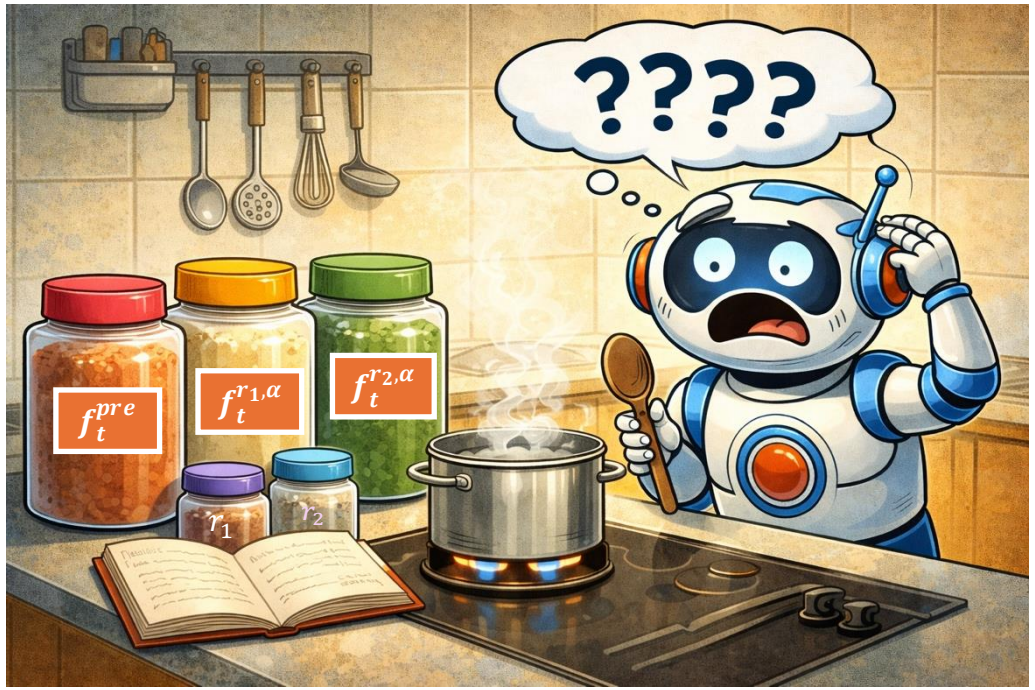
- Real users balance multiple conflicting objectives!
- Preferences vary across users and even prompts
- Fine-tuning for each preference combination?
    a. **Time-consuming**
    b. **computationally expensive!**



Inference-Time Multi-Preference Alignment

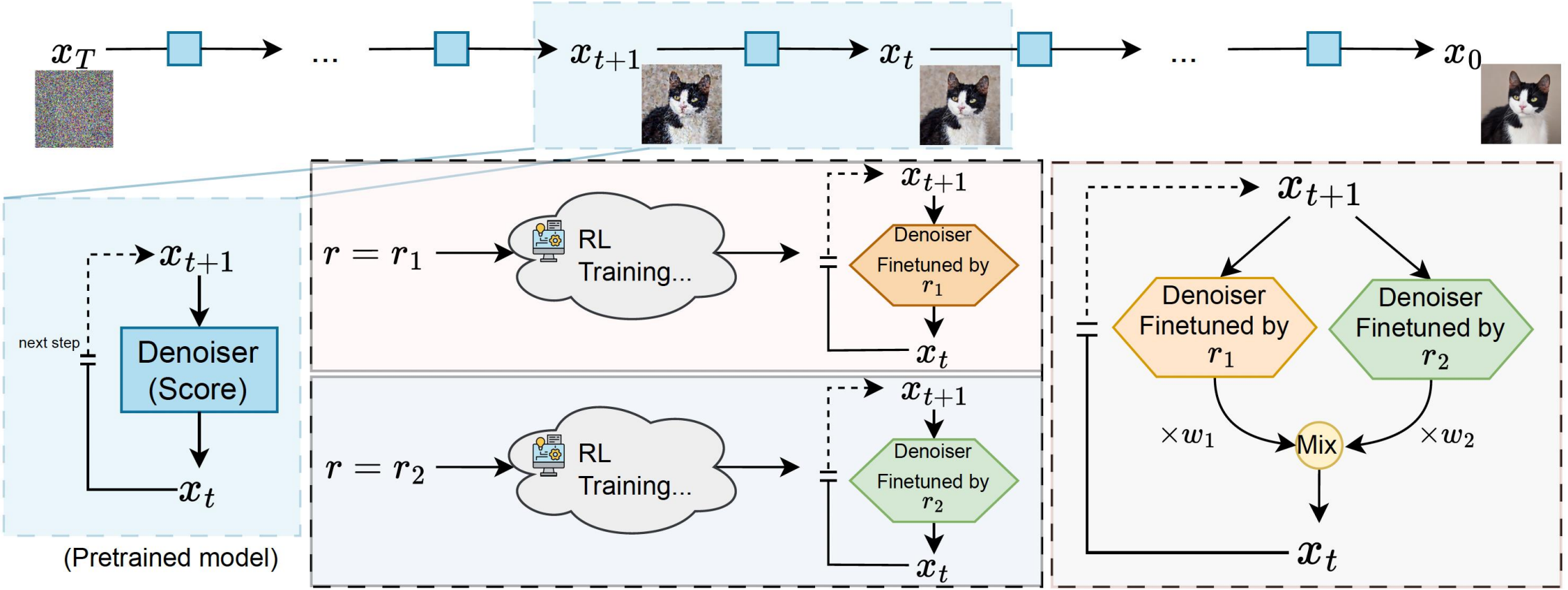
Given:

- Basis reward functions $r_1, \ldots, r_m$
- A pretrained diffusion model f_t^{pre}
- RL-aligned models for each reward $f_t^{r_i,\alpha}$



Can we generate samples aligned with any user-specified preference $r(w) = \sum_{i=1}^{m} w_i r_i$ and regularization level α, at inference time, without additional fine-tuning?

