Vision Transformers Need Registers

Understanding and Fixing Artifacts in ViT Feature Maps

Timothée Darcet, Maxime Oquab, Julien Mairal, Piotr Bojanowski, ICLR 2024

Presented By: Siddarth Chilukuri, Michael Ng, Dilli Bhaskar

Outline

- **Part 1: What is Vision Transformer**
- **Part 2: The Problem**
  - Artifacts in ViT attention maps
  - Characterizing high-norm outlier tokens
- **Part 3: The Solution**
  - Adding register tokens to ViT
  - Costs of method
- **Part 4: Experiments and Results**
- **Part 5: Conclusion**

Why Vision Transformers?

Transformers scaled insanely well in NLP.

Natural question: can we use the same recipe for images?

Vision was (and still is) CNN-dominated.

Earlier attention-in-vision was often “CNN + attention,” not pure attention.

ViT’s key claim: you can use a mostly-standard Transformer encoder on images.

The trick is to treat image patches like tokens (words).

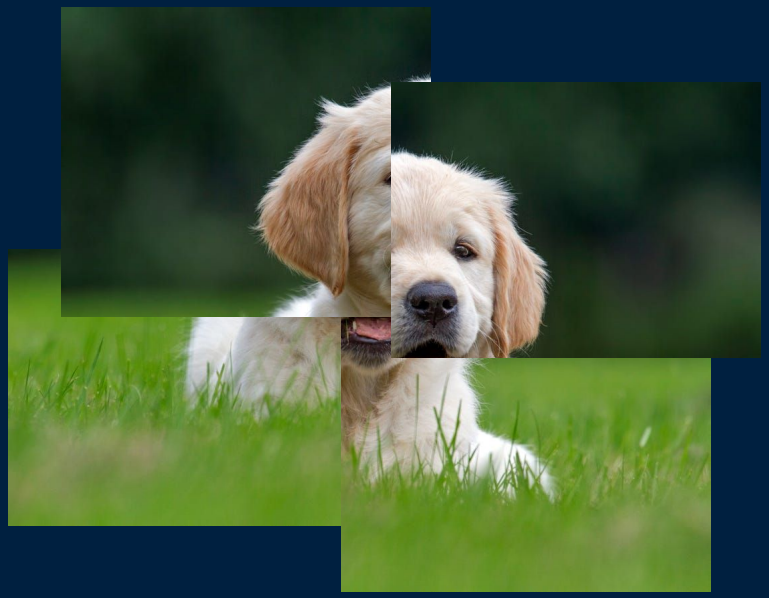
However, accuracy isn’t everything. Clean patch features matter for dense tasks and interpretation, and that’s where this paper comes in.

Dosovitskiy et al., "An Image Is Worth 16 X 16 Words: Transformers For Image Recognition At Scale", 2021

Going from Images to Tokens

Splitting into patches

Size: $P \times P$



Start with an Image: $H \times W \times C$

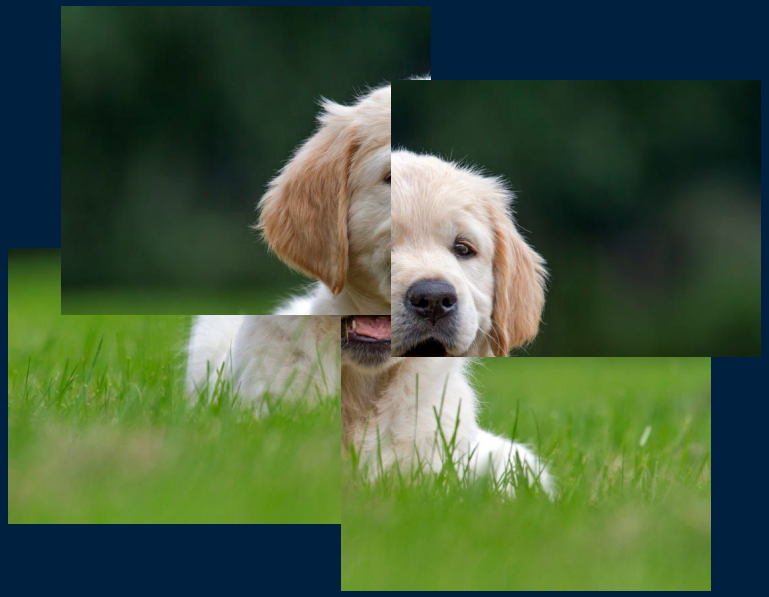
Eg. $240 \times 240 \times 3$

N patches of size $P \times P$

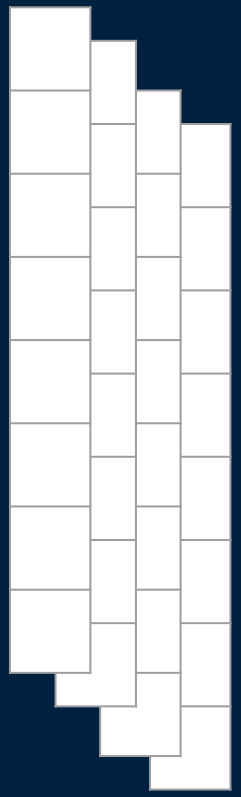
Eg. 225 patches of size 16×16

Dosovitskiy et al., "An Image Is Worth 16 X 16 Words: Transformers For Image Recognition At Scale", 2021

