FUSE: FINE-TUNED UNIFIED SEMANTIC EMBEDDINGS FOR VISION

ETHAN YU [ETHANYU@SEAS], KEVIN LIU [KLIU2360@SEAS], ARYAN ROY [ARYANROY@SEAS],

ABSTRACT. In this work, we propose a novel method to augment vision foundation models like DINOv2 with the semantic capabilities of CLIP by fine-tuning DINO to output embeddings that are smaller in dimension, language-conditioned, and semantically meaningful. Our method introduces a lightweight output head to the pretrained DINO model, effectively aligning DINO's representations with CLIP's text-conditioned features. This integration enhances vision tasks, including instance retrieval, depth estimation, semantic segmentation, and zero-shot learning. For instance retrieval, DINO's embeddings are fine-tuned with CLIP's image-text embeddings for aligned structural and semantic representation. Depth estimation employs global CLS-based features and localized patch details to predict depth maps, while segmentation uses DINO's patch embeddings for precise pixel-level accuracy. CLIP's multimodal embeddings enable zero-shot task generalization. By aligning DINO's hierarchical features with CLIP's semantic context, the framework bridges self-supervised learning and multimodal applications, improving both spatial precision and contextual understanding.

1. INTRODUCTION

In recent years, there has been a significant shift towards the development of large, pre-trained models that generalize well across a wide range of tasks. These foundation models, such as OpenAI's GPT [6] for natural language processing and Stable Diffusion [5] for generative image tasks, have demonstrated remarkable performance on diverse downstream applications. Their massive architectures enable them to learn generalized representations from vast datasets, unlocking the potential to transfer knowledge across tasks with minimal task-specific training [2]. Despite their impressive capabilities, these models often remain opaque in their inner workings, raising challenges in interpretability and adaptability across specialized domains.

This versatility has proven particularly compelling in the field of computer vision, where achieving robust and accurate performance across tasks like instance retrieval, depth estimation, semantic segmentation, and zero-shot learning remains a core challenge. Traditionally, addressing these tasks required developing specialized models or employing approaches tailored to specific data types or representations. In response, vision foundation models, such as DINOv2 [3], have emerged as a unified approach to tackling these diverse tasks. These models are trained on large-scale image datasets using self-supervised techniques, learning representations that excel across numerous downstream vision applications.

However, as real-world applications become increasingly complex, there is a growing demand for models that can integrate multiple modalities of information, such as language and vision, in a seamless and efficient manner. CLIP [7], introduced in 2021, was a groundbreaking effort to bridge vision and language by training on paired image-text data. While CLIP remains impressive, its vision performance now lags behind state-of-the-art (SOTA) vision foundation models on many purely visual tasks [2]. This raises the question: how can we combine the strengths of specialized vision models with the semantic richness of multimodal models like CLIP without training a multimodal foundation model from scratch?

1.1. Contributions.

Our work makes the following key contributions:

- We introduce a lightweight output head to DINOv2, aligning its structural embeddings with CLIP's semantic features, enabling language-conditioned representations.
- **Instance Retrieval:** Achieved noticeable improvements over the baseline, particularly in long-range retrieval (*Accuracy@100*) and overall ranking quality (mAP), leveraging CLIP's semantic insights.
- **Depth Estimation:** Maintained comparable performance to the baseline with minimal accuracy loss, achieving a 33% reduction in embedding dimensionality without significant sacrifices in inference time or quality.
- Demonstrated the potential of integrating vision foundation models with multimodal representations to enhance downstream tasks such as segmentation and zero-shot learning.

2. Background

Recent advancements in machine learning have focused on combining self-supervised learning and multimodal models to improve vision system accuracy, robustness, and versatility. Self-supervised learning techniques like DINO (Distillation of Knowledge from Self-Supervised Learning) excel at extracting hierarchical and structural representations from visual data without requiring large labeled datasets. DINO generates embeddings that capture both global and local image features, making it effective for tasks requiring detailed spatial understanding.

Multimodal models such as CLIP (Contrastive Language-Image Pretraining) align vision and language by learning joint embeddings from paired image-text data, enabling impressive zero-shot performance across various tasks. However, CLIP struggles with purely visual tasks like depth estimation and segmentation due to its focus on language-vision alignment rather than visual precision.

This work aims to combine the strengths of DINO and CLIP by aligning their embedding spaces. The goal is to enhance DINO's structural embeddings with semantic insights from CLIP and explore whether language can help reduce the dimensionality of vision representations without sacrificing task performance. This is done by adding an auxiliary output head to DINO, fine-tuned with a mix of self-supervised and multimodal losses. The approach is evaluated by benchmarking against two DINO baselines on downstream tasks.

- **Instance Retrieval:** The task of retrieving similar instances from a large dataset based on a query image. This requires both fine-grained visual features (e.g., structural details) and semantic understanding (e.g., context or object meaning).
- **Depth Estimation:** The challenge of estimating the 3D structure of a scene from a 2D image, requiring both global scene understanding (augmented by CLIP) and localized details (captured by patch-based information).

By integrating CLIP's semantic conditioning into DINO's output, we aim to bridge the gap between self-supervised and multimodal vision systems, unlocking new possibilities for compact and efficient vision-language fusion.