Diffusion Blend Overview

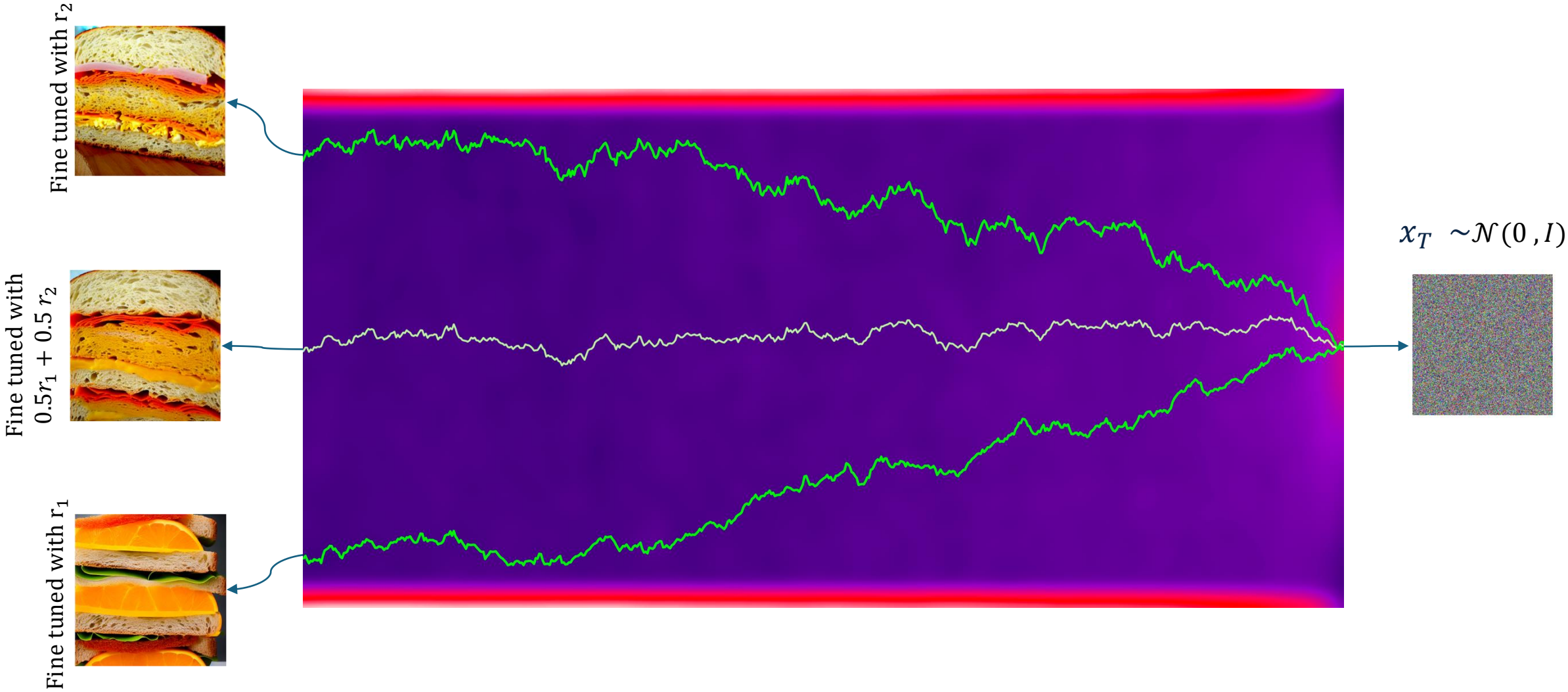


(a) RL Fine tuning for each objective.

(b) Blending diffusions.

Takeaway: $f_t^{blend} \approx \sum_i w_i f_t^{r_i,\alpha}$.

Blending the diffusion



r_1 denotes the aesthetics reward and r_2 denotes the text-image alignment reward.

Drift Blending Theorem

Proposition 1. *The aligned drift decomposes as:*

$$f^{r,\alpha}(x_t, t) = f^{pre}(x_t, t) - \beta(t) u^{(r,\alpha)}(x_t, t),$$

where the control term is:

$$u^{(r,\alpha)}(x_t, t) = \nabla_{x_t} \log E_{(x_0 \sim p^{pre}_{(0|t)}(\cdot|x_t))} \left[\exp\left(\frac{r(x_0)}{\alpha} \right) \right].$$

Jensen Approximation (interchange E and exp) :

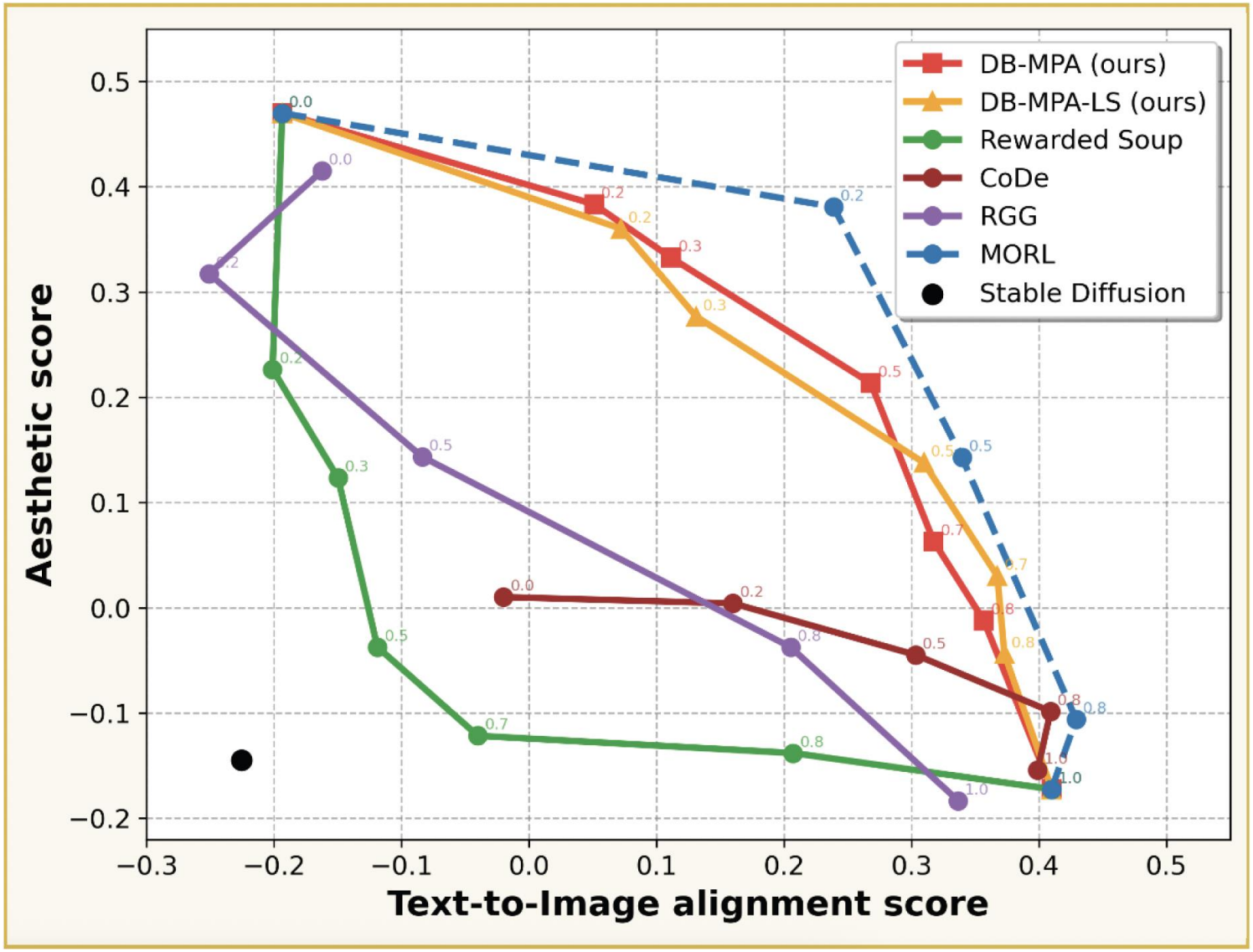
$$u^{(r,\alpha)} \approx \nabla_x \mathbb{E}[\frac{r(x_0)}{\alpha}]$$

