SnapFusion: Text-to-Image Diffusion Model on Mobile Devices within Two Seconds

Yanyu Li[1,2,†] Huan Wang[1,2,†] Qing Jin[1,†] Ju Hu[1] Pavlo Chemerys[1] Yun Fu[2] Yanzhi Wang[2] Sergey Tulyakov[1] Jian Ren[1,†]

[1]Snap, USA [2]Northeastern University, USA [†]Equal contribution

Webpage, ArXiv

Talk @JD, 06/21/2023, Wed

Short Bio: Huan Wang, BE'16@ZJU, MS'19 @ZJU, now 4th-year Ph.D. candidate @NEU. Interned at Alibaba/MERL/Snap. Now interning at Google. Work on **efficient deep learning** (network sparsity & distillation), primarily in the vision domain.

Background & Motivation

Rise of Diffusion Models

- **2015**-ICML-Deep Unsupervised Learning using Nonequilibrium Thermodynamics (Stanford & UCB) – CIFAR10
- **2020**-NIPS-Denoising diffusion probabilistic models (UCB) – DDPM, 1st demonstration of DM generating high-quality images
- **2021**-ICLR-Denoising Diffusion Implicit Models (Stanford) – DDIM
- **2021.01**-DALL-E 1 (OpenAI)
- **2021.05**-Diffusion Models Beat GANS on Image Synthesis (OpenAI)
- **2022.04**-DALL-E 2 (OpenAI)
- **2022.05**-Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding (Google Imagen)
- **2022.08**-Stable Diffusion first release (CVPR'22, Runway + Stability AI)
- **2022.11**-eDiff-I: Text-to-Image Diffusion Models with Ensemble of Expert Denoisers (NVIDIA)
- Papers are exponentially exploding now...

Prerequisites: Diffusion Model in the Generative Family

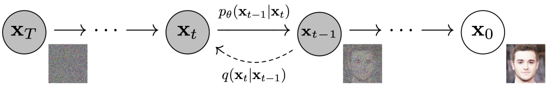
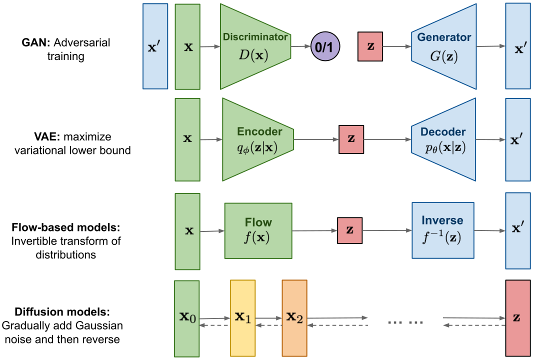


Figure 2: The directed graphical model considered in this work.



- Above: from DDPM (Ho et al., 2020), Below: from *What are Diffusion Models?* by Lilian Weng @OpenAI
- DM is featured by the *gradual (iterative)* diffusion and denoising process.
- DM: Feature or latent ($\mathbf{z}$) has the same shape as the input ($\mathbf{x}$).

Prerequisites: Latent Diffusion Model / Stable Diffusion

- LDM: Apply DM in **feature space** (latent space), for less required resources.
- SD: Train LDM on the large-scale **LAION-5B** dataset (Schuhmann et al.).

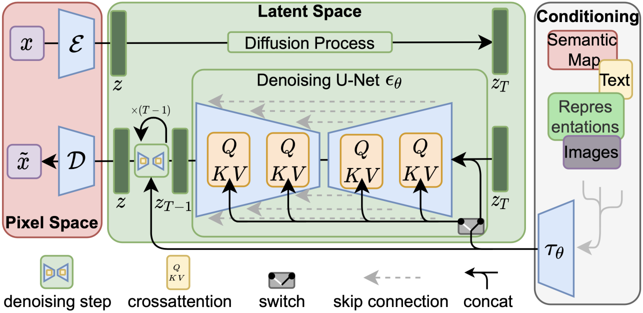


Figure: Overview of LDM (Rombach et al., 2022). 3 parts: VAE encoder/decoder (frozen), **UNet** (key!), text encoder (from CLIP, frozen). Inference: z_0=noise, c=TextEnc(prompt) ⇒ z'=UNet(t, z, c) (iterative) ⇒ img=VAEDec(z).

