APPROXIMATE CACHING FOR EFFICIENTLY SERVING TEXT-TO-IMAGE DIFFUSION MODELS

Shubham Agarwal,[1] Subrata Mitra,[*1] Sarthak Chakraborty,[2] Srikrishna Karanam,[1] Koyel Mukherjee,[1] Shiv K. Saini[1]

Adobe Research,[1] UIUC[2]

Shubham Agarwal

Research Associate, Adobe Research

Networked Systems Design and Implementation

(NSDI 2024)

* corresponding author

Popularity of Text-to-Image

AI Has Already Created As Many Images As Photographers Have Taken in 150 Years. Statistics for 2023*

15 billion images created using AI*

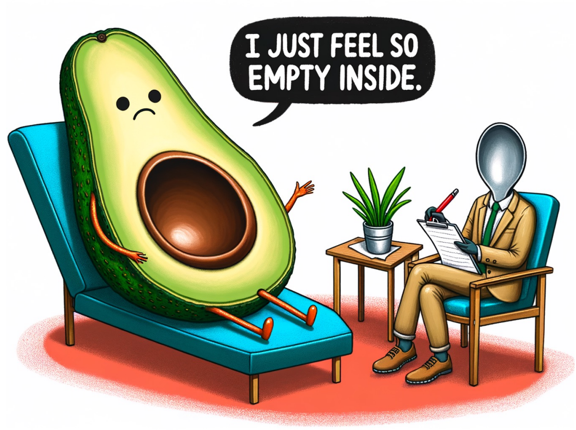
DALLE-E 2: 916 Million

Adobe Firefly**: 1 Billion

Stable Diffusion: 12.59 Billion

Midjourney: 964 Million

Text Art



Cartoons

Abstract Arts

Flyers and templates

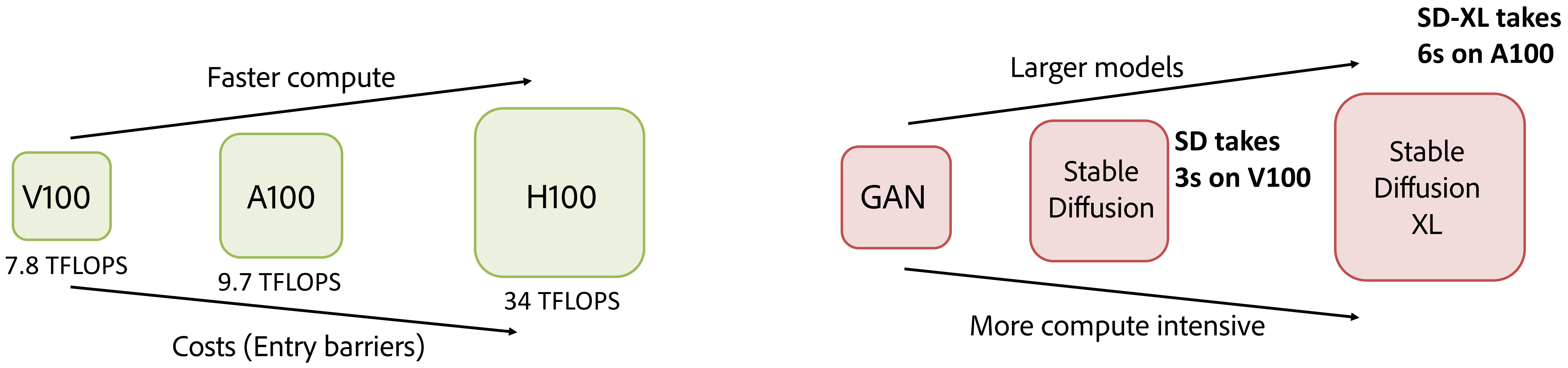
* As of Aug 2023, [https://journal.everypixel.com/ai-image-statistics]