Key consequence — Multi-Reward Setting:

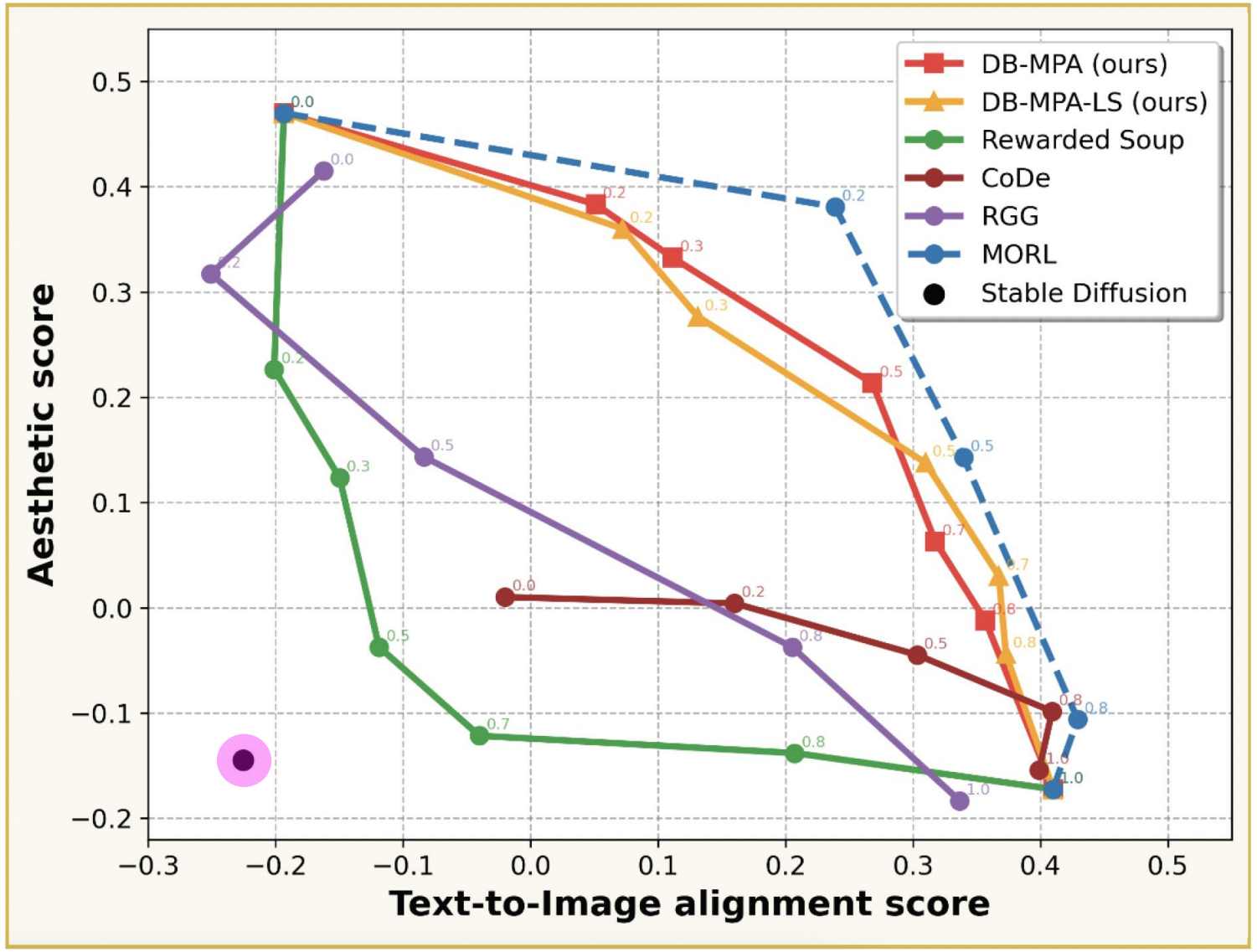
$$f^{(r(w),\alpha)} = \sum_i w_i f^{(r_i,\alpha)}$$

