LiT: Delving into a Simple Linear Diffusion Transformer for Image Generation

Supplementary Material

A. Model Configuration

We report the configurations of LiT variants in Tab. 1, which basically follow the hyper-parameters of DiT [23], except for using few heads. For class-conditional image generation, we set 2/3/4/4 heads for linear attention used in LiT. For LiT-XL/2, used for text-to-image tasks, we use 2 heads for linear attention.

Model	Layers	Hidden size	Heads	Patch size
LiT-S	12	384	2	2
LiT-B	12	768	3	2
LiT-L	24	1024	4	2
LiT-XL	28	1152	4	2
LiT-XL$^\diamond$	28	1152	2	2

Table 1. **Configuraions of LiT** for class-conditional image generation and text-to-image generation (denoted by $^\diamond$). Apart from using few heads, we generally follow the DiT [23] setting.

B. Latency Analysis in Diffusion Transformer

We conduct a component-wise latency analysis of the Diffusion Transformer, with results shown in Fig. 1. The analysis was performed using the DiT-B/4 [23] model on an NVIDIA A100 GPU. The results indicate that the self-attention module accounts for 42.6% of the total latency of a DiT block. We attribute the observed considerable latency proportion to the quadratic computational complexity of the self-attention.

C. Detailed Latency and Theoretical GMACs

We use one NVIDIA V100 GPU to evaluate the latency and theoretical GMACs of the DiT-S/B models with different numbers of heads. The task was to generate 256×256 resolution images with a batch size of 8, following the experimental setup of Fig. 5 in the main paper (except for the GPU type), and the results are shown in Fig. 3.

We observe that both the small and base models on the V100 GPU exhibit a phenomenon similar to that on the A100 GPU: as theoretical GMACs increased, practical latency does not follow the same trend and even descend (in the B/2 model). This finding supports the generalizability of the free lunch effect of linear attention.

D. Detailed Results on Attention Head Similarities

We take LiT-S/2 with 6 heads for visualization. Average cosine similarity among the attention maps of different heads is illustrated in Fig. 4.

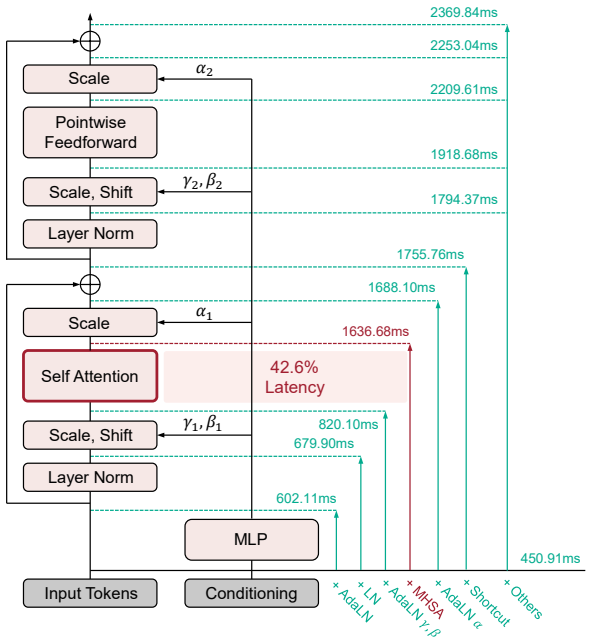

Figure 1. **Latency analysis of different components** in DiT-B/4 [23] with a batch size of 8 using NVIDIA A100 GPU. Latency of the vanilla self-attention occupies about **42.6%** of the backbone. Our LiT successfully replaces the heavy attention module with simple linear attention, by using the proposed architectural design and training guidelines.

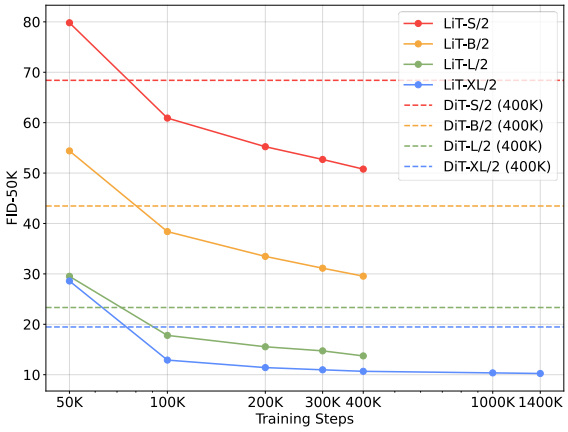

Figure 2. **Comparing training efficiency between our LiT and DiT**. LiT outperforms DiT (400K training steps) with only 100K training steps for different model sizes.