** To date, **Firefly generated over 6.5 billion images and counting!**

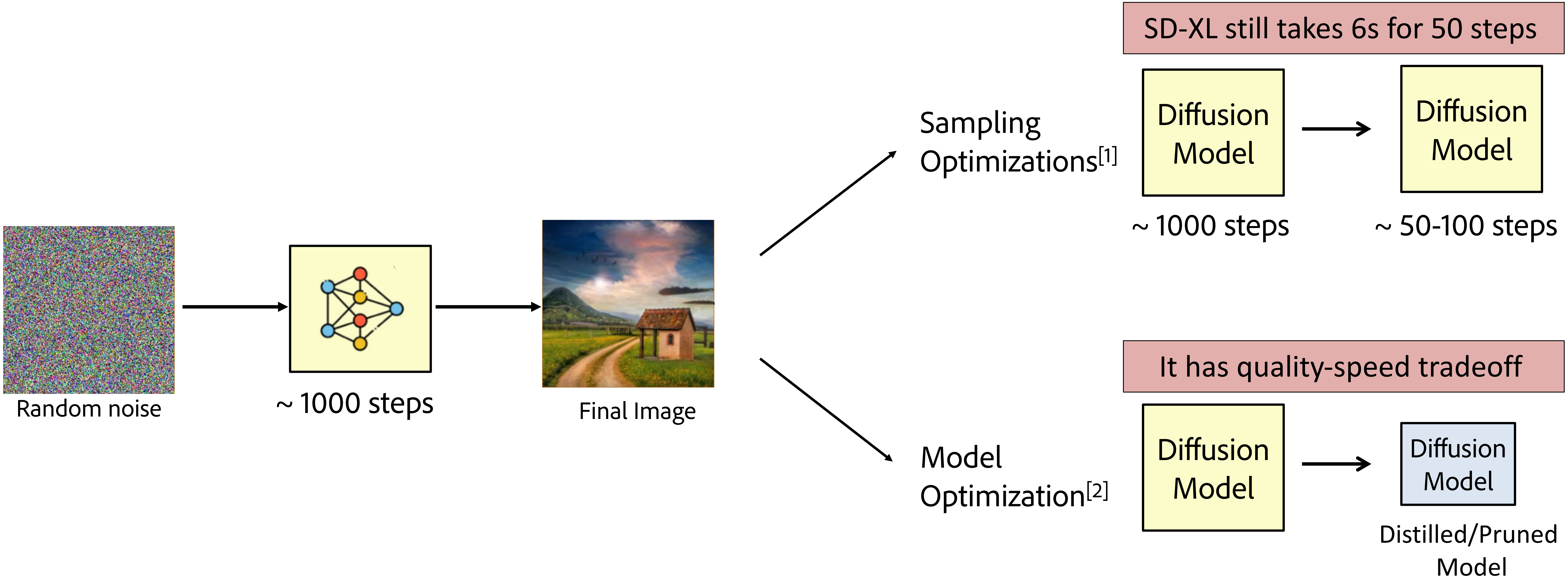
Soaring Costs with Popularity

Evolution of Compute and Model Complexity

Increasing Model Complexity can outpace the Hardware Evolution



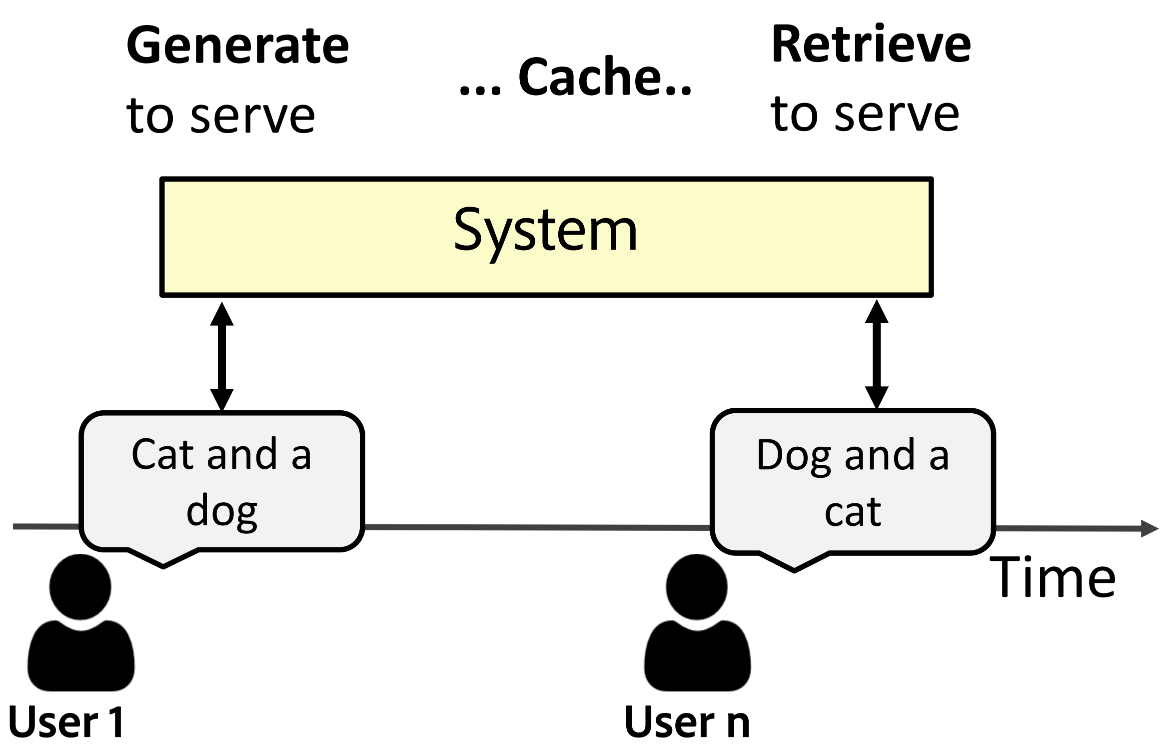
Efficiency of Diffusion models



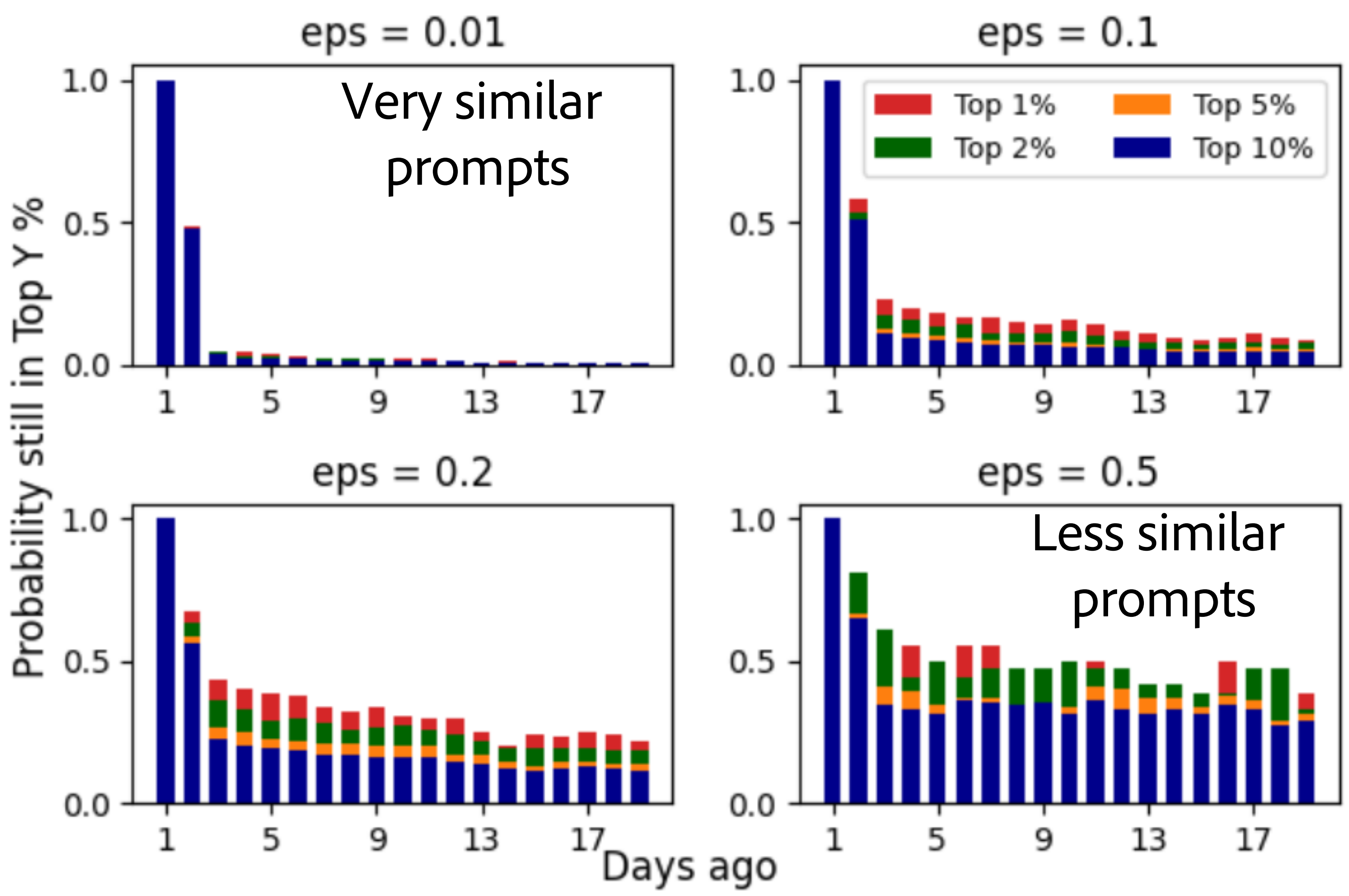
[1] Hongkai Zheng, Weili Nie, Arash Vahdat, Kamyar Azizzadenesheli, and Anima Anandkumar. Fast sampling of diffusion models via operator learning. In ICML 2023.
[2] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In CVPR, 2023.