Prerequisites: Latent Diffusion Model / Stable Diffusion

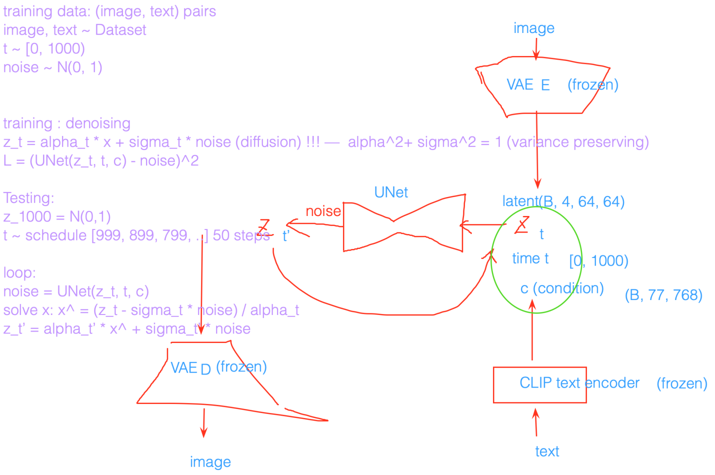


Figure: A rough sketch of SD training, and testing with DDIM (Song et al., 2021)

Motivation: SD is Great, but **Huge and Slow**

- **Huge** model size (fp16, in CoreML), 1B params:
  - Text encoder: 246.3 MB
  - UNet: 1720.7 MB
  - Image Decoder: 99.2 MB
- Prohibitively **slow** (in CoreML):
  - Text encoder: 4.2 ms
  - UNet: unable to profile as one, chunked into 2 parts: $\sim$1,700ms
  - Image Decoder: 370.66 ms

Existing Works (Qualcomm & Google)

OnQ Blog

SD-v1.4, 15s, via full-stack AI optimization

World's first on-device demonstration of Stable Diffusion on an Android phone

Qualcomm AI Research deploys a popular 1B+ parameter foundation model on an edge device through full-stack AI optimization

FEB 23, 2023 | Snapdragon and Qualcomm branded products are products of Qualcomm Technologies, Inc. and/or its subsidiaries.

Google Research Philosophy Research Areas Publications People Resources Outreach

SD-v1.5, 12s, via mobile GPU optimization

BLOG ›

Speed is all you need: On-device acceleration of large diffusion models via GPU-aware optimizations

THURSDAY, JUNE 15, 2023

Posted by Juhyun Lee and Raman Sarokin, Software Engineers, Core Systems & Experiences

The proliferation of large diffusion models for image generation has led to a significant increase in model size and inference workloads. On-device ML inference in mobile environments requires meticulous performance optimization and consideration of trade-offs due to resource constraints. Running inference of large diffusion models (LDMs) on-device, driven by the need for cost efficiency and user privacy, presents even greater challenges due to the substantial memory requirements and computational demands of these models.

More of engineering optimizations – not change the UNet arch., not optimize loss, no new training pipeline.

Profiling - Where is the Speed Bottleneck?

Latency Comparison: SD-v1.5 *vs.* Ours

Table: **Latency Comparison** between Stable Diffusion v1.5 and our proposed efficient diffusion models (UNet and Image Decoder) on **iPhone 14 Pro**. *We notice the latency varies depending on the phones and use three phones to get the average speed.

Stable Diffusion v1.5	Text Encoder	UNet	VAE Decoder
Input Resolution	77 tokens	64 × 64	64 × 64
#Parameters (M)	123	860	50
Latency (ms)	4	∼1,700*	369
Inference Steps	2	50	1
Total Latency (ms)	8	85,000	369
Our Model	Text Encoder	**Our UNet**	**Our Image Decoder**
Input Resolution	77 tokens	64 × 64	64 × 64
#Parameters (M)	123	**848**	**13**
Latency (ms)	4	**230**	**116**
Inference Steps	2	**8**	1
Total Latency (ms)	8	**1,840**	**116**

- Speedup comes from **two major aspects**: reduce the latency of per inference (7.4×) + reduce the number of inferences (6.25×).
- A small speedup from VAE decoder: structured pruning (not covered in this presentation).

