Contents

A. Related Work

Our work builds on and contributes to four major areas of prior research: self-supervised learning for visual representation learning, scaling strategies for data and model capacity in vision models, open-source foundation models, and techniques for disentangling semantic content from positional or representational biases.

Self-Supervised Learning (SSL) for Vision. Self-supervised learning has emerged as a powerful paradigm for visual representation learning without any manual annotations. By designing pretext tasks that utilize image structure as supervision signals, SSL methods enable models to learn transferable features. Early approaches used handcrafted objectives such as context prediction [22], patch reordering [57], colorization [88, 89], inpainting [63], geometric transformation prediction [28], and instance discrimination [23, 85]. Modern SSL methods primarily focus on learning invariances across augmented data views. While early approaches leverage contrastive learning [12, 14, 15, 36, 55, 58] by aligning positive pairs and separating negatives, bootstrap-based [13, 29, 34] and distillation-based methods [11, 60] refine targets through teacher-student networks, often removing the need for negative pairs. More recently, Masked Image Modeling (MIM) has emerged as a dominant SSL strategy, where models learn to reconstruct masked patches [5, 37, 45, 84, 91]. Beyond these, clustering-based methods [9, 10, 43, 75] have gained prominence, assigning pseudo-labels through algorithms like K-means or Sinkhorn-Knopp. The combination of MIM with clustering has shown particular promise, as exemplified by recent works such as MOCA [30] and CAPI [21]. This hybrid approach leverages the strengths of both paradigms.

While current vision foundation models often fall into categories like vision-language or MAE-like architectures, which have their own strengths and limitations (e.g., reliance on text supervision or need for task-specific adaptations), DINOv2 [60] stands out as a powerful pretrained model employing a clustering-based approach. However, DINOv2 has two key limitations: as a clustering method, it doesn't inherently capture the ambiguity often present in assignments at a fixed granularity, nor does it explicitly incorporate the benefits of modern hierarchical masking strategies. Our work addresses these concerns by integrating nested *Matryoshka* projections [49] directly into its objective. This allows each subspace to perform clustering at a *different* granularity, yielding diverse pseudo-labels efficiently (see Figure 1). Combined with an improved input masking strategy, our approach enables the joint learning of coarse-to-fine semantics without increasing model size, leading to strongly improved performances and reduction in memory.

Open Foundation Vision Models. The reliance on proprietary datasets in the training of current vision foundation models raises critical concerns regarding transparency, reproducibility, and the disentanglement of contributions. Models such as SEER [32], DINOv2 [60], CLIP [65], and billion-scale MAE [74] are all trained on proprietary data. This practice makes it challenging for the research community to isolate the true impact of model's novelty and training strategies from the unique characteristics and biases of the datasets themselves. The lack of access to these datasets hinders independent verification, fair comparison, and a comprehensive understanding of what truly drives model performance. Inspired by the success and large-scale utilization of CLIP, OpenCLIP [17, 42] and MetaCLIP [18, 86] are the first to release models trained on public data. OpenCLIP [17, 56] stands out with the fully reproducible pipeline leveraging the ready-to-use LAION dataset [51, 70], study of scaling laws and release of intermediate checkpoints. On vision-only foundation models, the performance of DINOv2 has proven difficult to match by public data models. Web-SSL [26] extends the study of large-scale self-supervised pretraining by training models on publicly available MetaCLIP-2B [86] dataset, showing that models trained on open data can approach the performance of those trained on proprietary data on VLM tasks, but still below DINOv2 on visual perception tasks.

Building on this, we present a fully open-source vision foundation model using publicly available datasets, Re-

LAION [51], as it represents the most popular and safe public dataset for large-scale vision model training.

Spatial correlations in learned representations. A common issue in dense self-supervised learning is the entanglement of semantic content with positional cues, causing models to rely on location rather than object identity. For instance, a model trained on "cows in grassy fields" and "camels in deserts" may misclassify a cow on a beach as a camel, due to learned associations with background context [1]. Such spatial biases reduce generalization and can hinder performance when objects appear in atypical locations (e.g., a cow in the sky) [73].

Several works have addressed this by proposing methods invariant to positional information. Lenc and Vedaldi [52] enforce equivariance to geometric transformations; Wang et al. [80] disentangle representations into orthogonal subspaces for content and style. Invariant Risk Minimization [1] seeks features stable across environments, minimizing reliance on spurious cues. We propose a simple post-training strategy that learns a linear projection to identify and remove spatial information from features. Since, we use it as a post-training strategy, it requires no architectural changes and can be easily adapted to any pretrained model to reduce spatial bias.

B. Implementation details

A summary of the different datasets used at different training stage in given in Table 6

Pretraining Datasets We pretrain `Franca` on large-scale, publicly available image-only datasets to ensure full reproducibility. We use the ImageNet-21K [67], which contains approximately 13.1M high-quality, hierarchically labeled images across 21,841 classes. This dataset offers broad visual coverage and is widely used in foundation model pretraining. To further scale up training and improve generalization, we also leverage LAION-600M[1], a subset of ReLAION-2B, which is a research-safe version of the LAION-5B dataset [51, 70]. While LAION-5B is originally paired image-text data, we discard the text and use only the image modality.

