

## A Convergence of DTM to flow matching

Here we want to prove the following fact: Assume we have a sequence of Markov chains  $\{X_0, X_h, X_{2h}, \dots, X_1\}$ , with an initial state  $X_0 = x$ , where  $h = \frac{1}{T}$  and  $T \rightarrow \infty$ . For convenience note that we index the Markov states with fractions  $\ell h$ ,  $\ell \in [T]$ , and we denote the RV

$$Y_t = \frac{X_{t+h} - X_t}{h}. \quad (19)$$

Assume the Markov chains satisfy:

1. The function  $f_t(x) = \mathbb{E}[Y_t | X_t = x]$  is Lipschitz continuous. By Lipschitz we mean that  $\|f_s(y) - f_t(x)\| \leq c_L(|s - t| + \|x - y\|)$ .
2. For  $\ell \in [k]$  the quadratic variation satisfies,  $\mathbb{E}[\|Y_{\ell h}\|^2 | X_0 = x] \leq c(x)$ .

Let  $k = k(h) \in \mathbb{N}$  be an integer-valued function of  $h$  such that  $k \rightarrow \infty$  and  $\frac{1}{2} \geq kh \rightarrow 0$  as  $h \rightarrow 0$ . We will prove that the random variable

$$\frac{X_{kh} - X_0}{kh} \quad (20)$$

converges in mean to  $f_0(x)$ . That is, we want to show

**Theorem 2.** *Considering a sequence of Markov processes  $\{X_0, X_h, X_{2h}, \dots, X_1\}$  satisfying the assumptions above, then*

$$\lim_{h \rightarrow 0} \mathbb{E} \left[ \left\| \frac{X_{kh} - X_0}{kh} - f_0(X_0) \right\|^2 \middle| X_0 = x \right] = 0. \quad (21)$$

*Proof.* First,

$$\mathbb{E} \left[ \left\| \frac{X_{kh} - X_0}{kh} - f_0(X_0) \right\|^2 \middle| X_0 = x \right] = \mathbb{E} \left[ \left\| \frac{1}{k} \sum_{\ell=0}^{k-1} Y_{\ell h} - f_0(X_0) \right\|^2 \middle| X_0 = x \right] \quad (22)$$

and if we open the squared norm we get three terms:

$$\mathbb{E} [\|f_0(X_0)\|^2 | X_0 = x] = \|f_0(x)\|^2. \quad (23)$$

$$\mathbb{E} \left[ \frac{1}{k} \sum_{\ell=0}^{k-1} Y_{\ell h} \cdot f_0(X_0) \middle| X_0 = x \right] = f_0(x) \cdot \frac{1}{k} \sum_{\ell=0}^{k-1} \mathbb{E} [Y_{\ell h} | X_0 = x] \quad (24)$$

$$\begin{aligned} \mathbb{E} \left[ \frac{1}{k^2} \sum_{\ell=0}^{k-1} \sum_{m=0}^{k-1} Y_{\ell h} \cdot Y_{mh} \middle| X_0 = x \right] &= \frac{1}{k^2} \sum_{\ell=0}^{k-1} \mathbb{E} [\|Y_{\ell h}\|^2 | X_0 = x] \\ &\quad + \frac{2}{k^2} \sum_{\ell=0}^{k-1} \sum_{m=0}^{\ell-1} \mathbb{E} [Y_{\ell h} \cdot Y_{mh} | X_0 = x] \end{aligned}$$

We will later show that  $\mathbb{E}[Y_{\ell h} | X_0 = x] = f_0(x) + O(kh)$  and for  $\ell \neq m$  we have  $\mathbb{E}[Y_{\ell h} \cdot Y_{mh} | X_0 = x] = \|f_0(x)\|^2 + O(kh)$ . Plugging these we get that equation 22 equals

$$\|f_0(x)\|^2 - 2\|f_0(x)\|^2 + \frac{k^2 - k}{k^2} \|f_0(x)\|^2 + O(kh + k^{-1}) \rightarrow 0, \quad (25)$$

508 as  $h \rightarrow 0$ , where we used assumption 2 above to bound  $\mathbb{E} [\|Y_{\ell h}\|^2 | X_0 = x] \leq c(x)$ . Now to  
 509 conclude we show

$$\|\mathbb{E} [Y_{\ell h} | X_0 = x] - f_0(x)\| = \|\mathbb{E} [\mathbb{E} [Y_{\ell h} | X_{\ell h}] | X_0 = x] - f_0(x)\| \quad (26)$$

$$= \|\mathbb{E} [f_{\ell h}(X_{\ell h}) | X_0 = x] - f_0(x)\| \quad (27)$$

$$= \|\mathbb{E} [f_{\ell h}(X_{\ell h}) - f_0(X_0) | X_0 = x]\| \quad (28)$$

$$= \mathbb{E} [\|f_{\ell h}(X_{\ell h}) - f_0(X_0)\| | X_0 = x] \quad (29)$$

$$\leq c_L \mathbb{E} [\ell h + \|X_{\ell h} - X_0\| | X_0 = x] \quad (30)$$

$$\leq c_L \mathbb{E} \left[ \ell h + \sum_{m=0}^{\ell-1} \|X_{(m+1)h} - X_{mh}\| \middle| X_0 = x \right] \quad (31)$$

$$\leq O(kh) + c_L h \sum_{m=0}^{\ell-1} \mathbb{E} [\|Y_{mh}\| | X_0 = x] \quad (32)$$

$$\leq O(kh) + c_L h k \sqrt{c(x)} \quad (33)$$

$$= O(hk). \quad (34)$$

510 Now for  $m < \ell$  we have

$$|\mathbb{E} [Y_{\ell h} \cdot Y_{mh} | X_0 = x] - f_0(x) \cdot \mathbb{E} [Y_{mh} | X_0 = x]| \quad (35)$$

$$= |\mathbb{E} [Y_{mh} \cdot (Y_{\ell h} - f_0(X_0)) | X_0 = x]| \quad (36)$$

$$= |\mathbb{E} [Y_{mh} \cdot \mathbb{E} [Y_{\ell h} - f_0(X_0) | X_{\ell h}] | X_0 = x]| \quad (37)$$

$$= |\mathbb{E} [Y_{mh} \cdot (f_{\ell h}(X_{\ell h}) - f_0(X_0)) | X_0 = x]| \quad (38)$$

$$\leq \mathbb{E} [|Y_{mh} \cdot (f_{\ell h}(X_{\ell h}) - f_0(X_0))| | X_0 = x] \quad (39)$$

$$\leq \mathbb{E} [\|Y_{mh}\| \|f_{\ell h}(X_{\ell h}) - f_0(X_0)\| | X_0 = x] \quad (40)$$

$$\leq \mathbb{E} \left[ \|Y_{mh}\| c_l (kh + \sum_{j=0}^{\ell-1} \|X_{(j+1)h} - X_{jh}\|) | X_0 = x \right] \quad (41)$$

$$\leq \mathbb{E} \left[ \|Y_{mh}\| c_l (kh + h \sum_{j=0}^{\ell-1} \|Y_{jh}\|) | X_0 = x \right] \quad (42)$$

$$\leq O(kh) + c_L h \mathbb{E} \left[ \sum_{j=0}^{\ell-1} \|Y_{mh}\| \|Y_{jh}\| | X_0 = x \right] \quad (43)$$

