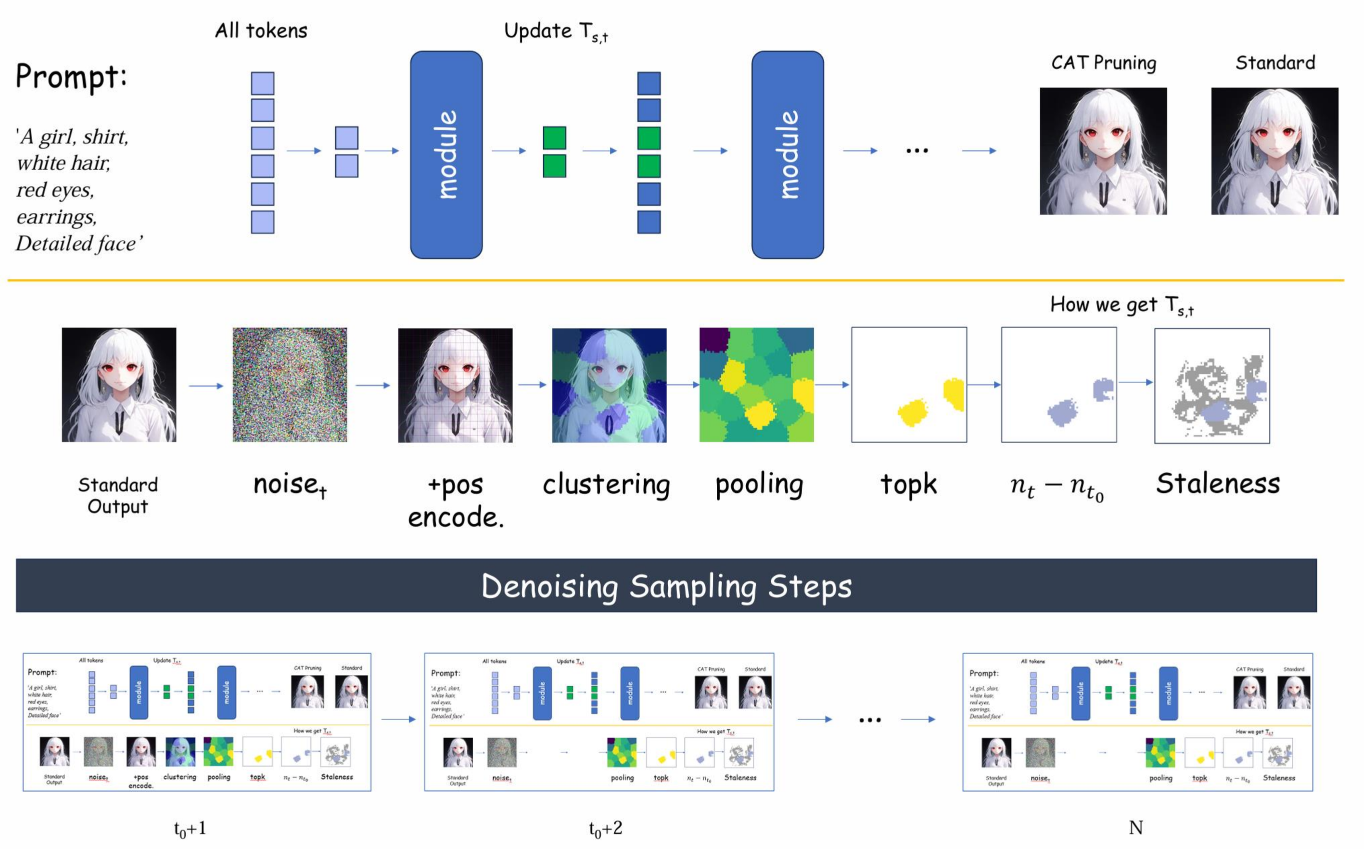
CAT Pruning:
Cluster-Aware Token Pruning For Text-to-Image Diffusion Models

Xinle Cheng[1], Zhuoming Chen[2], Zhihao Jia[2]

[1]Peking University [2]Carnegie Mellon University

Method Overview



At each iteration, tokens are dynamically selected using a combination of the clustering results, noise magnitude, and token staleness.

Token Pruning via Masking

Algorithm 1 is an example of how our method applies to the attention mechanism.

Algorithm 1 Attention Forward Pass in CAT Pruning

1: $Q, K, V \leftarrow \texttt{Update}(T_s)$
2: Compute attention:

$$\text{Attention}(Q,K,V) \leftarrow \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

3: **for** $i \in T_{s,t}$ **do**
4: $\quad h_t[i] \leftarrow MLP(Attention(Q,K,V))$ ▷ Update hidden states of selected tokens
5: **end for**
6: **for** $i \in T_{u,t}$ **do**
7: $\quad h_t[i] \leftarrow h_{t-1}[i]$ ▷ Reuse hidden states for unselected tokens
8: **end for**

Correlation Between Predicted Noise and Historical Noise

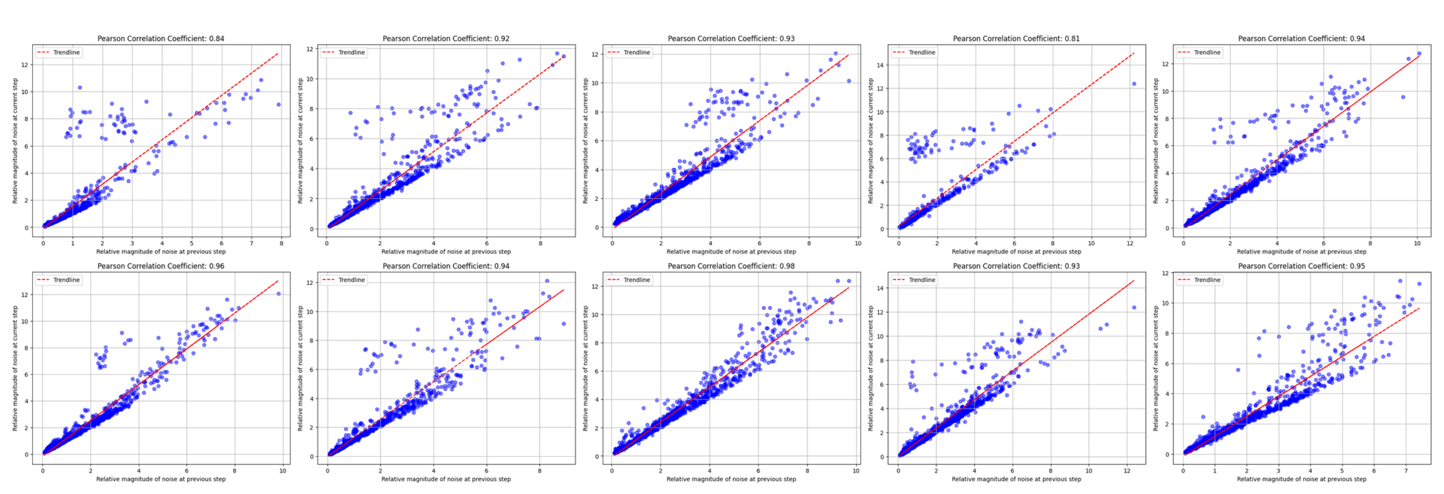


Figure 2: Scatter plot showing the norm of the relative noise at the current step versus the norm of the relative noise at the previous step. We calculate and visualize the Pearson correlation coefficient between these two values.

Proposition 1. Selecting tokens with larger relative noise in the current step increases the likelihood that these tokens will exhibit a larger relative noise in subsequent steps.

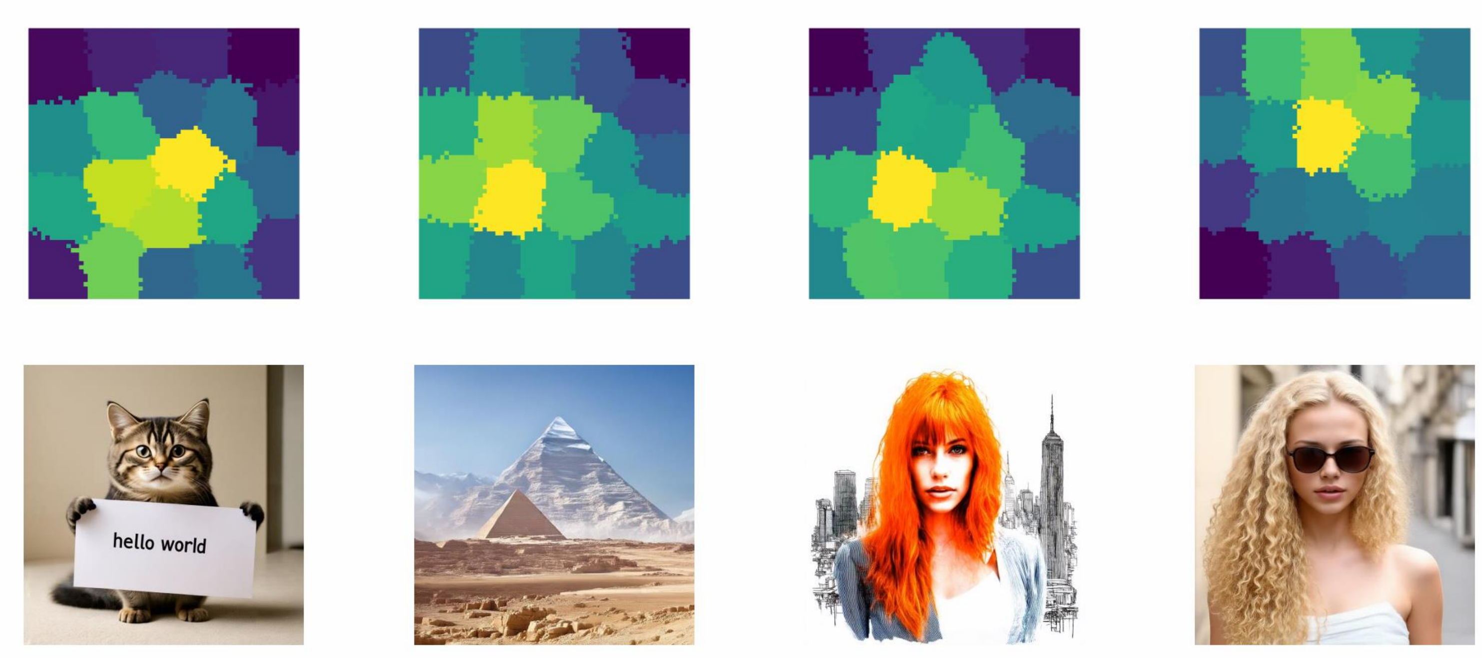
Balancing Noise-Based Token Selection with Distributional Considerations

Clustering Gives Rise to Spatial Details

Enforcing Spatial-awareness while Clustering

$$pos\_enc(i \cdot w + j, :) = \begin{cases} \frac{i}{h}, & \text{if } 1 \le k \le \frac{d}{2} \\ \frac{j}{w}, & \text{if } \frac{d}{2}+1 \le k \le d \end{cases}$$

After adding this positional encoding, we perform KMeans.



A cat holding a sign that says hello world | *The Great Pyramid of Giza situated in front of Mount Everest* | *a mixed media image with a photograph of a woman with long orange hair* | *a girl with long curly blonde hair and sunglasses*

Algorithm 2 Finding Indices for CAT Pruning

1: **Input:** i, t_0, n_i, n_{t_0}
2: $indices \leftarrow []$
3: $RN \leftarrow n_i - n_{t_0}$
4: **if** $i == t_0 + 1$ **then**
5: $\quad clusters \leftarrow KMeans(pos\_enc + n_i - n_{t_0})$ ▷ Cluster noise
6: $\quad graph\_scores \leftarrow pool(clusters, n_i - n_{t_0} + pos\_enc)$ ▷ Aggregate cluster scores
7: $\quad top\_clusters \leftarrow topk(graph\_scores)$
8: $\quad$ **for each** $c \in top\_clusters$ **do**
9: $\quad\quad indices \leftarrow indices \cup topk((n_i - n_{t_0})[j],\ \text{for } j \in c)$
10: $\quad$ **end for**
11: **else**
12: $\quad graph\_scores \leftarrow pool(clusters, pos\_enc + n_i - n_{t_0})$ ▷ Use clusters from $t_0 + 1$
13: $\quad top\_clusters \leftarrow topk(graph\_scores)$
14: $\quad$ **for each** $c \in top\_clusters$ **do**
15: $\quad\quad indices \leftarrow indices \cup topk((n_i - n_{t_0})[j],\ \text{for } j \in c)$
16: $\quad$ **end for**
17: $\quad indices \leftarrow indices \cup topk(-f_i(j), for j \notin indices)$ ▷ Add stale tokens
18: **end if**
19: **return** $indices$

Experiments

α = 0.1 | α = 0.2 | α = 0.3 | α = 0.5 | α = 0.8 | α = 1

Prompt : *a cute blue (kitten bee) looking up, psychedelic background, beautiful detailed eyes, chibi*

Prompt : *a frozen cosmic rose, the petals glitter with a crystalline shimmer, swirling nebulas, 8k unreal engine photorealism, ethereal lighting, red, nighttime, darkness, surreal art*

Method	PartiPrompts				COCO2017			
	MACs ↓	Throughput ↑	Speed ↑	CLIP Score ↑	MACs ↓	Throughput ↑	Speed ↑	CLIP Score ↑
SD3 - 28 steps	168.28T	0.119	1.00×	32.33	168.28T	0.113	1.00 ×	32.47
Ours - 28 steps	90.28T	0.217	1.82×	32.03	90.28T	0.212	1.87 ×	32.21
AT-EDM - 28 steps	93.48T	0.166	1.40×	31.07	93.48T	0.170	1.51×	30.59
Pixart-Σ - 28 steps	120.68 T	0.151	1.00 ×	31.12	120.68 T	0.143	1.00×	31.36
Ours - 28 steps	60.08 T	0.262	1.73 ×	31.06	60.08 T	0.258	1.80×	30.02
AT-EDM - 28 steps	62.08T	0.238	1.57 ×	24.30	62.08T	0.244	1.71×	14.66

Table 2: Comparison of different methods on PartiPrompts and COCO2017 datasets. All methods here adopt 28 sampling steps.

Method	PartiPrompts				COCO2017			
	MACs ↓	Throughput ↑	Speed ↑	CLIP Score ↑	MACs ↓	Throughput ↑	Speed ↑	CLIP Score ↑
SD3 - 50 steps	300.50 T	0.062	1.00 ×	32.92	300.50 T	0.062	1.00×	32.20
Ours - 50 steps	136.70 T	0.134	2.15 ×	32.72	136.70 T	0.130	2.08 ×	32.18
AT-EDM - 50 steps	143.42T	0.107	1.72 ×	28.48	143.42T	0.102	1.64×	28.20
Pixart-Σ - 50 steps	215.40T	0.079	1.00×	31.41	215.40T	0.078	1.00 ×	31.20
Ours - 50 steps	88.24 T	0.166	2.09 ×	31.36	88.24 T	0.160	2.04 ×	30.62
AT-EDM - 50 steps	92.44T	0.148	1.87 ×	17.08	92.44T	0.147	1.88 ×	11.00

Table 3: Comparison of different methods on PartiPrompts and COCO2017 datasets 50 Steps. All methods here adopt 50 sampling steps.

References

Karras, T., Aittala, M., Aila, T. and Laine, S. (2022). Elucidating the design space of diffusion-based generative models. In Proc. NeurIPS.

Ma, X., Fang, G., Mi, M. B. and Wang, X. (2024a). Learning-to-cache: Accelerating diffusion transformer via layer caching.

Twitter

山大王