Latency Breakdown of SD-v1.5

Cross-Attention modules of the early stages consume *a lot* of time, despite the small #params.

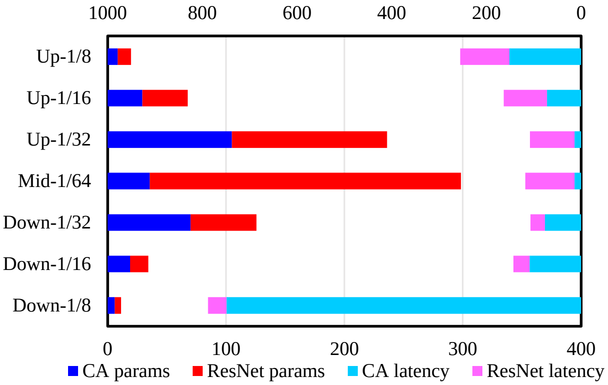


Figure: Latency (iPhone 14 Pro, ms) and parameter (M) analysis for cross-attention (CA) and ResNet blocks in the UNet of Stable Diffusion.

Proposed Method: SnapFusion (Efficient UNet + CFG-aware Distillation)

Efficient UNet (1) - Robust Training, Evaluation and Architecture Evolving

Algorithm 1 Optimizing UNet Architecture

Require: UNet: $\hat{\epsilon}_{\boldsymbol{\theta}}$; validation set: $\mathbb{D}_{val}$; latency lookup table $\mathbb{T}$: $\{Cross\text{-}Attention[i,j], ResNet[i,j]\}$.
Ensure: $\hat{\epsilon}_{\boldsymbol{\theta}}$ converges and satisfies latency objective S.
while $\hat{\epsilon}_{\boldsymbol{\theta}}$ not converged **do**
  Perform robust training.
  $\rightarrow$ **Architecture optimization**:
  if perform architecture evolving at this iteration **then**
    $\rightarrow$ **Evaluate blocks**:
    for each $block[i,j]$ **do**
      $\boxed{\Delta CLIP} \leftarrow \text{eval}(\hat{\epsilon}_{\boldsymbol{\theta}}, A^{-}_{block[i,j]}, \mathbb{D}_{val})$,
      $\boxed{\Delta Latency} \leftarrow \text{eval}(\hat{\epsilon}_{\boldsymbol{\theta}}, A^{-}_{block[i,j]}, \mathbb{T})$
    end for
    $\rightarrow$ **Sort actions based on** $\boxed{\frac{\Delta CLIP}{\Delta Latency}}$**, execute action, and evolve architecture to get latency** T:
    if T not satisfied **then**
      $\{\hat{A}^{-}\} \leftarrow \arg\min_{A^{-}} \frac{\Delta CLIP}{\Delta Latency}$,
    else
      $\{\hat{A}^{+}\} \leftarrow \text{copy}(\arg\max_{A^{-}} \frac{\Delta CLIP}{\Delta Latency})$,
      $\hat{\epsilon}_{\boldsymbol{\theta}} \leftarrow \text{evolve}(\hat{\epsilon}_{\boldsymbol{\theta}}, \{\hat{A}\})$
    end if
  end if

- General idea: remove the unimportant modules and retain the important ones.
- How to measure unimportant/important? By CLIP score drop + latency (lookup table).
- To accurately capture CLIP score drop, apply robust training first.
- So, the pipeline is: robust training + get a latency lookup table ⇒ Drop/keep a module via the automatic Evaluation and Architecture Evolving.

Efficient UNet (2) - Final Proposed Architecture

Table 3: Detailed architecture of our efficient UNet model.