Going from Images to Tokens



Flatten



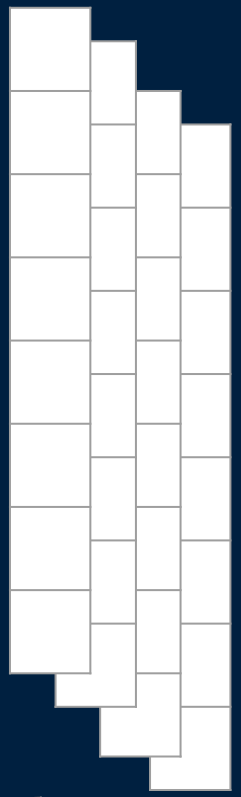
N patches of size $P \times P$

Eg. 225 patches of size 16×16

N flattened vectors of size $P^2 \times C$

Eg. 225 vectors of size 768

Going from Images to Tokens



Learnable Linear Projection

$R^{P^2C} \rightarrow R^D$

+ Positional Embeddings

0	1	2	3	...	196
CLS	t_1	t_2	t_3		t_{196}

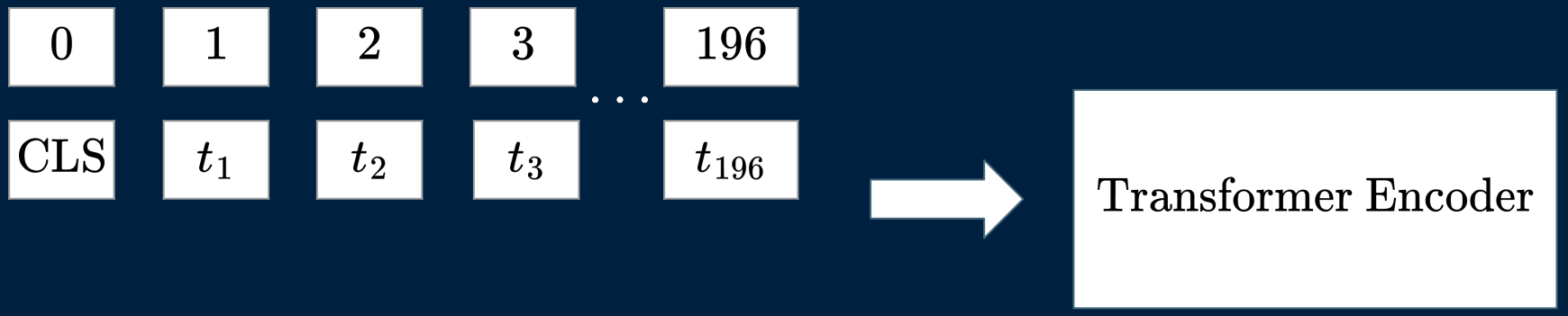
N flattened vectors of size $P^2 \times C$

Eg. 225 vectors of size 768

Sequence of $(N+1) - D$ dimensional vectors

Eg. $(225+1) - 768$ dimensional vectors

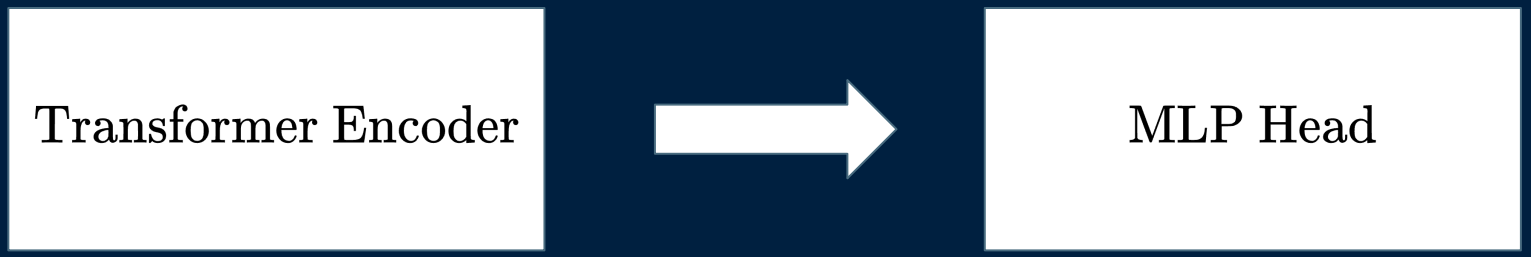
Onward!



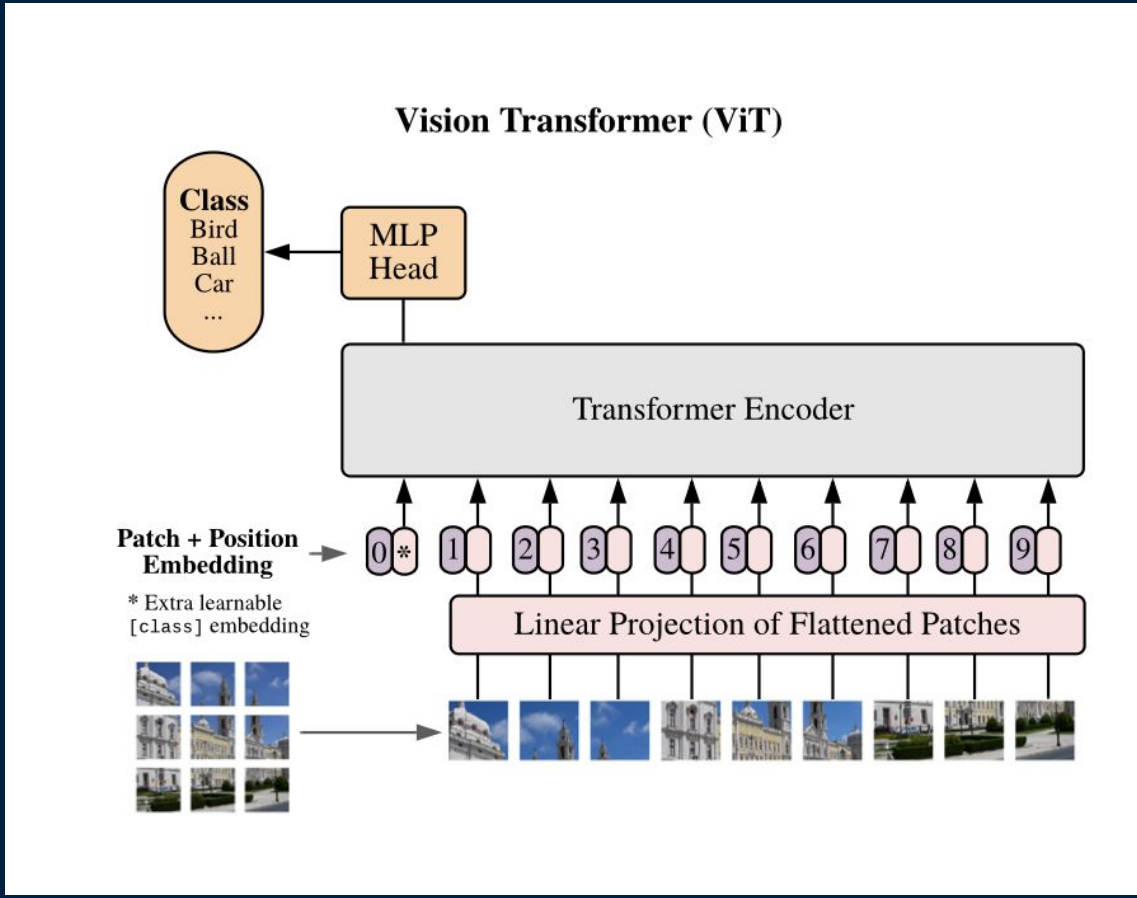
Sequence of $(N+1) - D$ dimensional vectors

