Supplementary Material

6. Implementation Details

For the training of the textual decoder of the memory-based configuration ϕ, we adopt a prefix GPT2-style decoder-only Transformer with 4 attention heads and 4 layers, following the architecture used by [33]. We train the model on captions from the COCO training set, which also serves as the memory bank M for the projection mechanism, comprising approximately 500k texts. We set the hyperparameters of the projection mechanism as in DeCap ($\tau = 0.01$), and use the AdamW optimizer with a weight decay of 0.01. Training proceeds for 10 epochs with a learning rate of 10^{-5} and a batch size of 64. A comprehensive overview of the framework operating in the DeCap setting (with the projection mechanism as modality gap mitigation strategy) is provided in Figure 4.

For the external knowledge-based captioning models we performed a training of 15 epochs with a batch size of 80 captions on the same GPT2-style textual decoder using a learning rate of 2×10^{-5} and a gaussian noise variance of 16×10^{-3} to replicate the experimental settings of [15] and [68].

All experiments were conducted on a single NVIDIA H100 GPU with 80GB of HBM3 memory. Training took approximately 25 minutes per epoch.

7. Backbone Details

We briefly summarize here the characteristics of the vision-language models we tested in our framework.

- **CLIP** [51]: A foundational model that learns a shared embedding space for images and text through contrastive learning. While being the most used model for global image-text alignment, its patch tokens are known to lack strong spatial and fine-grained semantic information. The input resolution of its training is 224 pixel.
- **DenseCLIP** [53]: A fine-tuned version of CLIP that incorporates a pixel-text matching loss to enhance the model's ability to understand local regions. The official implementation input resolution is 640 pixel for the ViT-B/16 version.
- **INViTE** [8]: This method modifies CLIP's vision transformer to bring patch tokens in the text space by disabling the self-attention mechanism. It employs the same visual encoder of CLIP, trained at 224 pixel input resolution.
- **ProxyCLIP** [29]: A model that leverages the local understanding of a DINO backbone to improve CLIP's patch-level representations. It achieves this by replacing the attention maps in CLIP's final layer with DINO's attention maps, effectively transferring DINO's fine-grained spatial awareness to the CLIP embedding space. The DINO ViT-B/8 version was tested with images at 296 pixel resolution, while the DINOv2 ViT-B/14 at 518 pixel.
- **SigLIP2** [60]: A multilingual vision–language model that improves upon CLIP by enhancing both global alignment and dense localization capabilities through refined training objectives and architectural adjustments. We employed the B/16 variant, which uses a ViT with 16-pixel patches and is trained at an input resolution of 512 pixels. To obtain text-aligned semantic patch representations, we follow the authors' dense feature extraction strategy and apply the MAP (Mean Attention Pooling) head—normally used to produce the global image representation embedding—individually to each patch token.
- **DINO.txt** [25]: This model builds upon a frozen DINOv2 backbone, adding learnable transformer blocks on top. It is then trained with a contrastive objective against a text encoder to align both global and patch-level representations with language. The DINOv2 backbone was trained at the resolution of 518 pixel.
- **Talk2DINO** [5]: This model creates a bridge between the CLIP and DINOv2 embedding spaces. It trains a projection to map CLIP text embeddings into the DINOv2 patch space, using DINOv2's highly meaningful attention maps to identify and align with the most relevant patches during training. The DINOv2 backbone was trained at the resolution of 518 pixel.

8. Comparison with LMMs

Our framework is designed as a zero-shot regional captioner, relying solely on text-only corpora to train the decoder, and operating without any paired image-text or region-text supervision. While most of the Large Multi-modal Models (LMMs) are trained without region-text supervision, they are trained on massive datasets using image-text pairs, making them inherently supervised solutions.

While this difference in training paradigm places LMMs outside the core assumptions of our zero-shot setup, we include a comparison with representative, high-performing LMMs — Llava1.5 OneVision (4B) [1], Qwen2.5 VL [3] (3B) and Qwen3 VL (4B) — to provide a strong, external supervised benchmark for our framework's capabilities.

8.1. LMM Adaptation for Regional Tasks

LMMs typically process a whole image and cannot natively process explicit regional coordinates (such as bounding boxes or traces) as additional inputs, unless specifically trained with region-level box or mask annotations. To enable a fair comparison of our regional tasks, we propose two zero-shot adaptation strategies to inject regional information into the LMM.

- **Visual Prompting:** The regional annotations (bounding boxes or traces) are drawn over the image, allowing the model to condition its generation on the spatial input. We then ask the model to describe the annotated image.

Captioning Task: (Dataset)	Adaptation strategy	Trace (COCO)		Dense (VG v1.2)		Region-Set (COCO Entities)		Image (COCO)		
		C	P	C	P	C	P	C	P	CLIP-S
LMMs										
Llava1.5 One-Vision (4B)	Visual Prompting	21.2	75.9	25.3	76.2	72.2	87.3	**91.6**	**91.1**	80.1
Qwen2.5 VL (3B)	Visual Prompting	17.4	74.0	19.1	74.2	59.6	84.5	77.9	89.9	81.2
Qwen3 VL (4B)	Visual Prompting	9.1	70.7	10.8	74.2	10.8	74.2	19.4	84.2	**85.3**
Llava1.5 One-Vision (4B)	Crop	19.7	75.4	15.1	72.7	75.9	86.7	**91.6**	**91.1**	80.1
Qwen2.5 VL (3B)	Crop	18.4	73.9	11.9	69.2	68.6	85.9	77.9	89.9	81.2
Qwen3 VL (4B)	Crop	7.6	69.3	3.5	68.8	20.2	80.4	19.4	84.2	**85.3**
Patch-ioner (Our Patch-based Framework)										
T2D + Mem. (0.21B)		**27.9**	**78.7**	**31.9**	**78.8**	**109.1**	**87.5**	69.2	87.4	72.8

Table 3. **Patch-ioner framework vs. LMMs.**

- **Cropping:** The image is cropped to the bounding box encompassing the regional annotation (be it a trace, a box, or a set of boxes). The cropped image is then fed to the LMM, forcing the model to focus its attention solely on the region of interest.

An example of adaptation for each task with the respective prompt is shown in Figure 3.