3. Related Work

SAM-CLIP:

The authors of the SAM-CLIP paper [9] introduce a unified model combining the strengths of SAM (Segment Anything Model) [10] and CLIP to address tasks requiring both semantic and spatial understanding. SAM excels in spatial localization and instance segmentation, while CLIP offers powerful semantic understanding through image-text associations. To merge these models effectively, the authors use multi-task training, and knowledge distillation, significantly reducing computational costs to less than 10% of traditional methods. Their approach not only retains the individual zero-shot capabilities of SAM and CLIP but also introduces a new use-case zero-shot semantic segmentation, enabling segmentation based on free-form text prompts. The SAM-CLIP model achieves state-of-the-art performance in zero-shot semantic segmentation, outperforming previous models by +6.8% mIoU on Pascal VOC and +5.9% mIoU on COCO-Stuff, and demonstrates improved feature representations, making it suitable for diverse downstream tasks.

This work bridges the gap between training-free model merging and computationally intensive multi-task learning, and serves as one of the main inspirations for our project.

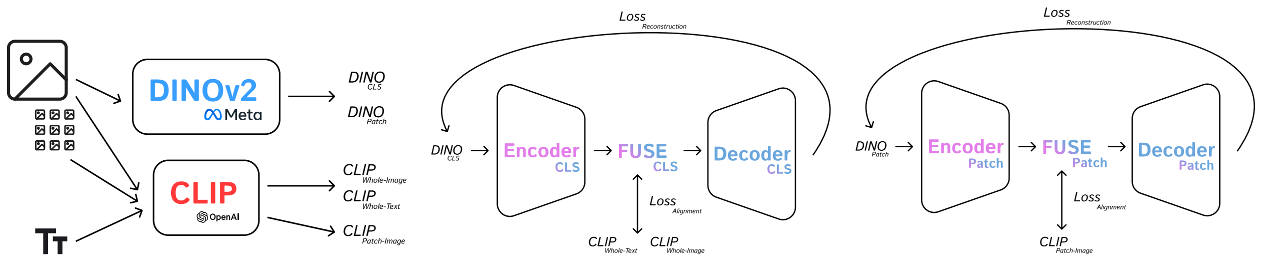
4. Approach

In this section, we describe our proposed method to efficiently integrate the structural embeddings of DINO with the semantic richness of CLIP, enabling improved instance retrieval and depth estimation while reducing the embedding dimensionality from 768 to 512. Our approach leverages two autoencoders trained on DINO embeddings with alignment losses derived from CLIP. The architecture and training process are described in detail below.

4.1. Architecture.

Our method consists of two key components:

(1) **CLS Autoencoder:** An autoencoder designed to operate on DINO's class (CLS) embedding, which captures global image features. The encoder takes as input the CLS embedding, processes it through a series of layers, and outputs a compressed representation. The decoder reconstructs the original CLS embedding from this compressed latent space.
(2) **Patch Autoencoder:** An autoencoder designed for DINO's patch embeddings, which capture localized features from image patches. The encoder processes individual patch embeddings into a compressed representation, and the decoder reconstructs the original patch embeddings.



(A) Feature Extraction Process (B) ClS Embedding Autoencoder (C) Patch Embedding Autoencoder

FIGURE 1. Model Approach

Both autoencoders are trained with a reconstruction loss on their respective embeddings, while additional alignment losses with CLIP embeddings introduce semantic alignment (details below). We just use layer-2 MLP for this section, with the patch encoder not having the final nonlinearity la

4.2. **Training Objectives.** We are trying to minimize 3 losses that can be further separated into subparts of image and patch. First is reconstruction loss which measures how well the models' reconstructed inputs from the latent space match the original inputs. Next, we have the CLIP visual losses which compare the image feature embeddings with the latent representations from the autoencoder encoding. This loss aims to align the visual feature representations from the 2 foundation models. In practice for this CLIP visual patch loss, we generate 256 14 by 14 patches and then use `torch.interpolate` to up-sample to 224 by 224 which is expected CLIP input. Finally, we have a CLIP text loss that compares the textual embedding from CLIP with the latent representation from the autoencoder, aiming to align visual and text feature representations.

1. Reconstruction Loss (CLS):

$$\mathcal{L}_{\text{rec}}^{\text{cls}} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2$$

where x_i are the original CLS tokens and $\hat{x_i}$ are the reconstructed CLS tokens.

2. Reconstruction Loss (Patch):

$$\mathcal{L}_{\text{rec}}^{\text{patch}} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{p}_i - \hat{\mathbf{p}}_i\|_2^2$$

where p_i are the original CLS tokens and $\hat{p_i}$ are the reconstructed CLS tokens.

3. CLIP Image Loss (CLS):

$$\mathcal{L}_{\text{clip\_image}}^{\text{cls}} = 1 - \frac{\mathbf{z}_{\text{cls}} \cdot \mathbf{z}_{\text{image}}}{\|\mathbf{z}_{\text{cls}}\|_2 \|\mathbf{z}_{\text{image}}\|_2}$$

where $\mathbf{z}_{\text{cls}}$ is the latent representation of the CLS token and $\mathbf{z}_{\text{image}}$ is the CLIP image embedding.

4. CLIP Image Loss (Patch):

$$\mathcal{L}_{\text{clip\_image}}^{\text{patch}} = 1 - \frac{\mathbf{z}_{\text{patch}} \cdot \mathbf{z}_{\text{image\_embed}}}{\|\mathbf{z}_{\text{patch}}\|_2 \|\mathbf{z}_{\text{image\_embed}}\|_2}$$

where $\mathbf{z}_{\text{patch}}$ is the latent representation of the patch tokens and $\mathbf{z}_{\text{image}}$ is the CLIP image embedding.