Training `Franca`'s architecture follows DINOv2 [60] without registers, using Vision Transformers [24] of varying model capacities: ViT-B with 86M parameters, ViT-L with 300M, and ViT-G with 1.1B. All models are trained from scratch for 625K iterations without distillation from larger models, unlike DINOv2, which distills from ViT-G into smaller variants. We use CyclicMask (pseudo-code in Algorithm 1) and employ Matryoshka [49] with five nested heads with feature dimensions $[d, \frac{d}{2}, \ldots, \frac{d}{16}]$ on top of the normal ViT backbone.

[1] https://huggingface.co/datasets/laion/laion-coco

Algorithm 1 Pseudo-code for CyclicMask

Require: input size (H, W), masking ratio $m \in [0, 1]$, roll flag `roll`, aspect ratio range $[r_{\min}, r_{\max}]$

$T \leftarrow H \times W$ ▷ total number of patches
$n \leftarrow \lfloor m \cdot T \rfloor$ ▷ number of masking patches
$c \leftarrow \max(1,\ T - n)$ ▷ number of complement patches
$r \sim \exp\left(\mathcal{U}[\log r_{\min},\ \log r_{\max}]\right)$ ▷ sample aspect ratio
$h \leftarrow \min\left(H,\ \lceil\sqrt{c \cdot r}\rceil\right)$
$w \leftarrow \min\left(W,\ \lceil\sqrt{c/r}\rceil\right)$
`top` $\sim \mathcal{U}\{0,\ H - h\}$, `left` $\sim \mathcal{U}\{0,\ W - w\}$
`mask` $\leftarrow$ `torch.zeros(`H, W`, , dtype=bool)`
`mask[top:top + h, left:left + w]` $\leftarrow$ `True`
`mask` $\leftarrow \neg$ `mask` ▷ invert block to mask everything else
if `roll` = `True` **then**
  `shift_x` $\sim \mathcal{U}\{0,\ H - 1\}$, `shift_y` $\sim \mathcal{U}\{0,\ W - 1\}$
  `mask` $\leftarrow$ `torch.roll(mask, (shift_x, shift_y), dims=(0,1))`
end if
return `mask` of size $H \times W$

For LAION-600M, we use global crops scale of $[0.48, 1.0]$, following DINOv2-style augmentations. Stochastic depth regularization is set to 0.1 for ViT-B and 0.4 for ViT-L and ViT-G. We use a total batch size of 2048 for ViT-B and 3072 for both ViT-L and ViT-G, distributed across 32, 64 and 128 H100 GPUs for ViT-B, ViT-L and ViT-G respectively. The learning rate is set to 1×10^{-3} for the Base model and 3.5×10^{-4} for the Large and Giant variants, using a cosine schedule with warmup of 100K iterations.

We train RASA on top of our frozen backbone on Pascal VOC using crops of resolution 518×518, batch size of 128, with AdamW [53] optimizer. For each of the 8 incremental head training iterations, we used a dual-head linear projection (one for predictings and the other the y-axis patch positions) with sigmoid activation. In every iteration, only the top head was trained for 5 epochs with no weight decay and an initial learning rate of 2×10^{-3}, while all heads from previous iterations are absorbed into the weights of the final ViT layer, for iteratively removing the positional information from the features of the last ViT layer.

High-resolution adaptation We initialize the model with pretrained weights and perform incremental high-resolution finetuning. First, we finetune the model at input resolution 364×364 with a local crop resolution of 112×112, using a base learning rate of 1.25×10^{-5} for 30K iterations. The resulting checkpoint is then used to initialize the model for a second finetuning stage at input resolution 518×518 with a local crop size of 168×168, again with

TRAINING TYPE	ARCHITECTURE	DATASET	#IMAGES
Pretraining	ViT-B	ImageNet-21K	13,153,000
Pretraining	ViT-L	LAION-COCO	599,187,600
Pretraining	ViT-G	LAION-COCO	599,187,600
High-Res Finetuning	ViT-B, ViT-L	IN-1K, ADE20K, COCO, VOC, KITTI	1,444,270
RASA Post-training	ViT-B/L/G	Pascal VOC	17,125

Table 6. Summary of training setup across architectures, datasets, and image counts.

the same learning rate, for 10K iterations. All schedules are preserved but temporally compressed to fit within the shorter training horizons. The teacher network undergoes a warmup phase during the first 10K iterations of the initial 364-resolution stage to stabilize early training dynamics. We use a dataset mix comprising only the training set of ImageNet-1K, ADE20K, COCO, KITTI, and VOC. Due to computational constraints, high-resolution finetuning is performed only for the ViT-B and ViT-L model.

C. Evaluation details

Overclustering In the overclustering setting, we follow the protocol of [93]. Specifically, we perform K-Means clustering (via faiss [44]) on the spatial tokens extracted from the backbone, discarding the projection head. To align clusters with ground-truth semantic labels, we first apply greedy matching based on pixel-level precision and then refine the assignment with Hungarian matching [48], ensuring permutation-invariant evaluation as in [43]. Inputs are cropped to 448×448, while clustering operates on downsampled 100×100 masks to reduce the computational cost of Hungarian matching. Results are reported as the mean Intersection-over-Union (mIoU), averaged over five seeds, across twp datasets: COCO-Thing, and Pascal VOC 2012 [25].