8.2. Results Analysis

Table 3 presents the comparison results of our Patch-ioner framework against state-of-the-art LMMs across various captioning granularities.

In the finer-level tasks, specifically Dense Captioning (VG v1.2) and the Trace Captioning (COCO), the pronounced performance gap between our framework and LMMs highlights the fundamental inability of LMMs to effectively reason at the precise region level based solely on a visual annotation or a simple crop without sufficient global context. For these granular tasks, the Visual Prompting strategy is generally preferable for LMMs because it preserves the essential context of the entire image. Furthermore, simple cropping is problematic for concave traces as it fails to precisely delineate the intended region of the image.

In tasks involving broader regions, such as Region-Set Captioning (COCO Entities), the Patch-ioner framework achieves better performance despite LMMs showing closer results, highlighting our higher capability to generate captions specifically guided by the set of input regions. For these broader regions, the Cropping adaptation strategy generally leads to better LMM performance than Visual Prompting, suggesting that Region-Set Captioning demands a lower level of context from the uncropped parts of the image with respect to Dense and Trace Captioning.

For the Image Captioning task, which relies on global understanding, LMMs generally demonstrate superior performance compared to our zero-shot captioner baseline. This is largely expected, as LMMs are bigger and extensively trained on massive image-text pairs.

8.3. Comparison Highlights

The comparison with powerful LMM baselines adapted for regional tasks clearly underscores the unique advantages of the Patch-ioner framework in terms of design, data requirements, parameter efficiency, and inference speed.

Designed for Granularity The fundamental difference lies in the design objective: our Patch-ioner framework is explicitly engineered for multi-granularity, region-level captioning , while LMMs are built primarily for global image understanding. The need for ad-hoc adaptation strategies for LMMs (Visual Prompting or Cropping) inherently limits their precision, resulting in significantly lower performance on fine-grained regional tasks –as shown in Table 3– compared to our method.

Data and Training Efficiency Unlike LMMs, which are trained on massive paired image-text datasets and are therefore inherently supervised solutions, our approach adheres to a strict zero-shot paradigm. The text decoder is trained solely on text corpora, eliminating the dependency on costly, large-scale image-text supervision for training the generative module.

Parameter and Computational Efficiency The Patch-ioner framework is dramatically more lightweight and parameter-efficient than LMMs. We provide a compact model (e.g., 0.21B parameters for the Talk2DINO configuration), contrasting sharply with the multi-billion parameter counts typical of LMMs (e.g., 3B to 4B parameters in the tested models).

Inference Speed and Scalability One of our framework's most critical advantages is its inference efficiency for multiple regions within a single image. For an image containing multiple annotations (such as in the Dense Captioning task, which can have over a hundred boxes), we require only a single forward pass of the frozen vision backbone to extract all patch features. These pre-computed patch features can then be reused to caption arbitrary regions. Conversely, an LMM adapted through Cropping or Visual Prompting requires a full inference of the vision-language model for every single annotation, leading to dramatically slower performance when scaling to dense or complex regional tasks

9. Ablations on Text Decoder Architectures

We tested different architectures and sizes for the text decoder network. In particular, we trained GPT-2 small [50], Gemma 3 270M [58], Qwen 3 0.6B [64], LLaMa 3.2 1B [17] and Qwen 3 1.7B [64]. Details on training hyperparameters are provided in §6. The dataset adopted for these trainings is made of the ground truth captions of COCO train Karpathy split. The total number of different textual captions for that split is 566,747. Table 4 reports the scores obtained when varying the decoder in our framework.

Across all tasks, we observe that increasing the capacity of the textual decoder does not translate into consistent improvements. GPT-2 small (124M) performs surprisingly strongly, outperforming larger models on Trace and Region-Set captioning in terms of CIDEr, while maintaining competitive RefPAC-S scores. Larger decoders such as Gemma 3 (270M), Qwen3 (0.6B), and Llama 3.2 (1B) yield similar or slightly worse results, and the largest tested model, Qwen3 1.7B, exhibits the lowest performance on all CIDEr metrics. These findings suggest that, under our training regime and dataset size (COCO train Karpathy split, 566k captions), model capacity is not the limiting factor. Larger decoders tend to overfit more easily and provide little benefit in a setting where the captioning supervision is narrow in distribution and relatively small compared to their pretraining scale. Moreover, since our overall architecture relies on a strong visual encoder capable of providing semantic local representations, the decoder's primary role is to map visual representations to fluent text; in this context, compact autoregressive models appear sufficiently expressive. Interestingly, RefPAC-S shows minimal variation across decoder sizes, indicating that semantic alignment with the image features (rather than surface-level text quality) is largely preserved even with larger models. However, the trend of decreasing CIDEr with increasing model size suggests that bigger decoders may introduce unnecessary linguistic diversity that hurts match-based metrics. Overall, these results highlight that changing the decoder affects only marginally the performances.

10. Ablations on Text-only Training Dataset

In this section, we compare the results obtained when using the classical training strategy adopted by the models such as [33, 43, 68] in the zero-shot captioning task — which consists of training for 10 epochs on the collection of texts from the COCO dataset — with training on a different dataset, to assess generalization capabilities of our framework.

For this comparison, we took the best Patch-ioner configuration — which consists of using Talk2DINO [5] as visual backbone and a memory bank of texts borrowed from COCO as in [33]— and we trained it on a subset of ReLaion.

10.1. Selected dataset: ReLaion 600M

ReLaion is a large-scale image-text dataset containing approximately 600 million image-caption pairs sourced from LAION-2B [55]. Each entry includes multiple machine-generated captions of varying quality, along with a "best caption" field obtained by ranking the alternatives using a CLIP-based scoring function. In our ablation we exclusively use this "best caption" attribute, since it provides a higher-quality textual signal while avoiding noisy or inconsistent descriptions that typically arise in large crawled datasets.

To ensure a fair comparison with the standard COCO-based training strategy from prior work [33, 43, 68], we adopt a subset of ReLaion sized so that one training epoch over that matches the number of training steps of the standard COCO training lasting 10 epochs. Since the COCO training split contains roughly 560k captions, one epoch on a 5.6M-sample subset of ReLaion results in approximately the same number of optimization steps, while a 28.3M-sample subset corresponds to five times more steps. In all experiments, we maintain the same Patch-ioner configuration (our best-performing model) and apply identical optimization settings.