Eg. $(225+1) - 768$ dimensional vectors

Onward!

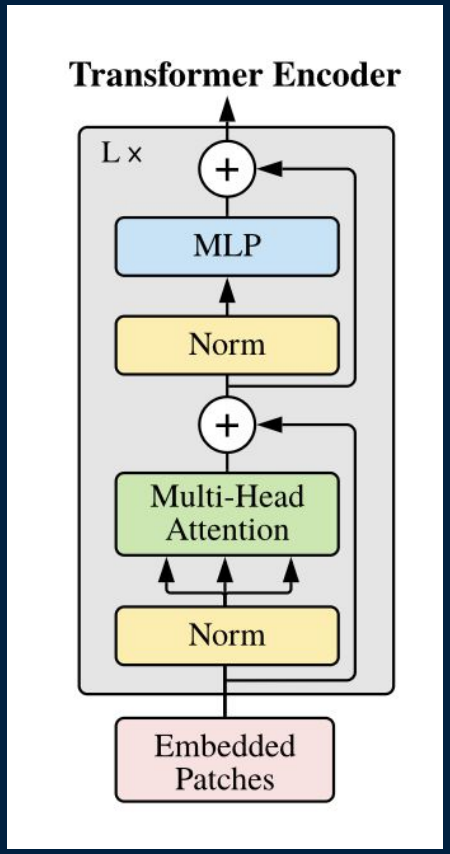


Summing it up



Dosovitskiy et al., "An Image Is Worth 16 X 16 Words: Transformers For Image Recognition At Scale", 2021

Summing it up



Dosovitskiy et al., "An Image Is Worth 16 X 16 Words: Transformers For Image Recognition At Scale", 2021

Inside the Transformer Encoder

Input is a sequence: $[\mathrm{CLS}], t_1, t_2, t_3, \ldots t_N$

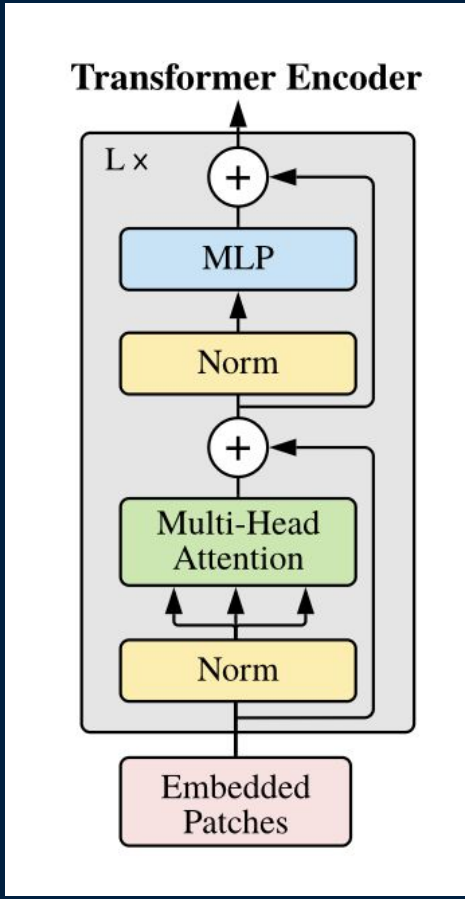
Each layer alternates: multi-head self-attention and an MLP block.

LayerNorm + residual connections stabilize training.

Self-attention allows tokens to exchange information globally, rather than only with nearby regions as in CNNs.

The model is trained so that the CLS token aggregates information useful for classification.

Patch tokens start as local features, but through attention they incorporate global context while still being associated with spatial locations.



Dosovitskiy et al., "An Image Is Worth 16 X 16 Words: Transformers For Image Recognition At Scale", 2021

CNNs vs ViTs. Inductive Bias Tradeoff

CNNs look at small neighborhoods first (like sliding filters) and gradually build up to bigger patterns.

ViTs let every patch “talk” to every other patch using attention, even early on. Much less “inductive bias”.

CNNs have built-in image structure (they naturally handle “nearby pixels matter most”).

ViTs have fewer built-in assumptions, so they often need more data / training to learn those patterns.

With enough training, ViTs can learn flexible global relationships and become strong general-purpose backbones.

Dosovitskiy et al., "An Image Is Worth 16 X 16 Words: Transformers For Image Recognition At Scale", 2021

What do ViT tokens represent?

A ViT produces one token per image patch (plus the CLS token).

You can think of each patch token as a feature summary of that patch region.

Attention lets tokens share information, so patch tokens can use context from the whole image.

These patch tokens are often reused for dense tasks (segmentation, detection) by reshaping them back into a grid.