Where $r(w) = \sum_i w_i r_i$,

Results: Pareto Front



Results: Visual Comparison

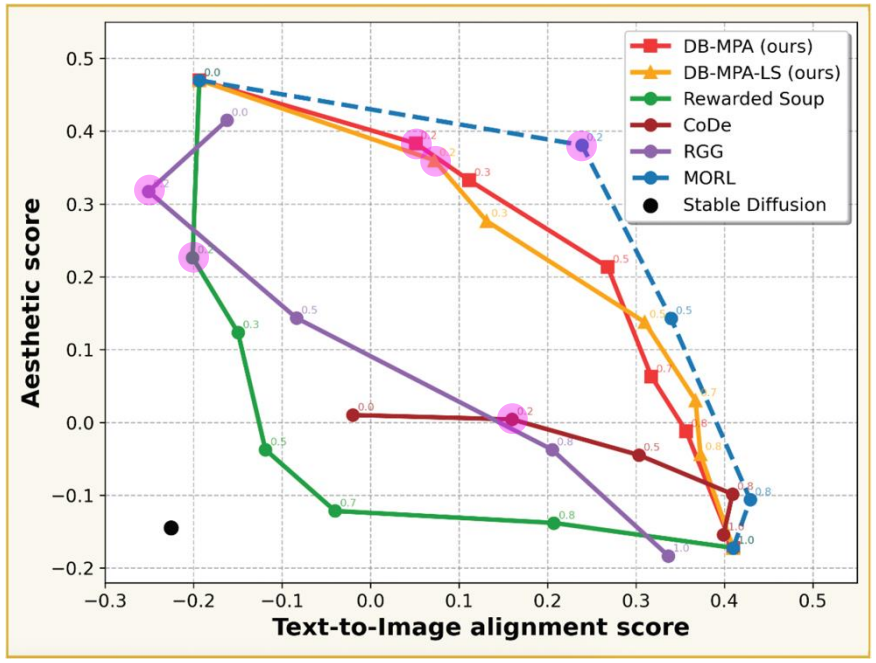


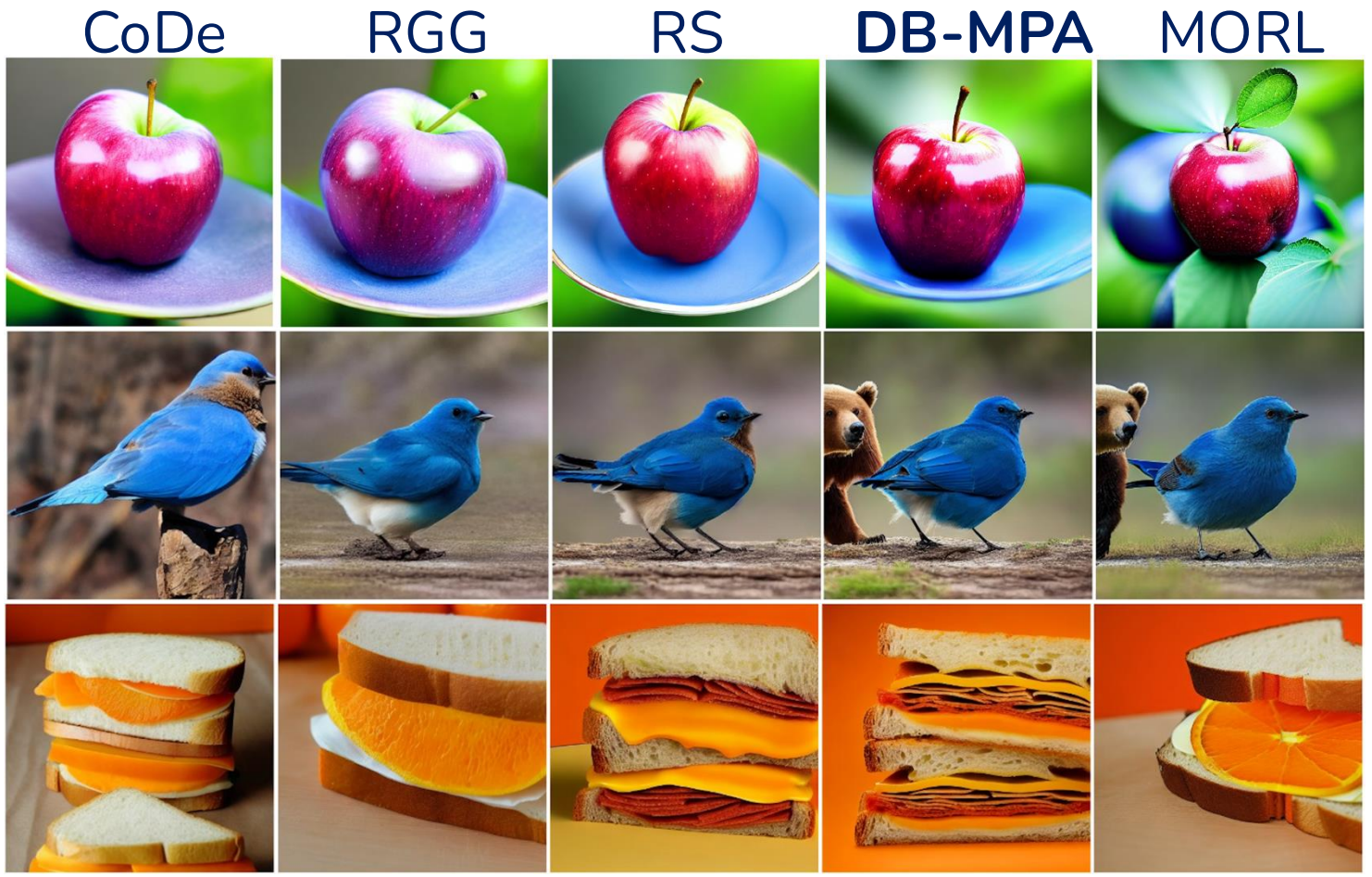
“A blue colored apple.”

“A blue bird and a brown bear.”

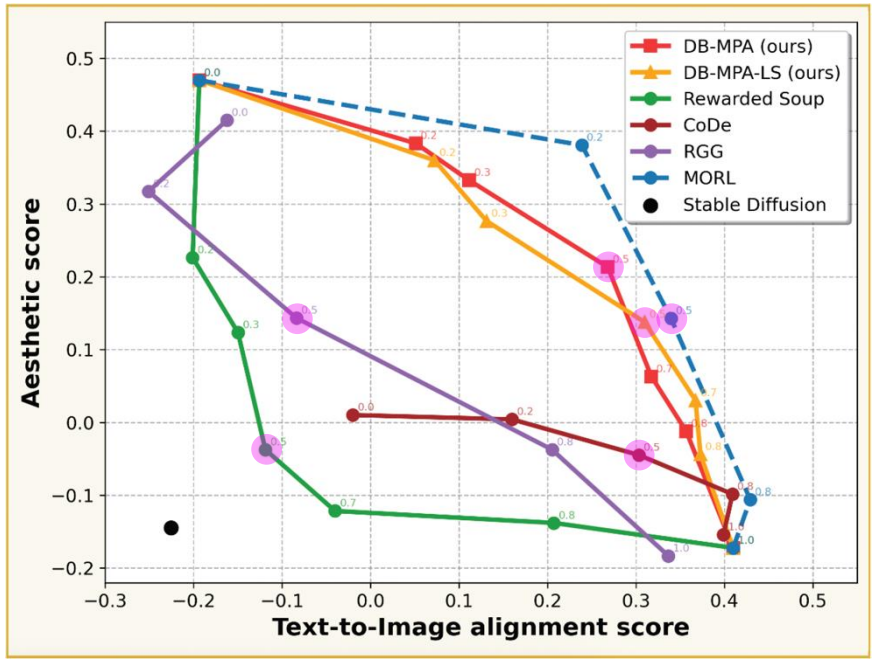
“An orange-colored Sandwich.”

