

A DETAILS ABOUT FACILITY LOCATION OPTIMIZATION

A.1 GAIN FUNCTION COMPUTATION IN FL

In the greedy algorithm we select the token v^* from $V \setminus A$ that maximizes the following gain function:

$$v^* = \arg \max_{v \in (V-A)} f(v|A)$$

where $f(v|A)$ is defined as:

$$\begin{aligned} f(v^*|A) &= f(\{v^*\} \cup A) - f(A) \\ &= \sum_{v \in V} \max_{v' \in (\{v^*\} \cup A)} \text{sim}(v, v') - \sum_{v \in V} \max_{v'' \in A} \text{sim}(v, v'') \end{aligned}$$

Since for each $v \in V$, find the maximum corresponding v' in the updated representatative set $\{v^*\} \cup A$ is equivalent to compare v' in A and v^* , namely:

$$\sum_{v \in V} \max_{v' \in (\{v^*\} \cup A)} \text{sim}(v, v') = \sum_{v \in V} \max \left[\left(\max_{v' \in A} \text{sim}(v, v'), \text{sim}(v, v^*) \right) \right]$$

Therefore

$$\begin{aligned} f(v^*|A) &= \sum_{v \in V} \max \left[\left(\max_{v' \in A} \text{sim}(v, v'), \text{sim}(v, v^*) \right) \right] - \sum_{v \in V} \max_{v'' \in A} \text{sim}(v, v'') \\ &= \sum_{v \in V} \max \left[\left(\max_{v' \in A} \text{sim}(v, v'), \text{sim}(v, v^*) \right) - \max_{v'' \in A} \text{sim}(v, v'') \right] \\ &= \sum_{v \in V} \max \left(0, \text{sim}(v, v^*) - \max_{v'' \in A} \text{sim}(v, v'') \right) \end{aligned}$$

Eventually,

$$v^* = \arg \max_{v' \in (V-A)} \sum_{v \in V} \max \left(0, \text{sim}(v', v^*) - \max_{v'' \in A} \text{sim}(v, v'') \right)$$

A.2 FACILITY LOCATION OPTIMIZATION ALGORITHM

Algorithm 2: Facility Location Token Selection Algorithm**Input:** Similarity matrix $S \in \mathbb{R}^{N \times N}$, number of tokens to select D **Output:** Selected token indices T **Initialize:** $T \leftarrow \{\}$;**for** $i = 1$ **to** N **do** Compute row sums: $s_i = \sum_{j=1}^N S_{i,j}$;**end**Select the first token: $t_1 \leftarrow \arg \max_i s_i$;Add t_1 to T : $T \leftarrow T \cup \{t_1\}$;Initialize the largest row: $c \leftarrow S_{t_1}$;Set $S_{t_1} \leftarrow 0$;**for** $k = 2$ **to** D **do** **for each token** i **not in** T **do** Compute gain: $g_i = \sum_{j=1}^N \max(0, S_{i,j} - c_j)$; **end** Select next token: $t_k \leftarrow \arg \max_{i \notin T} g_i$; Add t_k to T : $T \leftarrow T \cup \{t_k\}$; Update largest row: $c_j \leftarrow \max(c_j, S_{t_k,j})$ for all $j = 1$ to N ; Set $S_{t_k} \leftarrow 0$;**end****return** T

B OVERALL DETAILED ALGORITHM OF TOMA

Algorithm 3: ToMA with local regions

Input: Tensor $\mathbf{X} \in \mathbb{R}^{B \times N \times d}$ (input tensor), D (number of destinations), τ (attention temperature), F (computational layer)