Proposed Caching Technique

Previous works have overlooked the use of text-to-image systems across multiple generations



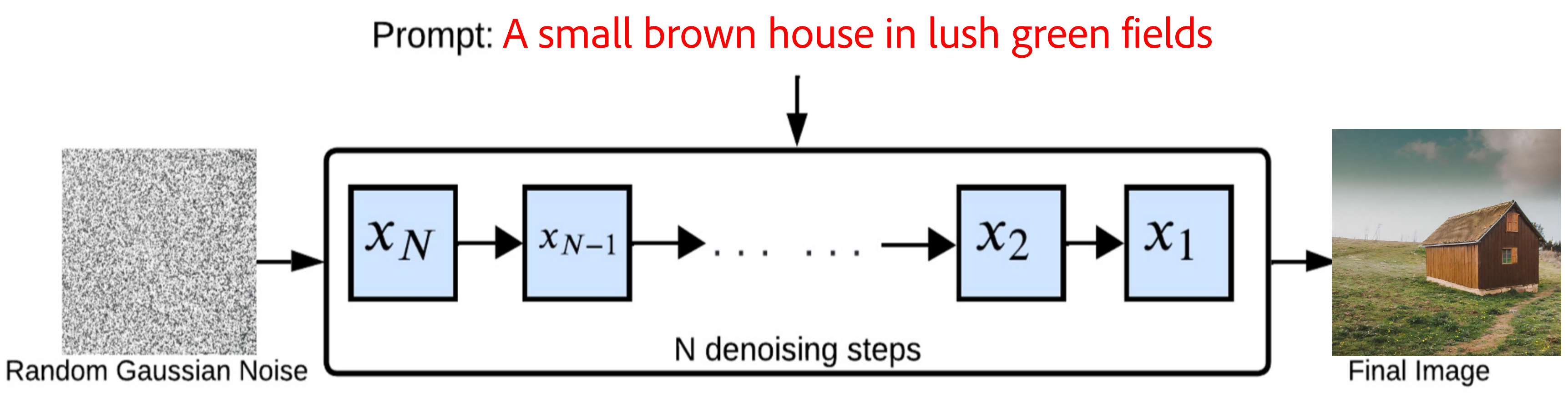
Generate vs. Retrieve

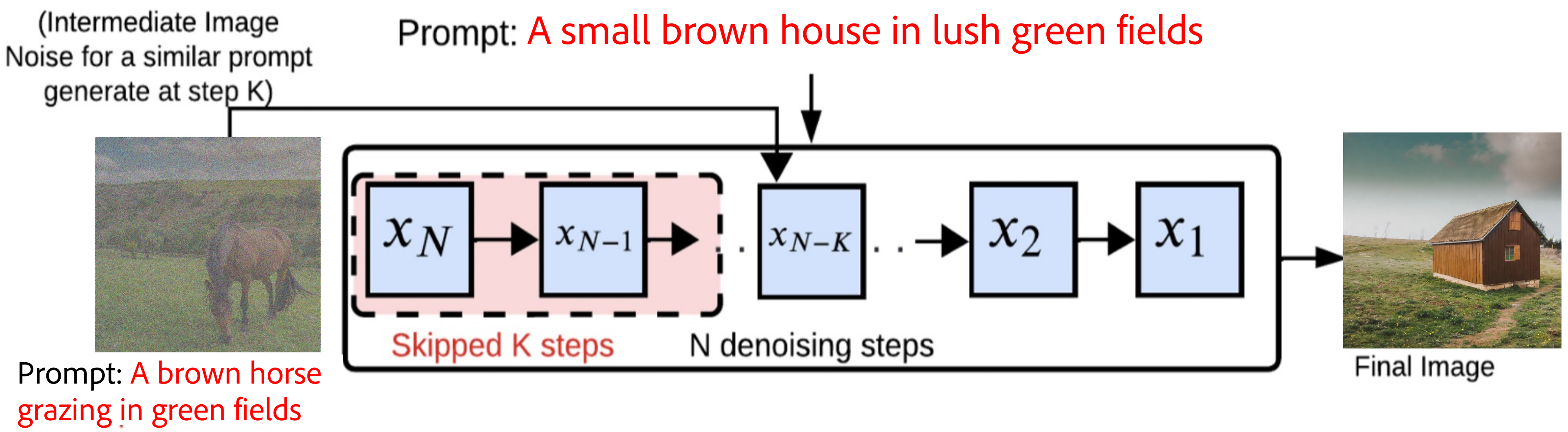


Top Prompt Clusters' Popularity Over Days

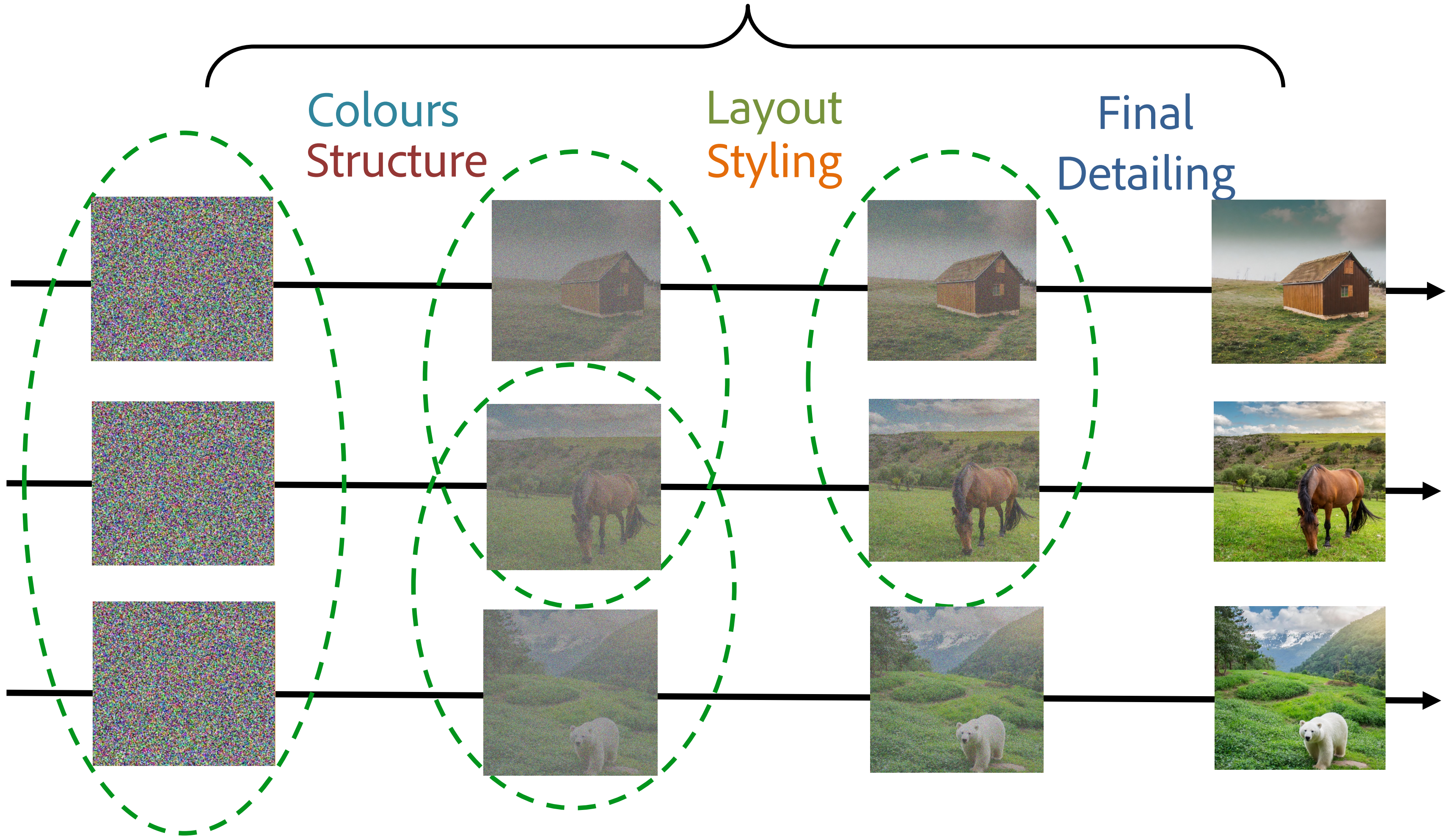
In this work, we reduce the generation time by using a simple-yet-novel approach of **caching**