$$\leq O(kh) + \frac{c_L h}{2} \mathbb{E} \left[ \sum_{j=0}^{\ell-1} \|Y_{mh}\|^2 + \|Y_{jh}\|^2 | X_0 = x \right] \quad (44)$$

$$= O(kh). \quad (45)$$

511 Therefore,

$$|\mathbb{E} [Y_{\ell h} \cdot Y_{mh} | X_0 = x] - f_0(x) \cdot f_0(x)| \quad (46)$$

$$\leq |\mathbb{E} [Y_{\ell h} \cdot Y_{mh} | X_0 = x] - f_0(x) \cdot \mathbb{E} [Y_{mh} | X_0 = x]| \quad (47)$$

$$+ |f_0(x) \cdot \mathbb{E} [Y_{mh} | X_0 = x] - f_0(x) \cdot f_0(x)| \quad (48)$$

$$\leq O(kh), \quad (49)$$

512 where we used equation 34 and equation 45, and the proof is done since  $kh \rightarrow 0$  as  $h \rightarrow 0$ .  $\square$

### 513 A.1 The DTM case

514 We note show that the DTM process satisfies the two assumptions above. We recall that the DTM  
515 process is defined by  $Y_t \sim q_{Y|t}(\cdot|X_t)$  where  $Y = X_1 - X_0$ .

516 First we check the Lipchitz property.

$$f_t(x) = \mathbb{E}[Y_t|X_t = x] \quad (50)$$

$$= \mathbb{E}[X_1 - X_0|X_t = x] \quad (51)$$

$$= u_t(x) \quad (52)$$

$$= \int \frac{x_1 - x}{1 - t} p_{1|t}(x_1|x) dx_1 \quad (53)$$

$$= \int \frac{x_1 - x}{1 - t} \frac{p_{t|1}(x|x_1)p_1(x_1)}{\int p_{t|1}(x|x'_1)p_1(x'_1)dx'_1} dx_1 \quad (54)$$

517 which is Lipschitz for  $t < 1$  as long as  $p_{t|1}(x|x_1) > 0$  for all  $x$ , and is continuously differentiable in  
518  $t$  and  $x$ , both hold for the Gaussian kernel  $p_{t|1}(x|x_1) = \mathcal{N}(x|tx_1, (1-t)I)$ .

519 Let us check the second property. For this end we make the realistic assumption that our data is  
520 bounded, i.e.,  $\|X_1\| \leq r$  for some constant  $r > 0$ . Then, consider some RV  $X'_1 - X'_0 = Y_t \sim$   
521  $p_{Y|t}(\cdot|X_t)$ . Then by definition we have that  $X_{t+h} = X_t + h(X'_1 - X'_0)$  and  $X_t = (1-t)X'_0 + tX'_1$ .  
522 Therefore,

$$\|X_{t+h}\| = \|X_t + h(X'_1 - X'_0)\| \quad (55)$$

$$= \left\| \frac{(1-t-h)X_t + h(1+t)X'_1}{(1-t)} \right\| \quad (56)$$

$$\leq \frac{(1-(t+h))}{(1-t)} \|X_t\| + h \frac{1+t}{1-t} r. \quad (57)$$

523 We apply this to  $t+h = \ell h$  where  $\ell \in [k]$  and therefore

$$\|X_{\ell h}\| \leq \|X_{(\ell-1)h}\| + h \frac{1+kh}{1-kh} r \quad (58)$$

$$\leq \|X_0\| + kh \frac{1+kh}{1-kh} r \quad (59)$$

$$\leq \|x\| + \frac{3r}{2} = \tilde{c}(x) \quad (60)$$

524 where we used  $kh \leq \frac{1}{2}$ . Finally,

$$\mathbb{E}[\|Y_t\|^2 | X_0 = x] = \mathbb{E}[\mathbb{E}[\|Y_t\|^2 | X_t] | X_0 = x] \quad (61)$$

$$= \mathbb{E}[\mathbb{E}[\|X'_1 - X'_0\|^2 | X_t] | X_0 = x] \quad (62)$$

$$\leq \mathbb{E}[\mathbb{E}[\left\| X'_1 - \frac{X_t - tX'_1}{(1-t)} \right\|^2 | X_t] | X_0 = x] \quad (63)$$

$$\leq \mathbb{E}[\mathbb{E}[\left\| \frac{X_t - (1+t)X'_1}{(1-t)} \right\|^2 | X_t] | X_0 = x] \quad (64)$$

$$\leq 2\mathbb{E}\left[\frac{1}{(1-t)^2} \|X_t\|^2 + \frac{(1+t)^2}{(1-t)^2} r^2 | X_0 = x\right], \quad (65)$$

525 where we used again  $X_t = (1-t)X'_0 + tX'_1$ . Lastly, applying this to  $t = \ell h \leq kh \leq \frac{1}{2}$  and using  
526 equation 60 we get

$$\mathbb{E}[\|Y_{k\ell}\|^2 | X_0 = x] \leq \frac{2}{(1-t)^2} \tilde{c}(x)^2 + 2 \frac{(1+t)^2}{(1-t)^2} r^2 = c(x). \quad (66)$$

## B Experiments

### B.1 Implementation details

**DiT architecture** The DiT architecture [27] uses a 24 blocks of a self-attention layer followed by cross attention layer with the text embedding [31], with a 2048 hidden dimension, 16 attention heads, and utilize a 3D positional embedding [4]. Embedded image [28] size is  $32 \times 32 \times 4$  and input to the DiT through a patchify layer with patch size of  $2 \times 2 \times 4$ . The total number of parameters is 1.7B.

**LLM architecture** The LLM architecture [25] is similar to the DiT with the following differences: (i) time injection is removed, (ii) cross attention layer is removed and text embedding is input as a prefix (iii) it uses a simple 1D instead of 3D positional embedding. To compensate for reduction in number of parameters, we increase the number of self-attention layers to 34, reaching 1.7B total number of parameters (comparable to the DiT).

**Flow head architecture** Following [21] we use an MLP with 6 layers and a hidden dimension of 1024. to convert from the backbone hidden dimension (2048) to the MLP hidden dimension (1024) we use a simple linear layer. Finally, we replace the time input with AdaLN [27] time injection.

**Optimization** The models are trained for 500K iterations, with a 2048 total batch size,  $1 * e^{-4}$  constant learning rate and 2K iterations warmup.

**Classifier guidance free** To support classifier guidance free (CFG), during training, with probability of 0.15, we drop the text prompt and replace it with empty prompt. Following [21], during sampling, we apply CFG to the velocity of the flow head ( $g^\theta$ ) with a guidance scale of 6.5.

### B.2 Main result

**Flow head NFE** We ablate the number of NFE required by the flow head ( $g^\theta$ ) to reach best performance for each model. As shown in Figure 9, we observe the models reach saturation with relatively low NFE, and decide to report results on Tables 1, 3 and 2 with 64 NFE for the flow head.

**TM steps for DTM** We test the performance of the DTM variant as function of TM steps. As shown in Figure 8, our DTM model achieve reach saturation about 32 TM steps according to CLIPScore and PickScore.