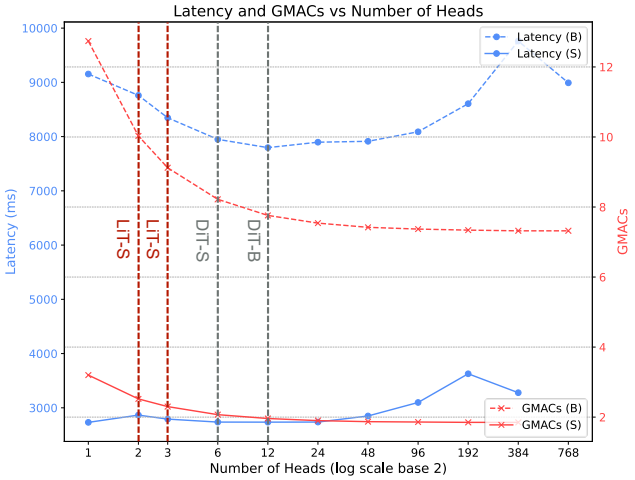

Figure 3. **Free lunch in linear attention.** Comparison of latency and theoretical GMACs for linear attention with different number of heads. We test the latency to generate 256×256 resolution images using one NVIDIA V100 GPU with a batch size of 8. Results of S/2 and B/2 model were averaged over 30 times. Results for the case of the V100 GPU demonstrate a similar phenomenon to the A100 GPU.

Figure 4. **Redundancy in linear attention heads.** Attention maps of different heads of LiT-S/2 (6 heads) show high average cosine similarity.

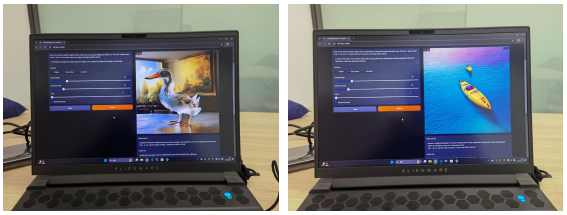

Figure 5. **Offline deployment of on a Windows 11 laptop.** LiT runs swiftly on the edge, generating 1K resolution images.

E. Detailed Results on Class-Conditional Image Generation

Detailed results on Sec. 4. We provide detailed results for Tab. 1, Tab. 2, Tab. 3 and Tab. 4 in the main text in Tab. 2, Tab. 3, Tab. 4, and Tab. 5, respectively. In each table, we report results involving FID-50K [13] (*without* classifier-free guidance), Inception Score (IS) [32] and Precision/Recall [17]. IS, and Precision/Recall show results similar to FID-50K. As a result, the conclusions drawn in Sec. 4 of the main paper apply not only to metrics evaluating the distance between generated images and real images (*e.g.*, FID-50K) but also to metrics reflecting the quality of the generated images themselves (*e.g.*, IS).

F. More Results on Text-to-Image Generation

We provide more text-to-image results in Fig. 7 and Fig. 8. As shown, LiT can accurately generate 512px photo-realistic images in various styles, themes, and content, whether the human instructions are simple or complicated. These results demonstrate that LiT effectively learns useful knowledge from the teacher model while maintaining exceptional computational efficiency, validating the effectiveness of our proposed cost-effective training strategy. Results of the offline laptop development are shown in Fig. 5.

G. Pseudo-code

As presented in Alg. 1, we provide an example of the pseudo-code for the linear attention used in our LiT model for text-to-image generation for the laptop development. We set the kernel size of the depthwise convolution to 5.

H. Full Related Work

Linear attention. As a computationally efficient alternative to self-attention, linear attention [16] reduces computational complexity from quadratic to linear and has been proven effective in both visual understanding domain [1, 2, 9–11] and language domain [4, 27–29, 36]. EfficientViT [2] introduces a multi-scale linear attention with hardware-efficient operations to obtain a general vision backbone. Flatten Transformer [10] introduces focused linear attention to address the deficiencies in focus ability and feature diversity of linear attention, incorporating a focused function and depthwise convolution (DWC). SLAB [9] simplifies focused linear attention by retaining only the DWC component and introduces a progressive re-parameterized batch normalization to adapt offline batch normalization [15] for achieving low inference latency. These studies have been validated on visual perception tasks. Meanwhile, our work refines a linear attention module tailored for image generation tasks and identifies the free lunch of using few heads.