Proposed Approximate Caching



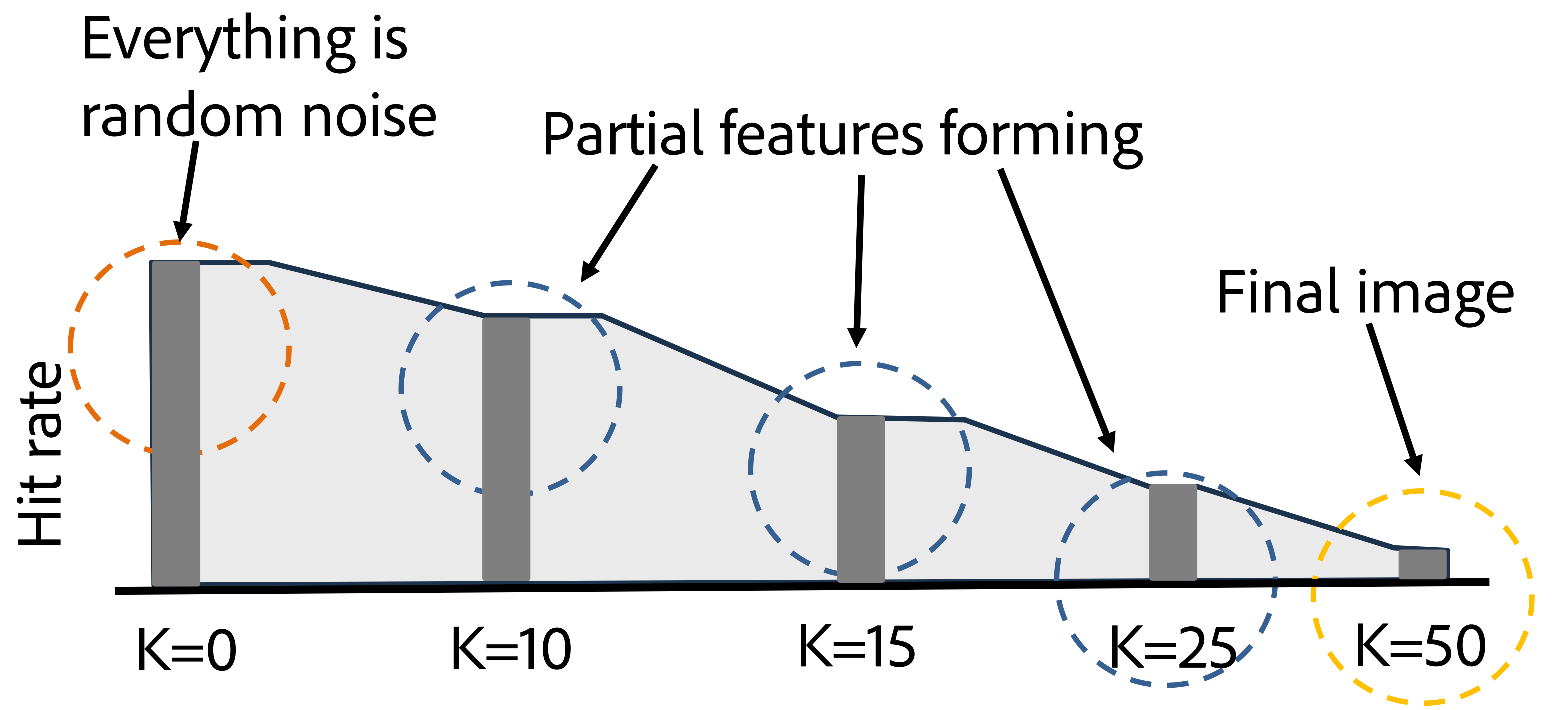


Different aspects of image forms across different steps



Opportunity: Cache and generate from intermediate step

$$\boxed{\text{Maximize compute savings } f_C = h(K) \cdot \frac{K}{N}}$$



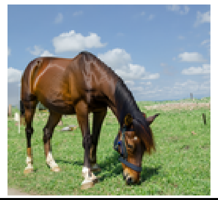
Hit rate is different for different steps of generation