5. CLIP Text Loss (CLS):

$$\mathcal{L}_{\text{clip\_text}}^{\text{cls}} = 1 - \frac{\mathbf{z}_{\text{cls}} \cdot \mathbf{z}_{\text{text}}}{\|\mathbf{z}_{\text{cls}}\|_2 \|\mathbf{z}_{\text{text}}\|_2}$$

where $\mathbf{z}_{\text{patch}}$ is the latent representation of the patch tokens and $\mathbf{z}_{\text{text}}$ is the CLIP text embedding.

Our final loss function is the weighted sum of these 5 loss terms based on coefficients we define in `training_config.yaml`.

Total Loss Function:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{clip\_text}} \mathcal{L}_{\text{clip\_text}}^{\text{cls}} + \lambda_{\text{cls\_rec}} \mathcal{L}_{\text{rec}}^{\text{cls}} + \lambda_{\text{clip\_image}} \mathcal{L}_{\text{clip\_image}}^{\text{cls}} + \lambda_{\text{patch\_rec}} \mathcal{L}_{\text{rec}}^{\text{patch}} + \lambda_{\text{patch\_clip}} \mathcal{L}_{\text{clip\_image}}^{\text{patch}}$$

4.3. Data Preprocessing.

- **DINO Baseline**

  To evaluate the effectiveness of our proposed FUSE alignment approach, we establish a comprehensive baseline using DINO's pre-trained self-supervised embeddings.

  For the instance retrieval task, we use DINO's pretrained weights to extract CLS and patch embeddings, which capture global and local structural features of images. These embeddings are indexed using FAISS [11], a scalable similarity search library, with L2 distance as the similarity metric. Queries are then matched against this index to retrieve visually similar images from a large dataset. The baseline evaluation leverages the COCO 2017 dataset, which provides a diverse set of images to test retrieval performance under various conditions. Retrieval accuracy is quantified using key metrics, such as Accuracy@K (the proportion of images of the same class retrieved out of K) and Mean Average Precision (average accuracy across all queries).

  For the depth estimation task, we build on DINO's capabilities by adding a regression head atop its global CLS embeddings to predict pixel-wise depth maps. The baseline model is trained on the NYU Depth V2 dataset, which includes detailed depth annotations paired with RGB images. During training, the regression head is optimized using the Mean Squared Error (MSE) loss function, with images and depth maps resized to 224x224 to match DINO's input requirements. Training is performed over 20 epochs using the Adam optimizer, with an initial learning rate of 0.03 that is halved every five epochs via a learning rate scheduler. Depth estimation accuracy is evaluated using metrics such as Mean Absolute Error (MAE) and Root Mean Square Error (RMSE).
- **FUSE**

  We use the COCO 2017 dataset [8] for training our FUSE model. The entire COCO training dataset consists of 118k images with their corresponding captions. For text-image pairs with multiple captions, one caption is randomly selected as the "gold" standard per iteration. Training is done on a Google colab A100 GPU. With a batch size of 2, the FUSE model takes half the 40 GB GPU RAM. We acknowlege that this likely leads to to very erratic gradient updates. Training runs for 3 epochs for approximately 8,500 weight updates. A 5% subset of the COCO dataset is sampled and used a training data. The training runs for 10 epochs with a batch size of 8, constrained by the GPU RAM limit of the A100 40 GB card used on Colab. A learning rate of 0.0001 is employed. While the setup effectively handles the task, higher computational resources—unfortunately limited by AWS's vCPU provisioning—could have enabled more extensive exploration of the CLIP forward pass, likely yielding more impressive results. An examination of the loss curve further shows that the loss was still trending down.

5. Experimental Results

5.1. Instance Retrieval. The results in Table 1 (Appendix) indicate that FUSE generally outperforms the Baseline, with performance margin increasing as we increase the number of retrieved images to 100. However for Accuracy@50, the FUSE model significantly outperforms the baseline. We choose to use Accuracy@k as a metric because it indicates how often the correct item appears within the top-k retrieved items. Our model also has a higher mean average precision than the baseline which indicates that FUSE is not just good at retrieving the right items in the top few ranks, but it also performs well across a broader range of retrieval positions. This means that, on average, it is better at ensuring relevant items appear higher in the list, even when the retrieval set expands.

Accuracy@K measures how many of the returned images are relevant, making it crucial for assessing a model's effectiveness in practical retrieval tasks. mAP, on the other hand, provides a more comprehensive measure of performance. It averages precision at different recall levels, offering a more balanced evaluation of a model's retrieval capability over multiple queries. Combining Accuracy@k and mAP helps assess both the ranking quality and overall retrieval effectiveness. For images showcasing the difference between baseline and FUSE, see figures 2 and 3 in the Appendix.