Visual In-Context Learning The Dense Nearest Neighbor Retrieval Evaluation, introduced by [4] and openly implemented by [61], is designed to measure the scene understanding ability of dense image encoders through a retrieval-based protocol. The evaluation proceeds in three stages:

1. **Memory Bank Construction**: Given a training dataset with dense annotations, two memory banks are built. The first stores patch-level features obtained from the spatial outputs of a dense encoder, while the second stores the corresponding patch-level labels.
2. **Query Processing**: For each validation image, we extract patch embeddings from the encoder's spatial output. Each query patch searches for its k nearest neighbors in the feature memory bank, and the associated labels of these neighbors are aggregated to infer the query patch label.
3. **Evaluation**: After generating a predicted dense annotation for the full image, the result is compared against ground-truth labels to compute performance.

Since the original implementation of [4] is not publicly available, we rely on the open-source reimplementation from [61], which follows the authors' description and allows leveraging either the ScaNN library [35] or the faiss library [44] for efficient nearest-neighbor search. In our experiments, we adhere closely to this setup but make two modifications: (1) instead of restricting memory to a fixed capacity of 10,240,000 entries, we index all patch embeddings extracted from images resized to 518×518; and (2) we employ GPU-accelerated FAISS for nearest-neighbor retrieval, configured to approximate the ScaNN setup used in the original evaluation (e.g., with $k = 30$ neighbors).

We report results as mean Intersection-over-Union (mIoU) on two benchmark datasets: Pascal VOC 2012 [25] and ADE20K [90], averaging over five random seeds.

Linear Segmentation Linear segmentation is a common protocol to assess the linear separability of learned spatial representations. The setup freezes the encoder and trains a lightweight segmentation head, typically a single linear layer with batch normalization, using pixel-wise cross-entropy loss. In practice, the spatial patch embeddings produced by the backbone are bilinearly upsampled to match the resolution of the ground-truth masks, and the linear head is trained on top. This procedure is implemented in `mmsegmentation` and was used in the DINOv2 paper [59] for evaluating Pascal VOC and ADE20K, where the backbone is kept frozen and only the linear head is optimized.

In this paper, we adopt the DINOv2 setup as a baseline and introduce a few modifications tailored to our experimental needs:

- **General changes (both datasets):** We switch from iteration-based training (`max_iters = 40000`) to epoch-based training with a maximum of 50 epochs. The crop size is increased from 512×512 to 518×518, and the stride is adjusted accordingly to half the crop size. Data augmentation uses `RandomResize` instead of `Resize`. The learning rate schedule is also defined by epoch rather than iteration.
- **Pascal VOC 2012 [25]:** We replace the AdamW optimizer (used in DINOv2) with SGD (lr = 0.001, momentum = 0.9, weight decay = 5×10^{-4}). The resize target is

set to $(2048, 518)$ with a ratio range of $(0.5, 2.0)$, consistent with the crop size change.

- **ADE20K [90]:** We keep AdamW as the optimizer but wrap it with an `OptimWrapper`. The random resize ratio range is expanded from $(0.5, 2.0)$ to $(1.0, 3.0)$, making the scale augmentation more aggressive. At test time, instead of single-scale evaluation, we adopt multi-scale testing with image ratios $[1.0, 1.32, 1.73]$.

These modifications preserve the core linear segmentation evaluation protocol of DINOv2 while adapting it for our framework and experimental goals. We generally have seen the modifications to work better and improve further the results of all the models we tested.

Evaluation Datasets **Pascal VOC 2012** [25]. We use the most recent `trainaug` split, which contains 10,582 annotated images across 21 categories, including one background class. The validation set includes 1,449 images. We exclude unlabeled objects and the boundary class from evaluation. For hyperparameter tuning of the fully unsupervised segmentation method of [93], we rely on the original `train` split (1,464 images).

COCO-Stuff 164K [8]. The full COCO-Stuff dataset provides semantic annotations for both "stuff" (background, amorphous regions) and "thing" (foreground, countable objects) categories, with 91 stuff and 80 thing classes, respectively. It contains 118,000 training images and 5,000 validation images. In prior work [62, 93], two variants of this dataset are considered:

1. *COCO-Stuff*, where the 91 stuff categories are grouped into 15 broader classes (with all thing categories collapsed into a single "other" label that is ignored during evaluation).
2. *COCO-Thing*, where the 80 thing categories are consolidated into 12 broader object classes, and background regions are excluded from evaluation.

In this work, we only make use of the *COCO-Thing* variant. Specifically, we follow [46] to obtain panoptic instance annotations, which are then merged into object-level categories using the official conversion script. The resulting 12-class setup emphasizes object-level reasoning in cluttered natural scenes and serves as our benchmark for over-clustering experiments. Although we refer to the dataset as "COCO-Stuff 164K" for consistency with the literature, only the COCO-Thing portion is used in our evaluations.

ADE20K [90]. ADE20K is a large-scale scene parsing benchmark with 150 semantic categories, ranging from background classes (e.g., sky, grass) to fine-grained objects (e.g., person, car). The dataset contains 20,210 training images and 2,000 validation images with detailed annotations. Due to its diversity and fine-grained structure, ADE20K is widely regarded as one of the most challenging datasets for dense prediction tasks. In our experiments, we use the full dataset but ignore the *others* label during evaluation.