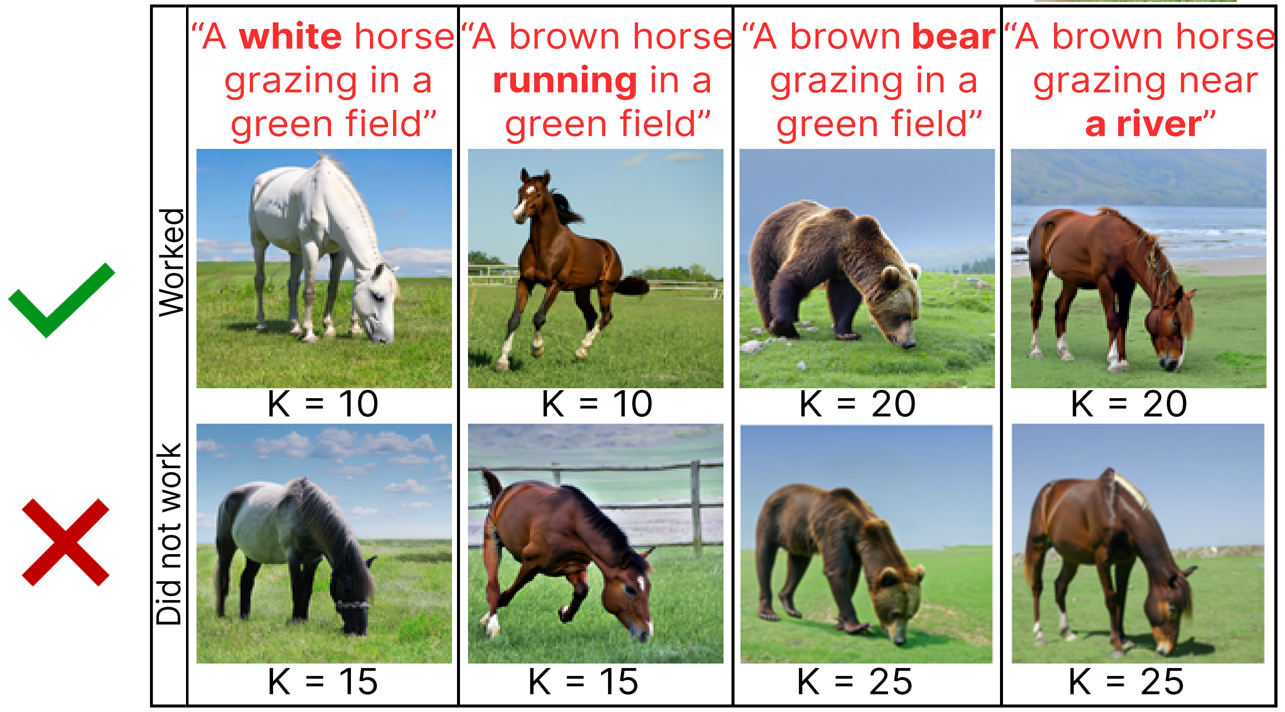
Cache Selection

Cache Selector Heuristic

Hypothesis – More steps can be skipped for a prompt if its cache is more similar

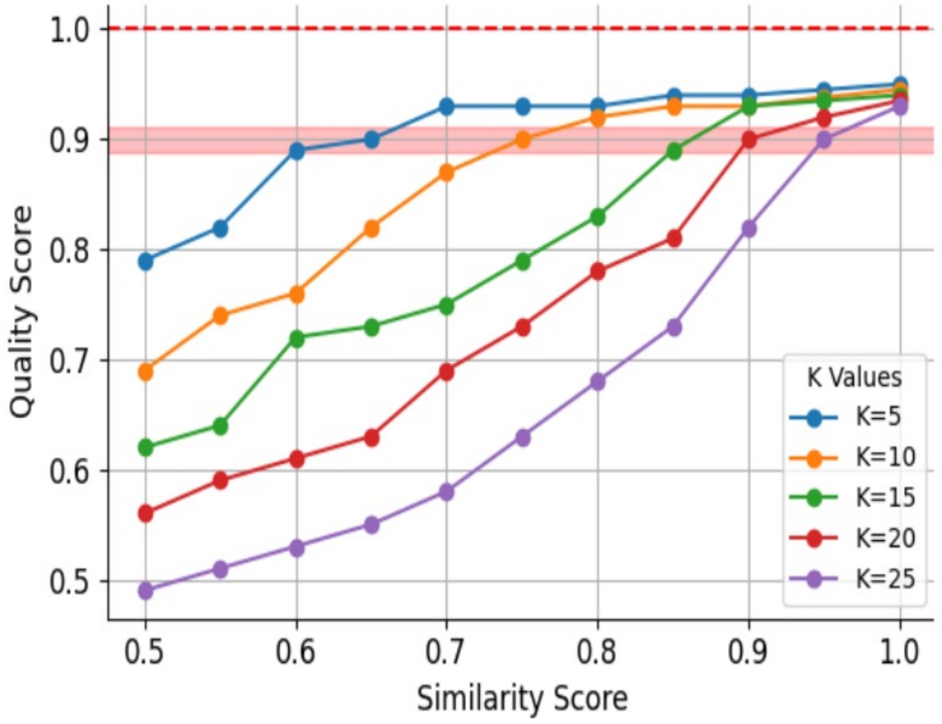
Retrieved prompt:
A **brown** horse **grazing** in a green **field**





K is Determined by Prompt-Cache Similarity:
Noise from a Brown Horse Transforms into Different Prompts, Limited by K

Determine K in terms of Similarity score



- Generate Images for queries at different K values
- Profile Image Quality at different K values for different Prompt-Cache similarity levels

Example output

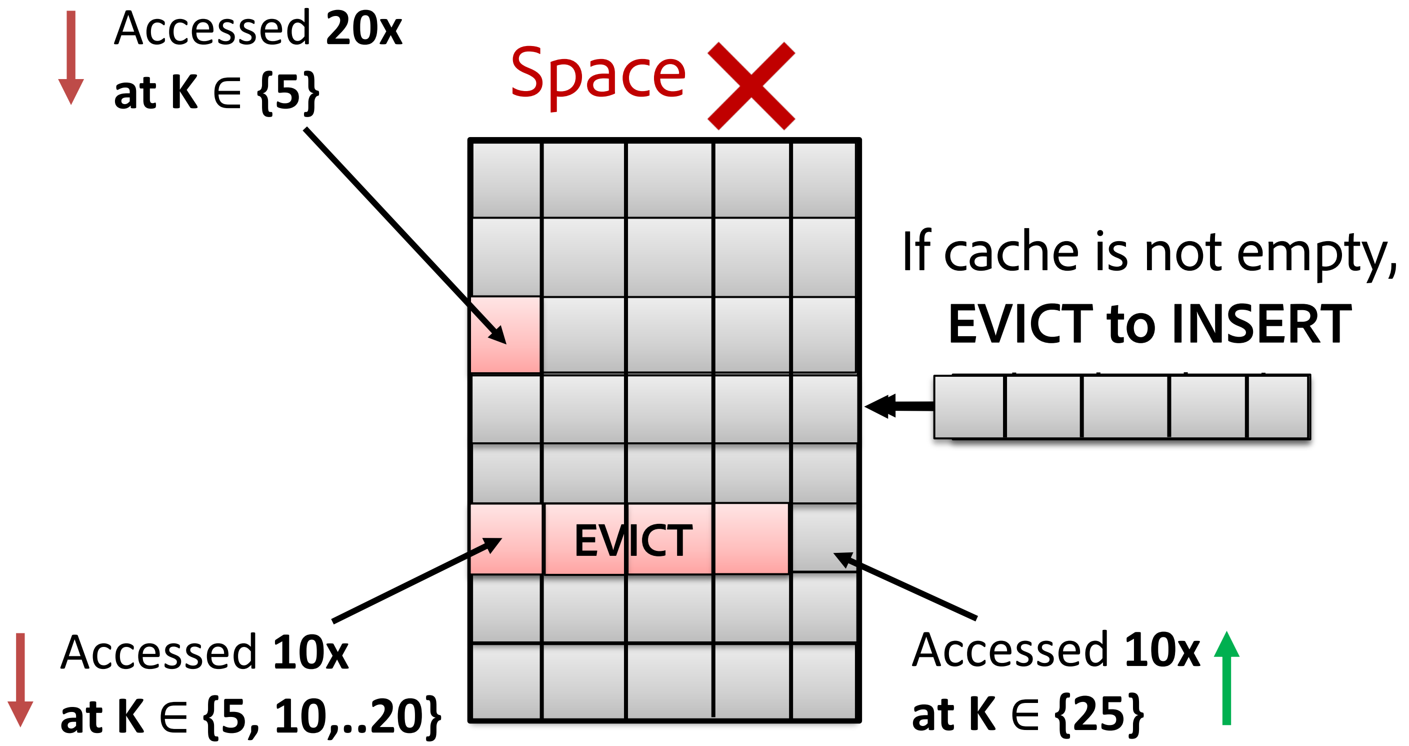
```
def cache_selector(s):
  if s > 0.95: k = 25
  elif s > 0.9: k = 20
  elif s > 0.85: k =
     15
  elif s > 0.75: k =
     10
  elif s > 0.65: k = 5
  else: k = 0
  return k
```

- Map from similarity score to the max K that can be skipped

Cache Management

LCBFU Policy How to maintain the cache for storing noises ?