5.2. Depth Estimation. The depth estimation results (See Table 1 in Appendix) show that FUSE performs comparably to the baseline model, with only marginal differences across metrics. The Mean Absolute Error (MAE) shows a slight increase from 0.186 to 0.189, while the Root Mean Square Error (RMSE) remains constant at 0.226, indicating that both models have similar overall prediction accuracy. For the threshold accuracy metrics (δ), which measure the proportion of pixels where the ratio between predicted and ground truth depth falls within specific thresholds, we observe small decreases in performance. The 1.25 metric decreases from 0.274 to 0.259, and 1.5625 shows a similar slight decline from 0.524 to 0.518. Interestingly, the 1.953125 metric shows nearly identical performance, with the baseline at 0.693 and FUSE at 0.695. These results suggest that while our FUSE approach successfully maintains most of the depth estimation capability of the baseline model, the addition of CLIP's semantic features and dimensionality reduction to 512 dimensions introduces a small trade-off in depth prediction accuracy.

5.3. **Training Loss.** In the appendix is a plot of losses over about 8500 weight updates. The CLS reconstruction is the noisiest loss. Total loss is steadily decreasing over epochs and the plot patch reconstruction loss and cls reconstruction loss show no signs of having converged. Total loss at the end of the 3 epochs were $1.938, 1.661, 1.525$ respectively.

6. Discussion

6.1. **Instance Retrieval.** Our results highlight the potential and limitations of aligning DINO's structural understanding with CLIP's semantic insights through an autoencoder-based approach. For instance retrieval, the FUSE model exhibits:

- **Improved Performance:** Significant gains over the baseline in long-range retrieval (*Accuracy@100*) and overall ranking quality (mAP). These improvements suggest that CLIP's semantic information aids in distinguishing visually similar but semantically distinct images.
- **Challenges in Fine-Grained Retrieval:** A slight degradation in performance at *Accuracy@50*, indicating that the model may prioritize semantic similarity at the expense of fine-grained visual details. This trade-off reflects the delicate balance between structural and semantic information, consistent with findings in the SAM-CLIP paper.

6.2. **Depth Estimation.** Depth estimation reveals an interesting efficiency-performance trade-off:

- **Spatial Information Loss:** Compressing DINO's embeddings from 768 to 512 dimensions led to small but consistent accuracy reductions, despite the semantic conditioning from CLIP.
- **Preserved Performance:** Minimal differences across metrics, including identical RMSE values, indicate that our method retains most of the depth estimation capabilities while achieving a 33% reduction in embedding dimensionality.

These findings align with cross-modal variational alignment literature, where dimensionality reduction typically improves efficiency at the cost of slight performance drops.

6.3. **Future Work.** Several areas for improvement were identified:

- **Training Enhancements:** Due to GPU memory constraints, small batch sizes were used, likely resulting in noisy gradient updates. Loss curves suggest incomplete convergence after three epochs, indicating potential for better performance with extended training.
- **Architectural Improvements:** Introducing attention mechanisms or skip connections in the autoencoder could help preserve fine-grained visual details while maintaining semantic alignment.
- **Loss Weighting Exploration:** Adjusting the weighting of reconstruction and alignment loss terms could enhance the final representation quality.
- **Expanded Evaluation:** Evaluating on additional vision tasks, such as zero-shot classification or semantic segmentation, would provide a more comprehensive understanding of the method's capabilities.
- **Location-Conditioned Autoencoder:** The patch autoencoder currently is agnostic to the location of the input patch - further work could investigate passing as input the patch location as well.

References

[1] T. Theodoridis, T. Chatzis, V. Solachidis, K. Dimitropoulos, and P. Daras, “Cross-modal Variational Alignment of Latent Spaces,” in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Seattle, WA, USA: IEEE, Jun. 2020, pp. 4127–4136. doi: 10.1109/CVPRW50498.2020.00488.

[2] G. Zhou, H. Pan, Y. LeCun, and L. Pinto, “DINO-WM: World Models on Pre-trained Visual Features enable Zero-shot Planning,” Nov. 07, 2024, arXiv: arXiv:2411.04983. doi: 10.48550/arXiv.2411.04983.

[3] M. Oquab et al., “DINOv2: Learning Robust Visual Features without Supervision,” Feb. 02, 2024, arXiv: arXiv:2304.07193. doi: 10.48550/arXiv.2304.07193.

[4] E. Latif and X. Zhai, “Fine-tuning ChatGPT for automatic scoring,” Computers and Education: Artificial Intelligence, vol. 6, p. 100210, Jun. 2024, doi: 10.1016/j.caeai.2024.100210.

[5] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, “High-Resolution Image Synthesis with Latent Diffusion Models,” Apr. 13, 2022, arXiv: arXiv:2112.10752. doi: 10.48550/arXiv.2112.10752.

[6] T. B. Brown et al., “Language Models are Few-Shot Learners,” Jul. 22, 2020, arXiv: arXiv:2005.14165. doi: 10.48550/arXiv.2005.14165.

[7] A. Radford et al., “Learning Transferable Visual Models From Natural Language Supervision,” Feb. 26, 2021, arXiv: arXiv:2103.00020. doi: 10.48550/arXiv.2103.00020.

[8] T.-Y. Lin et al., “Microsoft COCO: Common Objects in Context,” Feb. 21, 2015, arXiv: arXiv:1405.0312. doi: 10.48550/arXiv.1405.0312.