Table 3: Evaluation of TM versus baselines on MS-COCO. <sup>†</sup> Inference is done with activation caching. NFE\* counts only backbone model evaluation ( $f^\theta$ ). LLM and DiT have comparable number of parameters.

	Attention	Kernel	Arch	NFE*	CLIPScore $\uparrow$	PickScore $\uparrow$	ImageReward $\uparrow$	UnifiedReward $\uparrow$	AestheticScore $\uparrow$	DeQAScore $\uparrow$
Baseline	Full	MAR	DiT	256	26.1	20.7	0.17	4.62	5.06	2.34
		MAR-Fulid	DiT	256	25.5	20.5	-0.11	3.94	4.86	2.38
		FM	DiT	256	25.8	21.1	0.09	5.00	5.45	2.47
TM		<b>DTM</b>	DiT	32	26.2	21.2	0.22	5.38	5.55	2.58
Baseline		AR <sup>†</sup>	DiT	256	24.8	20.1	-0.48	3.60	4.76	2.34
TM	Causal	<b>ARTM-2<sup>†</sup></b>	DiT	$2 \times 256$	25.9	20.8	0.07	4.70	5.19	2.41
		<b>FHTM-2<sup>†</sup></b>	DiT	$2 \times 256$	25.9	20.8	0.07	4.78	5.27	2.45
		<b>ARTM-3<sup>†</sup></b>	DiT	$3 \times 256$	26.1	20.9	0.11	4.99	5.35	2.46
		<b>FHTM-3<sup>†</sup></b>	DiT	$3 \times 256$	26.1	21.0	0.15	5.23	5.38	2.41
		<b>FHTM-3<sup>†</sup></b>	LLM	$3 \times 256$	26.1	21.1	0.24	5.51	5.53	2.51

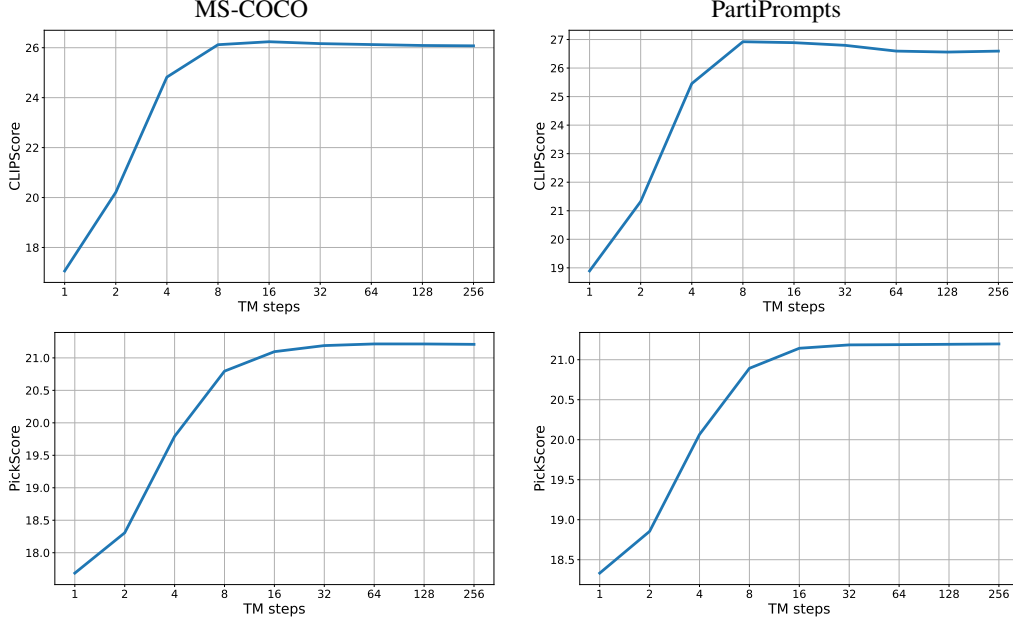


Figure 8: Comparison of TM steps with DTM variant vs. CLIPScore (top), and PickScore (bottom) computed on both MS-COCO (left) and PartiPrompts (right) datasets.

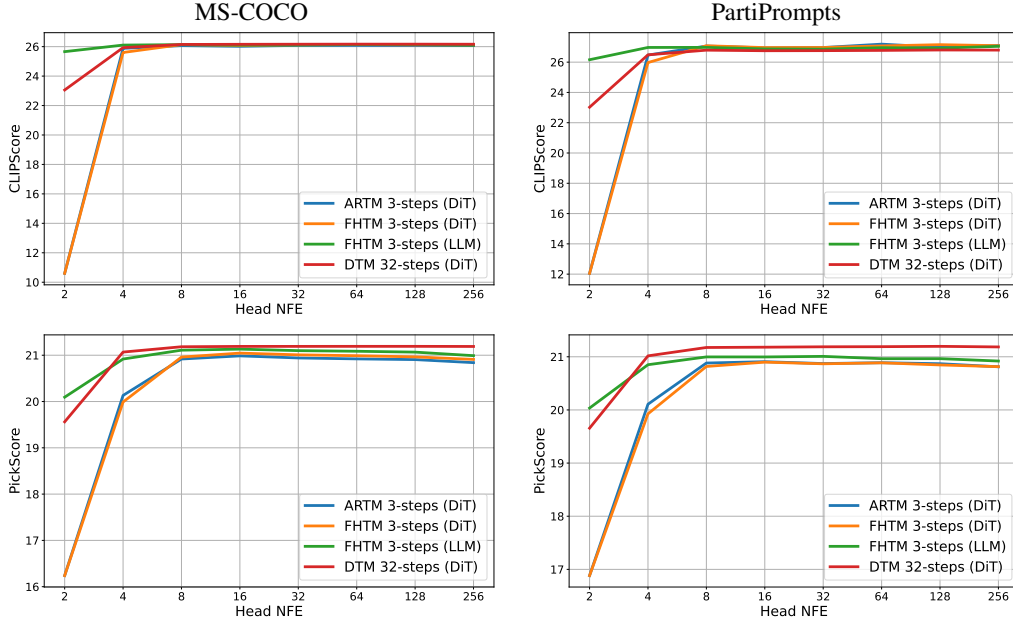


Figure 9: Comparison of flow head NFE vs. CLIPScore (top), and PickScore (bottom) computed on both MS-COCO (left) and PartiPrompts (right) datasets.

### 553 B.3 Dependent vs. independent linear process

554 Further analysis of the generated images reveals that the AR kernels are unable to learn the linear  
555 process, resulting in low quality image generation. We hypothesize that the AR kernels exploit the  
556 linear relationship between  $X_t$  and  $X_{t+1}$  during training, which leads the model to learn a degenerate  
557 function and causes it to fail in inference.