- 1 $\mathbf{X} \leftarrow (\mathbf{X}_1, \dots, \mathbf{X}_P)$; /* Reorganize \mathbf{X} as local regions */
- where $\mathbf{X}_p \in \mathbb{R}^{B \times N_{local} \times d}$ for $p = 1 \dots P$ and $N_{local} \times P = N$;
- 2 $D_{local} \leftarrow D/P$; $\mathbf{X} \leftarrow \mathbf{X}.reshape(B \times P, N_{local}, d)$;
- Step 1: Facility Location**
- 3 GPU Greedy to get: $(T_1, T_2, \dots, T_{B \times P}) \leftarrow \text{Greedy}(f_{FL}, D_{local}, \mathbf{X})$;
- 4 Gather $\mathbf{X}_T \leftarrow (\mathbf{X}_{1,T_1}, \mathbf{X}_{2,T_2}, \dots, \mathbf{X}_{B \times P, T_{B \times P}})$; /* Shape: $B \times P, D_{local}, d$ */
- Step 2: Merge**
- 5 $\mathbf{A} \leftarrow \text{SDPA}(\mathbf{X}_T, \mathbf{X}, \mathbf{I}, \tau)$; /* Shape: $B \times P, D_{local}, N_{local}$ */
- 6 $\tilde{\mathbf{A}} \leftarrow \mathbf{A} / \mathbf{A}.\text{sum}(-1)$; /* Normalize each row */
- 7 $\mathbf{X}_{merged} \leftarrow \tilde{\mathbf{A}} \mathbf{X}$; /* Apply Merge, Shape: $B \times P, D_{local}, d$ */
- Computational Layer:**
- 8 $\mathbf{X}' \leftarrow F(\mathbf{X}_{merged}.reshape(B, D, d))$;
- Step 3: Unmerge**
- 9 $\mathbf{X}'_{unmerged} \leftarrow \tilde{\mathbf{A}}^\top \mathbf{X}'$;
- 10 Group $\mathbf{X}'_{unmerged}$ back to reverse the local region split;
- 11 **return** $\mathbf{X}'_{unmerged}$

C MORE ABLATION RESULTS

C.0.1 UNMERGE COMPARISON (TRANSPOSE VS PSEUDO-INVERSE)

Unmerge method	CLIP	DINO	MSE	Time (s)
transpose	31.0273	0.0569	1296.3734	4.75
pseudo-inverse	30.9972	0.0571	1288.2609	10.07

Table 6: Comparison of unmerge methods (transpose vs. pseudo-inverse) under the condition: 50 recompute steps, ratio=0.5, global attention (globalAttn)

From Tab. 6 we find the difference between transpose and pseudo-inverse method of unmerge shows similar outcome in scores while transpose is significantly faster than pseudo-inverse, we then set transpose as our default setting.

C.0.2 SCHEDULE

In this section, we compare the metric of different sharing schedules of destination selections and attention weights computation

Dst steps	Attn steps	CLIP	DINO	MSE
first step	first step	30.0429	0.0773	2488.5682
every 10 steps	every 10 steps	30.8171	0.0729	1735.4429
every 10 steps	every 5 steps	30.8652	0.0699	1632.3441
every 10 steps	every 1 step	30.9971	0.0668	1524.6436
every 5 steps	every 5 steps	30.8923	0.0692	1608.6099
every 1 step	every 1 step	30.9196	0.0668	1551.5814

Table 7: Recompute schedule comparison with CLIP, DINO, and MSE metrics

From Tab. 7 we observe that recomputing attention and destination (Dst) steps more frequently generally results in slightly improved performance across the CLIP, DINO, and MSE metrics. Specifically, recomputing attention every step yields the best scores in all metrics, while less frequent recomputation (e.g., every 10 steps) results in slightly worse scores but still competitive results. The difference between recomputation frequencies becomes more noticeable in the MSE metric, where recomputing more frequently leads to lower error values. We select a recomputation schedule of computing attention every 5 steps and destination every 10 steps because this provides nearly similar performance to the most frequent recomputation (every step) while likely being faster due to reduced computation overhead. This approach strikes a good balance between performance and efficiency.



Figure 7: Comparison of pseudo-inverse and transpose

D THEORETICAL COMPLEXITY

We keep necessary constants for the complexity estimate as they are essential factors in practical speedup calculation. Also, we count the total number of multiplications by treating matrix multiplication as multiple dot products, ignoring algorithms with better theoretical complexity. The complexity is $7d^2N + 2dN^2$ for a self-attention block. After using token merge, the complexity is: $(7d^2D + 2dD^2)$ as we reduce the input size from N to D . We also define $r := D/N$ as the reduction ratio. Thus, the speedup in terms of the reduction ratio is $\text{Speedup} = \frac{7d+2N}{7d \cdot r + 2N \cdot r^2}$.