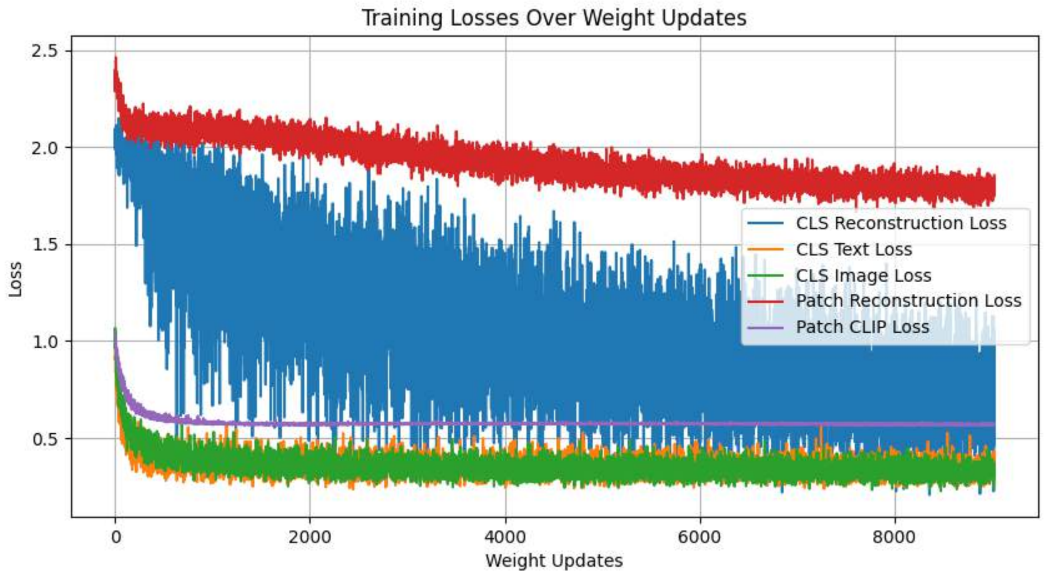
[9] H. Wang et al., “SAM-CLIP: Merging Vision Foundation Models towards Semantic and Spatial Understanding,” Jun. 10, 2024, arXiv: arXiv:2310.15308. doi: 10.48550/arXiv.2310.15308.

[10] A. Kirillov et al., “Segment Anything,” Apr. 05, 2023, arXiv: arXiv:2304.02643. doi: 10.48550/arXiv.2304.02643.

[11] M. Douze et al., “The Faiss library,” Sep. 06, 2024, arXiv: arXiv:2401.08281. doi: 10.48550/arXiv.2401.08281.

[12] H. Yang, Y. Zhang, J. Xu, H. Lu, P. A. Heng, and W. Lam, “Unveiling the Generalization Power of Fine-Tuned Large Language Models,” Mar. 14, 2024, arXiv: arXiv:2403.09162. doi: 10.48550/arXiv.2403.09162.

APPENDIX



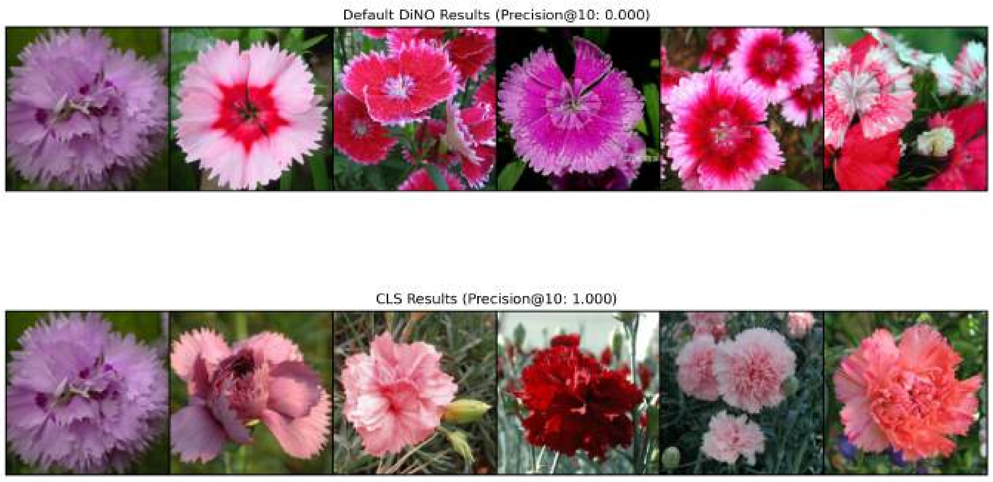

Default DiNO Results (Precision@10: 0.000)
CLS Results (Precision@10: 1.000)

FIGURE 2. Baseline vs DINO + CLIP for a sample image

Metric	**FUSE**	**Baseline**
Accuracy@1	1.000	1.000
Accuracy@5	**0.998**	0.990
Accuracy@10	**0.998**	0.988
Accuracy@10	**0.990**	0.985
Accuracy@50	0.953	**0.961**
Accuracy@100	**0.934**	0.898
mAP	**0.998**	0.898

TABLE 1. Comparison of instance retrieval metrics before and after modification. Bold values indicate better performance.