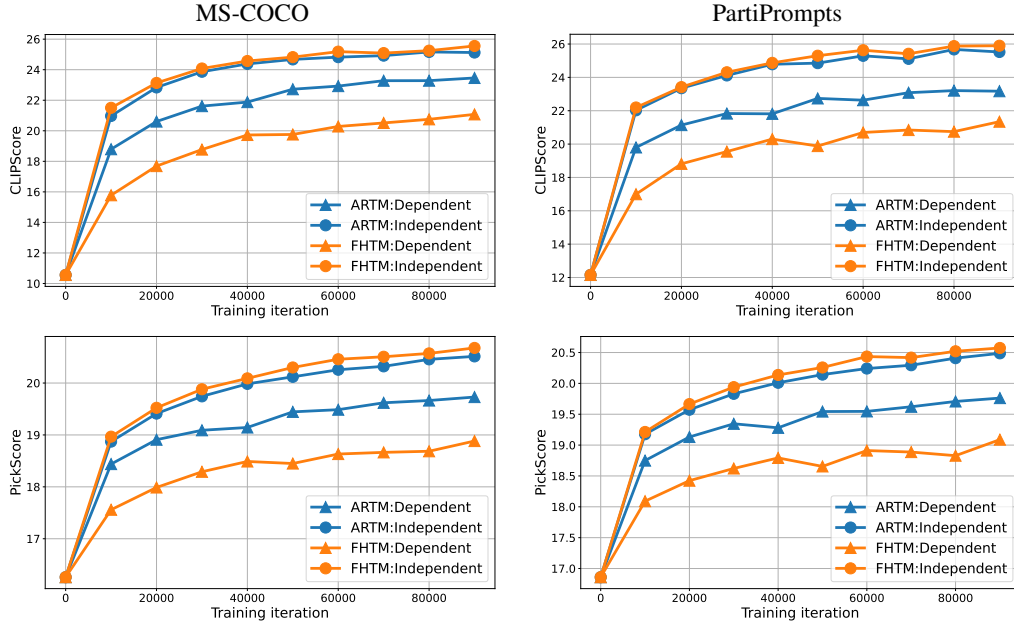


Figure 10: Dependent linear process (10) vs. Independent linear process (15) on the AR kernels: ARTM-3 and FHTM-3. The models are evaluated on the MS-COCO (left) and PartiPrompts (right) with CLIPScore and PickScore every 10K training iterations across 100K iterations. Observe that on the AR kernels trained with the independent linear process are far superior to the ones trained with the dependent linear process.

## 558 B.4 DTM Kernel expressiveness

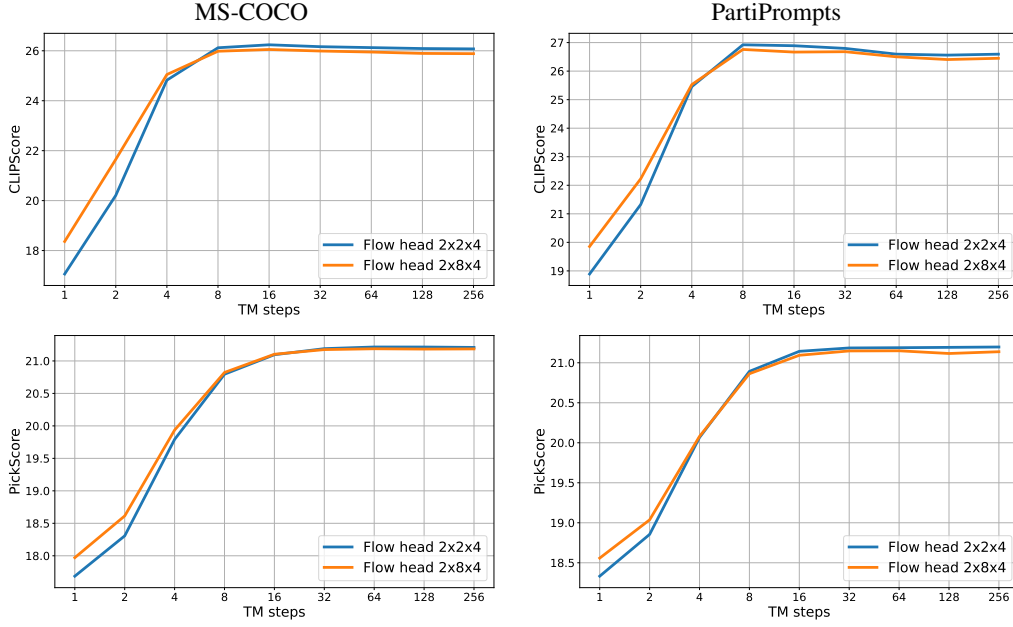


Figure 11: Impact of flow head patch size:  $2 \times 2 \times 4$  vs.  $2 \times 8 \times 4$ , on the DTM performance, evaluated across varying numbers of TM steps. The metrics are CLIPScore vs. TM steps and PickScore vs. TM steps computed on both MS-COCO (left) and PartiPrompts (right) datasets. On low number of TM steps, the larger flow head patch size shows an advantage in both metrics. On high number of TM steps, both patch sizes yield comparable results. This aligns with Theorem 1, which predicts that for infinitesimal steps size, the entries of  $Y \in \mathbb{R}^d$  defined in equation 11 become independent.

## 559 B.5 Scheduler ablation for independent linear process

560 We have experimented with two transition scheduler options: uniform (as described in 15) and  
 561 "exponential", i.e.,  $\frac{t}{T} \in \{0, 0.5, 0.75, 1\}$ . The results for ARTM and FHTM are reported in Table 4  
 562 and show almost the same performance with a slight benefit towards exponential in DiT architecture and these are used in our main implementations.

Kernel	Arch	TM Steps	MS-COCO		PartiPrompts		
			Scheduler	ClipScore $\uparrow$	PickScore $\uparrow$	ClipScore $\uparrow$	PickScore $\uparrow$
ARTM	DiT	3	Uniform	26.0	20.8	26.8	20.8
			Exponential	26.1	20.9	27.0	20.9
FHTM	DiT	3	Uniform	25.9	21.0	26.9	20.9
			Exponential	26.1	21.0	27.0	20.9
FHTM	LLM	3	Uniform	26.1	21.0	27.0	21.0
			Exponential	26.1	21.1	27.0	21.0

Table 4: Comparison of uniform and exponential transition steps.