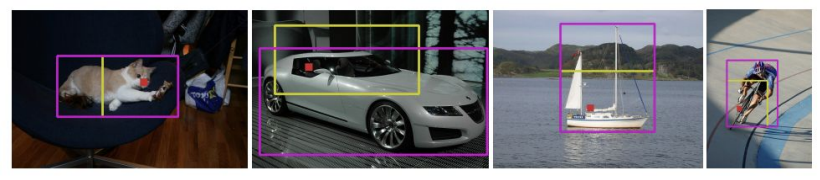
So ideally, patch tokens should stay meaningful and stable (they should describe image content, not random noise).

Many downstream vision tasks reuse patch tokens directly, so the quality and stability of these token representations matters.

Dosovitskiy et al., "An Image Is Worth 16 X 16 Words: Transformers For Image Recognition At Scale", 2021
Darcet et al., "Vision Transformers Need Registers", 2024

When patch tokens stop being “patch” tokens

- ViT patch tokens are reused as a spatial grid for dense / local tasks (segmentation, detection, object discovery).



Siméoni, Oriane, et al. "Localizing objects with self-supervised transformers and no labels."

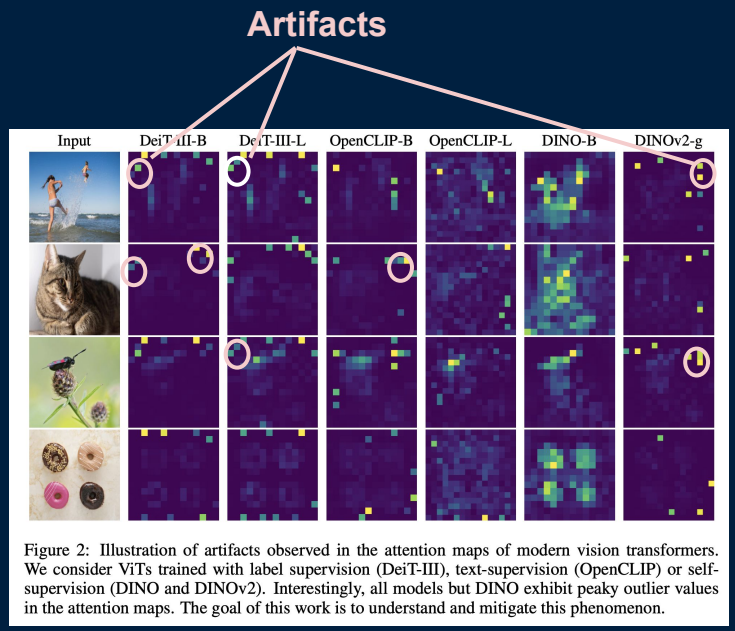


Figure 2: Illustration of artifacts observed in the attention maps of modern vision transformers. We consider ViTs trained with label supervision (DeiT-III), text-supervision (OpenCLIP) or self-supervision (DINO and DINOv2). Interestingly, all models but DINO exhibit peaky outlier values in the attention maps. The goal of this work is to understand and mitigate this phenomenon.

- But for some ViTs, some patch tokens become high-norm outliers → visible artifacts in attention/feature maps.
- These outliers corrupt local feature geometry, hurting methods that rely on patch similarity (e.g., LOST object discovery).

Darcet, Timothée, et al. "Vision transformers need registers."

The Problem: Artifacts in ViT Attention Maps

- Modern ViTs show strange artifacts in their attention maps
- Affects multiple models: DeiT-III (supervised), OpenCLIP (text-supervised), DINOv2 (self-supervised)
- Exception: Original DINO has clean attention maps
- Impact: Breaks object discovery methods like LOST

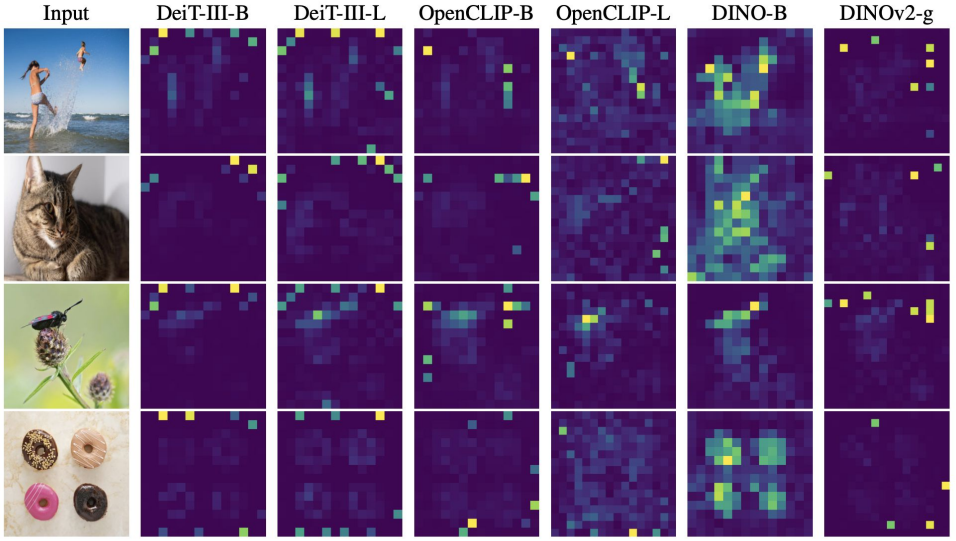


Figure 2: Illustration of artifacts observed in the attention maps of modern vision transformers. We consider ViTs trained with label supervision (DeiT-III), text-supervision (OpenCLIP) or self-supervision (DINO and DINOv2). Interestingly, all models but DINO exhibit peaky outlier values in the attention maps. The goal of this work is to understand and mitigate this phenomenon.

Darcet, Timothée, et al. "Vision transformers need registers."

Why this matters: local features are used everywhere

- Patch tokens are expected to represent local image content.
- Dense tasks + unsupervised grouping methods assume neighboring patches **have meaningful similarity**.
- Outliers create “spikes” that dominate dot products / similarity introduce unstable seeds and noisy grouping.