Least Computationally Beneficial Frequently Used



Fixed Size Cache storage

Cache Size(GB)	#noises in cache	*FIFO*	*LRU*	*LFU*	LCBFU
1*GB*	1500	0.11	0.12	0.12	0.12
10*GB*	15000	0.13	0.14	0.14	0.15
100*GB*	150000	0.14	0.16	0.16	**0.18**
1000*GB*	1500000	0.17	0.20	0.19	**0.23**

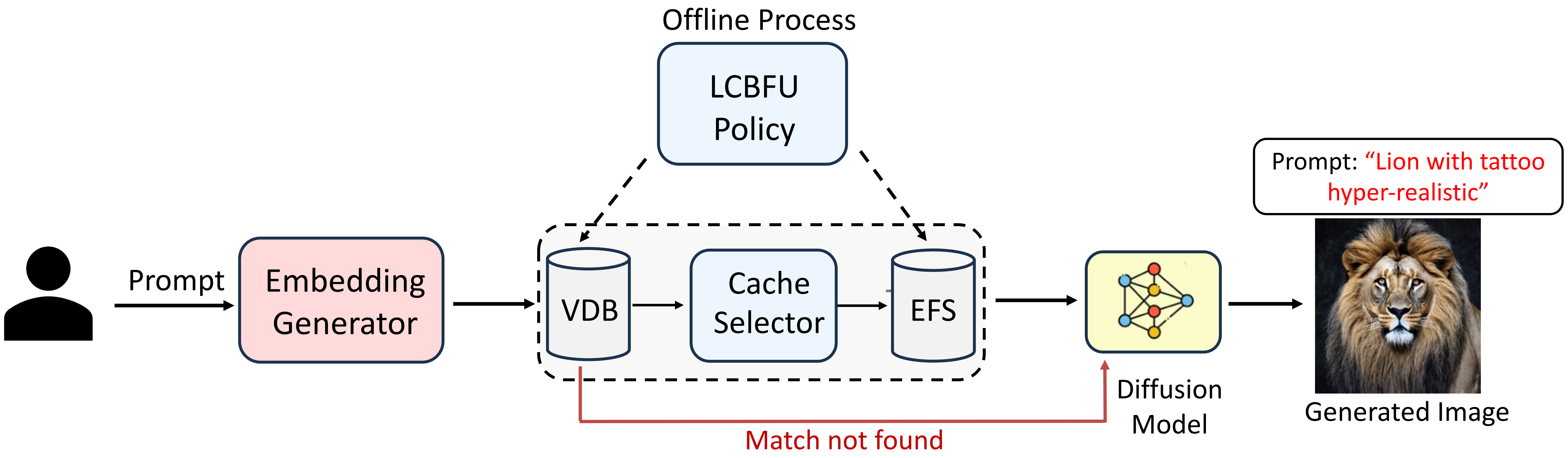
Compute Savings facilitated by LCBFU compared to other eviction techniques

Eviction Policy?

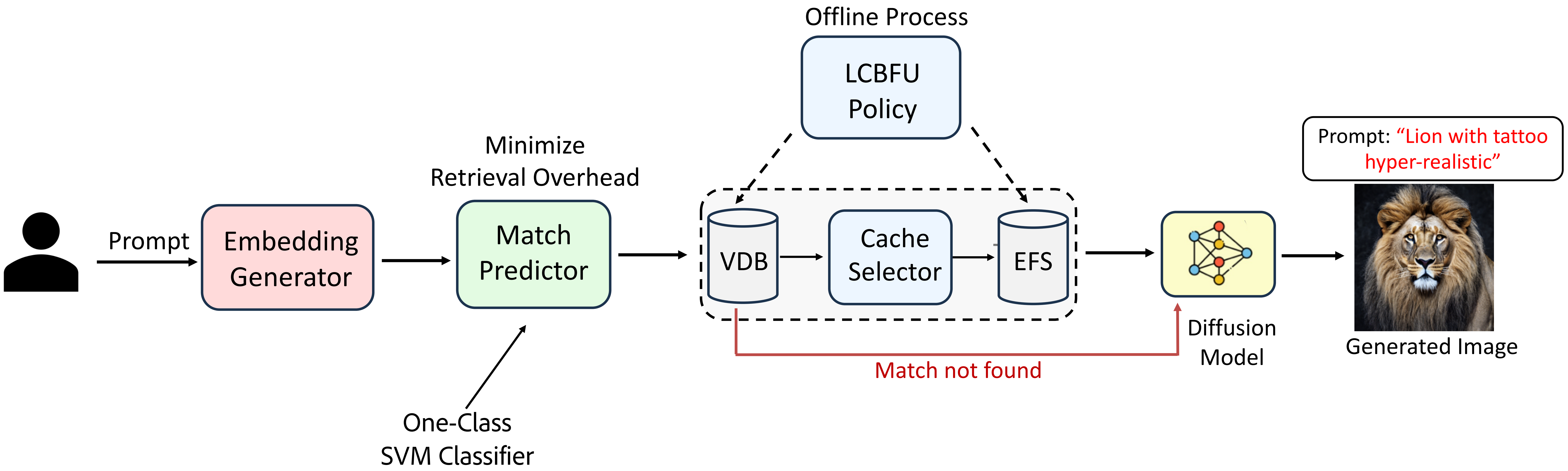
Least Computationally Beneficial Cache (***LCBFU***)

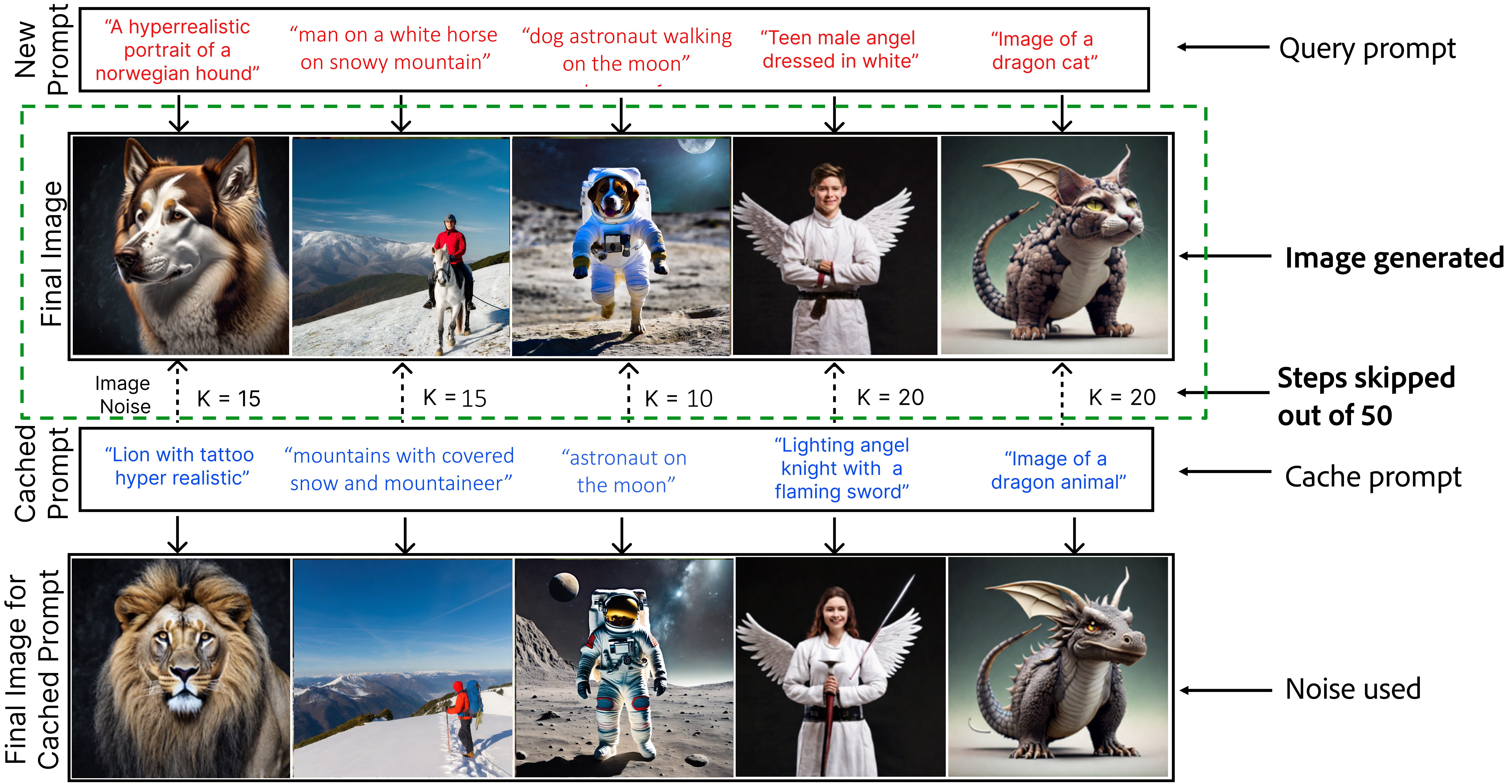
Score = ***K (potential savings)* * f (*frequency of use*)**

NIRVANA: Proposed pipeline



NIRVANA: Proposed pipeline





New Prompt
"A hyperrealistic portrait of a norwegian hound"
"man on a white horse on snowy mountain"
"dog astronaut walking on the moon"
"Teen male angel dressed in white"
"Image of a dragon cat"
Query prompt
Final Image
Image generated
Image Noise
K = 15
K = 15
K = 10
K = 20
K = 20
Steps skipped out of 50
Cached Prompt
"Lion with tattoo hyper realistic"
"mountains with covered snow and mountaineer"
"astronaut on the moon"
"Lighting angel knight with a flaming sword"
"Image of a dragon animal"
Cache prompt
Final Image for Cached Prompt
Noise used

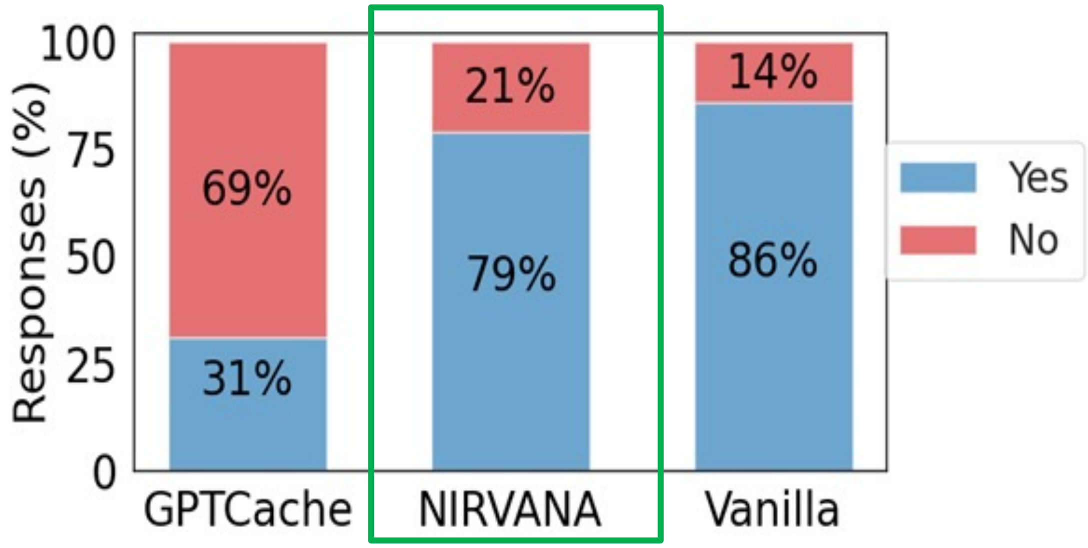
NIRVANA: Results

- We evaluate Nirvana quality using FID, CLIP and PickScore.
- We use real prompt traces from production.
- We conduct a user study with 60 participants.

Dataset	Models	Quality		
		FID ↓	*CLIP Score* ↑	*PickScore* ↑
DiffusionDB*	GPT-CACHE	7.98	25.84	19.04
	PINECONE	10.92	24.83	18.92
	CRS	8.43	24.05	18.84
	SMALLMODEL	11.14	25.64	18.65
	NIRVANA – *w/o*MP	4.94	28.65	20.35
	NIRVANA	**4.68**	**28.81**	**20.41**
	VANILLA	6.12-6.92	30.28	20.86
SYSTEM-X	GPT-CACHE	8.15	26.32	19.11
	PINECONE	10.12	24.43	18.83
	CRS	8.38	23.81	18.78
	SMALLMODEL	11.35	25.91	18.92
	NIRVANA – *w/o*MP	4.48	28.94	20.31
	NIRVANA	**4.15**	**29.12**	**20.38**
	VANILLA	5.42-6.12	30.4	20.71

Comparison of NIRVANA against retrieval-based baselines.
We also compare against a small distilled model.

NIRVANA favored over GPTCache, almost matching Vanilla.