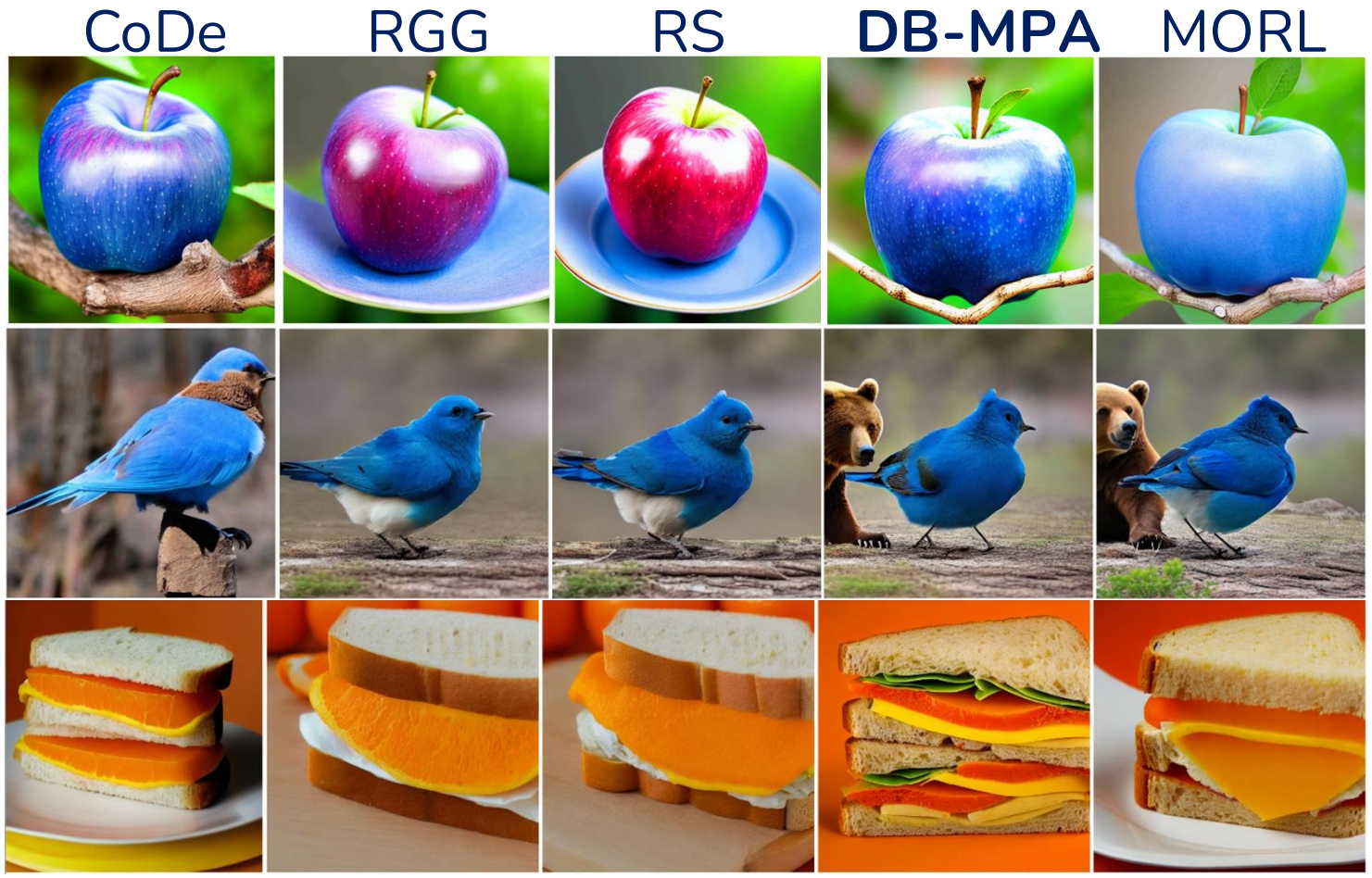
Results: Visual Comparison



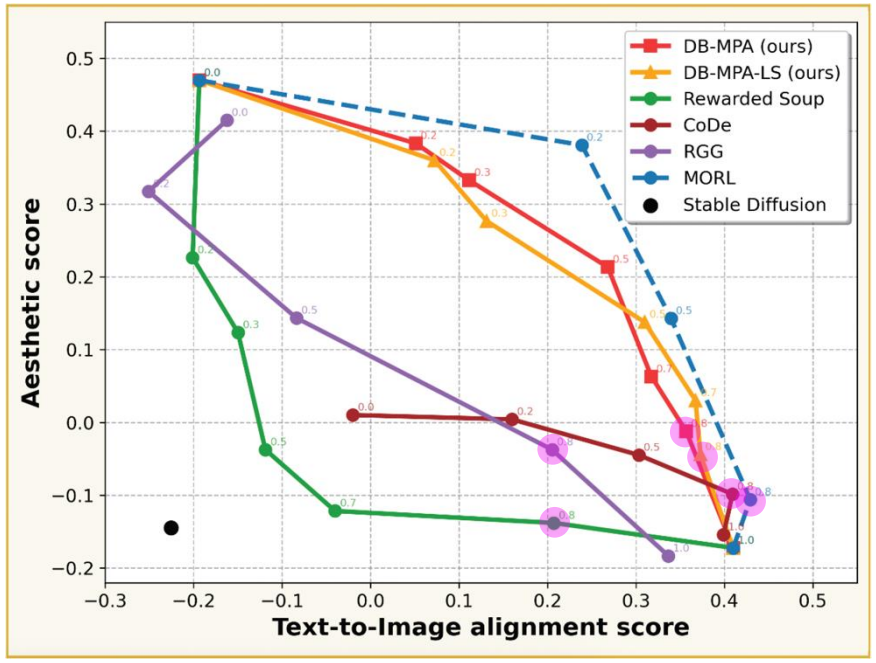


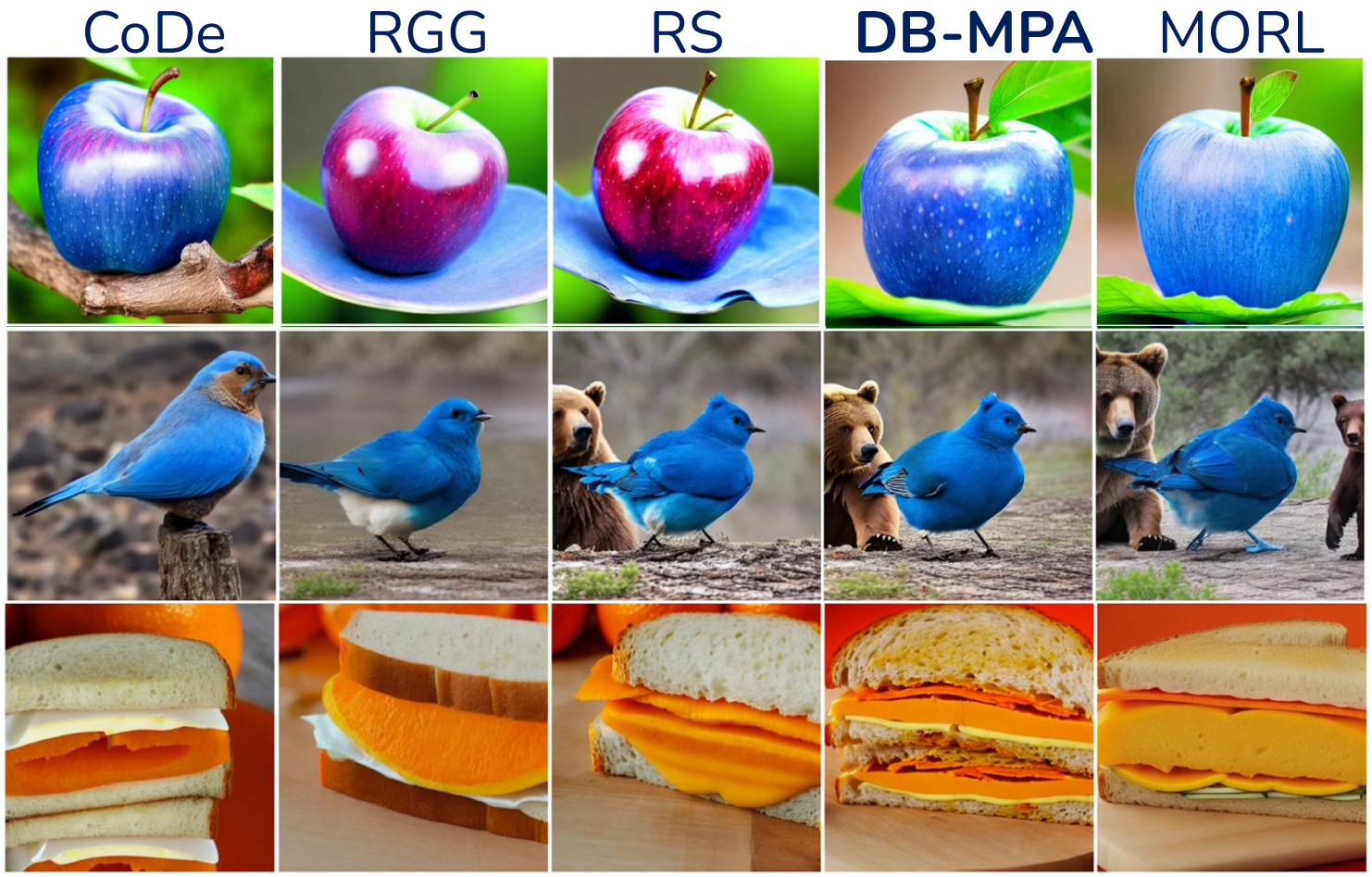
Results: Visual Comparison





Results: Visual Comparison





Controlling KL Regularization at Inference Time

Users may also want to control regularization strength:

$$\alpha(\lambda) = \frac{\alpha}{\lambda}$$

Problem:

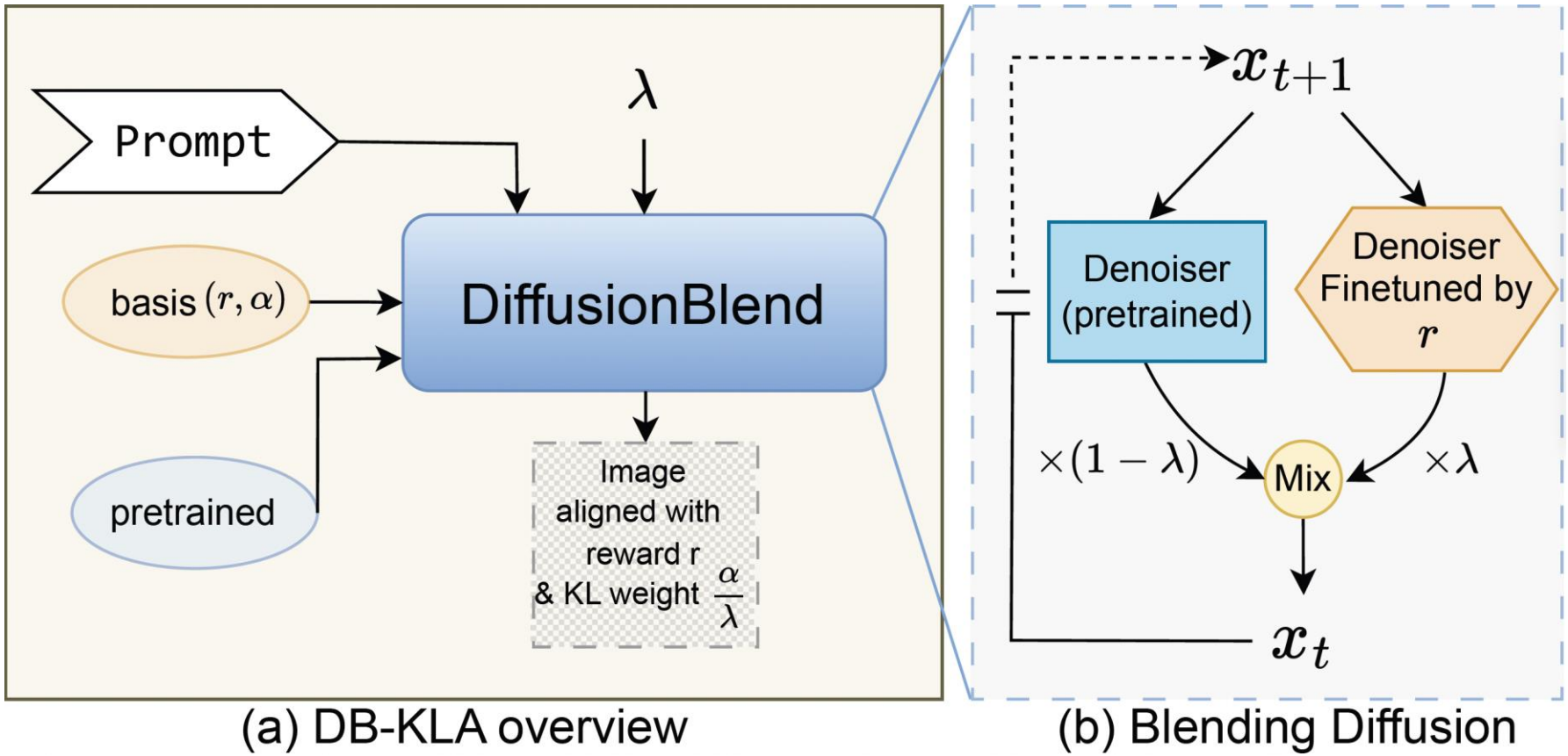
- Can we approximate models for arbitrary λ at inference time without additional fine-tuning?

Solution:

We show that $f^{(r,\alpha(\lambda))} \approx (1 - \lambda) f^{pre} + \lambda f^{(r,\alpha)}$