This setup allows us to assess the generalization capability of our framework when trained on broader, noisier web-scale text compared to the highly curated COCO captions that are commonly used in zero-shot captioning pipelines.

10.2. Discussion

The results in Table 5 highlight three main trends.

(1) ReLaion training improves cross-dataset generalization. Training on ReLaion — even for a single epoch — consistently improves performance across captioning tasks compared to the classical COCO-only training. The 5.6M-sample setting yields clear gains on Trace and Dense Captioning, and the larger 28.3M subset further enhances Region-Set and Image Captioning scores. These improvements indicate

Figure 3. **LMM Adaptation Strategies for Zero-Shot Regional Captioning.** This figure visualizes the two input-adaptation strategies used to benchmark Large Multimodal Models (LMMs) on Region-level Captioning tasks: Trace, Dense, and Region-Set Captioning. **Top Row:** The original image corresponding to each task. **Middle Row (Visual Prompting):** The task region is spatially indicated by superimposed red annotations (traces or bounding boxes). The LMM input includes this visually prompted image along with a detailed instruction (shown below the images) to describe the marked region while suppressing any mention of the annotations. **Bottom Row (Cropping):** The input is a minimal crop tightly encompassing the annotated region. The LMM is given a general instruction (shown below the images) to describe the presented image, implicitly forcing focus onto the region of interest.

that Patch-ioner benefits substantially from the wider linguistic and visual variability captured in ReLaion, despite its noisier nature.

(2) Gains are not uniform across tasks. While the Trace and Dense tasks benefit the most from ReLaion (e.g., +2.4 CIDEr on Trace and +1.7 on Dense when moving from COCO to ReLaion 5.6M), the COCO-based Region-Set task exhibits a more nuanced behavior. The larger ReLaion sub-

Captioning Task: (Dataset)		Trace (COCO)		Dense (VG v1.2)		Region-Set (COCO Entities)		Image (COCO)		
Textual Decoder	# Parameters	C	P	C	P	C	P	C	P	CLIP-S
GPT2	124M	**27.9**	**78.7**	**31.9**	**78.8**	**109.1**	**87.5**	**69.2**	**87.4**	72.8
Gemma 3	270M	23.5	78.4	25.5	78.5	98.7	**87.5**	61.0	87.2	73.2
Qwen3	0.6B	23.5	78.4	25.0	78.4	98.7	**87.5**	59.7	87.2	73.4
LLama 3.2	1B	24.2	78.3	26.0	78.4	101.0	87.4	60.3	87.1	73.5
Qwen3	1.7B	22.9	78.2	24.5	78.2	95.6	87.4	57.9	87.1	**73.7**

Table 4. **Training different decoders.** CIDEr (C) and RefPAC-S (P) across four captioning tasks. The model adopted is T2D + Memory (≈ DeCap) trained on COCO train Karpathy split.

set (28.3M) significantly boosts Region-Set CIDEr, suggesting that recognizing localized entities and relations requires broader caption diversity, which becomes available only at larger scale.

(3) Memory Bank from ReLaion is less effective than the one from COCO. When employing 500k captions randomly sampled from ReLaion as memory bank, performance drops across many tasks. This suggests that using a collection of texts for the projection mechanism sampled from COCO leads the model to provide captions that are closer to the COCO ground-truth ones, particularly from a syntactic standpoint. In fact, we can notice how the larger performance drop is on the CIDEr metric.

Overall, these findings show that Patch-ioner adapts well to large-scale noisy text, outperforming COCO-trained baselines on most tasks, and that the advantages grow with the size of the ReLaion subset. This demonstrates that our framework can leverage broad web-scale text distributions to improve zero-shot captioning, even when training the textual decoder for a single epoch. However, we pick the model trained on COCO as reference to have a fair comparison with existing models.

11. Additional Results: Aggregation Strategies, Input Resolution, Text Collection

We tested several patch aggregation strategies and input resolutions for our model and the other baselines.

Patch Aggregation. In cases where we are not captioning a single patch, we test different aggregation functions for merging the $\mathbf{v}_i$ in a selected set S of visual patches:

a) **uniform**, the average box patch representations;
b) **gaussian**, for rectangular configurations of contiguous patches — i.e., either the full image or a bounding box; we consider a weighted average of patches representations where central patches weigh more; specifically, we assign to each patch (a, b) coordinates in a uniform square grid $[-1, 1]^2$ (i.e., the top-left and bottom-right patches have $(-1, -1)$ and $(1, 1)$ coordinates, respectively), and a weight of $e^{-(a^2+b^2)}$ in the average, and
c) **attention**, a weighted average of box patches representations, with patch weights defined as the average attention map of the last layer of ψ_{v}.

Input Resolution. For patch-based captioning, we followed Talk2DINO [5] and used an input image resolution of 518x518, obtaining 37 14x14 patches per side when using the Talk2DINO backbone. The original DeCap [33], ViECap [15], MeaCap [68] implementations uses the CLIP B/32 backbone with 224x224 input images with 7 patches per side. We also tested with the CLIP B/16 backbone, resulting in 14 patches per side at 224x224 resolution, and with 592x592 input image size, to obtain the same number of patches as in our framework (37 per side).

We report results of these additional configurations for all baselines: DeCap, ViECap, and MeaCap. While the main paper reports only the best configuration per task and model, in this section, we report and discuss the results of all the tested configurations. We perform these tests on COCO-derived datasets and on VG v1.2 for dense captioning. We highlight the rows in the tables corresponding to the configurations reported in the main paper.

Trace Captioning. Table 6 reports trace captioning results. We did not apply the *gaussian* weighting scheme for this task, as the sparse discontinuous traces often do not identify a rectangular region needed to apply this scheme. We notice that a) the simple average of the trace patches provides the best performance in our framework b) as expected, using the CLIP B16 backbone, that extracts finer patches, improves over the standard CLIP B32 backbone used in the baseline methods, and c) resolutions higher than 224 only marginally improve performance for baselines in this task.