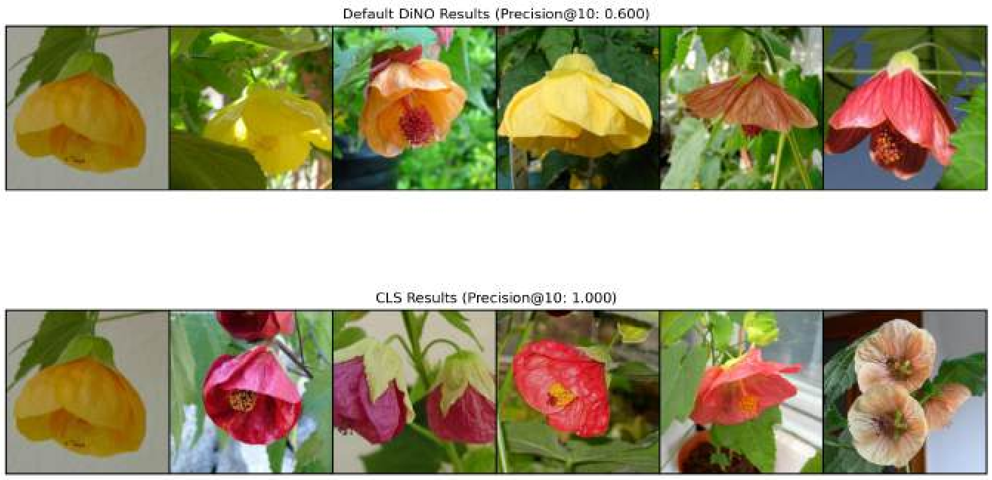

Default DiNO Results (Precision@10: 0.600)
CLS Results (Precision@10: 1.000)

FIGURE 3. Baseline vs DINO + CLIP for a sample image

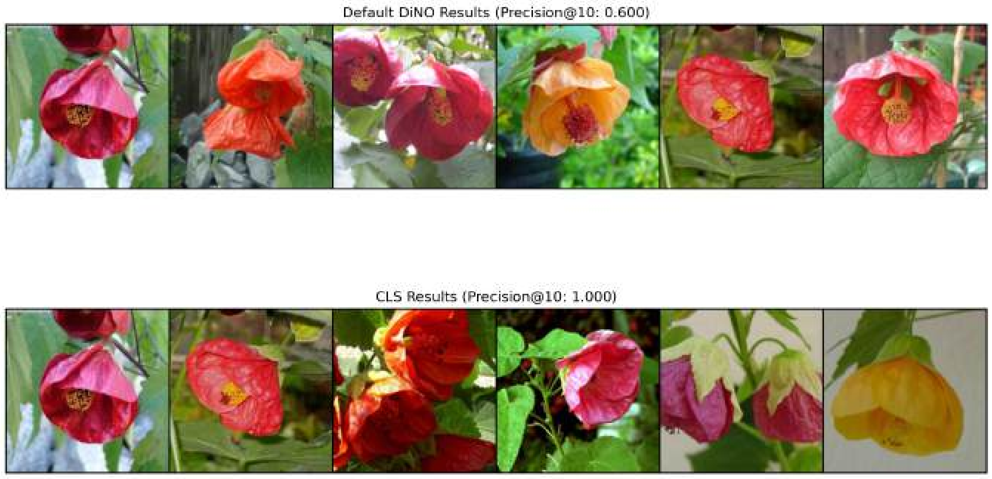

Default DiNO Results (Precision@10: 0.600)
CLS Results (Precision@10: 1.000)

FIGURE 4. Baseline vs DINO + CLIP for a sample image

Metric	**Baseline**	**FUSE**
MAE	**0.186**	0.189
RMSE	0.226	0.226
$\delta_{1.25}$	**0.274**	0.259
$\delta_{1.5625}$	**0.524**	0.518
$\delta_{1.953125}$	**0.693**	0.695

TABLE 2. Comparison of depth estimation metrics before and after modification. Bold values indicate better performance.

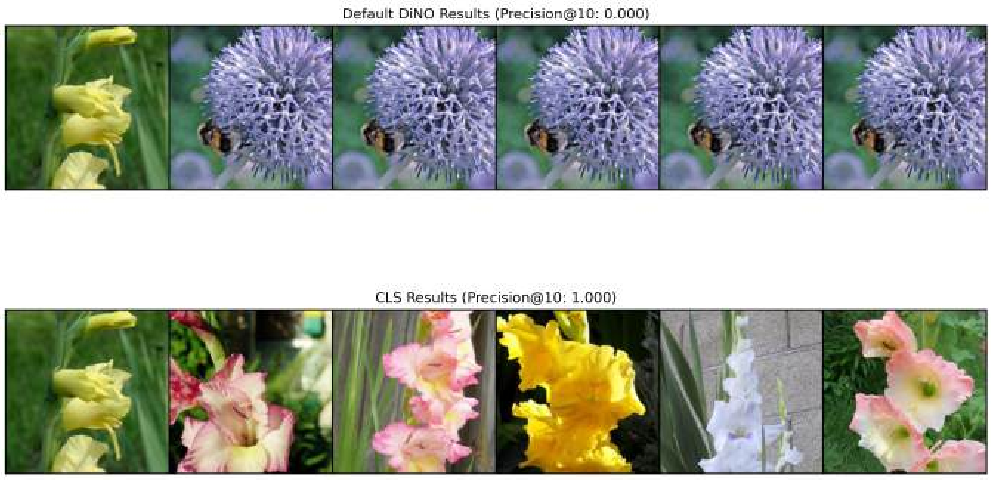

Default DiNO Results (Precision@10: 0.000)
CLS Results (Precision@10: 1.000)