Algorithm 1 Linear Attention in LiT, Pseudo-code

```
import torch
import torch.nn as nn

class LinearAttention(nn.Module):
   def __init__(
      self,
      dim,
      num_heads=8,
      qkv_bias=False,
      attn_drop=0.,
      proj_drop=0.,
      kernel_function=nn.ReLU,
      kernel_size=5,
      fp32_attention=True,
      **block_kwargs,
   ):
      super().__init__()
      assert dim % num_heads == 0, f"dim␣{dim}␣should␣be␣divisible␣by␣num_heads␣{num_heads}."

      self.dim = dim
      self.num_heads = num_heads
      head_dim = dim // num_heads

      self.q = nn.Linear(dim, dim, bias=qkv_bias)
      self.kv = nn.Linear(dim, dim * 2, bias=qkv_bias)
      self.attn_drop = nn.Dropout(attn_drop)
      self.proj = nn.Linear(dim, dim)
      self.proj_drop = nn.Dropout(proj_drop)

      self.dwc = nn.Conv2d(in_channels=head_dim, out_channels=head_dim, kernel_size=kernel_size,
                           groups=head_dim, padding=kernel_size // 2)
      self.kernel_function = kernel_function()
      self.fp32_attention = fp32_attention

   def forward(self, x, HW=None):
      B, N, C = x.shape
      new_N = N
      if HW is None:
         H = W = int(N ** 0.5)
      else:
         H, W = HW

      q = self.q(x)
      dtype = q.dtype

      kv = self.kv(x).reshape(B, N, 2, C).permute(2, 0, 1, 3)
      k, v = kv[0], kv[1]

      q = self.kernel_function(q) + 1e-6
      k = self.kernel_function(k) + 1e-6

      q = q.reshape(B, N, self.num_heads, -1).permute(0, 2, 1, 3).to(dtype)
      k = k.reshape(B, N, self.num_heads, -1).permute(0, 2, 1, 3).to(dtype)
      v = v.reshape(B, N, self.num_heads, -1).permute(0, 2, 1, 3).to(dtype)

      use_fp32_attention = getattr(self, 'fp32_attention', False)
      if use_fp32_attention:
         q, k, v = q.float(), k.float(), v.float()

      with torch.cuda.amp.autocast(enabled=not use_fp32_attention):
         z = 1 / (q @ k.mean(dim=-2, keepdim=True).transpose(-2, -1) + 1e-6)
         kv = (k.transpose(-2, -1) * (N ** -0.5)) @ (v * (N ** -0.5))
         x = q @ kv * z

         x = x.transpose(1, 2).reshape(B, N, C)
         v = v.reshape(B * self.num_heads, H, W, -1).permute(0, 3, 1, 2)
         x = x + self.dwc(v).reshape(B, C, N).permute(0, 2, 1)

      x = x.type(torch.float16)

      x = self.proj(x)
      x = self.proj_drop(x)

      return x
```

Efficient diffusion Transformer for image generation. Limited by the quadratic computational complexity of self-attention, recent studies focus on developing efficient Transformer-style architectures for diffusion models. For example, DiM [33], ZigMa [14], and DiMSUM [25] explore implementing Mamba-based [6, 8] DiT-style [23]

models. Diffusion-RMKV [7] studies RWKV-style [24] models for diffusion. Mediator [26] introduces an attention mediator to obtain an efficient diffusion Transformer with linear complexity. DiG [41] replaces the self-attention in DiT with gated linear attention to speed up training. LinFusion [21] and Sana [35] apply linear attention to U-Net-based [30] and Transformer-based [34] diffusion models, respectively, and train these models from scratch to generate high-quality images based on user instructions. Other studies [19, 20] explore efficient diffusion models through perspectives of low-bit quantization [12], feature map reusing [18, 22], and lightweight architecture design [40]. Differently, our work not only refines the design of linear attention but also introduces cost-effective training strategies, including weight inheritance and a novel hybrid diffusion distillation approach.

Advanced training method for diffusion models. Some studies explores improved training strategies to enhance the optimization of diffusion models or achieve more efficient models. For example, CAN [3] introduces a condition-aware weight generation module to the diffusion Transformer, and demonstrate the technique can be further equipped with EffcientViT [2] to achieve both effectiveness and efficiency. REPA [39] proposes aligning the intermediate features of the denoising model with those extracted by a pre-trained visual encoder during the training of diffusion models. Some studies [31, 37, 38] explore distillation techniques to reduce the sampling steps of the diffusion model. Unlike the goal of reducing sampling steps, our proposed hybrid knowledge distillation focuses on an architectural perspective, exploring how a lightweight student model can learn from a computationally intensive teacher model.

DiT	Attention	Resolution	Batch Size	Training Steps	FID-50K (↓)	IS (↑)	Precision (↑)	Recall (↑)
S/2	softmax	256	256	400K	68.40	-		
S/2	ReLU linear	256	256	400K	88.46	15.11	0.29	0.45
S/2	Simplified linear (ReLU)	256	256	400K	63.66	22.16	0.38	0.58
S/2	focused linear (ReLU)	256	256	400K	**63.05**	**22.49**	**0.39**	**0.58**
S/2	focused linear (GELU)	256	256	400K	70.83	19.41	0.36	0.54
B/2	softmax	256	256	400K	43.47	-		
B/2	ReLU linear	256	256	400K	56.92	25.80	0.42	0.59
B/2	Simplified linear (ReLU)	256	256	400K	42.11	34.60	0.48	0.63
B/2	focused linear (ReLU)	256	256	400K	**40.58**	**35.98**	**0.50**	**0.63**
B/2	focused linear (GELU)	256	256	400K	58.86	24.23	0.42	0.57

Table 2. **Detailed results of Tab. 1 in the main paper.** We report FID-50K [13] (*without* classifier-free guidance), Inception Score [32] and Precision/Recall [17] metrics.

DiT	Head	Resolution	Batch Size	Training Steps	FID-50K (↓)	IS (↑)	Precision (↑)	Recall (↑)
S/2	1	256	256	400K	64.42	21.54	0.380	0.574
S/2	2	256	256	400K	63.24	22.07	0.385	0.570
S/2	3	256	256	400K	**63.21**	**22.08**	**0.386**	**0.583**
S/2	6	256	256	400K	63.66	22.16	0.383	0.580
S/2	48	256	256	400K	78.76	17.46	0.322	0.482
S/2	96	256	256	400K	116.00	11.49	0.224	0.261
B/2	1	256	256	400K	41.77	34.78	0.487	0.631
B/2	2	256	256	400K	41.39	35.59	0.494	0.631
B/2	3	256	256	400K	**40.86**	**35.79**	**0.497**	**0.629**
B/2	12	256	256	400K	42.11	34.60	0.484	0.631
B/2	96	256	256	400K	68.30	20.45	0.375	0.531
B/2	192	256	256	400K	112.39	12.07	0.240	0.282
L/2	1	256	256	400K	24.46	57.36	0.600	0.637
L/2	2	256	256	400K	24.37	57.02	0.599	0.622
L/2	4	256	256	400K	**24.04**	**59.02**	**0.597**	**0.636**
L/2	16	256	256	400K	25.25	54.67	0.587	0.632
XL/2	1	256	256	400K	21.13	65.06	0.619	0.632
XL/2	2	256	256	400K	**20.66**	**65.39**	**0.624**	**0.636**
XL/2	4	256	256	400K	20.82	65.52	0.619	0.632
XL/2	16	256	256	400K	21.69	63.06	0.617	0.628

Table 3. **Detailed results of Tab. 2 in the main paper.** We report FID-50K [13] (*without* classifier-free guidance), Inception Score [32] and Precision/Recall [17] metrics. DiTs [23] setting.

Load	Iterations	FFN	Modulation Layer	Attention	FID-50K (↓)	IS (↑)	Precision (↑)	Recall (↑)
model	400K	✓	✓	✗	56.07	25.62	0.418	0.608
ema	400K	✓	✓	✗	56.07	25.61	0.416	0.601
model	200K	✓	✓	✗	57.84	24.72	0.408	0.600
model	300K	✓	✓	✗	56.95	25.04	0.414	0.608
model	400K	✓	✓	✗	56.07	25.62	0.418	0.608
model	600K	✓	✓	✗	54.80	26.65	0.424	0.613
model	800K	✓	✓	✗	**53.83**	**27.16**	**0.425**	**0.614**
model	600K	✓	✓	Q, K, V	55.29	26.09	0.419	0.619
model	600K	✓	✓	K, V	55.07	26.38	0.422	0.609
model	600K	✓	✓	V	54.93	26.44	0.427	0.612
model	600K	✓	✓	Q	54.82	26.72	0.423	0.605
model	600K	✓	✓	O	54.84	26.33	0.425	0.607

Table 4. **Detailed results of Tab. 3 in the main paper.** We report FID-50K [13] (*without* classifier-free guidance), Inception Score [32] and Precision/Recall [17] metrics.

Model Size	Iterations	Teacher	λ_1	λ_2	Training Steps	FID-50K (↓)	IS (↑)	Precision (↑)	Recall (↑)
S/2	800K	DiT-S/2	0.1	0.0	400K	55.11	26.28	0.419	0.614
S/2	800K	DiT-XL/2	0.0	0.0	400K	<u>53.83</u>	<u>27.16</u>	<u>0.425</u>	<u>0.614</u>
S/2	800K	DiT-XL/2	0.1	0.0	400K	53.05	27.43	0.431	0.609
S/2	800K	DiT-XL/2	0.05	0.0	400K	53.41	27.26	0.427	0.610
S/2	800K	DiT-XL/2	0.5	0.0	400K	51.13	28.89	0.438	0.616
S/2	800K	DiT-XL/2	0.1	0.05	400K	52.76	27.70	0.431	0.620
S/2	800K	DiT-XL/2	0.0	0.05	400K	53.49	27.26	0.429	0.609
S/2	800K	DiT-XL/2	0.05	0.05	400K	53.14	27.46	0.431	0.609
S/2	800K	DiT-XL/2	0.5	0.05	400K	**50.79**	**29.17**	**0.443**	**0.618**

Table 5. **Detailed results of Tab. 4 in the main paper.** We report FID-50K [13] (*without* classifier-free guidance), Inception Score [32] and Precision/Recall [17] metrics.

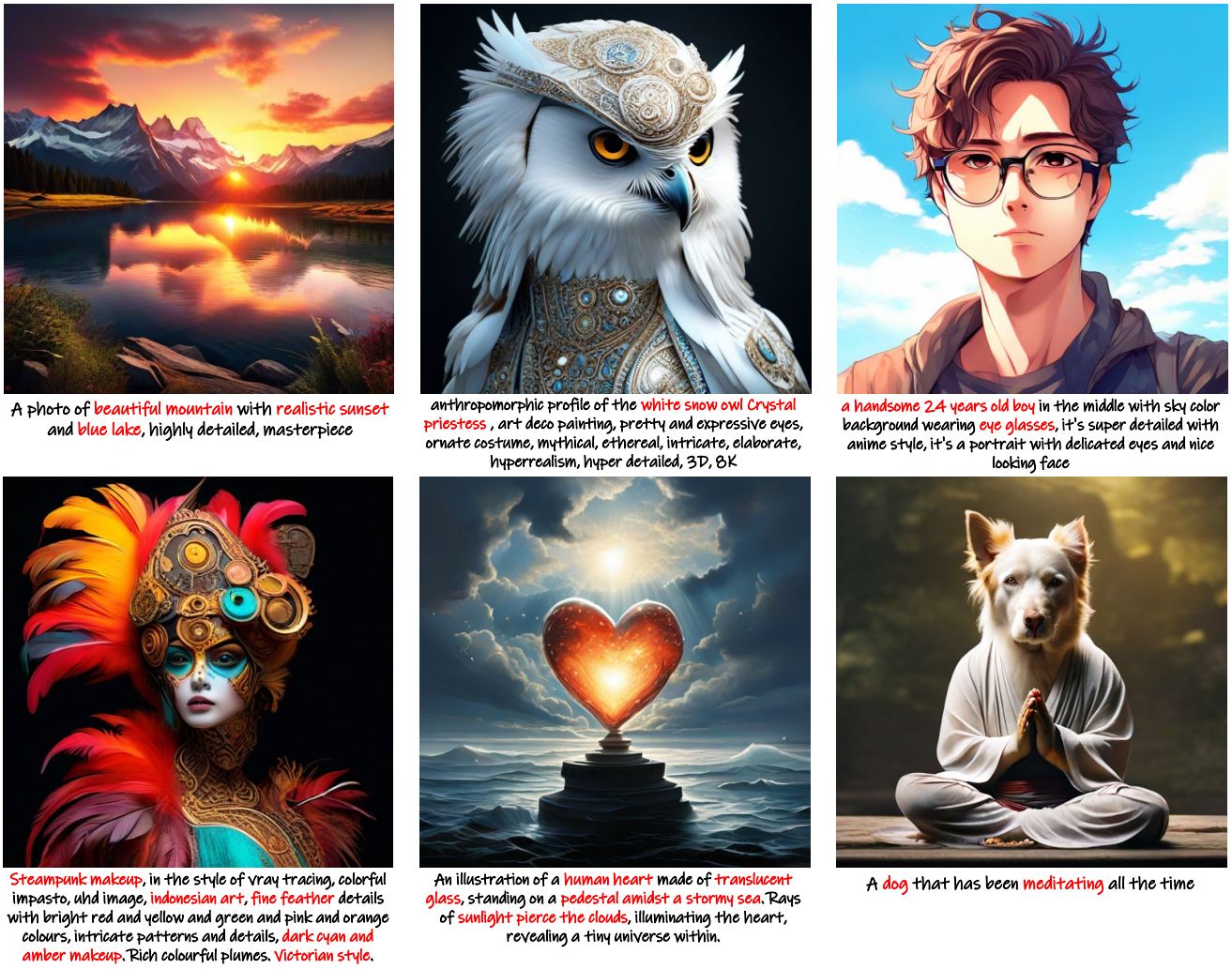

Figure 6. **512px Generated samples of LiT following user instructions.** Converted from PixArt-Σ [5], LiT adopts the same macro- and micro-level architecture, maintaining alignment with the PixArt-Σ framework while elegantly replacing all self-attention with efficient linear attention. While being simple and efficient, LiT can generate exceptional high-resolution images following user instructions.