D. Additional Results

D.1. Training with registers

Registers mitigate feature artifacts (high-norm outlier tokens) in DINOv2. However, Franca's Nested Matryoshka Clustering intrinsically prevents these artifacts by enforcing multi-scale semantic structure, evidenced by lower token norms in Figure 10 compared to even DINOv2-R. Consequently, adding registers to Franca is redundant and subpar, degrading k-NN on IN-1K (79.6→78.7) and linear segmentation on ADE20K (46.2→45.5). As registers primarily benefit DINOv2's dense tasks without improving classification, and Franca naturally yields coherent dense features as shown in Figure 8 and sharp attention maps (Figure 11), we use DINOv2 *without* registers (except Tab 2(b)) as the primary baseline to ensure a fair comparison.

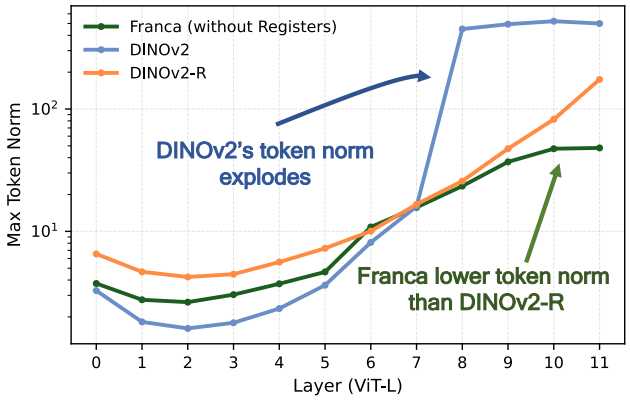

Figure 10. Layer-wise maximum token norms of Franca (LAION-600M), DINOv2 (LVD-142M) and DINOv2-R (LVD-142M) with ViT-L backbone on validation set of ImageNet-1K.

D.2. Fine-grained classification

We evaluate the transferability of the learned representations on 11 classification benchmarks introduced in SimCLR [12]. These benchmarks cover a variety of tasks, including scene recognition, fine-grained object classification (such as food, cars, and aircraft), and texture recognition. Following [60], we train a logistic regression classifier on features extracted from a frozen backbone. This approach focuses solely on the quality of the visual features and provides a fair way to compare performance across different tasks. Although some of these benchmarks tend to favor models trained with text supervision, our features perform strongly and competitively across many categories.

As shown in Table 7, our method, `Franca`, transfers well across a wide range of downstream tasks. Our ViT-G/14 model (`Franca-G`) achieves the same performance as DINOv2-G and outperforms Web-SSL-G by 1.1% despite being trained on much less data. It also matches the performance of larger models like OpenCLIP-G. On

Method	Arch	Food	C10	C100	SUN	Cars	Aircr	DTD	Pets	Cal	Flwrs	CUB	Avg
MAE	ViT-H/14	78.4	96.1	83.9	63.9	56.1	63.4	75.4	89.4	95.9	92.3	57.2	77.5
DINOv2[†]	ViT-L/14	93.4	99.2	93.9	78.1	89.9	81.7	82.9	95.2	87.2	99.6	90.3	90.1
DINOv2[§]	ViT-L/14	94.3	99.3	93.4	78.7	89.9	81.5	84.0	96.5	97.5	99.7	90.5	91.4
Web-SSL	ViT-L/14	91.0	98.9	90.7	77.5	88.9	80.2	83.6	93.1	95.1	98.8	90.9	89.9
DINOv2	ViT-G/14	94.7	**99.5**	94.4	78.7	91.4	**87.2**	84.5	96.7	**97.6**	**99.7**	**91.6**	**92.3**
Web-SSL	ViT-G/14	94.1	99.4	93.1	78.0	90.3	83.7	84.7	92.4	96.8	99.4	91.2	91.2
OpenCLIP	ViT-G/14	94.5	98.7	91.0	84.0	**96.1**	80.2	**86.0**	95.7	98.1	99.5	89.9	92.2
Franca (ours)	ViT-B/14	90.6	98.7	90.9	77.0	88.7	75.2	81.7	94.1	96.2	99.7	86.2	88.9
Franca (ours)	ViT-L/14	94.3	99.4	94.1	79.9	89.5	81.3	84.1	95.1	97.4	99.8	91.1	91.5
Franca (ours)	ViT-G/14	**95.0**	**99.5**	**95.1**	**78.9**	91.3	85.5	85.0	**97.2**	97.5	**99.7**	91.3	**92.3**

Table 7. *Linear evaluation of frozen features on fine-grained datasets*. top-1 accuracy measured across 11 benchmarks across objects, scenes, and textures, following [12]; [†]: reproduced on IN-21K without distillation; [§]: distilled from DINOv2-G on LVD-142M.

