CORE: *C*ompact *O*bject-centric *RE*presentations as a New Paradigm for Token Merging in LVLMs

Supplementary Material

A. Ablation Study

In Sec. 4.2, we introduce our primary merging heuristic which prioritizes large objects in the case of fixed-rate compression. To investigate the model's sensitivity to this choice, we conduct an ablation study with an inverted *small-object-first* strategy. The results, presented in Tab. S1, show that reversing the order leads to only a minimal drop in average performance. This not only demonstrates CORE's strong robustness to the merging order but also validates our *large-object-first* heuristic as a slightly superior choice. For completeness, the pseudo code for this inverted strategy is detailed as Algorithm 2 in Sec. B.

Table S1. **Ablation Study of Small-object-first Strategy.** Red (Blue) font indicates the performance drop (gain) relative to the large-object-first merging strategy.

tokens	POPE	MME	MMB^{CN}	SQA^{I}	$SEED^{I}$	MMMU	Avg.
640	85.3	1509.3	59.5	70.5	66.1	37.3	65.7
	-1.6	-12.3	-0.5	+1.3	-1.5	-1.0	-0.7
320	86.4	1486.5	58.0	69.8	64.6	38.2	65.2
	+0.1	-11.4	+0.7	+0.4	-1.3	-0.2	-0.2
160	85.7	1378.0	56.5	70.2	63.6	38.6	63.9
	-0.3	-27.3	-0.2	+0.4	-1.1	+2.0	-0.1

B. Algorithms

Algorithm 1 first calculates the area of all segmentation regions (the number of tokens contained) and sorts them in descending order by area. Subsequently, it sets a merging budget Δ which means the total number of tokens to be reduced, based on the difference between the original token count and the target number. During the merging phase, the algorithm iterates through these regions in descending order of size. As long as the budget Δ is sufficient, it fuses all tokens within a region into a single token via averaging and deducts the corresponding cost $(A_n - 1)$ from the budget. If the budget is insufficient to merge the entire current region, the algorithm performs a partial merging to exhaust the remaining budget. Once the budget reaches zero, all tokens from the remaining regions, typically smaller objects, are kept intact without merging. Finally, the algorithm returns the token set composed of newly merged tokens and preserved original tokens, totaling N_{target}. Algorithm 2 is similar to Algorithm 1 but changes the sorting criterion in line 7, prioritizing the merging of small objects.

Algorithm 1 Fixed-rate Token Merging. Larger objects are merged earlier.

Require: Set of N hard masks $\mathcal{Q}_{\text{valid}} = \{Q_1, \ldots, Q_N\}$, Vision features $F \in \mathbb{R}^{HW \times C}$, Target token number N_{target}
Ensure: Merged visual tokens $F_{\text{merged}} \in \mathbb{R}^{N_{\text{target}} \times C}$
1: *// 1. Analyze and Prioritize Segments*
2: $\mathcal{S} \leftarrow$ EmptyList
3: **for** $n = 1$ to N **do**
4: $A_n \leftarrow \text{Area}(Q_n)$
5: Add tuple (n, A_n, Q_n) to $\mathcal{S}$
6: **end for**
7: Sort segments $\mathcal{S}$ primarily by descending area A_n, then by ascending mask ID n
8: *// 2. Perform Budgeted Merging*
9: $\Delta \leftarrow |F| - N_{\text{target}}$
10: *// Initialize merging budget (number of tokens to remove)*
11: $F_{\text{merged}} \leftarrow$ EmptyList
12: **for** each segment (n, A_n, Q_n) in sorted $\mathcal{S}$ **do**
13: $F_n \leftarrow \text{SelectFeatures}(F, Q_n)$
14: **if** $\Delta > 0$ and $(A_n - 1) \leq \Delta$ **then**
15: *// Case 1: Fully merge the mask*
16: $\bar{f}_n \leftarrow \text{AverageFeatures}(F_n)$
17: $d_n \leftarrow \text{Centroid}(Q_n)$
18: Add token $(\bar{f}_n, d_n)$ to F_{merged}
19: $\Delta \leftarrow \Delta - (A_n - 1)$
20: **else if** $\Delta > 0$ and $(A_n - 1) > \Delta$ **then**
21: *// Case 2: Partially merge the mask*
22: $\bar{f} \leftarrow \text{AverageFeatures}(\text{the first } (\Delta + 1) \text{ tokens of } F_n)$
23: $d \leftarrow \text{Centroid}(\text{the first } (\Delta + 1) \text{ tokens of } Q_n)$
24: Add token $(\bar{f}, d)$ to F_{merged}
25: Add the remaining $A_n - (\Delta + 1)$ tokens from F_n to F_{merged}
26: $\Delta \leftarrow 0$
27: **else**
28: *// Case 3: No budget left, keep all tokens*
29: Add all tokens from F_n (with their original positions) to F_{merged}
30: **end if**
31: **end for**
32: *// 3. Finalize Output*
33: Sort F_{merged} by spatial position (original or averaged centroid)
34: $F_{\text{merged}} \leftarrow \text{StackFeatures}(F_{\text{merged}})$
35: **return** F_{merged}

Algorithm 2 Fixed-rate Token Merging. Smaller objects are merged earlier.