563

## 564 B.6 Additional generated images comparison

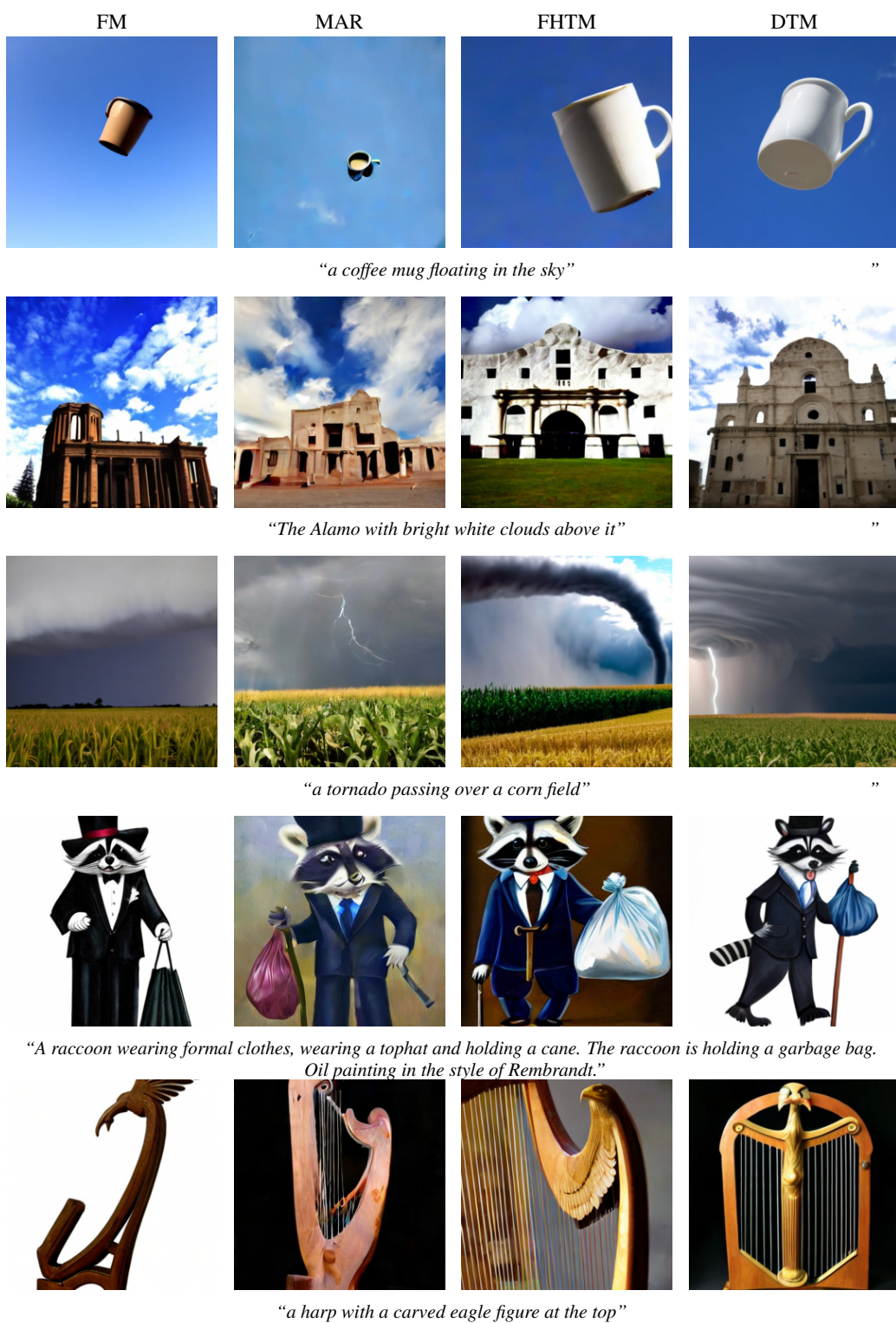


Figure 12: Additional generated samples of FM, MAR, FHTM, and DTM with models that is trained for 1M iterations.

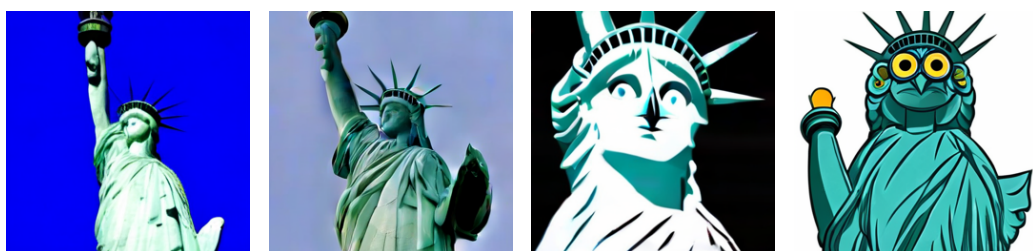




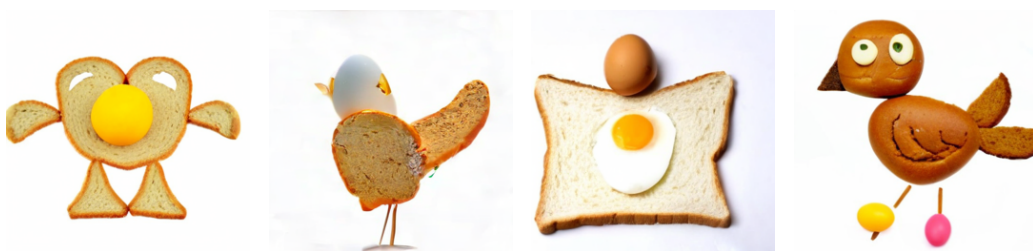
*"A close-up photo of a wombat wearing a red backpack and raising both arms in the air. Mount Rushmore is in the background."*



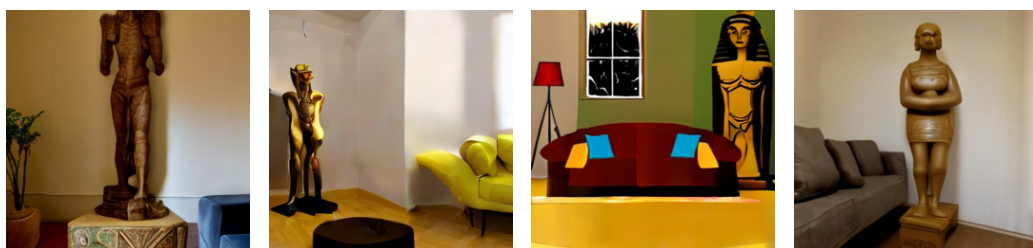
*"a blue airplane taxiing on a runway with the sun behind it"*



*"The Statue of Liberty with the face of an owl"*

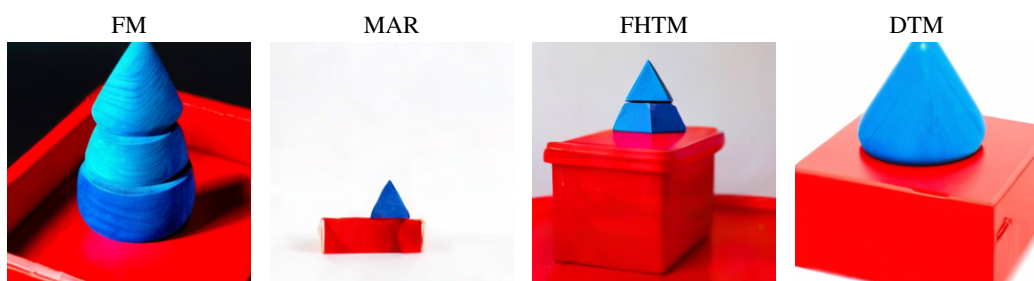


*"A photograph of a bird made of wheat bread and an egg."*



*"a living room with a large Egyptian statue in the corner"*

Figure 13: Additional generated samples of FM, MAR, FHTM, and DTM with models that is trained for 1M iterations.