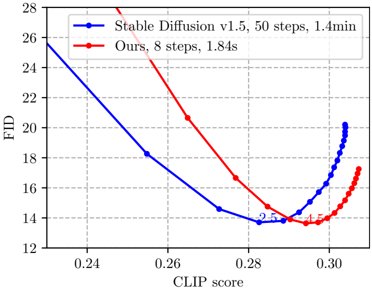
Stage	Resolution	Type	Config	UNet Model Origin	UNet Model Ours
Down-1	$\frac{H}{8} \times \frac{W}{8}$	Cross Attention	Dimension	320	
			# Blocks	$\boxed{2}$	$\boxed{0}$
		ResNet	Dimension	320	
			# Blocks	2	2
Down-2	$\frac{H}{16} \times \frac{W}{16}$	Cross Attention	Dimension	640	
			# Blocks	2	2
		ResNet	Dimension	640	
			# Blocks	2	2
Down-3	$\frac{H}{32} \times \frac{W}{32}$	Cross Attention	Dimension	1280	
			# Blocks	2	2
		ResNet	Dimension	1280	
			# Blocks	$\boxed{2}$	$\boxed{1}$
Mid	$\frac{H}{64} \times \frac{W}{64}$	Cross Attention	Dimension	1280	
			# Blocks	1	1
		ResNet	Dimension	1280	
			# Blocks	$\boxed{7}$	$\boxed{4}$
Up-1	$\frac{H}{32} \times \frac{W}{32}$	Cross Attention	Dimension	1280	
			# Blocks	3	3
		ResNet	Dimension	1280	
			# Blocks	$\boxed{3}$	$\boxed{2}$
Up-2	$\frac{H}{16} \times \frac{W}{16}$	Cross Attention	Dimension	640	
			# Blocks	$\boxed{3}$	$\boxed{6}$
		ResNet	Dimension	640	
			# Blocks	3	3
Up-3	$\frac{H}{8} \times \frac{W}{8}$	Cross Attention	Dimension	320	
			# Blocks	$\boxed{3}$	$\boxed{0}$
		ResNet	Dimension	320	
			# Blocks	3	3

- Remove the *Cross-Attention* module at high resolution (the 1st downsample and last upsample).
- Add more modules for the upsample stage (Up-2).

CFG-aware Distillation (1) - What is CFG?

CFG formulation:

$$\tilde{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(t, \mathbf{z}_t, \mathbf{c}) = w\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(t, \mathbf{z}_t, \mathbf{c}) - (w-1)\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(t, \mathbf{z}_t, \varnothing)$$



- CFG: classifier-free guidance, introduced in (Ho & Salimans, 2022), is an important technique to improve image quality of DMs.
- w: CFG scale, a scalar (default: 7.5 in HF diffusers[a]), used to tradeoff quality (CLIP score) and diversity (FID), see left.
- $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(t, \mathbf{z}_t, \varnothing)$: *unconditional* output obtained by using null text $\varnothing$ as input.

[a] https://github.com/huggingface/diffusers

CFG-aware Distillation (2): Proposed Scheme

Problem: CFG is used in testing while not used in training, *i.e.*, CFG unaware!

Vanilla step distillation:

$$\hat{\mathbf{v}}_t = \hat{\mathbf{v}}_{\boldsymbol{\theta}}(t, \mathbf{z}_t, \mathbf{c}) \Rightarrow \mathbf{z}_{t'} = \alpha_{t'}(\alpha_t \mathbf{z}_t - \sigma_t \hat{\mathbf{v}}_t) + \sigma_{t'}(\sigma_t \mathbf{z}_t + \alpha_t \hat{\mathbf{v}}_t),$$
$$\hat{\mathbf{v}}_{t'} = \hat{\mathbf{v}}_{\boldsymbol{\theta}}(t', \mathbf{z}_{t'}, \mathbf{c}) \Rightarrow \mathbf{z}_{t''} = \alpha_{t''}(\alpha_{t'} \mathbf{z}_{t'} - \sigma_{t'} \hat{\mathbf{v}}_{t'}) + \sigma_{t''}(\sigma_{t'} \mathbf{z}_{t'} + \alpha_{t'} \hat{\mathbf{v}}_{t'}).$$

CFG-aware distillation (ours): Before distillation, apply CFG first to the latent:

$$\tilde{\mathbf{v}}_t^{(s)} = w \hat{\mathbf{v}}_{\boldsymbol{\eta}}(t, \mathbf{z}_t, \mathbf{c}) - (w-1) \hat{\mathbf{v}}_{\boldsymbol{\eta}}(t, \mathbf{z}_t, \varnothing).$$

CFG is applied to both the teacher and the student, with the same w; w is randomly sampled from a uniform distribution ($[2, 14]$ in the paper).