Require: Set of N hard masks $\mathcal{Q}_{\text{valid}} = \{Q_1, \ldots, Q_N\}$, Vision features $F \in \mathbb{R}^{HW \times C}$, Target token number N_{target}
Ensure: Merged visual tokens $F_{\text{merged}} \in \mathbb{R}^{N_{\text{target}} \times C}$

// 1. Analyze and Prioritize Segments
$\mathcal{S} \leftarrow$ EmptyList
for $n = 1$ to N **do**
  $A_n \leftarrow \text{Area}(Q_n)$
  Add tuple (n, A_n, Q_n) to $\mathcal{S}$
end for
Sort segments $\mathcal{S}$ primarily by ascending area A_n, then by ascending mask ID n
// 2. Perform Budgeted Merging
$\Delta \leftarrow |F| - N_{\text{target}}$
// Initialize merging budget (number of tokens to remove)
$F_{\text{merged}} \leftarrow$ EmptyList
for each segment (n, A_n, Q_n) in sorted $\mathcal{S}$ **do**
  $F_n \leftarrow \text{SelectFeatures}(F, Q_n)$
  if $\Delta > 0$ and $(A_n - 1) \leq \Delta$ **then**
    // Case 1: Fully merge the mask
    $\bar{f}_n \leftarrow \text{AverageFeatures}(F_n)$
    $d_n \leftarrow \text{Centroid}(Q_n)$
    Add token $(\bar{f}_n, d_n)$ to F_{merged}
    $\Delta \leftarrow \Delta - (A_n - 1)$
  else if $\Delta > 0$ and $(A_n - 1) > \Delta$ **then**
    // Case 2: Partially merge the mask
    $\bar{f} \leftarrow \text{AverageFeatures}(\text{the first } (\Delta + 1) \text{ tokens of } F_n)$
    $d \leftarrow \text{Centroid}(\text{the first } (\Delta + 1) \text{ tokens of } Q_n)$
    Add token $(\bar{f}, d)$ to F_{merged}
    Add the remaining $A_n - (\Delta + 1)$ tokens from F_n to F_{merged}
    $\Delta \leftarrow 0$
  else
    // Case 3: No budget left, keep all tokens
    Add all tokens from F_n (with their original positions) to F_{merged}
  end if
end for
// 3. Finalize Output
Sort F_{merged} by spatial position (original or averaged centroid)
$F_{\text{merged}} \leftarrow \text{StackFeatures}(F_{\text{merged}})$
return F_{merged}

C. Soft masks Visualization

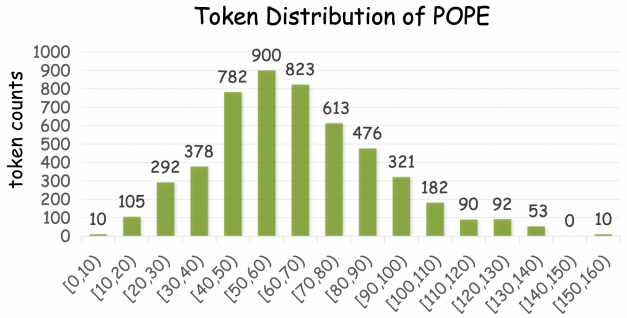
Presented below is the complete set of soft masks for the sample image.

Figure S1. **All Soft Masks.** We can see one object may have multiple soft masks.