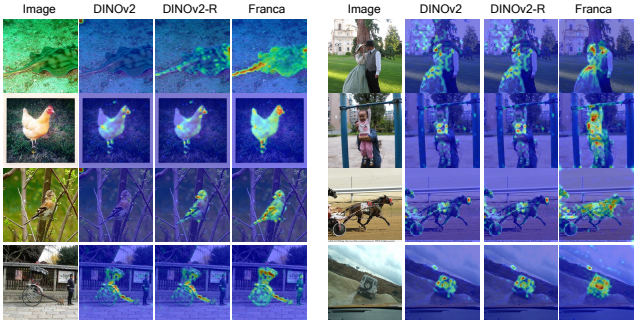

Figure 11. *Self-attention maps* utilizing 14 × 14 patches. These maps are visualized using the [CLS] token on the last layer's heads on the validation set of ImageNet-1K [69]. Franca has better localization than DINOv2 with registers [20] without requiring the use of registers, where the nested Matryoshka clustering captures fine-grained details, e.g., feathers, beaks of bird.

datasets such as CIFAR-100 and Oxford Pets, Franca-G achieves 0.7% and 0.5% gains over DINOv2G respectively, demonstrating Franca's strong generalization ability across both natural and fine-grained classification tasks.

D.3. Computational Cost

We report the computational cost in Table 8 providing estimates that cover the full training pipeline, including finetuning at 364 × 364 and 518 × 518 resolutions. Compute cost scales roughly linearly with model size: Franca-G requires about 13× the GPU-hours of Franca-B, while Franca-L lies between at 4×. This reflects both the higher per-step cost of larger models and the increased GPU parallelism needed to maintain throughput. Compared to prior large-scale self-supervised training efforts such as MetaCLIP and DINOv2-G, our models use fewer resources, Franca-G trains in ∼20K GPU-hours on H100s, over an order of magnitude less than MetaCLIP's 368K GPU-hours for a similar training horizon.

Reported numbers include practical overheads (e.g., hyperparameter tuning, checkpoint restarts, failed runs), providing a realistic measure of total compute. High-resolution finetuning accounts for a notable portion of the cost, especially for Franca-B and Franca-L, underscoring the need to balance resolution scaling with available compute.

Model Arch.	# GPUs	Trainable Params (in M)	Iterations (Pretrain+ HRFT)	GPU hours
DINOv2-B	32	184	625K + 20K	1,324
Franca-B	32	236	625K + 40K	1,504
DINOv2-L	64	404	625K + 20K	5,677
Franca-L	64	457	625K + 40K	5,952
MetaCLIP-G	256	1800	390K	368,640
DINOv2-G	128	1240	625K + 20K	22,016
Franca-G	128	1295	625K	19,992

Table 8. We estimate the overall GPU hours for our project, including hyper-parameter search, restarts from checkpoints, and failed runs, amounting to a total of 160K GPU-hours on NVIDIA H100 GPUs (80GB, NVLink interconnect). The table provides a breakdown across model scales (ViT-B, ViT-L, ViT-G) and training phases (pretraining vs. finetuning). Reported values reflect actual wall-clock consumption rather than idealized FLOP counts.

D.4. Effectiveness of slicing features

We compare both Franca-L (LAION-600M) and DINOv2-L (LVD-142M) features after applying PCA and slicing the first k components. Linear probing on IN-1K in Table 9 shows DINOv2-L degrades quickly (86.3 → 22.0 from 1024 → first 50 components), while Franca remains more robust (retaining 84.9 → 37.2), showing that Matryoshka preserves useful structure without post-hoc dimensionality reduction.

:DIM (AFTER PCA)	200	100	50	25	10
DINOv2-L	71.2	48.6	22.8	8.7	2.0
Franca-L	70.1	54.4	37.2	23.4	12.8

Table 9. Franca encodes better semantics across the sliced features as compared to DINOv2.

D.5. Attention maps visualizations

We compare self-attention maps from the final layer's `[CLS]` token of DINOv2, DINOv2R (with registers) [20] and `Franca` in Figure 11. DINOv2 often fails to localize objects, especially under clutter or occlusion, while DINOv2R offers only minor improvements. In contrast, `Franca` yields sharply focused attention maps aligned with object boundaries, even for small or partially occluded instances. This suggests that our Matryoshka-style multi-head clustering promotes semantically rich features and finer-grained representations.

D.6. Probing 3D understanding

We evaluate the geometric understanding of `Franca` on two tasks: keypoint correspondence (SPair-71k [54]) and monocular depth estimation (NYUv2 [71]). For SPair-71k, the goal is to establish dense keypoint correspondences under varying viewpoint changes, where we report accuracy across all keypoints and under increasing viewpoint disparity. For depth estimation, we follow the AdaBins protocol [6] and measure performance using the scale-invariant root-mean-square error (SI-RMSE). All evaluations are conducted at high resolution: 800×800 for SPair-71k and 480×480 for NYUv2.

As shown in Table 10, `Franca` achieves strong performance across both tasks. On SPair-71k, it outperforms DINOv2 trained on the ImageNet-21K dataset, both at ViT-B and ViT-L models. On NYUv2, `Franca` achieves comparable performance as DINOv2 distilled from larger DINOv2 model. This is also due to the fact that DINOv2 uses NYU Depth v2 during pretraining (as part of LVD 142M). These results indicate that `Franca` learns a spatial representation that captures both fine-grained 2D alignment and coarse 3D structure, generalizing well from object-centric pretraining to downstream geometric tasks.