Paper contributions

- **Characterization**: artifacts correspond to high-norm outlier tokens
- **Hypothesis**: large, well-trained ViTs recycle redundant patch tokens as internal compute/memory slots
- **Method**: add [REG] register tokens (learnable, appended after patch embed and discarded at output)
- **Results**:
  a. outliers disappear
  b. dense/object discovery improves
  c. small compute overhead (<2% FLOPs for 4 regs).

Characterizing the Artifacts: High-Norm Tokens

- **Key observation:** Artifact patches have ~10x higher L2 norm at output
- **Bimodal distribution:** Most tokens have norm 0-100, outliers have norm >150
- **Proportion:** Only ~2.37% of tokens are outliers
- **Detection method:** Simple threshold on token norm (e.g., norm > 150)

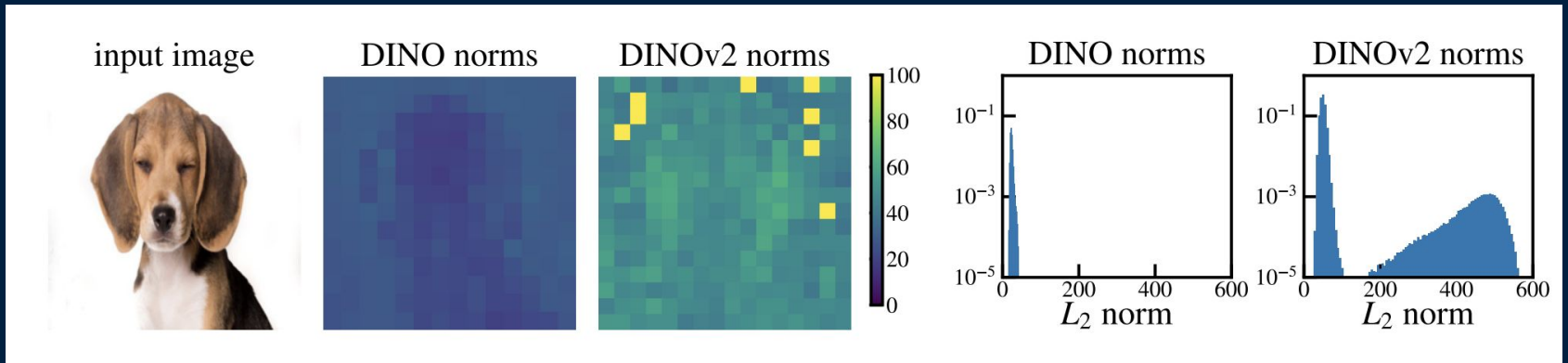
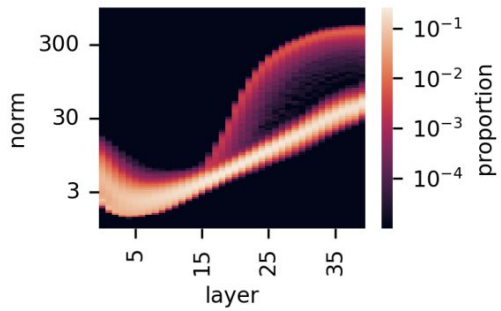


Figure 3: Comparison of local feature norms for DINO ViT-B/16 and DINOv2 ViT-g/14. We observe that DINOv2 has a few outlier patches, whereas DINO does not present these artifacts. For DINOv2, although most patch tokens have a norm between 0 and 100, a small proportion of tokens have a very high norm. We measure the proportion of tokens with norm larger than 150 at 2.37%.

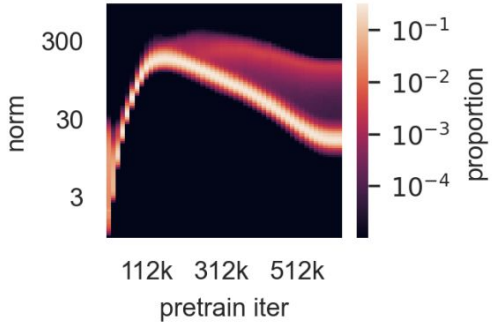
Darcet, Timothée, et al. "Vision transformers need registers."

When and Where Do Artifacts Appear?

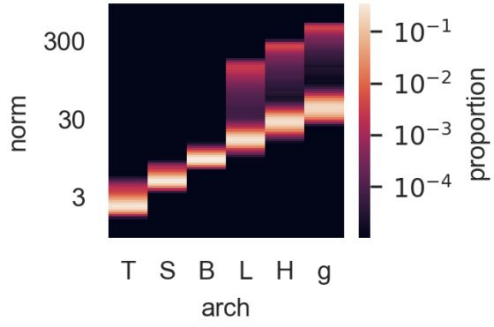
- **During training:** Outliers appear after ~1/3 of training
- **Network depth:** Emerge around middle layers (layer 15 of 40)
- **Model size:** Only in large models (ViT-L, ViT-H, ViT-g)
- **Spatial location:** Appear in redundant, low-information patches (similar to neighbors)



(a) Norms along layers.



(b) Norms along iterations.



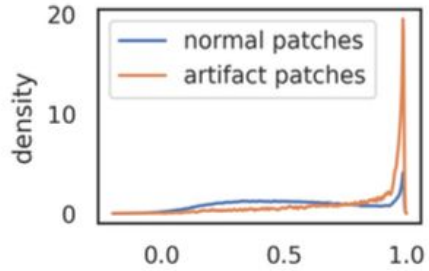
(c) Norms across model size.

Figure 4: Illustration of several properties of outlier tokens in the 40-layer DINOv2 ViT-g model. **(a)**: Distribution of output token norms along layers. **(b)**: Distribution of norms along training iterations. **(c)**: Distribution of norms for different model sizes. The outliers appear around the middle of the model during training; they appear with models larger than and including ViT-Large.

Darcet, Timothée, et al. "Vision transformers need registers."

Outlier tokens carry less local information

- **Local smoothness breaks:** artifact patches have lower cosine similarity to their 4 neighbors than normal patches → they don't behave like "local patch" features.
- **Probing confirms it**: outlier tokens perform much worse at
  a. position prediction
  b. patch reconstruction → they encode less local patch info.