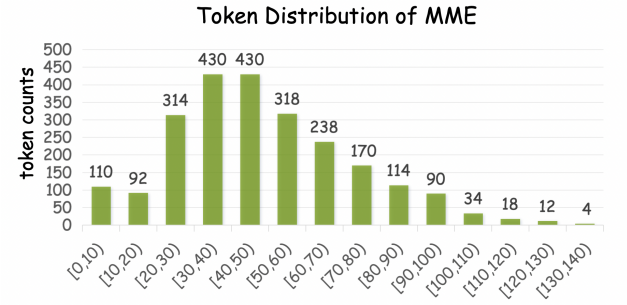
D. Token Distribution of Datasets

This section visually illustrates the detailed distribution of the number of tokens generated by the CORE model across multiple datasets when using its adaptive compression strat-

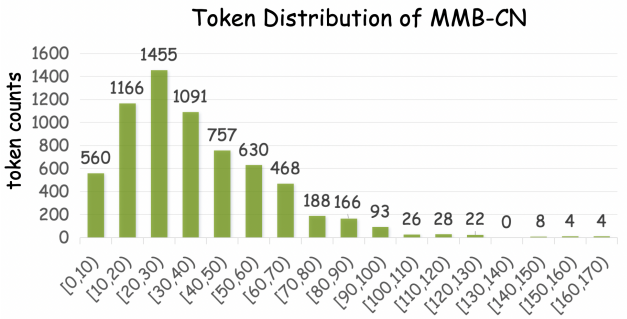
egy. These plots, which use token count intervals as the x-axis and image frequency as the y-axis, consistently exhibit a clear unimodal distribution. This indicates that the vast majority of images are compressed into a relatively concentrated range. This section demonstrates CORE's ability to dynamically adjust its compression rate based on the semantic complexity of the image content or the number of objects, rather than relying on a fixed token count.



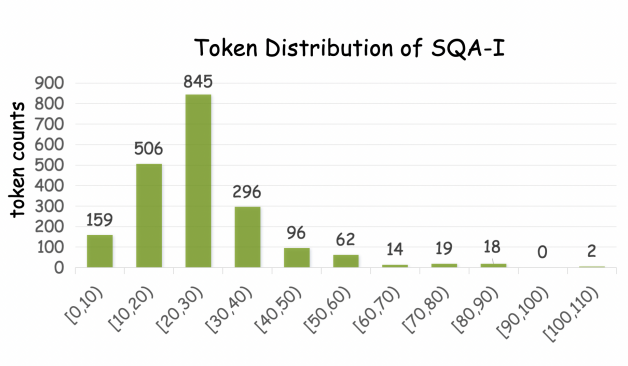
(a) Token count analysis on the POPE benchmark.



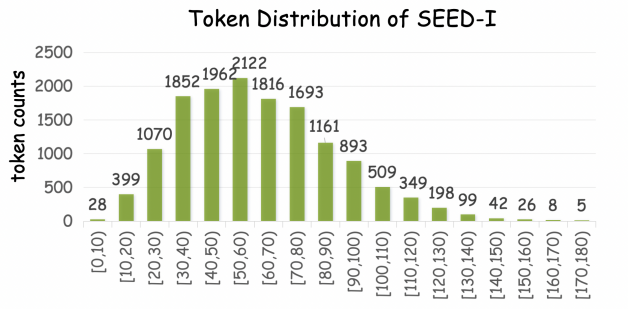
(b) Token count analysis on the MME benchmark.



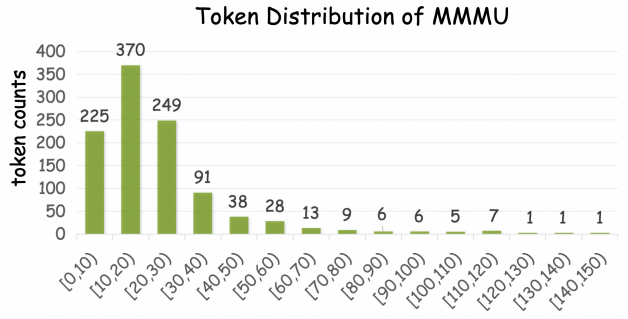
(c) Token count analysis on the MMB-CN benchmark.



(d) Token count analysis on the SQA-I benchmark.



(e) Token count analysis on the SEED-I benchmark.

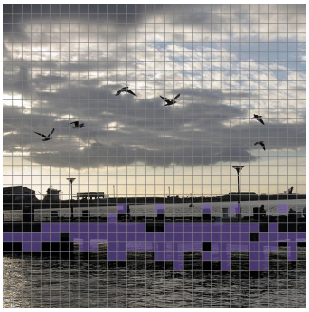


(f) Token count analysis on the MMMU benchmark.

Figure S2. Detailed token count analysis across the six evaluation benchmarks. Each subfigure shows the results for a specific dataset.

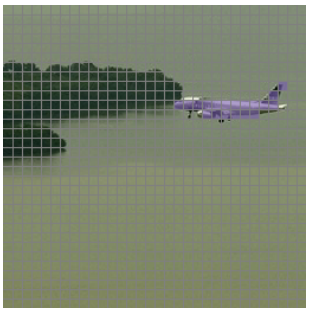
E. More Token Merging Examples

In this section, we give more token merging examples which include both things and stuff. Each caption briefly describes the corresponding object.