D.7. Overclustering

We evaluate `Franca` using the overclustering protocol from [93], which measures semantic alignment of spatial features in a label-free setting. Patch embeddings are clustered with K-Means and matched to ground-truth segmentation masks via Hungarian matching [48], and performance is reported as mean IoU (mIoU). This task highlights the ability of representations to capture fine-grained structure, which is crucial for dense prediction tasks such as semantic segmentation and object detection.

As shown in Table 11, `Franca` consistently outperforms strong baselines across backbones and clustering granularities. On ViT-B/14, it achieves the highest VOC performance at $K = 300$ (56.4 mIoU), surpassing DINOv2-B[§] (52.5) while remaining competitive on COCO-Things. With ViT-L/14, `Franca` delivers its strongest results: it reaches 47.4 and 58.9 mIoU on VOC ($K = 100/K = 300$), and 49.6 and 54.4 on COCO-Things, outperforming DINOv2-L[§], Web-SSL, and SigLIP 2. At ViT-G/14 scale, `Franca` maintains a clear lead over DINOv2-G and Web-SSL, reaching 49.2 on VOC ($K = 300$) and 31.9 on COCO-Things ($K = 300$), despite the challenging scaling regime.

These results underscore that `Franca` not only scales effectively to larger backbones but also delivers robust gains on both coarse (VOC) and complex (COCO-Things) datasets. Importantly, it rivals or surpasses multimodal baselines like EVA-CLIP and SigLIP 2, highlighting its strength as a unimodal self-supervised learner for dense prediction settings.

D.8. Ablating RASA

D.8.1. Dataset Size Requirements for Training the RASA Head

We ablate the dataset size required for training the **RASA** head by varying the fraction of images sampled from COCO (scene-centric) and IMAGENET100 (object-centric). Tables 12a and 12b summarize the results across in-context segmentation and linear segmentation.

Observations. On COCO, as little as 10%–20% of the data suffices to reach peak performance, with VOC and ADE20K scores saturating at 76.6–76.7 and 35.2–35.3 (in-context), and 89.3–89.4 and 46.2 (linear segmentation). Increasing the fraction up to 80% yields no further improvements, indicating that performance plateaus once ∼10k images are used.

A similar trend is observed on IMAGENET100, where 10%–20% of the dataset again provides optimal results (76.7 VOC / 35.3–35.4 ADE20K in-context, 89.4 VOC / 45.9–46.1 ADE20K for linear segmentation). Larger fractions show negligible variation, well within noise levels.

Conclusion. These findings demonstrate two key points. First, training the RASA head is highly *data-efficient*: roughly 10k images are sufficient to achieve strong disentanglement. Second, the type of dataset—COCO (scene-oriented) vs. IMAGENET100 (object-oriented)—does not materially affect downstream performance. This indicates that RASA is largely agnostic to the semantic bias of the training distribution and can be reliably trained with minimal data.

Based on these observations, we adopt PASCAL VOC as our default choice for training the RASA head, since it is compact yet sufficient to reach optimal performance. With this setup, we achieve strong results across multiple benchmarks (Table 13). This confirms that a lightweight dataset such as Pascal VOC is adequate for effective training, while maintaining competitive performance across diverse tasks.

MODEL	ARCH.	DATA	d=0	d=1	d=2	All	SI-RMSE ↓
IMAGE RESOLUTION 800 × 800 (SPAIR-71K) AND 480 × 480 (NYUV2)							
OpenCLIP	ViT-B/14	LAION-2B	18.31	16.78	17.05	17.63	0.39
SigLIP2	ViT-B/14	WebLI	9.29	5.96	5.82	7.83	0.56
DINOv2†	ViT-B/14	IN-21K	42.82	33.38	34.82	38.71	0.33
DINOv2§	ViT-B/14	LVD-142M	63.04	**51.09**	**48.76**	**55.98**	**0.25**
`Franca` (ours)	ViT-B/14	IN-21K	53.46	41.9	44.2	45.6	0.30
OpenCLIP	ViT-L/14	LAION-2B	22.73	20.18	20.37	21.26	0.37
SigLIP2	ViT-L/14	WebLI	8.66	5.38	5.05	6.88	0.55
DINOv2†	ViT-L/14	IN-21K	57.60	42.91	44.17	50.68	0.31
DINOv2§	ViT-L/14	LVD-142M	63.91	**53.11**	**51.91**	56.92	**0.23**
`Franca` (ours)	ViT-L/14	LAION-600M	58.50	46.74	48.28	51.76	0.25

Table 10. *Probing 3D understanding* via keypoint matching on SPair-71K and monocular depth estimation on NYUv2. We report correspondence accuracy (higher is better) under increasing viewpoint changes (d=0, 1, 2, and overall) on SPair-71K, and scale-invariant RMSE (lower is better) for depth estimation on NYUv2. Results are shown for high (800^2) resolution SPair-71K input. †: reproduced on IN-21K without distillation; §: distilled from DINOv2-G.