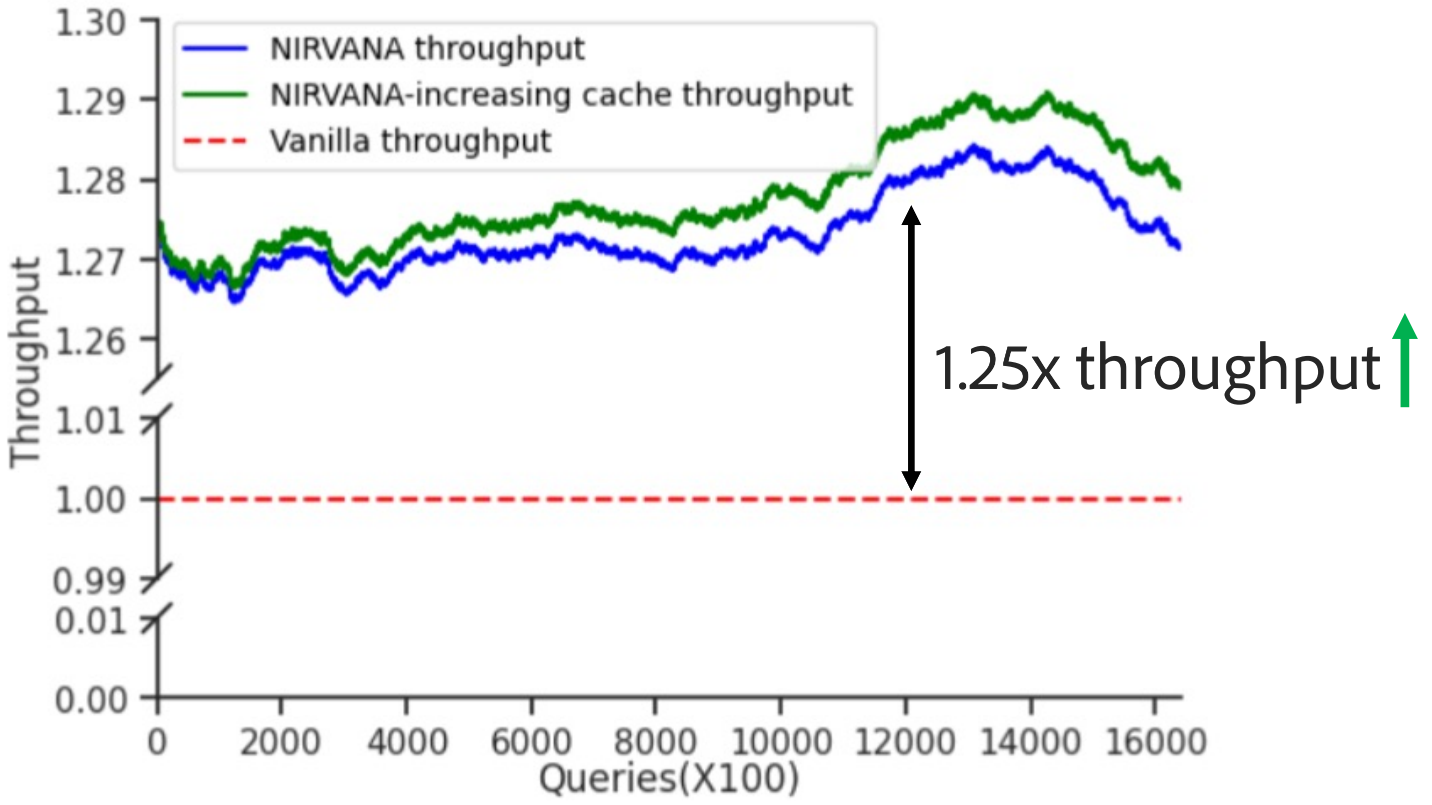


User study for qualitative analysis

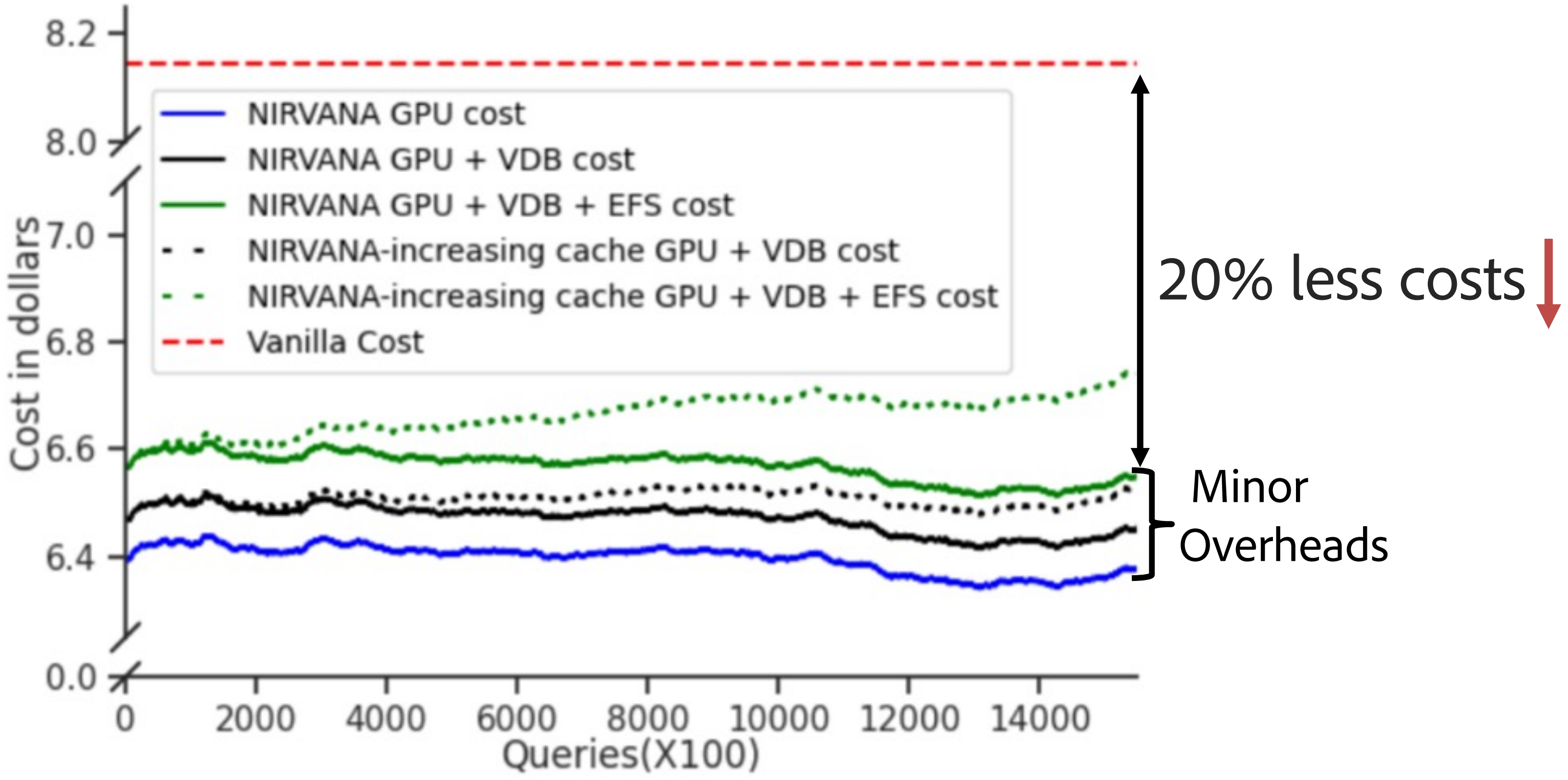
*Collected from Stable Diffusion public Discord server

NIRVANA: Results

- We assess the end-to-end speedup and cost reductions realized by NIRVANA.



Throughput attained by NIRVANA w.r.t. standard deployment



Cost analysis of deploying NIRVANA w.r.t. standard deployment

Analysis and Conclusion

We introduce **NIRVANA**: An **Approximate Caching Technique** for Diffusion models.

The idea is to **cache and reuse intermediate states** from previous text prompts.

We introduces a **new eviction technique** ***LCBFU*** *for* cache management.

We show the effectiveness of NIRVANA using **two real prompt logs***.

*Collected from Stable Diffusion public Discord server

Thank you

NSDI Paper link

Adobe Research

Contact: shagarw@adobe.com