(a) Bridge

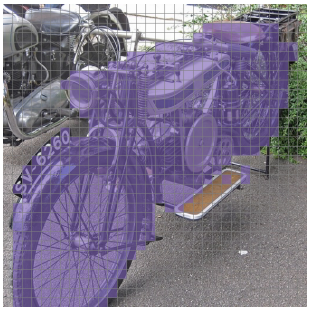
(b) Mirror



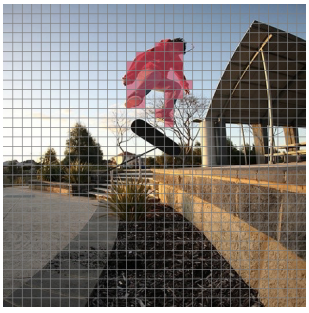
(i) Plane

(j) Bicycle

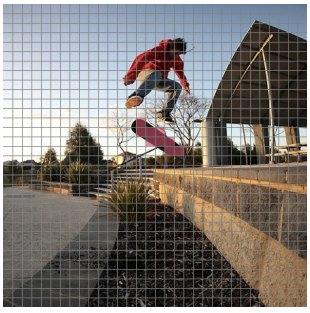
(c) Ground



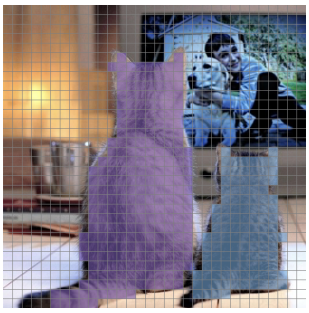
(d) Bicycle



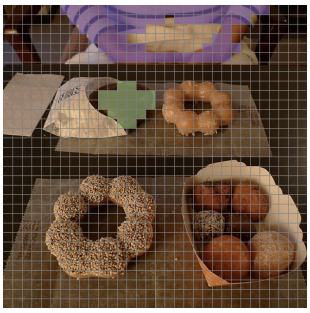
(k) Skateboarder



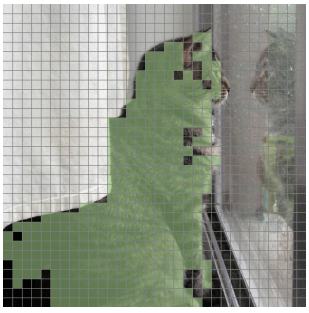
(l) Skateboard



(e) Two Cats



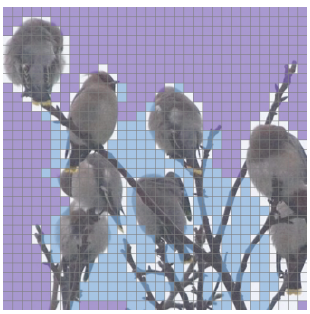
(f) Hands and Dessert



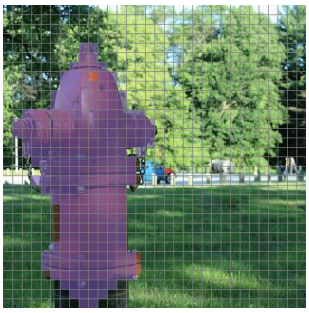
(m) Cat

(n) Cat's Reflection

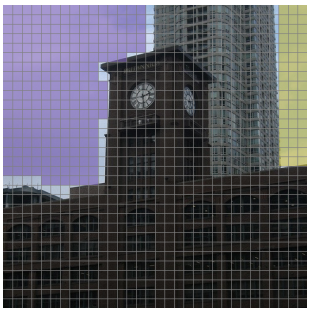
(g) Bottle and Glass



(h) Snowfield



(o) Hydrant



(p) Sky

Figure S3. **Token Merging Examples.** Tokens with the same color in an image are merged into one. A complex or blocked object may have more than one token after being merged. Stuff (e.g., sky, snowfield) is merged equally as things.

F. Dialogue Comparison

In this section, we compare our CORE model with another token comparison method, VisionZip [103], with the same number of retained tokens. We mainly examine the models' ability on background object recognition and objects' positional relationship detection. In Fig. S4 and Fig. S5, we also show the mask CORE generates internally as CORE's thinking process.

In Fig. S4, the image is that a large and a small cat are watching TV under a yellow light. CORE produces an object-centric representation for each object in the image, including those in the background, which leads to a correct answer (yellow light). In contrast, other methods may aggressively prune necessary tokens, resulting in an incorrect description (blue light). This comparison demonstrates CORE's comprehensiveness in object recognition, even for background elements.