(a) Cosine similarity to neighbors.

	position prediction		reconstruction
	top-1 acc	avg. distance ↓	L2 error ↓
normal	**41.7**	**0.79**	**18.38**
outlier	22.8	5.09	25.23

(b) Linear probing for local information.

Figure 5: **(a)**: Distribution of cosine similarity between input patches and their 4 neighbors. We plot separately artifact patches (norm of the *output token* over 150) and normal patches. **(b)**: Local information probing on normal and outlier patch tokens. We train two models: one for predicting position, and one for reconstructing the input patch. Outlier tokens have much lower scores than the other tokens, suggesting they are storing less local patch information.

Darcet, Timothée, et al. "Vision transformers need registers."

The Solution: Register Tokens

Hypothesis: Why Do Artifacts Appear?

- **The model needs internal computation space**
- Large models learn to recognize redundant patches
- These tokens get recycled to store, process, and retrieve global information
- **Problem:** This discards local patch information needed for dense tasks
- **Solution:** Give the model dedicated tokens for this purpose!

Method: Adding Register Tokens

- **Simple architectural change:**
  - Add N learnable tokens after patch embedding (like [CLS] token)
  - Input: [CLS] + [REG1] + [REG2] + ... + [REGN] + patch tokens
  - Output: Discard register tokens, use only [CLS] and patches
- **Implementation:** Authors use 4 register tokens in experiments

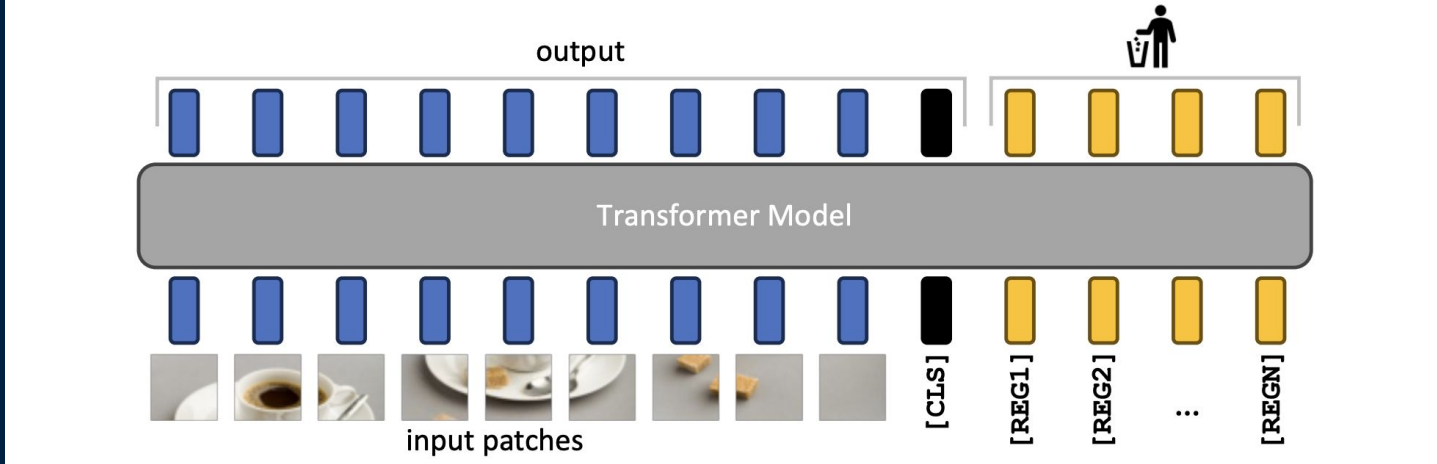


Figure 6: Illustration of the proposed remediation and resulting model. We add N additional learnable input tokens (depicted in yellow), that the model can use as *registers*. At the output of the model, only the patch tokens and `[CLS]` tokens are used, both during training and inference.

Darcet, Timothée, et al. "Vision transformers need registers."

Cost of adding registers is small

- Adds tokens → increases FLOPs/params, but params negligible.
- For N = 4 registers (used in most experiments), FLOPs increase is < 2%.

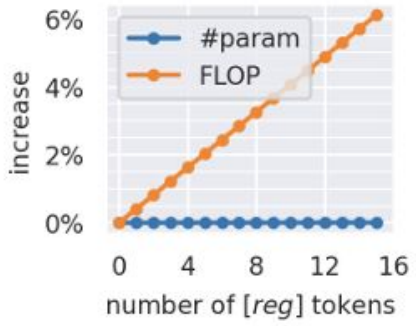


Figure 12: Increase in model parameter and FLOP count when adding different numbers of registers. Adding registers can increase model FLOP count by up to 6% for 16 registers. However, in the more common case of using 4 registers, that we use in most of our experiments, this increase is below 2%. In all cases, the increase in model parameters is negligible.

Darcet, Timothée, et al. "Vision transformers need registers."

Experiments

Training Algorithms and Data used

1. **OpenCLIP [1]** (2021)
    - Represents multimodal / text-supervised training.
2. **DEIT-III [2]** (ECCV, 2022)
    - Represents label-supervised training.
3. **DINOv2 [3]** (2023)
    - Represents self-supervised learning

[1] Ilharco, Gabriel, Mitchell Wortsman, Ross Wightman, Cade Gordon, et al. OpenCLIP. 2021.
[2] Touvron, Hugo, Matthieu Cord, and Hervé Jégou. DeiT III: Revenge of the ViT. European Conference on Computer Vision (ECCV), 2022.
[3] Oquab, Maxime, Timothée Darcet, Thibaut Lavril, et al. DINOv2: Learning Robust Visual Features without Supervision. 2023.