Dense Captioning. Table 7 reports the results of the dense captioning task. For our framework, changing the weighting

Captioning Task: (Dataset)	Trace (COCO)		Dense (VG v1.2)		Region-Set (COCO Entities)		Image (COCO)		
Text-only Training Dataset	C	P	C	P	C	P	C	P	CLIP-S
COCO Train	27.9	78.7	31.9	78.8	109.1	87.5	69.2	87.4	72.8
ReLaion 5.6M	**30.3**	**79.0**	**34.1**	**79.0**	109.0	87.7	69.3	87.5	72.5
ReLaion 28.3M	29.7	**79.0**	33.6	78.9	**113.5**	**87.9**	**70.6**	**87.7**	**73.0**
Using a Memory Bank of 500k ReLaion Captions									
Relaion 5.6M	26.7	78.4	33.8	79.2	86.3	85.3	54.1	85.0	70.1
ReLaion 28.3M	26.8	78.3	33.4	79.0	86.6	85.3	54.6	85.0	70.5

Table 5. **Training on different datasets.** CIDEr (C) and RefPAC-S (P) across four captioning tasks. The model adopted is T2D + Memory ($\approx$ DeCap) using the GPT2 textual decoder.

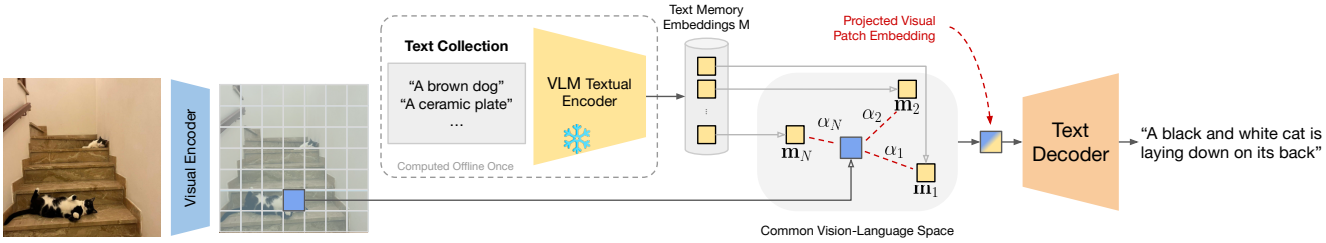

Figure 4. **Patch-level Captioning**. Given an input image, we first extract dense patch-level representations using a vision transformer backbone. For a selected patch, we apply the projection-based mechanism introduced by [33] to mitigate the modality gap and align its representation with the text embedding space. Finally, the transformed embedding is fed into a text decoder trained on a text-only corpus, generating a zero-shot caption for the patch.

strategy does not cause significant performance changes. The best baselines are the ones Region-based, which consist of applying the captioners to the CLS tokens of image crops specified by the bounding boxes (e.g., DeCap@224 Crop). The difference between the CLIP B/16 and B/32 versions is usually small or negligible.

Region-Set Captioning. Table 8 shows the results in the region-set captioning task on COCO Entities. The gap between zero-shot image captioners and our framework's captioners narrows in this task due to the more global nature of it, which requires the model to produce a caption for the whole image while focusing on certain regions. Also in this task, the choice of weighting strategy only marginally affects the performance of the models in our framework.

Image Captioning. In Table 9, we report the results of standard zero-shot image captioning. In addition to the already described weighting schemes, we test one additional configuration for our framework that is *central patch*, where the decoding is applied to the central patch of the image. We can observe that the most effective strategy for the image captioning task is *attention*. This is coherent with results from [5], where they suggest the attention-weighted patch means to use Talk2DINO for global tasks such as image-text retrieval.

Memory Bank. Considering that in the memory-based model of our framework (that is similar to [33]) we tackle the modality gap through a projection based on a collection of texts, we tested how much the selection of the texts in the memory bank influences performance. In Table 9, we also report the results obtained by that model when in its memory bank there are also ground-truth captions of the test set (rows marked with *GT Memory*). This provides a sort of upper bound to performance when varying the text collection used as memory. We observe that in this configuration, the performance only slightly improves (+0.5%), indicating that the model is robust to the choice of the memory bank.

12. Modality Gap: Projection to Textual Space vs Training with Noise

In this section, we quantitatively assess the performance of two state-of-the-art solutions to overcome the modality gap. In particular, we compared the configuration based on a memory bank of texts — the one introduced in §3 — with an alternative solution based on noise injection during the decoder training. Additionally, we include in our comparison a baseline with no modality gap mitigation (no mitig.), to highlight the benefits brought by each strategy.

Model	# Patches	Backbone	Input	Weighting	B	M	R	C	S	P
Image-based										
DeCap@224	7	CLIP B32	CLS	-	2.1	9.7	21.7	21.1	8.8	75.2
DeCap@224	14	CLIP B16	CLS	-	2.2	9.8	21.8	21.3	8.7	75.4
DeCap@592	37	CLIP B16	CLS	-	2.0	9.6	21.5	20.5	8.7	75.3
ViECap@224	7	CLIP B32	CLS	-	2.5	9.8	22.4	24.8	9.3	74.1
ViECap@224	14	CLIP B16	CLS	-	2.5	9.9	22.3	24.7	9.5	74.5
ViECap@592	37	CLIP B16	CLS	-	2.3	9.6	22.1	24.3	9.5	74.4
MeaCap@224	7	CLIP B32	CLS	-	2.3	9.4	21.5	23.1	8.9	74.2
MeaCap@224	14	CLIP B16	CLS	-	2.3	9.3	20.8	23.4	9.0	74.6
MeaCap@592	37	CLIP B16	CLS	-	2.2	9.1	20.3	22.5	9.0	74.4
Patch-based (Our Framework)										
T2D + Mem. ($\simeq$ DeCap) @518	37	DINOv2 B14	Patches	uniform	**2.5**	**10.7**	**23.2**	**27.9**	**12.6**	**78.7**
T2D + Mem. ($\simeq$ DeCap) @518	37	DINOv2 B14	Patches	attention	2.4	10.4	22.7	27.6	12.0	78.1

Table 6. Trace Captioning results on COCO test set.