METHOD	BACKBONE	OVERCLUSTERING			
		VOC		COCO-THINGS	
		$K = 100$	$K = 300$	$K = 100$	$K = 300$
SigLIP	ViT-B/16	29.5	36.9	41.5	53.4
iBOT	ViT-B/16	21.2	29.1	18.3	26.3
EVA-CLIP	ViT-B/16	**43.3**	49.1	41.3	52.0
DINOv2†	ViT-B/14	25.9	34.7	20.5	28.7
DINOv2§	ViT-B/14	39.2	52.5	**46.5**	**54.0**
`Franca` (ours)	ViT-B/14	37.5	**56.4**	38.8	51.1
SigLIP 2	ViT-L/16	24.3	40.4	43.5	50.7
Web-SSL	ViT-L/14	28.2	37.7	26.3	33.1
DINOv2†	ViT-L/14	25.9	34.7	24.1	35.1
DINOv2§	ViT-L/14	26.5	43.0	34.8	45.7
`Franca` (ours)	ViT-L/14	**47.4**	**58.9**	**49.6**	**54.4**
Web-SSL	ViT-G/14	26.0	33.4	15.5	21.5
DINOv2	ViT-G/14	19.5	27.7	20.7	29.2
`Franca` (ours)	ViT-G/14	**39.4**	**49.2**	**25.8**	**31.9**

Table 11. *OverClustering Performance.* Comparison of overclustering performance (mIoU) on Pascal VOC and COCO-Things datasets with $K = 100$ and $K = 300$. †: reproduced on IN-21K, without distillation; § : distilled from DINOv2-G on LVD-142M.

D.8.2. Learning Rate for Training the Dual-Linear Position Predictor

We also ablate the learning rate used when training each dual-linear position predictor in the RASA head. Results are reported on both PASCAL VOC and COCO in Tables 14a and 14b respectively.

Observations. On PASCAL VOC in Table 14a, the optimal performance is achieved with a learning rate of 0.002, reaching 76.7 VOC / 35.3 ADE20K for in-context and 89.4 VOC / 46.0 ADE20K for linear segmentation. Larger learning rates such as 0.005 yield slightly reduced segmentation accuracy, while very small rates (0.0001) underperform across both tasks. Interestingly, 0.0005 produces a competitive ADE20K score (46.0) but is less stable overall.

On COCO in Table 14b, the trend differs: the best results are obtained at the much smaller learning rate of 0.0001 (76.7 VOC / 35.3 ADE20K in-context, 89.4 VOC / 46.2 ADE20K linear segmentation). Higher learning rates (0.002, 0.005) still maintain strong performance but introduce minor drops or fluctuations, particularly in VOC segmentation. The consistency of 0.0001 across all metrics suggests that COCO benefits from more conservative updates during training.

Conclusion. These results indicate that while the RASA head is generally robust to a wide range of learning rates, the optimal configuration is dataset-dependent. For PASCAL VOC, a moderately large learning rate (0.002) works best, while for COCO, a smaller learning rate (0.0001) provides the most stable and accurate results. Overall, this highlights that tuning the learning rate can provide small but measurable gains, though the model remains relatively stable across settings.

D.8.3. Number of Epochs for Training the Dual-Linear Position Predictor

We further ablate the effect of the number of training epochs used for each dual-linear position predictor in the RASA head. Results on both COCO and PASCAL VOC are reported in Tables 15a and 15b.

Observations. Across both datasets, performance improves gradually from 1 to 3 epochs, with the most consistent gains observed when training for 5 epochs. On COCO in Table 15a, 5 epochs achieves the strongest overall performance (76.7 VOC / 35.3 ADE20K for in-context, 89.4 VOC / 46.2 ADE20K for linear segmentation). Beyond this point, additional training (7 or 10 epochs) does not yield further benefits and in some cases slightly reduces performance, particularly on ADE20K segmentation.

A similar trend is evident when training on PASCAL VOC in Table 15b. The 5-epoch configuration again pro-

Table 12. *Ablating the Dataset Size used for Training RASA Head*

	IN-CONTEXT		LIN. SEG.	
Fraction	VOC	ADE20K	VOC	ADE20K
0.1	76.6	35.2	89.3	46.2
0.2	**76.7**	**35.3**	**89.4**	**46.2**
0.3	76.7	35.3	89.3	46.0
0.4	76.7	35.3	89.3	46.1
0.5	76.7	35.3	89.3	46.2
0.6	76.7	35.3	89.3	46.1
0.8	76.7	35.3	89.3	46.2

(a) Fractions of the COCO Dataset

	IN-CONTEXT		LIN. SEG.	
Fraction	VOC	ADE20K	VOC	ADE20K
0.1	**76.7**	35.3	**89.4**	**46.1**
0.2	76.7	**35.4**	89.4	45.9
0.3	76.7	35.4	89.3	46.0
0.4	76.7	35.4	89.4	46.0
0.5	76.7	35.4	89.4	45.9
0.6	76.7	35.4	89.3	46.0
0.8	76.7	35.4	89.4	45.9

(b) Fractions of the Imagenet Dataset

Table 13. Training the RASA head on top of VIT-BASE of `Franca` on Pascal VOC.

SETUP	IN-CONTEXT		LIN. SEG.	
	VOC	ADE20K	VOC	ADE20K
`Franca`	76.2	35.0	89.4	46.0
`Franca` + RASA	**76.7**	**35.3**	**89.4**	**46.0**

vides the best balance across both metrics (76.7 VOC / 35.3 ADE20K in-context, 89.4 VOC / 46.0 ADE20K linear segmentation). Extending to 7 or 10 epochs produces only marginal changes, with scores fluctuating around the same plateau. Interestingly, ADE20K segmentation shows a minor uptick at 10 epochs (46.1), but the difference relative to 5 epochs is negligible and within noise.