Experiments

Training Algorithms and Data used

1. **OpenCLIP**
    a. Trains models by aligning images with text labels using CLIP-style image–text contrastive learning.
    b. Uses licensed Shutterstock-based image–text dataset
    c. Uses ViT-B/16 image encoder with patch size 16x16
        i. embedding dimension: 768
        ii. transformer layers: 12
        iii. attention heads: 12
    d. Represents multimodal / text-supervised training.
2. **DEIT-III**
3. **DINOv2**

Experiments

Training Algorithms and Data used

1. **OpenCLIP**
2. **DEIT-III**
    a. Uses labeled images for training (standard classification setup)
    b. Trained on ImageNet-22k dataset
    c. Uses base ViT architecture (ViT-B configuration) with patch size 32x32
        i. embedding dimension: 768
        ii. transformer layers: 12
        iii. attention heads: 12
    d. Represents label-supervised training.
3. **DINOv2**

Experiments

Training Algorithms and Data used

1. **OpenCLIP**
2. **DEIT-III**
3. **DINOv2**
    a. Learns visual features without labels based on DINO self-supervised framework.
    b. Trained on ImageNet-22k.
    c. Uses larger ViT-L configuration.
        i. embedding dimension: 1024
        ii. transformer layers: 24
        iii. attention heads: 16
    d. Represents self-supervised learning.

Experiments

Overview

1. Do register tokens remove artifacts?
2. Do they hurt performance?
3. How many registers are optimal?
4. Do they help downstream tasks like object discovery?
5. What do registers actually learn?

Results

Exp 1: Norm Outliers Disappearance

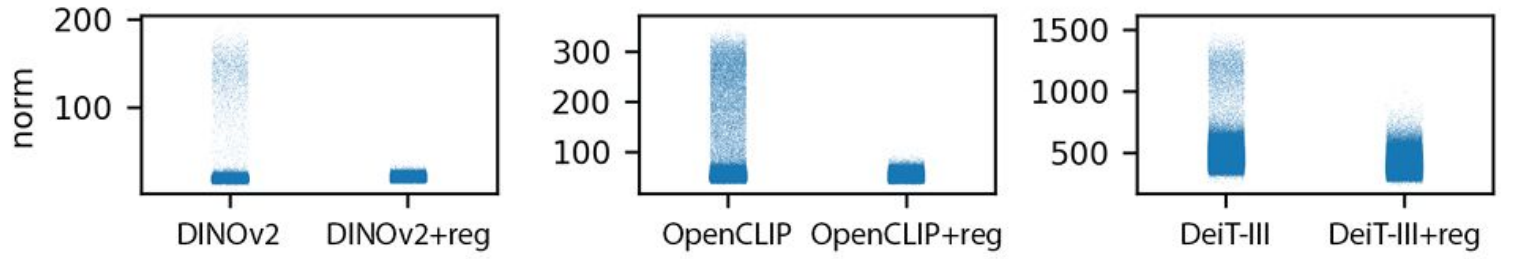


Figure 7: Effect of register tokens on the distribution of output norms on DINOv2, OpenCLIP and DeiT-III. Using register tokens effectively removes the norm outliers that were present previously.

- Without registers: some tokens have huge values (bad).
- With registers: distribution becomes smooth.
- Reason: registers absorb excess information.

Results

Exp 1: Reasoning

- **How often does this position become a high-norm outlier across many images?**

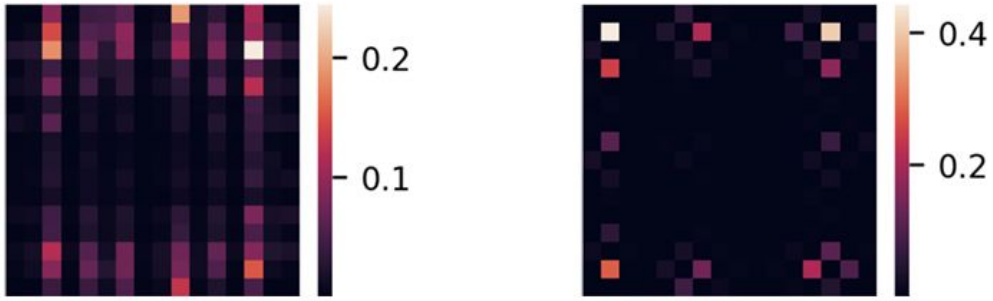


Figure 10: Feature norms along locations: proportion of tokens with norm larger than the cutoff value at a given location. Left: official DINOv2 model (no antialiasing), right: our models (with antialiasing). At some positions, more than 20% of tokens have a high norm.

- **Inference:** certain columns systematically produce outliers.
  - Real objects don’t naturally appear in stripes.
  - Generally the subject is in the middle of image

Results

Exp 1: Reasoning

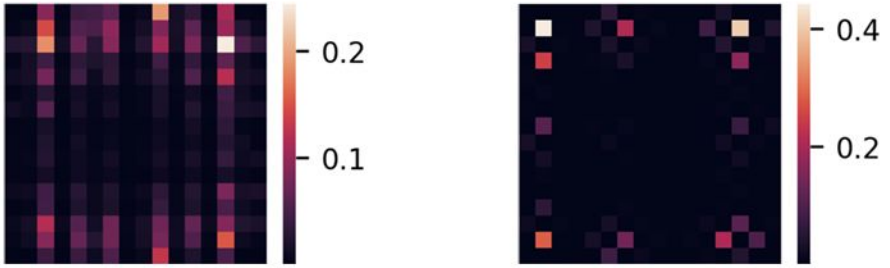


Figure 10: Feature norms along locations: proportion of tokens with norm larger than the cutoff value at a given location. Left: official DINOv2 model (no antialiasing), right: our models (with antialiasing). At some positions, more than 20% of tokens have a high norm.