The overhead of submodular optimization is: N^2d

The overhead of computing merge attention projection is: $NDd + Nd$

The overhead of merge is: NDd

The overhead of unmerge with transpose is: NDd

E DETAILED RESULTS ON GEMREC & IMAGENET1K

Method	Ratio	FID	CLIP	DINO	MSE	RTX6000	V100	RTX8000
baseline_SDXL	0	25.27	0.30	0.00	0.00	6.1	14.5	16.1
ToMe	0.25	25.65	0.30	0.05	1716.13	8.7	15.0	16.9
	0.5	26.73	0.30	0.07	2279.39	8.7	12.9	14.6
	0.75	41.23	0.29	0.08	2344.87	8.2	11.2	12.4
ToMA strip	0.25	25.17	0.30	0.05	1604.18	5.6	12.6	14.5
	0.5	29.11	0.30	0.07	2199.76	4.6	10.1	12.0
	0.75	89.93	0.27	0.11	3185.34	4.5	8.0	9.5
ToMA tile	0.25	25.43	0.30	0.05	1348.64	6.2	13.6	15.7
	0.5	29.19	0.30	0.06	1912.22	6.3	11.1	13.2
	0.75	58.90	0.28	0.09	2802.32	6.2	9.1	10.7
ToMA *	0.25	26.31	0.30	0.05	1866.68	5.5	12.3	13.5
	0.5	38.14	0.29	0.08	3451.15	4.9	9.7	11.5
	0.75	123.37	0.25	0.11	5440.23	4.9	7.6	8.9
ToMA	0.25	25.72	0.30	0.05	1432.56	6.0	14.3	15.9
	0.5	28.88	0.30	0.07	2012.13	5.0	11.0	12.8
	0.75	58.59	0.28	0.10	2785.68	4.3	8.5	9.8
LTB	0.25	—	—	—	—	5.2	12.1	3.1
	0.5	—	—	—	—	4.0	9.9	7.8
	0.75	—	—	—	—	3.1	8.3	6.5

Table 8: Comparison of different methods with respect to FID, CLIP, DINO, MSE, and various GPU performance metrics (RTX6000Ada, V100, RTX8000).

F MORE QUALITATIVE RESULTS

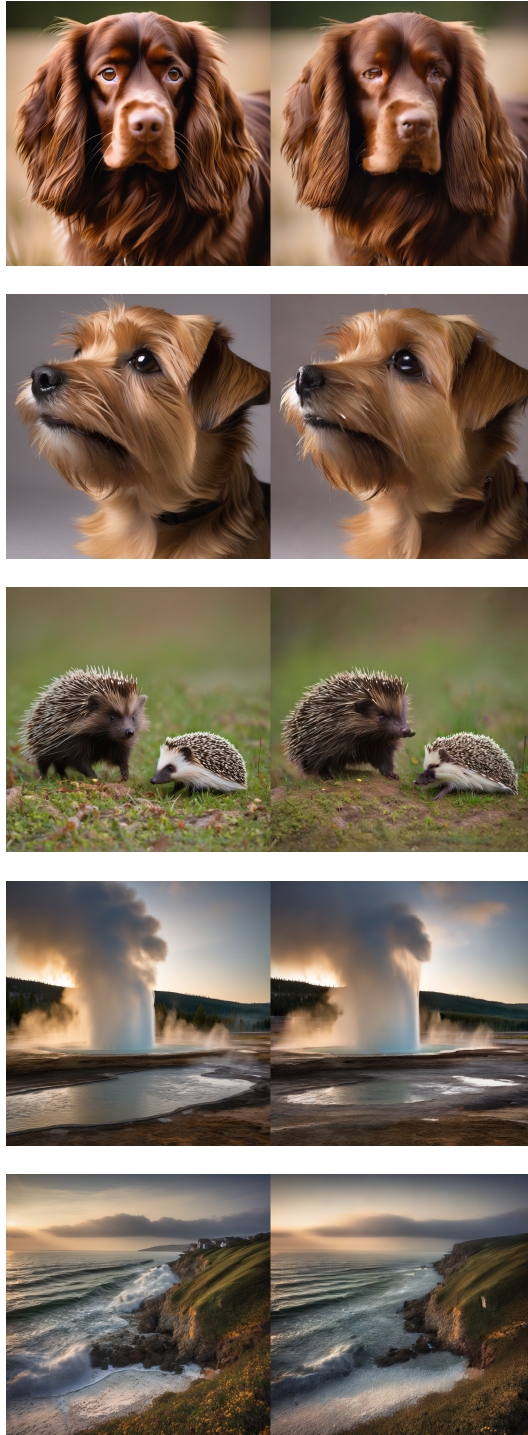


Figure 8: Image Comparison between left: original image generated by SDXL; right: Image generated by ToMA at ratio 0.5



Figure 9: Images generated by different ToMA variances. from left to right: ToMA_stripe, ToMA, ToMA_tile, ToMA