Model	# Patches	Backbone	Input	Weighting	mAP	M	B	R	C	S	P
Image-based											
DeCap@224	7	CLIP B32	CLS	-	0.15	8.40	0.94	15.61	19.38	9.38	73.71
DeCap@224	14	CLIP B16	CLS	-	0.14	8.48	0.95	15.70	19.11	9.40	73.94
DeCap@592	37	CLIP B16	CLS	-	0.15	8.37	0.92	15.67	18.53	9.26	73.91
ViECap@224	7	CLIP B32	CLS	-	0.13	8.25	1.02	16.06	24.18	9.97	73.03
ViECap@224	14	CLIP B16	CLS	-	0.14	8.30	1.01	15.86	23.81	9.91	73.49
ViECap@592	37	CLIP B16	CLS	-	0.14	8.17	1.00	15.86	23.26	9.82	73.35
MeaCap@224	7	CLIP B32	CLS	-	0.13	8.04	0.98	15.40	23.22	9.66	72.97
MeaCap@224	14	CLIP B16	CLS	-	0.13	8.01	1.03	15.15	23.37	9.49	73.55
MeaCap@592	37	CLIP B16	CLS	-	0.13	7.86	0.99	15.13	22.77	9.43	73.50
Region-based											
DeCap@224 Crop	7	CLIP B32	CLS	-	0.17	10.03	1.35	18.20	23.61	10.90	77.09
DeCap@224 Crop	14	CLIP B16	CLS	-	0.18	10.33	1.40	18.44	24.56	11.28	77.76
DeCap@592 Crop	37	CLIP B16	CLS	-	0.14	8.30	1.05	16.39	17.20	7.78	75.47
ViECap@224 Crop	7	CLIP B32	CLS	-	0.15	9.32	1.42	17.79	26.40	10.07	74.34
ViECap@224 Crop	14	CLIP B16	CLS	-	0.16	9.59	1.46	18.03	27.13	10.43	75.62
ViECap@592 Crop	37	CLIP B16	CLS	-	0.12	7.83	1.14	16.02	20.20	7.44	73.26
MeaCap@224 Crop	7	CLIP B32	CLS	-	0.15	9.64	1.46	18.00	28.62	10.98	75.08
MeaCap@224 Crop	14	CLIP B16	CLS	-	0.16	10.03	1.57	18.45	30.53	11.51	76.35
MeaCap@592 Crop	37	CLIP B16	CLS	-	0.12	7.93	1.19	16.17	21.32	7.86	73.69
Patch-based (Our Framework)											
T2D + Mem. ($\simeq$ DeCap) @518	37	DINOv2 B14	Patches	uniform	0.21	10.63	1.36	18.59	31.94	15.03	78.82
T2D + Mem. ($\simeq$ DeCap) @518	37	DINOv2 B14	Patches	gaussian	**0.22**	**10.82**	**1.43**	**18.82**	**32.80**	**15.48**	**79.14**
T2D + Mem. ($\simeq$ DeCap) @518	37	DINOv2 B14	Patches	attention	0.21	10.31	1.27	18.17	30.58	14.72	78.69

Table 7. Dense Captioning results on VG v1.2 test set.

Training with Noise. Various works [18, 43] proposed zero-shot image captioning solutions based on noise injection during the training of the text decoder. Through this strategy, the trained decoders are more effective in understanding semantic representations, even when those are not coming from the text modality. To implement this strategy in our framework, we trained the textual decoder on the same collection of captions as for the memory bank-based configuration. We adopted Talk2DINO [5] textual space for the decoder input space, which is aligned to DINOv2 [44] with registers [24]. Following the setting of [18], we added Gaussian noise with $\sigma^2 = 0.08$ to the textual embeddings while leaving the other parameters unchanged (as defined in §6). In the next paragraphs, we report and compare the results for each task of Talk2DINO within our framework with the memory bank ($\mathcal{M}emory$) and with the training with noise ($\mathcal{N}oise$).

In Table 10, we compare the two modality gap mitiga-

Model	# Patches	Backbone	Input	Weighting	B	M	R	C	S	P
Image-based										
DeCap@224	7	CLIP B32	CLS	-	10.1	19.0	38.0	94.4	26.4	86.9
DeCap@224	14	CLIP B16	CLS	-	10.0	19.4	38.3	95.1	26.8	87.4
DeCap@592	37	CLIP B16	CLS	-	9.6	18.6	37.5	91.4	25.9	86.7
ViECap@224	7	CLIP B32	CLS	-	11.2	18.2	38.9	102.7	27.0	85.0
ViECap@224	14	CLIP B16	CLS	-	11.3	18.3	38.6	102.2	26.9	85.4
ViECap@592	37	CLIP B16	CLS	-	10.8	17.8	37.9	99.2	26.5	85.0
MeaCap@224	7	CLIP B32	CLS	-	10.4	17.7	37.0	97.9	25.9	85.2
MeaCap@224	14	CLIP B16	CLS	-	10.1	17.5	35.5	96.5	25.7	85.4
MeaCap@592	37	CLIP B16	CLS	-	9.3	16.9	34.6	91.1	25.5	85.0
Patch-based (Our Framework)										
T2D + Mem. ($\simeq$ DeCap) @518	37	DINOv2 B14	CLS	-	9.1	16.9	35.0	89.4	25.4	85.5
T2D + Mem. ($\simeq$ DeCap) @518	37	DINOv2 B14	Patches	uniform	11.5	19.3	38.8	109.1	29.4	87.5
T2D + Mem. ($\simeq$ DeCap) @518	37	DINOv2 B14	Patches	gaussian	**11.6**	**19.6**	**39.3**	**111.6**	**30.1**	**87.7**
T2D + Mem. ($\simeq$ DeCap) @518	37	DINOv2 B14	Patches	attention	11.0	19.0	38.3	107.0	29.3	87.4

Table 8. Region-Set Captioning results for COCO Entities test set.