DB-KLA

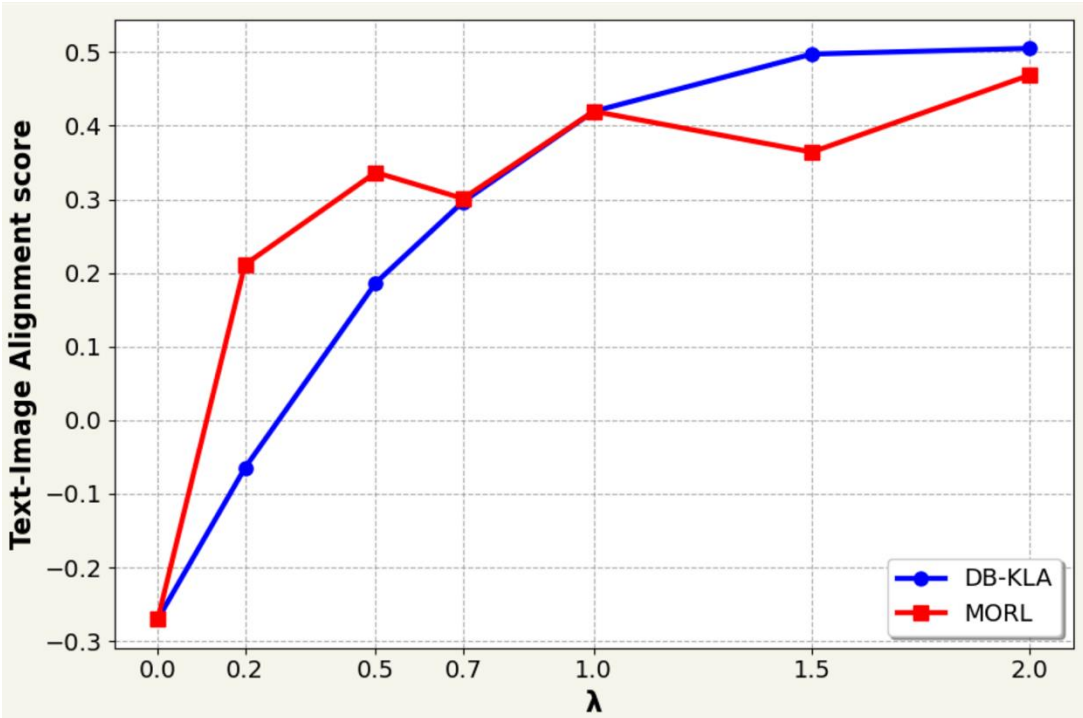
Users may also want to control regularization strength $\alpha(\lambda) = \frac{\alpha}{\lambda}$.



(a) DB-KLA overview

(b) Blending Diffusion

Results


Text-Image Alignment score
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
0.0
0.2
0.5
0.7
1.0
1.5
2.0
λ
DB-KLA
MORL


KL weight = 5α
KL weight = α
KL weight = 0.5α
Stable Diffusion
'A red apple and a purple backpack.'
Retrained Model
DB-KLA

PowerMamba: A Deep State Space Model and Comprehensive Benchmark for Time Series Prediction in Electric Power Systems

The sequence modeling Evolution:

RNNs/LSTMs: sequential, can't parallelize, vanishing gradients over long sequences.

Transformers: parallel, great at long-range dependencies, but attention is $O(n^2)$ which means for a 240-hour context window, that's 57,600 pairwise comparisons.

SSMs: model sequences as a dynamical system $h_t = \bar{A}h_{t-1} + \bar{B}x_t$.

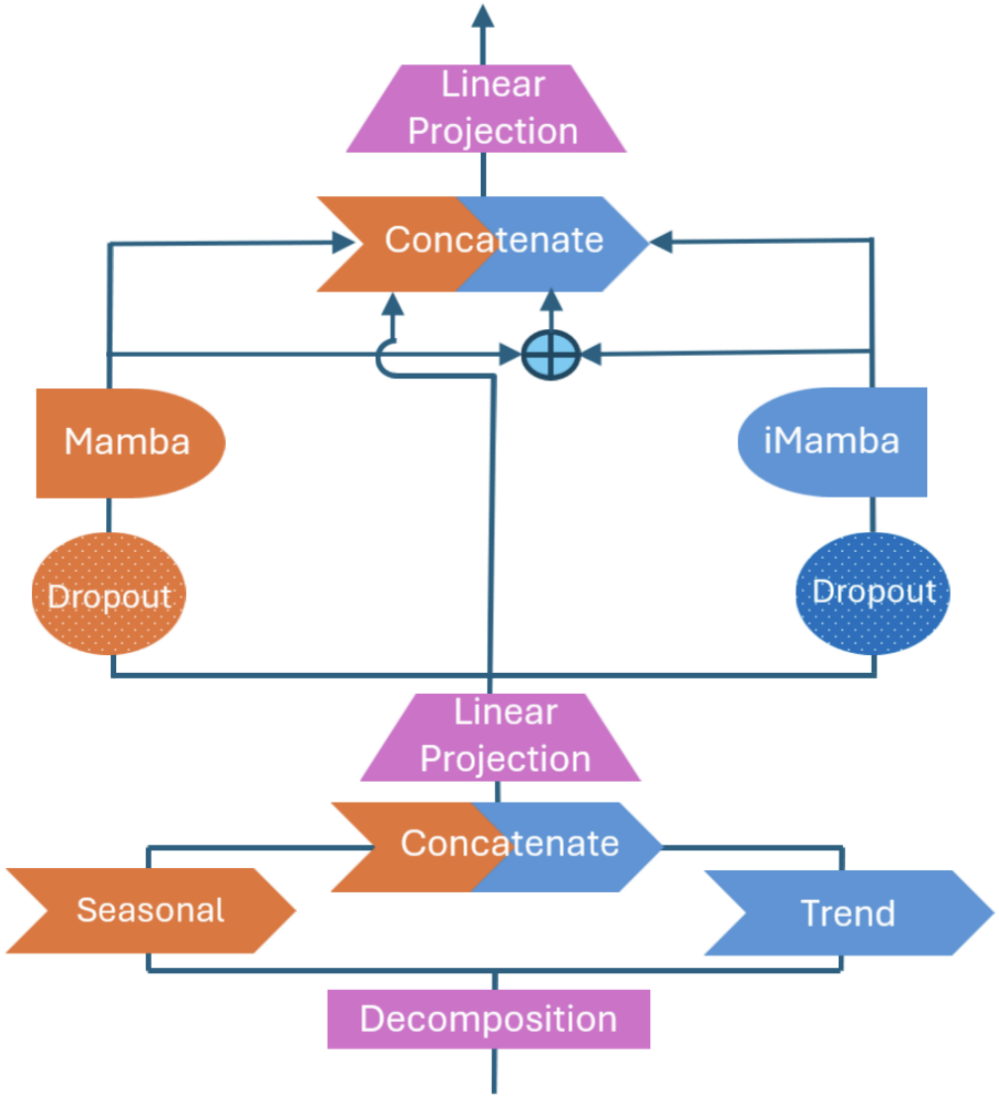
- Fixed parameters → output equals a convolution $y = K * x$ → parallel $O(n)$ training.
- At inference, just carry one hidden state forward → $O(1)$ per step.

Mamba: adds selectivity: A, B, C become input-dependent → the model learns what to remember and what to forget dynamically

Why Mamba for Power Grids?