Conclusion. These results highlight that training the dual-linear position predictor is *computationally efficient*, requiring only around 5 epochs to converge. Extending beyond 5 epochs yields no meaningful improvements and may even slightly degrade results. Moreover, the stability of the trends across both COCO (scene-centric) and PASCAL VOC (object-centric) datasets reinforces the robustness of this setting. In practice, we adopt 5 epochs as the default configuration for RASA training.

D.8.4. Number of Iterations for Training the RASA Head

Finally, we ablate the number of *iterations* used to train the RASA head, where each iteration corresponds to training an additional dual-linear position predictor and integrating the previously trained predictors into the final ViT block. Results on PASCAL VOC and COCO are reported in Tables 16a and 16b.

Observations. Even a small number of iterations (1–2) produces competitive results across both datasets, suggesting that the RASA head can already provide strong positional disentanglement with minimal overhead. For example, with only 2 iterations we obtain 76.2 VOC / 34.5 ADE20K (in-context) and 89.4 VOC / 46.1 ADE20K (linear segmentation) on PASCAL VOC, which is close to the final performance.

However, performance steadily improves as the number of iterations increases, with the most notable gains occurring between 2 and 8 iterations. At 8 iterations, results peak at 76.7 VOC / 35.3 ADE20K in-context and 89.4 VOC / 46.0 ADE20K for segmentation on PASCAL VOC (in Table 16a), and 76.7 VOC / 35.3 ADE20K in-context and 89.4 VOC / 46.2 ADE20K on COCO (in Table 16b). Beyond this point (10 iterations), performance saturates and in some cases slightly declines, suggesting diminishing returns from further iterative training.

Interestingly, across both datasets, the segmentation metrics appear to plateau earlier (around 4–6 iterations), while in-context segmentation continues to benefit until iteration 8. This indicates that additional iterations primarily refine segmentation-related representations, even though these could be sort of learned by the linear layer used in linear segmentation.

Conclusion. These experiments demonstrate that the RASA head achieves strong results even with a small number of iterations, but the optimal trade-off between performance and compute is obtained at 8 iterations. Beyond this, additional predictors provide no meaningful improvements, confirming that the gains saturate after moderate iterative refinement.

Table 14. *Ablating the learning rate used to train each dual linear position predictor.*

	IN-CONTEXT		LIN. SEG.	
lr	VOC	ADE20K	VOC	ADE20K
0.002	**76.7**	**35.3**	**89.4**	**46.0**
0.005	76.7	35.3	89.4	45.7
0.0001	76.3	34.7	89.4	45.8
0.0005	76.6	35.2	93.9	46.0

(a) Learning Rates on the Pascal VOC

	IN-CONTEXT		LIN. SEG.	
lr	VOC	ADE20K	VOC	ADE20K
0.002	76.7	35.3	89.4	45.9
0.005	76.5	35.0	89.4	46.1
0.0001	**76.7**	**35.3**	**89.4**	**46.2**
0.0005	76.6	35.2	89.4	45.9

(b) Learning Rates on the COCO Dataset

Table 15. *Ablating the number of Epochs used to train each dual linear position predictor.*

	IN-CONTEXT		LIN. SEG.	
Epochs	VOC	ADE20K	VOC	ADE20K
1	76.6	35.2	89.4	46.1
2	76.6	35.3	89.4	45.8
3	76.7	35.3	89.4	46.0
5	**76.7**	**35.3**	**89.4**	**46.2**
7	76.6	35.2	89.4	45.9
10	76.7	35.3	89.3	45.8

(a) Number of Epochs on COCO Dataset

	IN-CONTEXT		LIN. SEG.	
Epochs	VOC	ADE20K	VOC	ADE20K
1	76.1	34.5	89.4	45.9
2	76.3	34.6	89.3	46.2
3	76.4	34.9	89.4	45.8
5	**76.7**	**35.3**	**89.4**	46.0
7	76.5	35.0	89.3	46.0
10	76.6	35.0	89.3	**46.1**

(b) Number of Epochs on Pascal VOC

Table 16. *Ablating the number of dual predictor layers used for training RASA head (i.e., the number of iterations).*

	IN-CONTEXT		LIN. SEG.	
Iters	VOC	ADE20K	VOC	ADE20K
1	76.2	34.2	89.3	46.1
2	76.2	34.5	89.4	**46.1**
3	76.4	34.5	89.3	45.9
4	76.5	34.6	89.3	45.9
5	76.3	34.5	89.3	46.0
6	76.4	34.8	89.4	45.8
8	**76.7**	**35.3**	**89.4**	46.0
10	76.3	35.1	89.3	46.0

(a) Number of Iterations with Pascal VOC

	IN-CONTEXT		LIN. SEG.	
Iters	VOC	ADE20K	VOC	ADE20K
1	76.1	34.2	89.3	45.9
2	76.2	34.5	89.4	46.1
3	76.3	34.5	89.4	46.0
4	76.5	34.5	89.4	46.2
5	76.3	34.5	89.4	46.0
6	76.4	34.8	89.4	46.0
8	**76.7**	**35.3**	**89.4**	**46.2**
10	76.6	35.2	89.4	46.0

(b) Number of Iterations with Coco