FIGURE 5. Baseline vs DINO + CLIP for a sample image

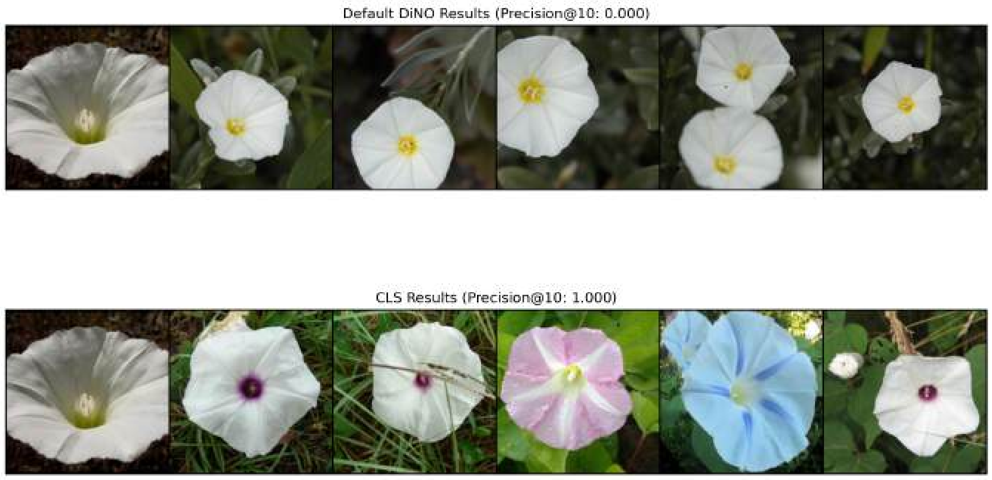

Default DiNO Results (Precision@10: 0.000)
CLS Results (Precision@10: 1.000)

FIGURE 6. Baseline vs DINO + CLIP for a sample image

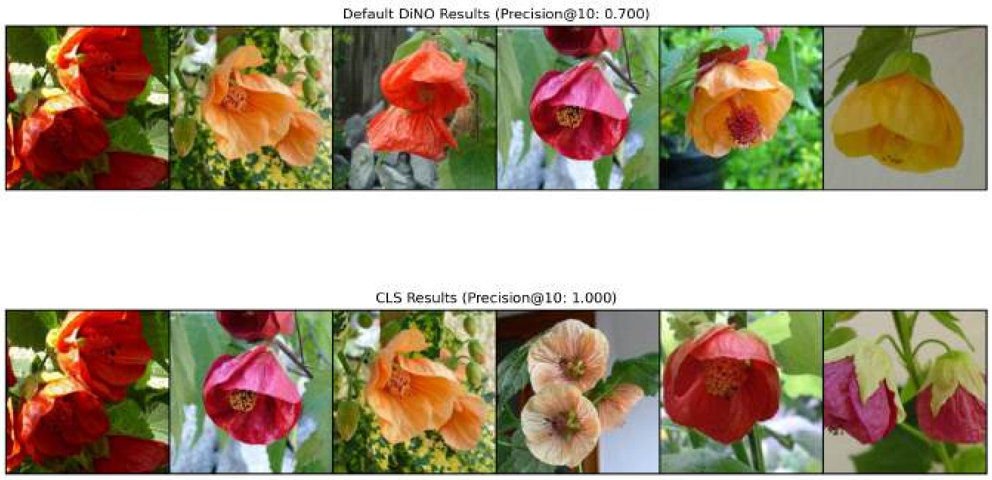

Default DiNO Results (Precision@10: 0.700)
CLS Results (Precision@10: 1.000)

FIGURE 7. Baseline vs DINO + CLIP for a sample image

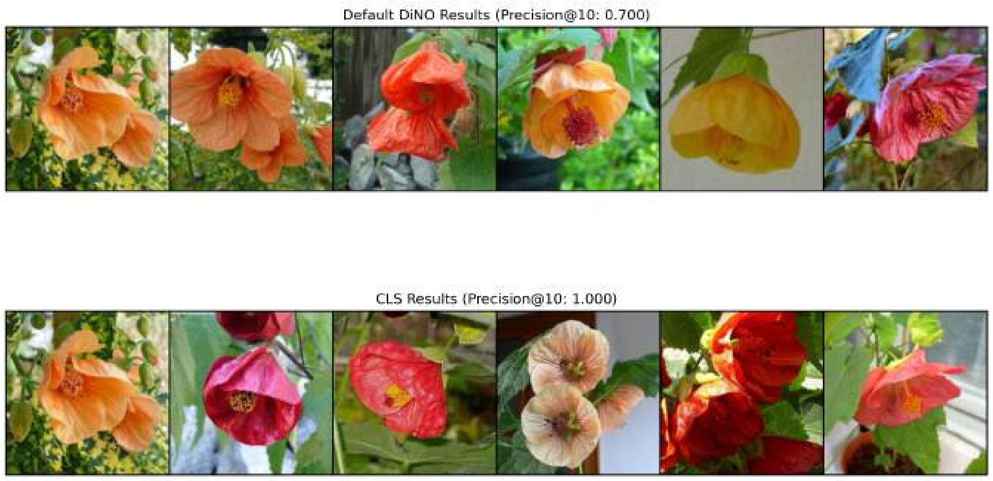

Default DiNO Results (Precision@10: 0.700)
CLS Results (Precision@10: 1.000)