- Power grid variables, load, price, wind, solar, evolve like **physical dynamical systems** governed by differential equations.
- Grid forecasting needs long context windows (240+ hours) to capture daily/weekly cycles → Transformer's $O(n^2)$ becomes prohibitive.
- Attention computes pairwise interactions across all time steps × all channels → for 22 channels × 240 hours = 5,280 tokens, attention matrix is 5,280 × 5,280 = 27.8M operations.
- To manage this, most Transformer forecasting models (like iTransformer) either attend across time or across channels ,not both simultaneously, so they still miss some interactions
- Mamba's dual-path architecture (Mamba + iMamba in parallel) handles both in one forward pass: one path sees L × D (time within each channel), the other sees D × L (across channels at each time step) → explicit joint modeling at linear cost

PowerMamba architecture



PowerMamba architecture

1. **Decomposition:**
   - split each time series into trend + seasonal components.
   - helps with highly volatile series like wind/solar.

2. **Dual-path Mamba + iMamba:** two parallel Mamba blocks:
   - Mamba sees L × D → captures **temporal patterns within each channel**
   - iMamba sees D × L (transposed) → captures **cross-channel dependencies**

3. **Fixed-size projections:**
   - linear projection maps input to fixed embedding size **E** before Mamba blocks.
   - Model size doesn't grow with context length or prediction window.

4. **External forecast integration:**
   - plugs in ERCOT's day-ahead forecasts for load and renewables without inflating model size.
   - 76% error reduction on renewables

GridSet

- 5 years of hourly data (2019–2023), ERCOT grid, Texas.
- 22 channels: 8 zonal loads, 8 zonal electricity prices, 4 ancillary service prices, 2 renewable generation (wind + solar).
- Extended version: 262 channels including ERCOT's day-ahead external forecasts for load and renewables.
- Captures real-world events: COVID demand drop, Winter Storm Uri, rapid renewable growth in Texas.

Why it matters:

- First open-access unified benchmark for multivariate power system forecasting
- Includes open-access toolbox with PowerMamba + 6 baseline models ready to benchmark

Results

TABLE III: Comparison of the prediction accuracy of our model and the baselines with and without external predictions. A fixed context size of $L = 240$ and a prediction window size of $W = 24$ are used for all the baselines.

Methods→		PowerMamba		TimeMachine		iTransformer		PatchTST		DLinear		TimesNET		Autoformer	
$\mathcal{D}$	PR	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
GridSet	✗	**0.129**	**0.166**	0.135	0.166	0.147	0.181	0.142	0.183	0.168	0.199	0.175	0.198	0.301	0.311
GridSet	✓	**0.074**	**0.123**	0.080	0.126	0.091	0.144	0.109	0.157	0.113	0.166	0.109	0.148	0.113	0.170
Load	✗	**0.098**	**0.223**	0.102	0.227	0.108	0.237	0.113	0.240	0.134	0.268	0.136	0.275	0.238	0.365
Load	✓	**0.065**	**0.183**	0.071	0.188	0.091	0.215	0.123	0.244	0.114	0.240	0.086	0.215	0.090	0.219
Price	✗	**0.100**	0.082	0.108	**0.079**	0.107	0.090	0.109	0.102	0.120	0.101	0.137	0.090	0.215	0.194
Price	✓	**0.093**	**0.075**	0.099	0.075	0.096	0.082	0.107	0.085	0.113	0.095	0.122	0.079	0.133	0.117
ASPrice	✗	**0.019**	**0.029**	0.020	0.029	0.021	0.033	0.022	0.038	0.022	0.037	0.028	0.032	0.089	0.153
ASPrice	✓	**0.018**	0.029	0.018	**0.026**	0.018	0.030	0.019	0.029	0.020	0.038	0.028	0.032	0.033	0.076
Renewables	✗	**0.590**	**0.545**	0.610	0.545	0.712	0.620	0.623	0.570	0.783	0.638	0.786	0.659	1.317	0.878
Renewables	✓	**0.142**	**0.267**	0.162	0.279	0.218	0.332	0.241	0.353	0.299	0.406	0.314	0.392	0.282	0.373

Results

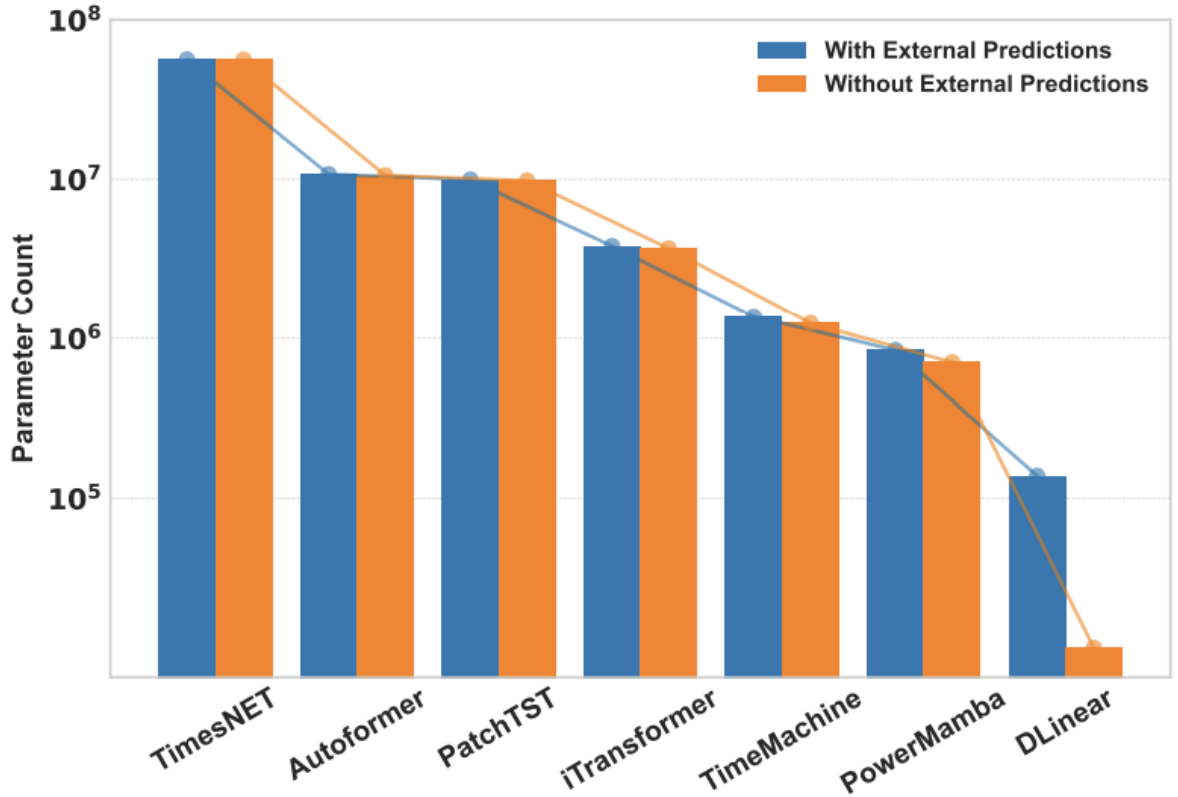


Fig. 5: The number of model parameters with and without external predictions, in log scale.