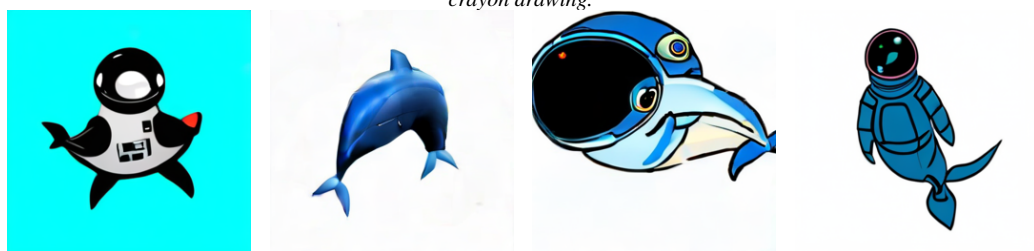
*“a blue wooden pyramid on top of a red plastic box”*



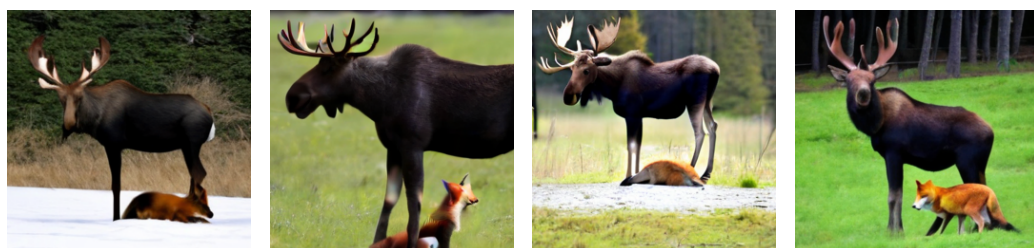
*“A bowl of soup that looks like a monster knitted out of wool”*



*“Portrait of a gecko wearing a train conductor's hat and holding a flag that has a yin-yang symbol on it. Child's crayon drawing.”*



*“a dolphin in an astronaut suit”*



*“a moose standing over a fox”*

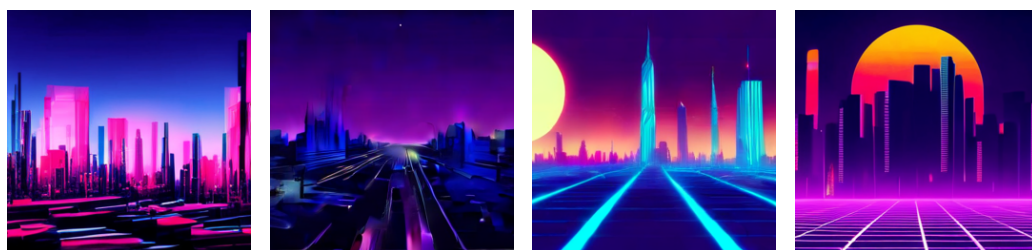
Figure 14: Additional generated samples of FM, MAR, FHTM, and DTM with models that is trained for 1M iterations.



*“a portrait of a statue of a pharaoh wearing steampunk glasses, white t-shirt and leather jacket. dslr photograph.”*



*“panda mad scientist”*



*“a futuristic city in synthwave style”*

Figure 15: Additional generated samples of FM, MAR, FHTM, and DTM with models that is trained for 1M iterations.



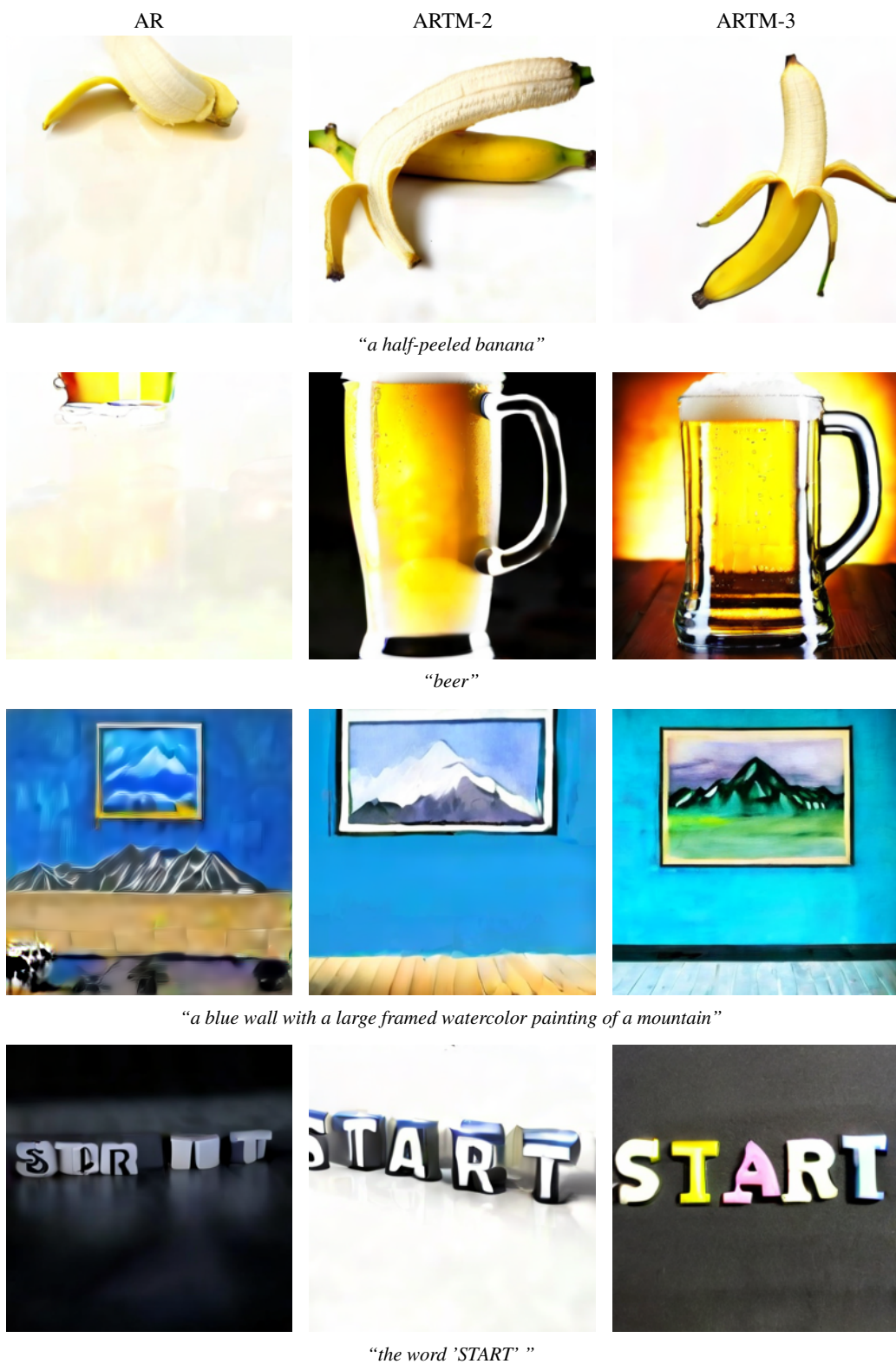


Figure 16: Samples comparison of AR (left) vs. ARTM-2 (middle) vs. ARTM-3 (right) on models trained for 500K iteration with the DiT architecture.

## 565 **C Training and sampling algorithms**

566 Algorithms 1 and 2 describe and training and sampling (resp.) of transition matching for a general  
567 supervision process, kernel parametrization, and kernel modeling. In this section, we provide training  
568 and sampling algorithms tailored to the specific design choices of our three variants: (i) DTM is  
569 described in Figure 17, (ii) ARTM is described in Figure 18, and (iii) FHTM is described in Figure 19.  
570 Additionally, we provide Python code of a training step for each variant: (i) DTM in Figure 20, (ii)  
571 ARTM in Figure 21, and (iii) FHTM in Figure 22.