Model	# Patches	Backbone	Input	Weighting	B	M	R	C	S	P
Image-based										
DeCap@224	7	CLIP B32	CLS	-	23.46	25.12	50.06	87.40	19.14	90.58
DeCap@224	14	CLIP B16	CLS	-	**23.89**	**25.51**	**50.34**	**89.64**	**19.52**	**91.05**
DeCap@592	37	CLIP B16	CLS	-	22.43	24.64	49.25	84.57	18.66	90.36
ViECap @224	7	CLIP B32	CLS	-	26.70	23.99	50.85	89.67	17.54	88.45
ViECap@224	14	CLIP B16	CLS	-	26.3	24.0	50.3	89.5	17.6	88.8
ViECap @592	37	CLIP B16	CLS	-	25.60	23.38	49.52	86.84	17.08	88.33
MeaCap@224	7	CLIP B32	CLS	-	24.57	23.12	47.68	86.66	17.27	88.76
MeaCap@224	14	CLIP B16	CLS	-	23.6	22.7	45.5	85.1	17.3	89.0
MeaCap@592	37	CLIP B16	CLS	-	22.01	21.87	44.72	80.84	16.69	88.41
Patch-based (Our Framework)										
T2D + Mem. ($\simeq$ DeCap) @518	37	DINOv2 B14	Patches	central patch	15.68	18.46	40.84	55.53	12.66	84.26
T2D + Mem. ($\simeq$ DeCap) @518	37	DINOv2 B14	Patches	uniform	19.52	21.49	44.88	69.19	15.59	87.36
T2D + Mem. ($\simeq$ DeCap) @518	37	DINOv2 B14	Patches	gaussian	21.17	22.62	46.62	76.79	16.73	88.36
T2D + Mem. ($\simeq$ DeCap) @518	37	DINOv2 B14	Patches	attention	23.64	23.93	48.54	88.46	18.21	90.21
T2D + Mem. ($\simeq$ DeCap) @518 GT Memory	37	DINOv2 B14	CLS	-	23.58	23.54	47.71	85.67	17.86	89.53
T2D + Mem. ($\simeq$ DeCap) @518 GT Memory	37	DINOv2 B14	Patches	attention	25.66	24.77	49.83	93.87	19.09	90.70

Table 9. Image Captioning results on COCO test set.

tion strategies across multiple captioning tasks, and also report the performance of a baseline without any mitigation (no mitig.). The baseline consistently underperforms compared to both the $\mathcal{M}emory$ and $\mathcal{N}oise$ configurations, indicating that, like other contrastively learned image-text encoders [36], Talk2DINO is also affected by the modality gap. These results highlight the importance of explicitly addressing this gap to achieve strong captioning performance. For Trace Captioning, the $\mathcal{M}emory$ method is slightly more effective in the semantic metric RefPAC-S, while the $\mathcal{N}oise$ variant achieves marginally better scores in CIDEr, ROUGE-L, METEOR, and BLEU@4, with a minimal gap between the two approaches. In Dense Captioning, the $\mathcal{M}emory$ model consistently outperforms the $\mathcal{N}oise$ model across all metrics. Similarly, for Region-Set Captioning, both methods achieve strong results, but the $\mathcal{M}emory$ method shows a clearer advantage, particularly in tasks closer to the patch level. Finally, in Image Captioning, the performance gap between the two architectures narrows, especially on the Flickr30k test split. In this scenario, the $\mathcal{M}emory$ method performs significantly better when applied to the CLS token, whereas patch aggregation produces comparable results. However, the metrics reveal conflicting trends across different datasets.

Chosen Strategy. Based on the observed results, we selected the projection-based approach ($\mathcal{M}emory$) as the primary strategy for overcoming the modality gap in our framework. While the noise injection method ($\mathcal{N}oise$) yielded competitive performance across multiple tasks, the $\mathcal{M}emory$

Table 10. **Mitigation of Modality Gap.** Comparison of Memory-based Projection ($\mathcal{M}emory$) vs Noise-trained Decoder ($\mathcal{N}oise$) across tasks.

	Trace Captioning (COCO)						Dense Captioning (VG v1.2)							Region-Set Captioning (COCO Entities)						Image Captioning (COCO)						
Mitigation	B	M	R	C	S	P	mAP	M	B	R	C	S	P	B	M	R	C	S	P	B	M	R	C	S	P	CLIP-S
no mitig.	1.2	9.1	18.3	14.7	8.5	75.1	0.18	9.7	0.7	15.9	17.8	10.2	75.2	5.0	15.0	29.4	59.4	21.1	82.2	9.9	17.7	36.8	43.7	12.3	82.2	69.6
$\mathcal{N}oise$	**3.0**	**11.5**	**24.7**	**29.3**	12.3	78.1	0.20	10.4	1.2	17.8	26.3	12.6	77.0	10.5	18.4	37.2	97.5	26.7	85.6	19.6	21.5	45.4	65.5	15.5	86.2	70.9
$\mathcal{M}emory$	2.5	10.7	23.2	27.9	**12.6**	**78.7**	0.21	10.6	1.4	18.6	31.9	15.0	78.8	11.5	19.3	38.8	109.1	29.4	87.5	19.5	21.5	44.9	69.2	15.6	87.4	72.8

method demonstrated superior performance in dense captioning and region-set captioning, as well as a clear advantage when applied to the CLS token in image captioning. Given these trends, and considering the stability of the projection-based approach across different evaluation settings, we adopted $\mathcal{M}emory$ as the default configuration for our framework.

13. Trace Captioning Benchmark Generation

We construct our Trace Captioning dataset from the Localized Narratives dataset [48]. This dataset consists of mouse traces and their corresponding speech transcriptions, where annotators describe objects in images while moving the mouse pointer over them.

The initial dataset samples include timestamped mouse traces and are composed of multiple sentences that thoroughly describe the trace, with the generated descriptions following the order of the mouse movement. However, our task does not require strict temporal coherence. Instead, we aim to generate a single, concise caption that describes the specific area covered by the localized trace, rather than a multi-sentence description.

To achieve this, we split the descriptions into individual sentences and align the traces accordingly. We then refine the traces by removing intermediate periods caused by transitions between sentences, which often occur when the annotator moves to a different region of the image. Specifically, we trim each trace by removing the first and last 15% of points, eliminating these transitional segments.