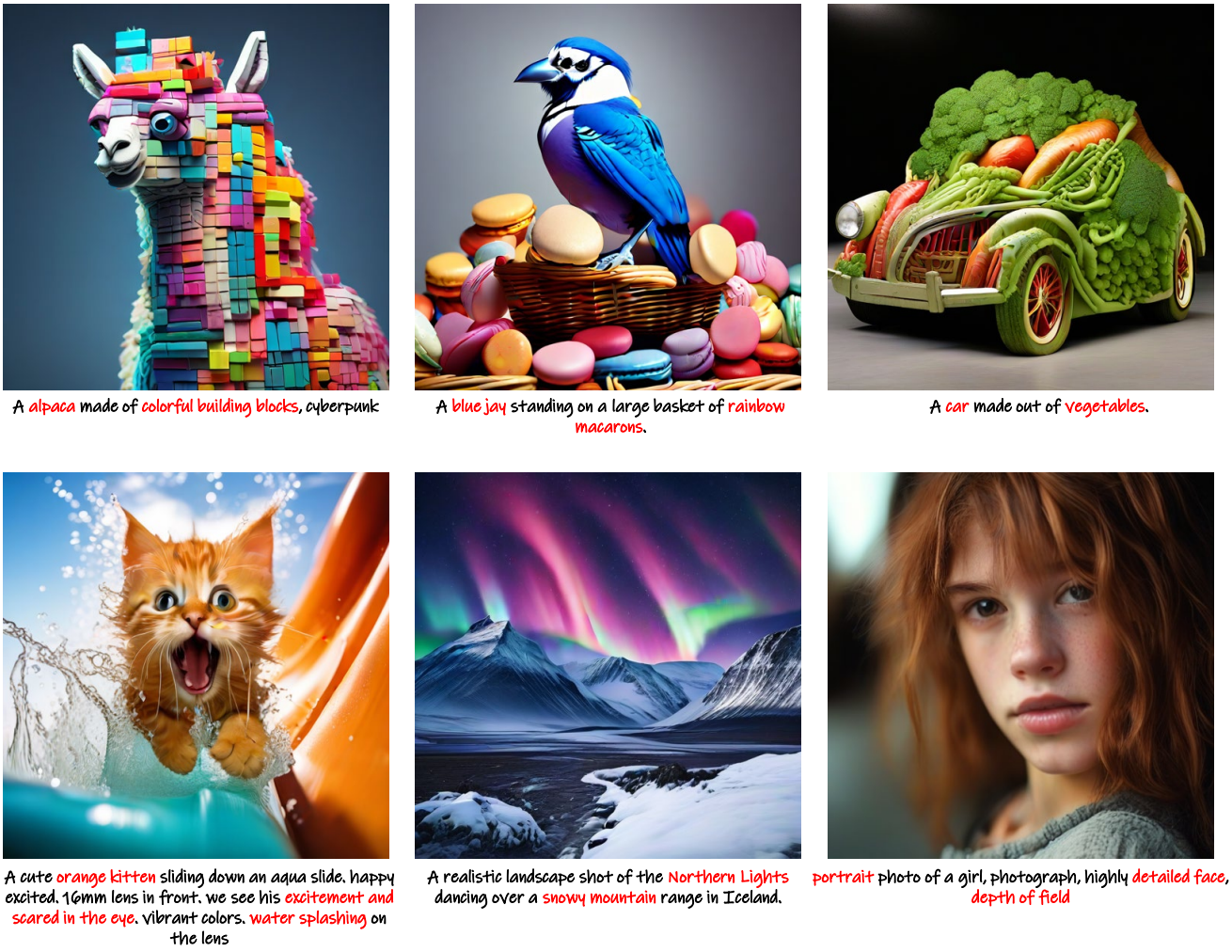

Figure 7. **512px Generated samples of LiT following user instructions.** Converted from PixArt-Σ [5], LiT adopts the same macro- and micro-level architecture, maintaining alignment with the PixArt-Σ framework while elegantly replacing all self-attention with efficient linear attention. While being simple and efficient, LiT can generate exceptional high-resolution images following user instructions.