---

**Algorithm 3** DTM Training

---

**Require:**  $p_T$  ▷ Data  
**Require:**  $T$  ▷ Number of TM steps

- 1: **while** not converged **do**
- 2:   Sample  $X_T \sim p_T$
- 3:   Sample  $t \sim \mathcal{U}([T-1])$
- 4:   Sample  $X_0 \sim N(0, I_d)$
- 5:    $X_t \leftarrow (1 - \frac{t}{T}) X_0 + \frac{t}{T} X_T$
- 6:    $Y \leftarrow X_T - X_0$
- 7:    $h_t \leftarrow f_t^\theta(X_t)$
- 8:   **parallel for**  $i = 1, \dots, n$  **do**
- 9:     Sample  $Y_0^i \sim N(0, I_{d/n})$
- 10:    Sample  $s \sim \mathcal{U}([0, 1])$
- 11:     $Y_s^i \leftarrow (1 - s) Y_0^i + s Y^i$
- 12:     $\mathcal{L}^i(\theta) \leftarrow \|g_{s,t}^\theta(Y_s^i, h_t^i) - (Y^i - Y_0^i)\|^2$
- 13:   **end for**
- 14:    $\mathcal{L}(\theta) \leftarrow \frac{1}{n} \sum_i \mathcal{L}^i(\theta)$
- 15:    $\theta \leftarrow \theta - \gamma \nabla_\theta \mathcal{L}$  ▷ Optimization step
- 16: **end while**
- 17: **return**  $\theta$

---

---

**Algorithm 4** DTM Sampling

---

**Require:**  $\theta$  ▷ Trained model  
**Require:**  $T$  ▷ Number of TM steps

- 1: Sample  $X_0 \sim \mathcal{N}(0, I_d)$
- 2: **for**  $t = 0$  **to**  $T - 1$  **do**
- 3:    $h_t \leftarrow f_t^\theta(X_t, t)$
- 4:   **parallel for**  $i = 1, \dots, n$  **do**
- 5:     Sample  $Y_0^i \sim N(0, I_{d/n})$
- 6:      $Y^i \leftarrow \text{ode\_solve}(Y_0^i, g_{\cdot, t}^\theta(\cdot, h_t^i))$
- 7:   **end for**
- 8:    $X_{t+1} \leftarrow X_t + \frac{1}{T} Y$
- 9: **end for**
- 10: **return**  $X_T$

---

Figure 17:  $n$  is the effective sequence length after patchify layer. The *parallel for* operations run simultaneously across the "sequence length" dimension of the tensor; `ode_solve` is any generic ODE solver for solving equation 8.

---

**Algorithm 5** ARTM Training

---

**Require:**  $p_T$   $\triangleright$  Data**Require:**  $T$   $\triangleright$  Number of TM steps

```
1: while not converged do
2:   Sample  $X_T \sim p_T$ 
3:   Sample  $t \sim \mathcal{U}([T-1])$ 

4:   Sample  $X_{0,t} \sim N(0, I_d)$ 
5:    $X_t \leftarrow (1 - \frac{t}{T}) X_{0,t} + \frac{t}{T} X_T$ 
6:   Sample  $X_{0,t+1} \sim N(0, I_d)$ 
7:    $X_{t+1} \leftarrow (1 - \frac{t+1}{T}) X_{0,t+1} + \frac{t+1}{T} X_T$ 
   } Sample  $(X_t, Y) \sim q_{t,Y|T}(\cdot|X_T)$ 

8:   parallel for  $i = 1, \dots, n$  do
9:      $h_{t+1}^i \leftarrow f_t^\theta(X_t, X_{t+1}^{<i})$ 
10:    Sample  $Y_0^i \sim N(0, I_{d/n})$ 
11:    Sample  $s \sim \mathcal{U}([0, 1])$ 
12:     $Y_s^i \leftarrow (1-s) Y_0^i + s X_{t+1}^i$ 
13:     $\mathcal{L}^i(\theta) \leftarrow \|g_{s,t}^\theta(Y_s^i, h_{t+1}^i) - (X_{t+1}^i - Y_0^i)\|^2$ 
14:  end for
15:   $\mathcal{L}(\theta) \leftarrow \frac{1}{n} \sum_i \mathcal{L}^i(\theta)$ 
   }  $\mathcal{L}(\theta) \leftarrow \hat{D}(Y, p_{Y|t}^\theta(\cdot|X_t))$ 

16:   $\theta \leftarrow \theta - \gamma \nabla_\theta \mathcal{L}$   $\triangleright$  Optimization step
17: end while
18: return  $\theta$ 
```

---

---

**Algorithm 6** ARTM Sampling

---

**Require:**  $\theta$   $\triangleright$  Trained model**Require:**  $T$   $\triangleright$  Number of TM steps

```
1: Sample  $X_0 \sim \mathcal{N}(0, I_d)$ 
2: for  $t = 0$  to  $T-1$  do
3:   for  $i = 1, \dots, n$  do
4:      $h_{t+1}^i \leftarrow f_t^\theta(X_t, X_{t+1}^{<i})$ 
5:     Sample  $Y_0^i \sim N(0, I_{d/n})$ 
6:      $X_{t+1}^i \leftarrow \text{ode\_solve}(Y_0^i, g_{\cdot,t}^\theta(\cdot, h_{t+1}^i))$ 
7:   end for
8: end for
9: return  $X_T$ 
   } Sample  $X_{t+1} \sim p_{t+1|t}^\theta(\cdot|X_t)$ 
```

---

Figure 18:  $n$  is the effective sequence length after patchify layer. The *parallel for* operations run simultaneously across the "sequence length" dimension of the tensor; `ode_solve` is any generic ODE solver for solving equation 8.

---

**Algorithm 7** FHTM Training

---

**Require:**  $p_T$   $\triangleright$  Data  
**Require:**  $T$   $\triangleright$  Number of TM steps

- 1: **while** not converged **do**
- 2:   Sample  $X_T \sim p_T$
- 3:   **parallel for**  $t = 0, \dots, T$  **do**
- 4:     Sample  $X_{0,t} \sim N(0, I_d)$
- 5:      $X_t \leftarrow (1 - \frac{t}{T}) X_{0,t} + \frac{t}{T} X_T$
- 6:   **end for**
- 7:   **parallel for**  $t = 0, \dots, T-1, i = 1, \dots, n$  **do**
- 8:      $h_{t+1}^i \leftarrow f_t^\theta(X_0, \dots, X_t, X_{t+1}^{<i})$
- 9:     Sample  $Y_0^i \sim N(0, I_{d/n})$
- 10:     Sample  $s \sim \mathcal{U}([0, 1])$
- 11:      $Y_s^i \leftarrow (1-s) Y_0^i + s X_{t+1}^i$
- 12:      $\mathcal{L}_t^i(\theta) \leftarrow \|g_{s,t}^\theta(Y_s^i, h_{t+1}^i) - (X_{t+1}^i - Y_0^i)\|^2$
- 13:   **end for**
- 14:    $\mathcal{L}(\theta) \leftarrow \frac{1}{nT} \sum_{i,t} \mathcal{L}_t^i(\theta)$
- 15:    $\theta \leftarrow \theta - \gamma \nabla_\theta \mathcal{L}$   $\triangleright$  Optimization step
- 16: **end while**
- 17: **return**  $\theta$