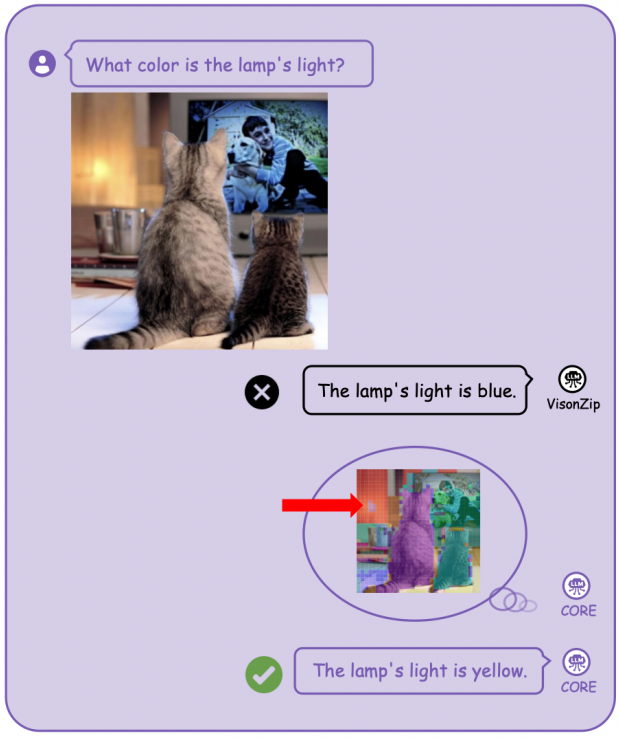


Figure S4. **Comparison on Background Object Recognition.** The red arrow emphasizes the object-centric token of the lamp, which helps CORE arrive at the correct answer.

Fig. S5 gives a comparison of different models' description on objects' positional relationship. The scene is from a movie and it shows the character riding a giant bird over the water. VisionZip gives an incorrect answer that the bird is on the character's shoulder. However, based on CORE's centroid-guided sorting strategy, the character's tokens are prior to the bird's. As a result, our model infers that the character rides the bird and gives the correct description. The thinking process shows the complete segmentation mask.

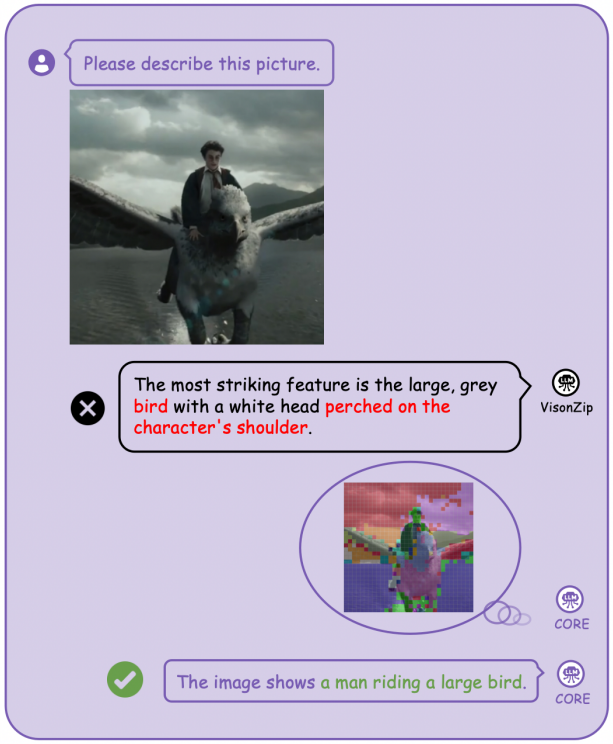


Figure S5. **Comparison on Objects' Positional Relationship Detection.** Incorrect and correct model responses are highlighted in red and green, respectively.

G. Long-tail Objects Concern

Tab. S2 summarizes the statistics under adaptive compression. The results indicate that, even for out-of-distribution (OOD) objects, CORE maintains an accuracy that is comparable to that of in-distribution (ID) objects. In fact, while the classification head is limited to 133 categories, the mask head and pixel decoder are class-agnostic. They rely on generic visual cues to achieve generalizable segmentation. Besides, Mask2Former performs panoptic segmentation and CORE could segment all semantically similar objects in both foreground and background.

Table S2. Statistics of ID and OOD objects in MME Dataset

	Count	Acc.
ID (within 133 COCO classes)	240	72.08%
OOD (beyond 133 COCO classes)	2134	68.70%