Furthermore, we refine the captions by prompting the Llama3 8B model to rephrase the sentences, removing vague or subjective phrases such as "there is," "we can see," or "on the left of the image," and replacing them with concise, objective descriptions that refer specifically to the region covered by the trace. This rephrasing is crucial to ensure that each caption adheres to the standard format of image-captioning datasets and focuses only on the precise part of the image that the trace corresponds to. The LLM also helps identify and remove irrelevant sentences (e.g., "the image is blurred," "the image is edited"), which are then discarded along with their associated traces from the final benchmark.

Figure 5 shows the full prompt used to guide the Llama model in refining and cleaning the descriptions. Figure 6 illustrates how the initial narrative samples are transformed into final trace captioning samples through the process of trace splitting and caption rephrasing.

14. Learned Patch Aggregation via Attention

In the main paper we employ a parameter-free aggregation strategy to combine patch embeddings belonging to a region. While this choice preserves the zero-shot nature of our framework, we additionally explored whether a lightweight learned aggregation module could further improve the quality of region representations when local supervision is available.

Attention-based aggregation. Let $S = \{v_i\}_{i=1}^{N}$, with $v_i \in \mathbb{R}^D$, denote the set of patch embeddings corresponding to a selected region. In the default formulation of our framework, the region representation is obtained through mean aggregation:

$$v_{\text{mean}} = \frac{1}{N} \sum_{i=1}^{N} v_i. \tag{3}$$

To investigate a learned alternative, we introduce a lightweight attention-based aggregation module that summarizes the patch set through a single transformer-style attention layer. The patch embeddings are processed jointly and produce a summary token that attends over the region patches. Formally, given the matrix of patch embeddings $V \in \mathbb{R}^{N \times D}$, we compute

$$v_{\text{att}} = \text{Attn}(V), \tag{4}$$

where $\text{Attn}(\cdot)$ denotes a self-attention block that outputs a single `[CLS]` token representing the aggregated region information.

Rather than replacing the mean representation, we combine the two signals to preserve the stability of the parameter-free aggregation while allowing the model to learn refinements. The final region embedding is defined as

$$v_R = v_{\text{mean}} + \alpha\, v_{\text{att}}, \tag{5}$$

where α is a learnable scalar controlling the contribution of the attention-based summary. The parameter α is initialized to 0.1, ensuring that the aggregation initially behaves close to the mean operator and gradually learns to incorporate the attention-based refinement during training.

```
I have image descriptions derived from spoken narratives. These need to be rewritten as concise,
↪  stand-alone captions in the style of the image-caption datasets. Follow these rules:

- Remove unnecessary narrative phrases like "we can see," "there is," "in this image," etc.
- Ensure the caption is standalone and descriptive.
- Use simple, objective language that highlights key elements.
- Keep it concise--just a single phrase.
- Follow the classical style of caption datasets.
- If the description is vague, subjective, or does not describe a concrete visual element (e.g., "The
↪  image is taken indoor," "This image is blurred"), return `<INVALID>`.
- Wrap the output in `{}` and add nothing else.

### **Examples:**
- **Input:** "We can see a young elephant stands which is near the water in a wooded area."
  **Output:** {A young elephant stands near the water in a wooded area.}

- **Input:** "In this image I can see some young children kicking a soccer ball in a field."
  **Output:** {A group of young children kicking a soccer ball around a field.}

- **Input:** "In the left of the image, we see a pole that has two green street signs on it."
  **Output:** {A pole has two green street signs on it.}

- **Input:** "We can see two surfboards which are stuck in the sand along the seashore."
  **Output:** {Two surfboards stuck in the sand along the seashore.}

- **Input:** "This image consists of a man which rides a wakeboard behind a boat."
  **Output:** {A man rides a wakeboard behind a boat.}

- **Input:** "In the background, there are a bunch of sticky notes and a pair of scissors."
  **Output:** {A bunch of sticky notes and a pair of scissors.}

- **Input:** "It looks like a sepia-toned photograph of a motorcycle underneath the shadow of a
tree."
  **Output:** {A sepia-toned photograph of a motorcycle underneath the shadow of a tree.}

- **Input:** "There is a sky"
  **Output:** {A sky.}

- **Input:** "She is smiling."
  **Output:** {A smiling girl.}

- **Input:** "The image is taken indoor."
  **Output:** {<INVALID>}

- **Input:** "This image is edited."
  **Output:** {<INVALID>}

- **Input:** "The image is blurred."
  **Output:** {<INVALID>}

- **Input:** "I think he is about to jump."
  **Output:** {<INVALID>}

Now, rewrite the following captions accordingly. Wrap each in `{}` and add nothing else:
<INPUT CAPTION>
```

Figure 5. **LLM Prompt for rephrasing trace captions**.

Training setup. The attention-based aggregator is trained with region-level supervision for a single epoch using the COCO Trace Captioning training set derived from the LocNar COCO train split. We optimize the parameters of the attention layer and the scalar weight α using AdamW with learning rate 10^{-4}, while keeping the visual backbone and the text decoder frozen.