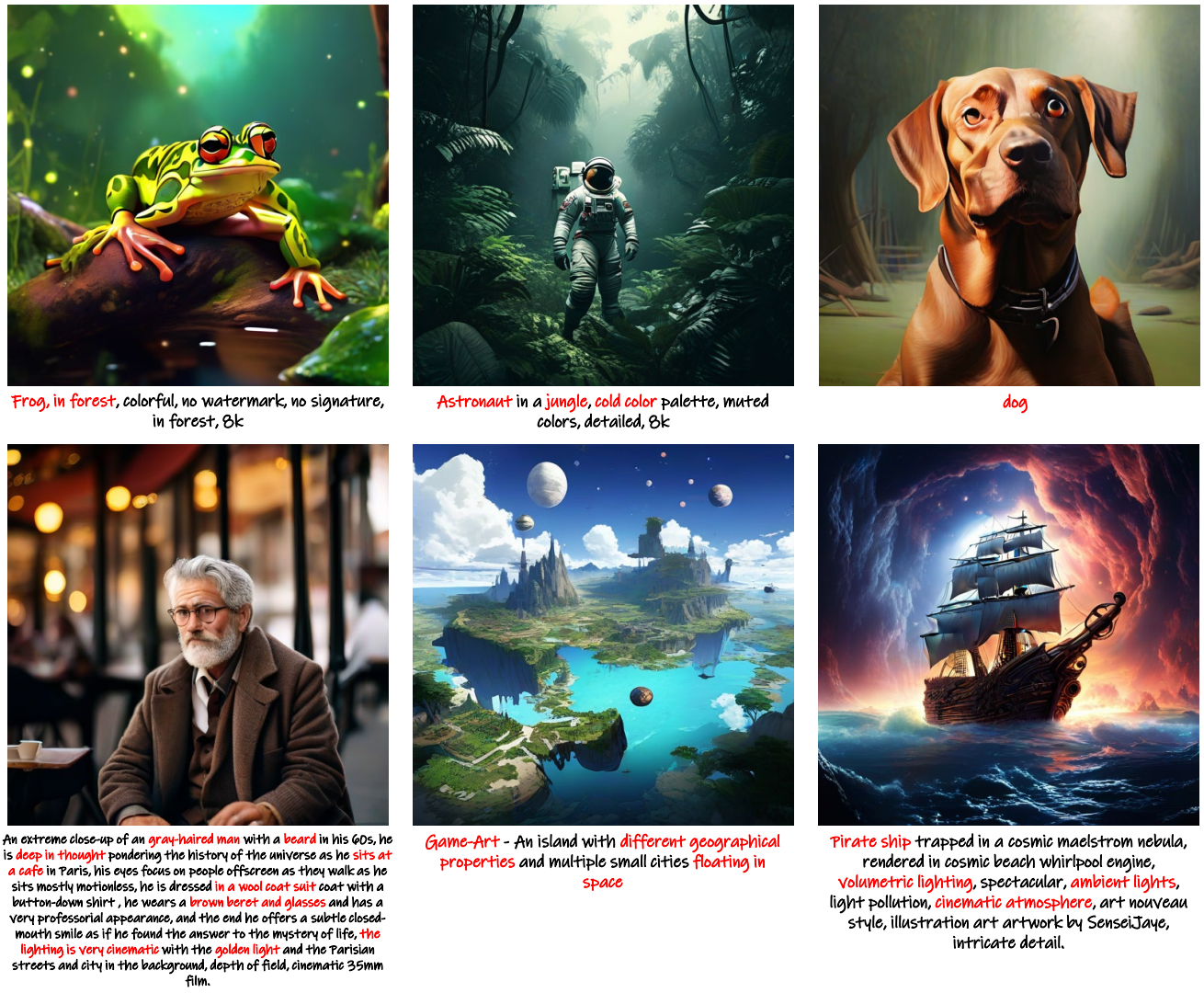

Figure 8. **512px Generated samples of LiT following user instructions.** Converted from PixArt-Σ [5], LiT adopts the same macro- and micro-level architecture, maintaining alignment with the PixArt-Σ framework while elegantly replacing all self-attention with efficient linear attention. While being simple and efficient, LiT can generate exceptional high-resolution images following user instructions.

References

[1] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, and Judy Hoffman. Hydra attention: Efficient attention with many heads. In *ECCV*, 2022. 2

[2] Han Cai, Junyan Li, Muyan Hu, Chuang Gan, and Song Han. Efficientvit: Lightweight multi-scale attention for high-resolution dense prediction. In *ICCV*, 2023. 2, 4

[3] Han Cai, Muyang Li, Zhuoyang Zhang, Qinsheng Zhang, Ming-Yu Liu, and Song Han. Condition-aware neural network for controlled image generation. In *CVPR*, 2024. 4

[4] Hanting Chen, Zhicheng Liu, Xutao Wang, Yuchuan Tian, and Yunhe Wang. Dijiang: Efficient large language models through compact kernelization. *arXiv preprint arXiv:2403.19928*, 2024. 2

[5] Junsong Chen, Chongjian Ge, Enze Xie, Yue Wu, Lewei Yao, Xiaozhe Ren, Zhongdao Wang, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-σ: Weak-to-strong training of diffusion transformer for 4k text-to-image generation. In *ECCV*, 2024. 7, 8, 9

[6] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. In *ICML*, 2024. 3

[7] Zhengcong Fei, Mingyuan Fan, Changqian Yu, Debang Li, and Junshi Huang. Diffusion-rwkv: Scaling rwkv-like architectures for diffusion models. *arXiv preprint arXiv:2404.04478*, 2024. 4