Other Contributions

The major contributions are two: Efficient UNet and CFG-aware Distillation, as presented above.

Please refer to the paper for more:

- Efficient VAE decoder via structured pruning (L_1-norm pruning).
- Training pipeline. *E.g.*, which teacher is used for distilling the 8-step student?

Experimental Results

Quantitative Comparison: FID and CLIP score

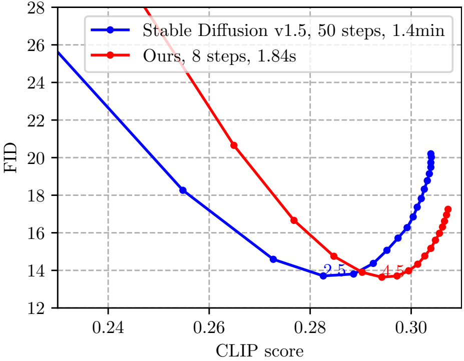


Figure: FID *vs.* CLIP score comparison of our method *vs.* the original SD-v1.5. Same FID, better CLIP score.

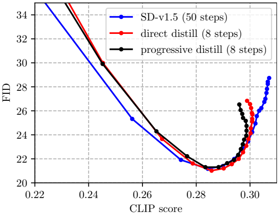
Method	Steps	FID	CLIP
DPM (Lu et al., 2022a)	8	31.7	0.32
DPM++ (Lu et al., 2022b)	8	25.6	0.32
Meng *et al.* (Meng et al., 2023)	8	26.9	0.30
Ours	8	**24.2**	0.30

Table: **Zero-shot** evaluation on MS-COCO 2017 5K subset. Our efficient model is compared against recent arts in the 8-step configuration. Note the compared works use the same model as SD-v1.5, which is much slower than our approach.

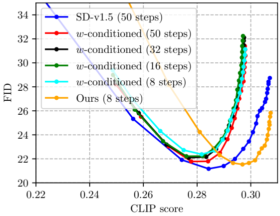
Visual Results

Figure: Generated samples by our SnapFusion. See more in the Appendix of the paper, or, our webpage.

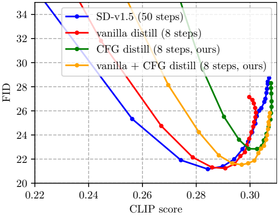
Ablation Studies



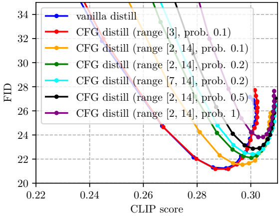
(a) Direct *vs.* progressive



(b) w-conditioned (Meng et al., 2023) *vs.* ours



(c) Vanilla *vs.* CFG distill



(d) CFG hyper-parameters

Conclusion and Discussions

Conclusion

- We present **SnapFusion**, a super-efficient text-to-image diffusion model on mobile devices, with a record-low inference time of **less than 2s**!
- SnapFusion achieves so through a comprehensive optimization in network architecture, objective loss function, and training pipeline.
  - Network architecture: efficient UNet;
  - Loss: CFG-aware distillation;
  - Training pipeline: (not covered in this talk).
- SnapFusion maintains the same quality (in terms of FID and CLIP score) while being much faster than the original SD-v1.5, heralding the encouraging future of *real-time* SD on mobile devices soon.

Limitations & Future Work

- Currently, we **only focus on text-to-image generation** while SD can be used in extensive applications, such as image super-resolution, LoRA (Hu et al., 2022), ControlNet (Zhang & Agrawala, 2023), *etc*.
- We focus on improving the inference **speed**. The #params and **model size** on disk and memory are not optimized.

Thanks! & Questions?

References

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

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

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *arXiv preprint arXiv:2206.00927*, 2022a.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022b.

Chenlin Meng, Ruiqi Gao, Diederik P Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *CVPR*, 2023.

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

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation