🍏PoM: A Linear-Time Replacement for Attention with the Polynomial Mixer

Supplementary Material

In this appendix, we report additional ablations (Sec. A) and analysis (Sec. B), and a proof of our main theoretical results (Sec. C). We also propose an extended related work in depth (Sec. D) and additional illustrations for the image generation experiment (Sec. E.)

A. Additional Ablations.

Higher Orders k. We study the effect of increasing the polynomial degree k in Tab. 1. Higher-order polynomials provide a modest improvement, at the cost of a slight speed reduction. The limited impact is expected, as the sequences evaluated in classic benchmarks are relatively short and do not fully stress the model's representational capacity. As suggested by our theoretical analysis, higher-order interactions could become more critical for distinguishing longer sequences.

Additional Baselines. We evaluate competing linear attention models and report their performance in Tab. 1.

- *Performer.* We attempted to train a GPT2-S model using Performer; however, both `pytorch-performer` and the Performer implementation from `torch_geometric` systematically diverged, producing NaN losses after a few iterations. At present, we are not aware of a well-maintained Performer implementation suitable for large-scale language model training, which prevented a fair comparison.
- *Mamba.* We trained a Mamba model of comparable size (124M parameters) on the NLP task; results are reported in Tab. 1. As expected, Mamba achieves excellent speed due to its $\mathcal{O}(1)$ complexity and dedicated CUDA kernel. PoM shares the same $\mathcal{O}(1)$ theoretical complexity, and we therefore expect that a dedicated CUDA implementation would yield comparable speedups. In terms of accuracy, however, **Mamba performs below MHA and PoM** on the evaluated benchmarks. We used the same classic training recipe inherited from attention-based models for all methods, which suggests that Mamba may require more tuning adaptation than PoM. We also observed that **Mamba consumed approximately $4\times$ more memory** during training, forcing smaller batch sizes and gradient accumulation, whereas PoM and MHA have a similar memory footprint.

B. Additional Analysis.

Comparison to FlashAttention. FlashAttention is a highly engineered implementation of attention, relying on custom CUDA kernels and carefully optimized memory management. In contrast, PoM is currently implemented entirely in high-level PyTorch, and yet already achieves consistent 2–4$\times$ speedups across tasks, and in some settings (notably NLP, see Tab. 1, and image generation), PoM is faster than Flash-Attention. We therefore view these results as a conservative estimate of PoM's speed, and expect further gains with dedicated low-level optimization.

Table 1. **Additional ablations and baselines.** MHA throughput uses Flash-Attention, Performer uses `torch_geometric`.

	ARC-easy Acc.Norm↑	HellaSwag Acc.Norm↑	Winogrande Acc.↑	Throughput 4096 tok., tok/s↑
PoM-hyb k=2	29.0	33.8	**51.9**	163
PoM-hyb k=3	**29.5**	33.4	49.6	161
PoM-hyb k=5	28.6	**33.9**	49.5	153
MHA	29.4	33.3	49.4	42
Performer		training failed		59
Mamba	29.3	30.8	48.0	**384**

Generalization Under Train–Test Length Mismatch. Since $H(x)$ is obtained by pooling over tokens, its dimensionality is independent of the sequence length. In our NLP experiments, PoM is trained with a fixed sequence length of 2048 tokens (with padding), while evaluation on HellaSwag involves sequences ranging from 48 to 742 tokens, with no observable impact on performance.

C. Proof of Lemma 3

We first need to show that set with different entries are mapped to different vectors. We first separate PoM into its two components:

$$s(X) = \sigma(W_s X) \tag{1}$$

$$H(X) = \left[\sum_{p=1}^{k} \boldsymbol{\alpha}_p \odot h(W_h X)^p\right] \mathbf{1} \tag{2}$$

$$\text{PoM}(X) = W_o \left[\sigma(W_s X) \odot H(X)\right] \tag{3}$$

Assuming $\ker(W_o) = \emptyset$, and noting that $H_k(X)$ is the same for every column, we just have to show that $s(X)$ has different columns. This is easily achieved by having $\ker(W_s) = \emptyset$ since σ is injective and the composition of injective functions is itself injective.

Second, we have to show that sets that differ by at least one element are mapped to all different entries. To simplify notations, we will consider the special case where all matrices are the identity or an identity block positioned such as to perform submatrix selection. All the matrices can thus be removed from the formula. A similar argument can be

made for matrices that are full rank as they preserve injectivity. We will also consider linear activations everywhere, which can be made as close as one wish by partitioning the image of the activation function and performing piecewise linear approximation.

With this simplified version of PoM, we have to show that for 2 sets X, X' differing by at least one element (i.e., $\exists x' \in X', \forall x \in X, x \neq x'$), then there exist k such that

$$\forall x \in X, x' \in X', x \sum_{x_i \in X} x_i^k \neq x \sum_{x_i \in X} x_i^k. \tag{4}$$

Consider the functions $P(t)$ and $P'(t)$ defined as follows:

$$P(t) = \sum_{x_i in X} x_i^t \tag{5}$$

$$P'(t) = \sum_{x_i in X'} x_i^t \tag{6}$$

Since X and X' differ by at least one element, there exists at least one $x_i \in X$ such that $x_i \neq x'_i, \forall x'_i \in X'$. This implies that the functions $P(t)$ and $P'(t)$ are not identical since are sums of exponentials with different bases.

Since $P(t)$ and $P'(t)$ are different functions, there must exist some k for which $P(k) \neq P'(k)$. In other words, there exists a k such that:

$$\sum_{x_i \in X} x_i^k \neq \sum_{x'_i \in X'} x_i'^k \tag{7}$$

For this k, let us denote $S_k = \sum_{x_i \in X} x_i^k$. We need to show that $xS_k \neq x'S'_k$ for all $x \in X$ and $x' \in X'$. Assume for the sake of contradiction that there exist $x \in X$ and $x' \in X'$ such that $xS_k = x'S'_k$. This implies:

$$x \sum_{x_i \in X} x_i^k = x' \sum_{x'_i \in X'} x_i'^k \tag{8}$$

Rearranging, we get:

$$\frac{x}{x'} = \frac{\sum_{x'_i \in X'} x_i'^k}{\sum_{x_i \in X} x_i^k} \tag{9}$$

Since $S_k \neq S'_k$, the right-hand side is not equal to 1. However, for this equality to hold for all $x \in X$ and $x' \in X'$, the ratio x/x' would need to be constant for all pairs (x, x'), which is not possible given that X and X' differ by at least one element.

Therefore, there exists a k such that $xS_k \neq x'S'_k$ for all $x \in X$ and $x' \in X'$.

D. Related work on Diffusion

Diffusion Diffusion models [13, 24, 29] learn a neural operator that produces natural images from noise using a forward-reverse set of processes. The forward process consists in pushing the distribution of natural images forward to a known distribution, typically Gaussian, which can be done by adding increasing level of noise to the image. The reverse process does not have an explicit solution, but can be approximated by a neural network by regressing the local inverse of the forward process, *i.e.*, solving

$$\min_\theta \mathbb{E}_{t \sim \mathcal{U}(0,1)} \left[\|\varepsilon_t - f_\theta(x_t, t)\|^2 \right], \tag{10}$$

$$\text{s.t. } x_t = \alpha_t x_0 + \gamma_t \varepsilon_t,\ \varepsilon_t \sim \mathcal{N}(0, 1). \tag{11}$$

Here, α_t and γ_t are chosen such that x_0 corresponds to a natural image whereas x_1 corresponds to pure Gaussian noise. A great amount of research has been put into finding better noise schedules (α_t and γ_t) [2, 11, 18], or improving the quantity that is regressed [20, 21, 27], keeping the general idea of learning to invert step by step the stochastic differential equation that transforms an image into noise.

For image and generation, most efforts have been poured into designing efficient architectures at the task. While the original DDPM papers [13, 24] sample images in pixel space, making it unsuitable for large resolution, the most groundbreaking improvement was introduced by Stable Diffusion [26] with the addition of a variational auto-encoder (VAE) that allows the diffusion process to be performed in a lower dimensional latent space. Stable Diffusion uses a U-Net architecture complemented by attention layers [26, 28]. To benefit more from the scaling properties of transformers [16, 32], simpler approaches based solely on transformer layers has been proposed in DiT [25] and the subsequent flow-matching version SiT [23]. Most modern text-to-image generation models are now based on Transformer layers rather than the U-Net [3, 7, 8, 12]. [5, 10], train efficient pixel space transformers models by leveraging multiscale training and SwinAttention. Similarly, RIN [4, 14] also proposes an approach using attention only, albeit in a Perceiver-IO [15] inspired architecture that uses cross-attention to perform most of the computation in a smaller latent space, and has been successfully extended to text-to-image [6]. In addition to architectures and sampling [1, 33, 34], the importance of training is also highlighted in recent works, from resampling the training data [9, 22] to RL [19, 30, 31] and model averaging [17].

E. Uncurated Image Samples

To show 🍏PoM versatility, with train a DiPoM-XL/2 with the diffusion loss instead of the flow-matching loss and show generated samples with CFG $\omega = 6$ in the following pages.

References

[1] Xingjian Bai and Luke Melas-Kyriazi. Fixed point diffusion models. In *CVPR*, 2024.

[2] Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, Tero Karras, and Ming-Yu Liu. eDiff-I: Text-to-image diffusion models with ensemble of expert denoisers. *arXiv:2211.01324*, 2022.

[3] Junsong Chen, Chongjian Ge, Enze Xie, Yue Wu, Lewei Yao, Xiaozhe Ren, Zhongdao Wang, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-\sigma: Weak-to-strong training of diffusion transformer for 4k text-to-image generation. In *ECCV*, 2024.

[4] Ting Chen and Lala Li. FIT: Far-reaching interleaved transformers. *arXiv:2305.12689*, 2023.

[5] Katherine Crowson, Stefan Andreas Baumann, Alex Birch, Tanishq Mathew Abraham, Daniel Z Kaplan, and Enrico Shippole. Scalable high-resolution pixel-space image synthesis with hourglass diffusion transformers. In *Int. Conf. Mach. Learn.*, 2024.

[6] Nicolas Dufour, Victor Besnier, Vicky Kalogeiton, and David Picard. Don't drop your samples! Coherence-aware training benefits conditional diffusion. In *CVPR*, 2024.

[7] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Int. Conf. Mach. Learn.*, 2024.

[8] Peng Gao, Le Zhuo, Ziyi Lin, Chris Liu, Junsong Chen, Ruoyi Du, Enze Xie, Xu Luo, Longtian Qiu, Yuhang Zhang, et al. Lumina-T2X: Transforming text into any modality, resolution, and duration via flow-based large diffusion transformers. *arXiv:2405.05945*, 2024.

[9] Aaron Gokaslan, A. Feder Cooper, Jasmine Collins, Landan Seguin, Austin Jacobson, Mihir Patel, Jonathan Frankle, Cory Stephenson, and Volodymyr Kuleshov. CommonCanvas: Open diffusion models trained on creative-commons images. In *CVPR*, 2024.

[10] Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Joshua M Susskind, and Navdeep Jaitly. Matryoshka diffusion models. In *ICLR*, 2023.

[11] Tiankai Hang and Shuyang Gu. Improved noise schedule for diffusion training. *ICCV*, 2024.

[12] Ali Hatamizadeh, Jiaming Song, Guilin Liu, Jan Kautz, and Arash Vahdat. DifFit: Diffusion vision transformers for image generation. In *ECCV*, 2024.

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[14] Allan Jabri, David J Fleet, and Ting Chen. Scalable adaptive computation for iterative generation. In *Int. Conf. Mach. Learn.*, 2023.

[15] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver IO: A general architecture for structured inputs & outputs. In *ICLR*, 2022.

[16] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[17] Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *CVPR*, 2024.

[18] Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *CVPR*, 2024.

[19] Seung Hyun Lee, Yinxiao Li, Junjie Ke, Innfarn Yoo, Han Zhang, Jiahui Yu, Qifei Wang, Fei Deng, Glenn Entis, Junfeng He, et al. Parrot: Pareto-optimal multi-reward reinforcement learning framework for text-to-image generation. In *ECCV*, 2025.

[20] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*, 2022.

[21] Xingchao Liu, Chengyue Gong, et al. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*, 2023.

[22] Yujian Liu, Yang Zhang, Tommi Jaakkola, and Shiyu Chang. Correcting diffusion generation through resampling. In *CVPR*, 2024.

[23] Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. SIT: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *ECCV*, 2024.

[24] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *Int. Conf. Mach. Learn.*, 2021.

[25] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

[26] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[27] Yuyang Shi, Valentin De Bortoli, Andrew Campbell, and Arnaud Doucet. Diffusion Schrödinger bridge matching. In *NeurIPS*, 2024.

[28] Chenyang Si, Ziqi Huang, Yuming Jiang, and Ziwei Liu. FreeU: Free lunch in diffusion U-net. In *CVPR*, 2024.

[29] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021.

[30] Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *CVPR*, 2024.

[31] Fanyue Wei, Wei Zeng, Zhenyang Li, Dawei Yin, Lixin Duan, and Wen Li. Powerful and flexible: Personalized text-to-image generation via reinforcement learning. In *ECCV*, 2024.

[32] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *CVPR*, 2022.

[33] Yang Zhao, Yanwu Xu, Zhisheng Xiao, Haolin Jia, and Tingbo Hou. MobileDiffusion: Instant text-to-image generation on mobile devices. In *ECCV*, 2024.

[34] Zhenyu Zhou, Defang Chen, Can Wang, and Chun Chen. Fast ode-based sampling for diffusion models in around 5 steps. In *CVPR*, 2024.

Figure 1. Uncurated 256^2 images for the class *loggerhead, loggerhead turtle, Caretta caretta* (33).

Figure 2. Uncurated 256^2 images for the class *macaw* (88).

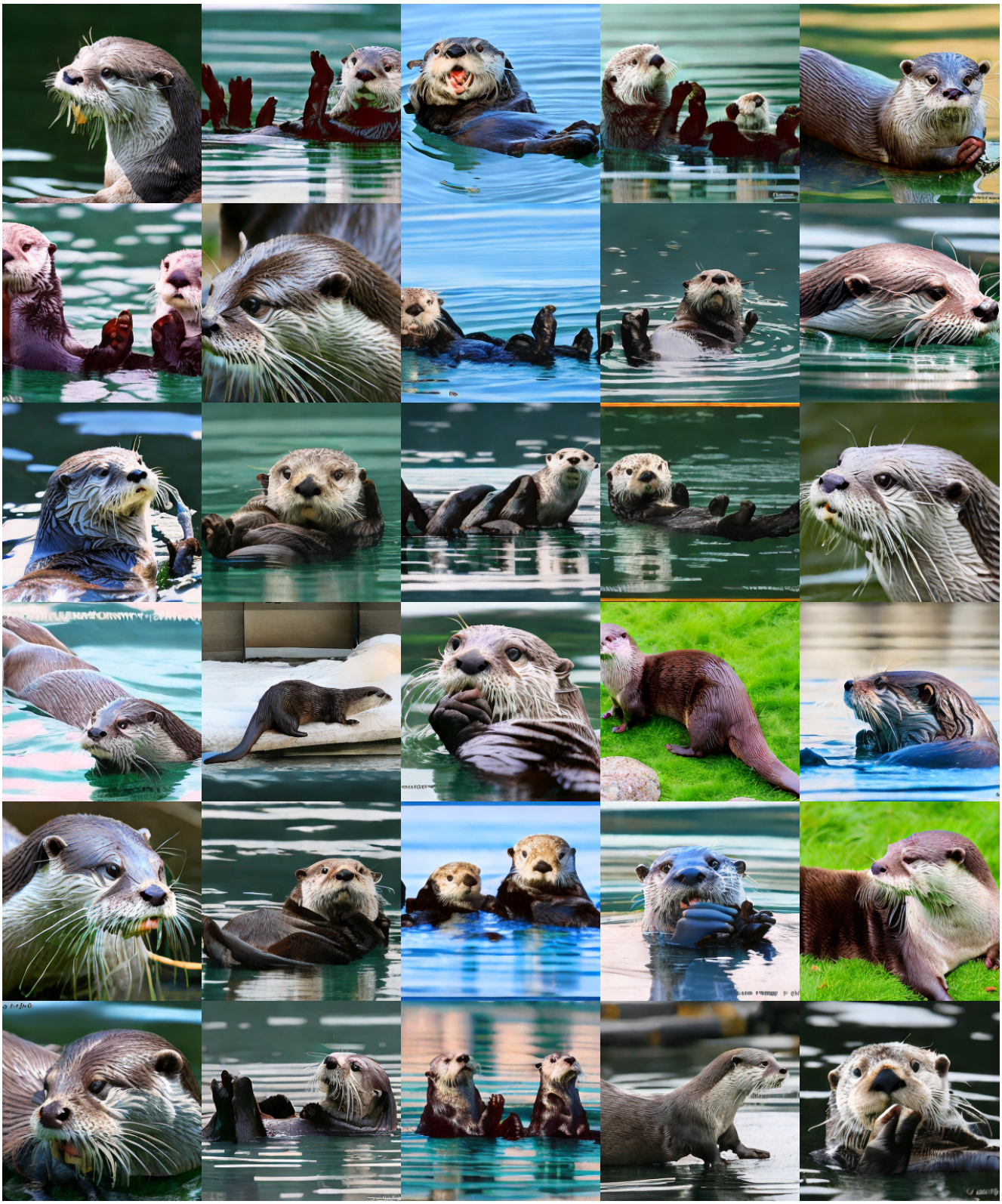

Figure 3. Uncurated 256² images for the class *otter* (360).

Figure 4. Uncurated 256² images for the class *volcano* (980).