---

---

**Algorithm 8** FHTM Sampling

---

**Require:**  $\theta$   $\triangleright$  Trained model  
**Require:**  $T$   $\triangleright$  Number of TM steps

- 1: Sample  $X_0 \sim \mathcal{N}(0, I_d)$
- 2: **for**  $t = 0$  **to**  $T-1$  **do**
- 3:   **for**  $i = 1, \dots, n$  **do**
- 4:      $h_{t+1}^i \leftarrow f_t^\theta(X_0, \dots, X_t, X_{t+1}^{<i})$
- 5:     Sample  $Y_0^i \sim N(0, I_{d/n})$
- 6:      $X_{t+1}^i \leftarrow \text{ode\_solve}(Y_0^i, g_{\cdot,t}^\theta(\cdot, h_{t+1}^i))$
- 7:   **end for**
- 8: **end for**
- 9: **return**  $X_T$

---

Figure 19:  $n$  is the effective sequence length after patchify layer. The *parallel for* operations run simultaneously across the "sequence length" dimension of the tensor; `ode_solve` is any generic ODE solver for solving equation 8.



```

1 import torch
2 from torch import nn, Tensor
3 from einops import rearrange
4
5 def dtm_train_step(
6     backbone:nn.Module, # Denoted as  $f^{\theta}$ 
7     head:nn.Module,     # Denoted as  $g^{\theta}$ 
8     X_T:Tensor,          # Image from training set  $X_T \sim p_T$ 
9     T:int,               # Number of TM steps
10    patch_size:int        # Patch size
11 ) -> Tensor:
12     # Convert image to sequence using patchify
13     X_T = rearrange(
14         X_T,
15         "b c (h dh) (w dw) -> b (h w) (dh dw c)",
16         dh=patch_size,
17         dw=patch_size,
18     )
19     bsz, seq_len = X_T.shape[:2]
20
21     # Sample time step  $t \sim U[T-1]$ 
22     t = torch.randint(0, T, (bsz,))
23
24     # Sample a pair  $(X_t, Y) \sim q_{\{t, Y|T\}}(.|X_T)$ 
25     X_0 = torch.rand_like(X_T)
26     X_t = (1-t/T).view(-1,1,1) * X_0 + (t/T).view(-1,1,1) * X_T
27     Y = X_T - X_0
28
29     # Backbone forward
30     h_t = backbone(X_t, t)
31
32     # Reshape sequence for head
33     h_t = h_t.view(bsz*seq_len, -1)
34     Y = Y.view(bsz*seq_len, -1)
35     t = t.repeat_interleave(seq_len)
36
37     # Flow matching loss with the head as velocity and Y as target
38     Y_0 = torch.rand_like(Y)
39     s = torch.rand(bsz*seq_len)
40     Y_s = (1-s).view(-1,1) * Y_0 + s.view(-1,1) * Y
41
42     # Head forward
43     u = head(h_t, t, Y_s, s)
44     loss = torch.nn.functional.mse_loss(u, Y - Y_0)
45
46     return loss

```

Figure 20: Python code for DTM training

```

1 import torch
2 from torch import nn, Tensor
3
4 def artm_train_step(
5     backbone:nn.Module, # Denoted as  $f^{\theta}$ 
6     head:nn.Module,     # Denoted as  $g^{\theta}$ 
7     X_T:Tensor,          # Image from training set  $X_T^{p_T}$ 
8     T:int                # Number of TM steps
9     patch_size:int       # Patch size
10 ) -> Tensor:
11     # Convert image to sequence using patchify
12     X_T = rearrange(
13         X_T,
14         "b c (h dh) (w dw) -> b (h w) (dh dw c)",
15         dh=patch_size,
16         dw=patch_size,
17     )
18     bsz, seq_len = X_T.shape[:2]
19
20     # Sample time step  $t \sim U[T-1]$ 
21     t = torch.randint(0, T, (bsz,))
22
23     # Sample a pair  $(X_t, Y) \sim q_{\{t, Y|T\}}(.|X_T)$ 
24     X_0_t = torch.rand_like(X_T)
25     X_t = (1-t/T).view(-1,1,1) * X_0_t + (t/T).view(-1,1,1) * X_T
26     X_0_tp1 = torch.rand_like(X_T)
27     Y = (1-(t+1)/T).view(-1,1,1) * X_0_tp1 + ((t+1)/T).view(-1,1,1) * X_T
28
29     # Backbone forward
30     output = backbone(torch.cat([X_t, Y], dim=1), t)
31     h_tp1 = output[:, seq_len-1:-1]
32
33     # Reshape sequence for head
34     h_tp1 = h_tp1.view(bsz*seq_len, -1)
35     Y = Y.view(bsz*seq_len, -1)
36     t = t.repeat_interleave(seq_len)
37
38     # Flow matching loss with the head as velocity and Y as target
39     Y_0 = torch.rand_like(Y)
40     s = torch.rand(bsz*n_tokens)
41     Y_s = (1-s).view(-1,1) * Y_0 + s.view(-1,1) * Y
42
43     # Head forward
44     u = head(h_tp1, t, Y_s, s)
45     loss = torch.nn.functional.mse_loss(u, Y - Y_0)
46
47     return loss

```

Figure 21: Python code for ARTM training

```

1 import torch
2 from torch import nn, Tensor
3
4 def fhtm_train_step(
5     backbone:nn.Module, # Denoted as  $f^{\theta}$ 
6     head:nn.Module,     # Denoted as  $g^{\theta}$ 
7     X_T:Tensor,          # Image from training set  $X_T \sim p_T$ 
8     T:int                # Number of TM steps
9     patch_size:int       # Patch size
10 ) -> Tensor:
11     # Convert image to sequence using patchify
12     X_T = rearrange(
13         X_T,
14         "b c (h dh) (w dw) -> b (h w) (dh dw c)",
15         dh=patch_size,
16         dw=patch_size,
17     )
18
19     bsz, seq_len, d = X_T.shape
20
21     # Sample a pair  $(X_t, Y) \sim q_{\{t, Y|T\}}(.|X_T)$ 
22     boi = torch.zeros(bsz,1,d) # begin of image token
23     X_FH = [boi]
24     for t in range(1,T+1):
25         X_0_t = torch.rand_like(X_T)
26         X_FH.append(
27             (1-t/T) * X_0_t + t/T * X_T
28         )
29     X_FH = torch.cat(X_FH, dim=1)
30     X_t = X_FH[:, :-1]
31     Y = X_FH[:, 1:]
32
33     # forward for teacher forcing
34     h_tp1 = backbone(X_t)
35
36     # Reshape sequence for head
37     h_tp1 = h_tp1.view(bsz*seq_len*T, -1)
38     Y = Y.view(bsz*seq_len*T, -1)
39
40     # Flow matching loss with the head as velocity and Y as target
41     Y_0 = torch.rand_like(Y)
42     s = torch.rand(bsz*seq_len*T)
43     Y_s = (1-s).view(-1,1) * Y_0 + s.view(-1,1) * Y
44
45     # Head forward
46     u = head(h_tp1, Y_s, s)
47     loss = torch.nn.functional.mse_loss(u, Y - Y_0)
48
49     return loss

```

Figure 22: Python code for FHTM training