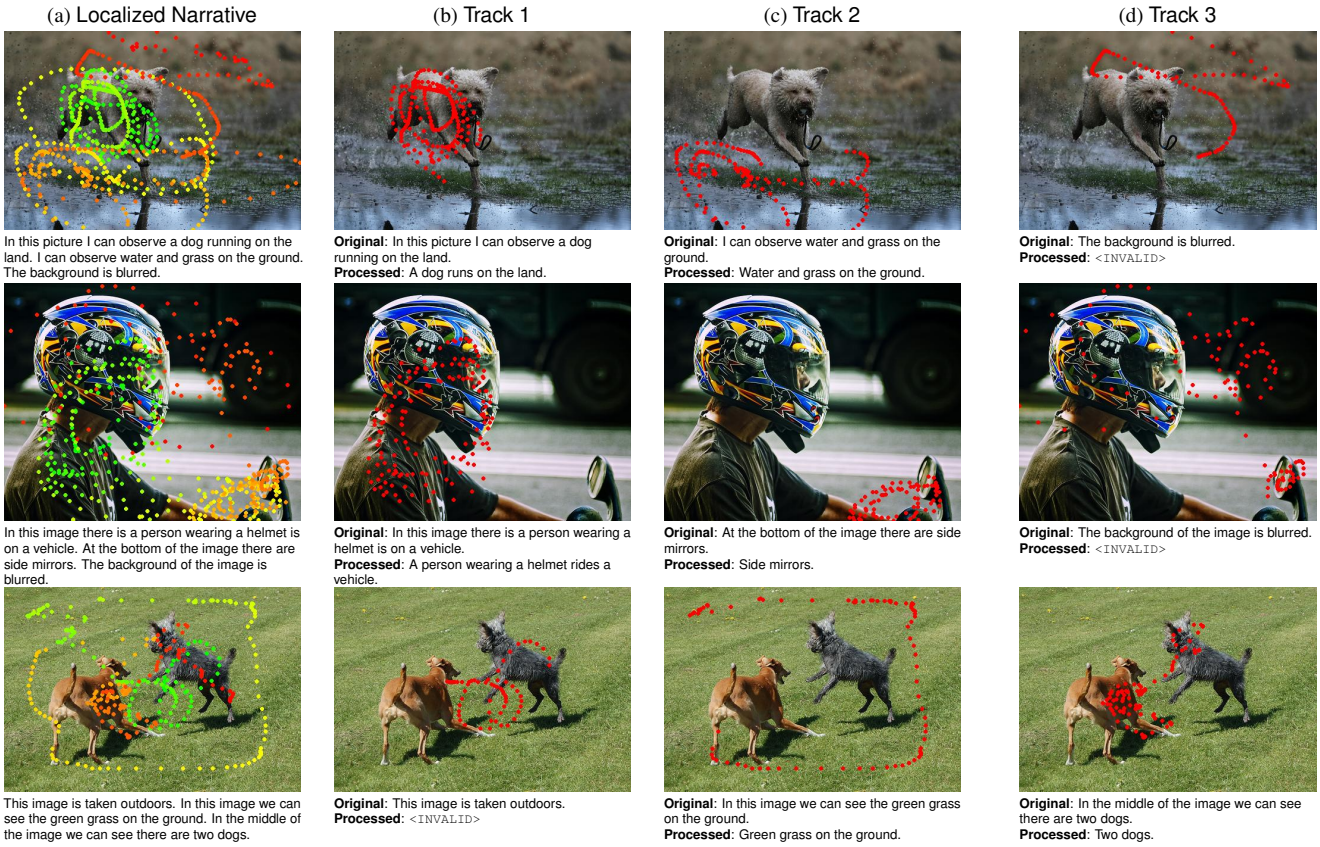

Figure 6. **Narrative vs. Trace Samples**. The first column displays sample images from the Localized Narrative dataset [49]. The remaining three columns show the corresponding mouse traces, along with the captions generated by the LLM. Captions marked with <INVALID> are removed from the dataset.

	Trace Captioning				Dense Captioning						Region-Set Captioning				Image Captioning					
	COCO		Flickr30k		VG v1.2			VG-COCO			COCO Entities		Flickr30k Entities		COCO			Flickr30k		
Model	C	P	C	P	mAP	C	P	mAP	C	P	C	P	C	P	C	P	CLIP-S	C	P	CLIP-S
Learned Aggregation (T2D + Noise)	**73.9**	**81.9**	**34.8**	**77.2**	17.8	**49.4**	**81.7**	17.7	**48.9**	**81.7**	67.8	83.6	25.7	73.3	**74.2**	86.5	70.9	27.3	79.7	66.0
Fixed Aggregation (T2D + Noise)	29.3	78.1	19.3	75.6	**20.3**	26.3	77.0	**20.3**	26.4	76.9	**97.5**	**85.6**	**37.1**	**76.5**	65.5	**86.2**	**70.9**	**27.8**	**80.8**	**67.0**

Table 11. Learned vs. Fixed Aggregation

Discussion. Table 11 reports the results of this experiment. The learned aggregator yields substantial improvements when evaluated on tasks with the same spatial granularity used during training (trace and dense captioning). However, these gains do not consistently transfer to tasks involving different region granularities (e.g., region-set or image captioning), where the fixed aggregation remains more balanced.

These findings suggest that when task-specific local supervision is available, learned aggregation can effectively refine region representations. However, in a unified zero-shot framework where no region-level annotations are assumed, the parameter-free aggregation strategy remains preferable due to its robustness across captioning granularities.

15. More Qualitative Results

Additional qualitative results are shown in Figures 7 and 8. Note that the first rows of Figures 7 and 8 contain also qualitative results for single patch captioning, for which we do not have annotated data to report quantitative results.

As can be noticed in Figures 7 and 8, the Region-Set Captioning task tends to align more closely with image-level captioning rather than strictly focusing on localized regions. This is expected since the ground-truth captions in the COCO Entities dataset originate from the image-level annotations of COCO, as stated in [10].

Figure 7. **Qualitative results.** We report four predictions of our model and compare baselines from the finer (top) to the coarser (bottom) task. For trace captioning examples, the trace time is color-coded from start (red) to end (yellow). **DeCap** = DeCap applied on the whole image. **DeCap (Crop)** = DeCap applied on cropped box. **ZeroCap** = ZeroCap [59] applied to the whole image. **CLOSE** = CLOSE [18] applied to the whole image. **Ours (CLIP + Mem.)** = Our patch-based framework using CLIP as backbone and the projection as modality gap mitigation strategy. **Ours (Talk2DINO + Mem.)** = Our patch-based framework using Talk2DINO as backbone and the projection as modality gap mitigation strategy. **GT** = ground-truth caption.

Figure 8. **Qualitative results.** We report four predictions of our model and compare baselines from the finer (top) to the coarser (bottom) task. For trace captioning examples, the trace time is color-coded from start (red) to end (yellow). **DeCap** = DeCap applied on the whole image. **DeCap (Crop)** = DeCap applied on cropped box. **ZeroCap** = ZeroCap [59] applied to the whole image. **CLOSE** = CLOSE [18] applied to the whole image. **Ours (CLIP + Mem.)** = Our patch-based framework using CLIP as backbone and the projection as modality gap mitigation strategy. **Ours (Talk2DINO + Mem.)** = Our patch-based framework using Talk2DINO as backbone and the projection as modality gap mitigation strategy. **GT** = ground-truth caption.