[8] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 3

[9] Jialong Guo, Xinghao Chen, Yehui Tang, and Yunhe Wang. Slab: Efficient transformers with simplified linear attention and progressive re-parameterized batch normalization. In *ICML*, 2024. 2

[10] Dongchen Han, Xuran Pan, Yizeng Han, Shiji Song, and Gao Huang. Flatten transformer: Vision transformer using focused linear attention. In *ICCV*, 2023. 2

[11] Dongchen Han, Tianzhu Ye, Yizeng Han, Zhuofan Xia, Siyuan Pan, Pengfei Wan, Shiji Song, and Gao Huang. Agent attention: On the integration of softmax and linear attention. In *ECCV*, 2024. 2

[12] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2016. 4

[13] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017. 2, 5, 6

[14] Vincent Tao Hu, Stefan Andreas Baumann, Ming Gui, Olga Grebenkova, Pingchuan Ma, Johannes S Fischer, and Björn Ommer. Zigma: A dit-style zigzag mamba diffusion model. In *ECCV*, 2024. 3

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015. 2

[16] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*, 2020. 2

[17] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *NeurIPS*, 2019. 2, 5, 6

[18] Muyang Li, Tianle Cai, Jiaxin Cao, Qinsheng Zhang, Han Cai, Junjie Bai, Yangqing Jia, Ming-Yu Liu, Kai Li, and Song Han. Distrifusion: Distributed parallel inference for high-resolution diffusion models. In *CVPR*, 2024. 4

[19] Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. Q-diffusion: Quantizing diffusion models. In *ICCV*, 2023. 4

[20] Yanjing Li, Sheng Xu, Xianbin Cao, Xiao Sun, and Baochang Zhang. Q-dm: An efficient low-bit quantized diffusion model. In *NeurIPS*, 2023. 4

[21] Songhua Liu, Weihao Yu, Zhenxiong Tan, and Xinchao Wang. Linfusion: 1 gpu, 1 minute, 16k image. *arXiv preprint arXiv:2409.02097*, 2024. 4

[22] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. In *CVPR*, 2024. 4

[23] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023. 1, 3, 5

[24] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, et al. Rwkv: Reinventing rnns for the transformer era. In *EMNLP*, 2023. 4

[25] Hao Phung, Quan Dao, Trung Dao, Hoang Phan, Dimitris Metaxas, and Anh Tran. Dimsum: Diffusion mamba–a scalable and unified spatial-frequency method for image generation. *arXiv preprint arXiv:2411.04168*, 2024. 3

[26] Yifan Pu, Zhuofan Xia, Jiayi Guo, Dongchen Han, Qixiu Li, Duo Li, Yuhui Yuan, Ji Li, Yizeng Han, Shiji Song, et al. Efficient diffusion transformer with step-wise dynamic attention mediators. In *ECCV*, 2024. 4

[27] Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. cosformer: Rethinking softmax in attention. In *ICLR*, 2022. 2

[28] Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion. In *COLM*, 2024.

[29] Zhen Qin, Songlin Yang, and Yiran Zhong. Hierarchically gated recurrent neural network for sequence modeling. In *NeurIPS*, 2024. 2

[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 4

[31] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *ICLR*, 2021. 4

[32] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NeurIPS*, 2016. 2, 5, 6

[33] Yao Teng, Yue Wu, Han Shi, Xuefei Ning, Guohao Dai, Yu Wang, Zhenguo Li, and Xihui Liu. Dim: Diffusion mamba for efficient high-resolution image synthesis. *arXiv preprint arXiv:2405.14224*, 2024. 3

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017. 4

[35] Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Yujun Lin, Zhekai Zhang, Muyang Li, Yao Lu, and Song Han. Sana: Efficient high-resolution image synthesis with linear diffusion transformers. In *ICLR*, 2025. 4

[36] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023. 2

[37] Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T Freeman. Improved distribution matching distillation for fast image synthesis. In *NeurIPS*, 2024. 4

[38] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *CVPR*, 2024. 4

[39] Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. *arXiv preprint arXiv:2410.06940*, 2024. 4

[40] Yang Zhao, Yanwu Xu, Zhisheng Xiao, and Tingbo Hou. Mobilediffusion: Subsecond text-to-image generation on mobile devices. *arXiv preprint arXiv:2311.16567*, 2023. 4

[41] Lianghui Zhu, Zilong Huang, Bencheng Liao, Jun Hao Liew, Hanshu Yan, Jiashi Feng, and Xinggang Wang. Dig: Scalable and efficient diffusion models with gated linear attention. *arXiv preprint arXiv:2405.18428*, 2024. 4