FIGURE 8. Baseline vs DINO + CLIP for a sample image

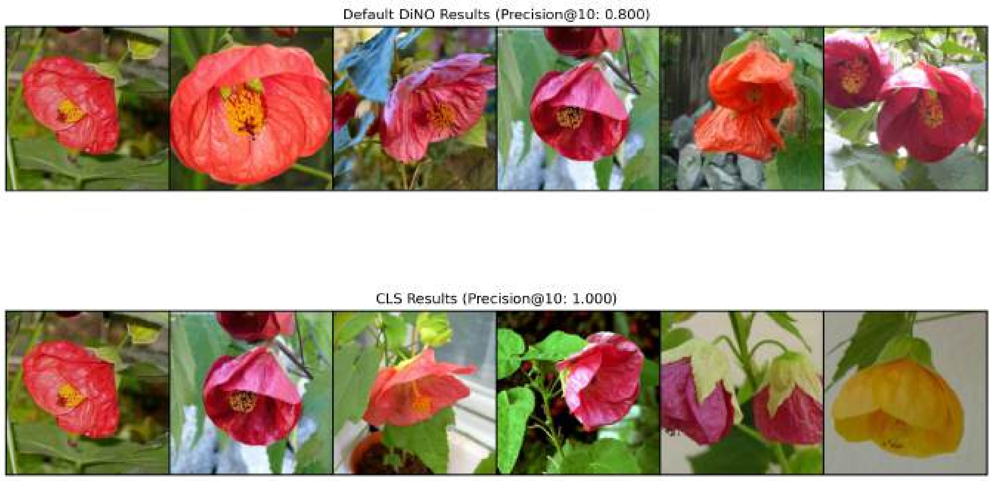

Default DiNO Results (Precision@10: 0.800)
CLS Results (Precision@10: 1.000)

FIGURE 9. Baseline vs DINO + CLIP for a sample image

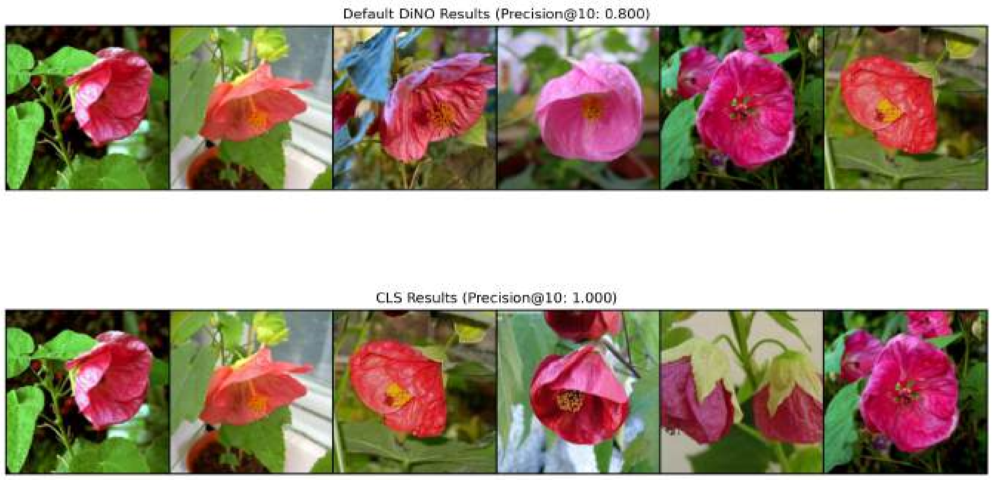

Default DiNO Results (Precision@10: 0.800)
CLS Results (Precision@10: 1.000)

FIGURE 10. Baseline vs DINO + CLIP for a sample image

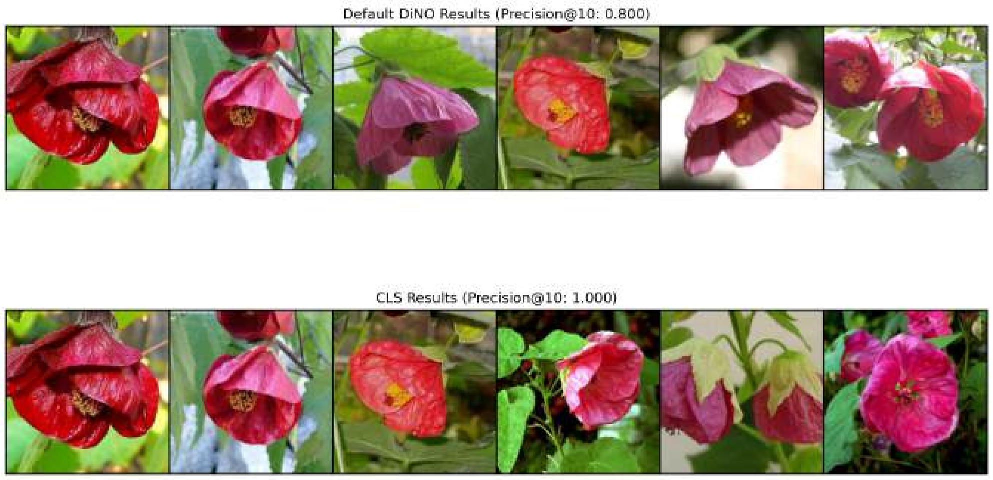

Default DiNO Results (Precision@10: 0.800)
CLS Results (Precision@10: 1.000)

FIGURE 11. Baseline vs DINO + CLIP for a sample image