- **Why does stripes happen?**
  - Positional embedding interpolation.
  - When image resolution or token count may change, no. of patches received changes so positional embedding must be interpolated accordingly
  - Since we use bicubic interpolation without antialiasing strips appears

Results

Exp 1: Proof for reason

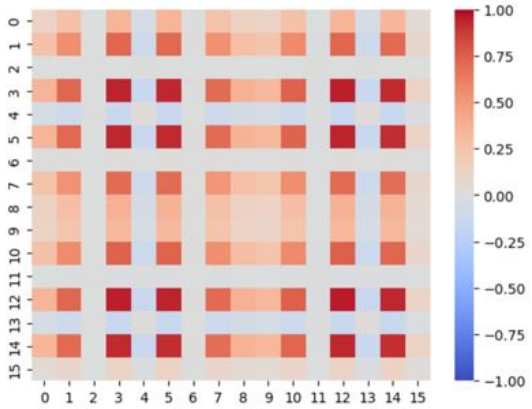


Figure 11: Propagating unit gradients through a bicubic interpolation ($16 \times 16 \rightarrow 7 \times 7$) without antialiasing. We observe a striping pattern similar to the one of Fig. 10 (left).

Results

Exp 1: Reasoning

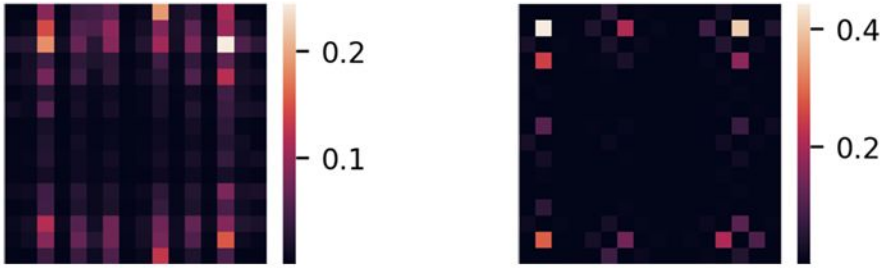
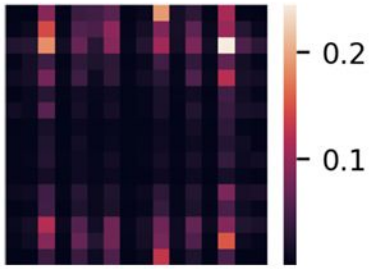


Figure 10: Feature norms along locations: proportion of tokens with norm larger than the cutoff value at a given location. Left: official DINOv2 model (no antialiasing), right: our models (with antialiasing). At some positions, more than 20% of tokens have a high norm.

- So, antialiasing solves the problem of stripes, fixing the artifacts
- **Why outliers appear near borders?**

Results

Exp 1: Reasoning



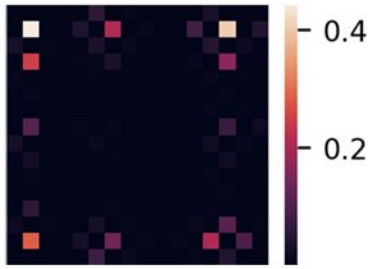
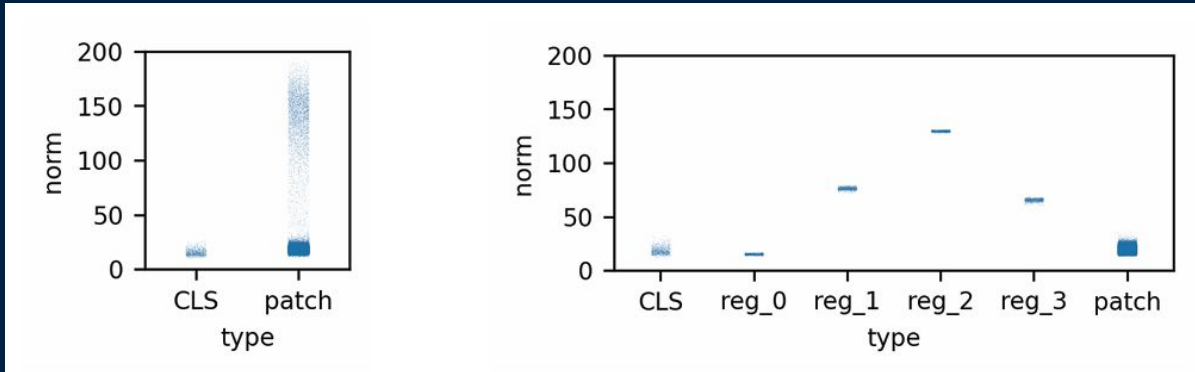


Figure 10: Feature norms along locations: proportion of tokens with norm larger than the cutoff value at a given location. Left: official DINOv2 model (no antialiasing), right: our models (with antialiasing). At some positions, more than 20% of tokens have a high norm.

- **Why outliers appear near borders?**
  - Transformer needs somewhere to store: global context, aggregated information.
  - Since no dedicated memory slots exist, model hijacks certain patch tokens.
  - Preferably the less important ones which are usually the background patches

Results

Exp 1: Reasoning



(a) DINOv2 - no register

(b) DINOv2 - 4 registers

Figure 15: Distribution of token norms for a DINOv2 model without (left) and with (right) 4 registers. Introducing registers entirely negates the high-norm outliers among the patch tokens.

- **Without registers:**
  - patch tokens contain extreme norms.
- **With registers:**
  - **p**atch tokens clean.
  - registers contain high norms.

Results

Exp 2: Performance Regression Check

	ImageNet Top-1	ADE20k mIoU	NYUd rmse ↓
DeiT-III	84.7	38.9	0.511
DeiT-III+reg	84.7	39.1	0.512
OpenCLIP	78.2	26.6	0.702
OpenCLIP+reg	78.1	26.7	0.661
DINOv2	84.3	46.6	0.378
DINOv2+reg	84.8	47.9	0.366

(a) Linear evaluation with frozen features.

	ImageNet Top-1
OpenCLIP	59.9
OpenCLIP+reg	60.1

(b) Zero-shot classification.

Table 2: Evaluation of downstream performance of the models that we trained, with and without registers. We consider linear probing of frozen features for all three models, and zero-shot evaluation for the OpenCLIP model. We see that using register not only does not degrade performance, but even improves it by a slight margin in some cases.

- Performance stays same or slightly improves.
- So, registers DO NOT harm performance.

Results

Exp 3: Ablating number of register tokens

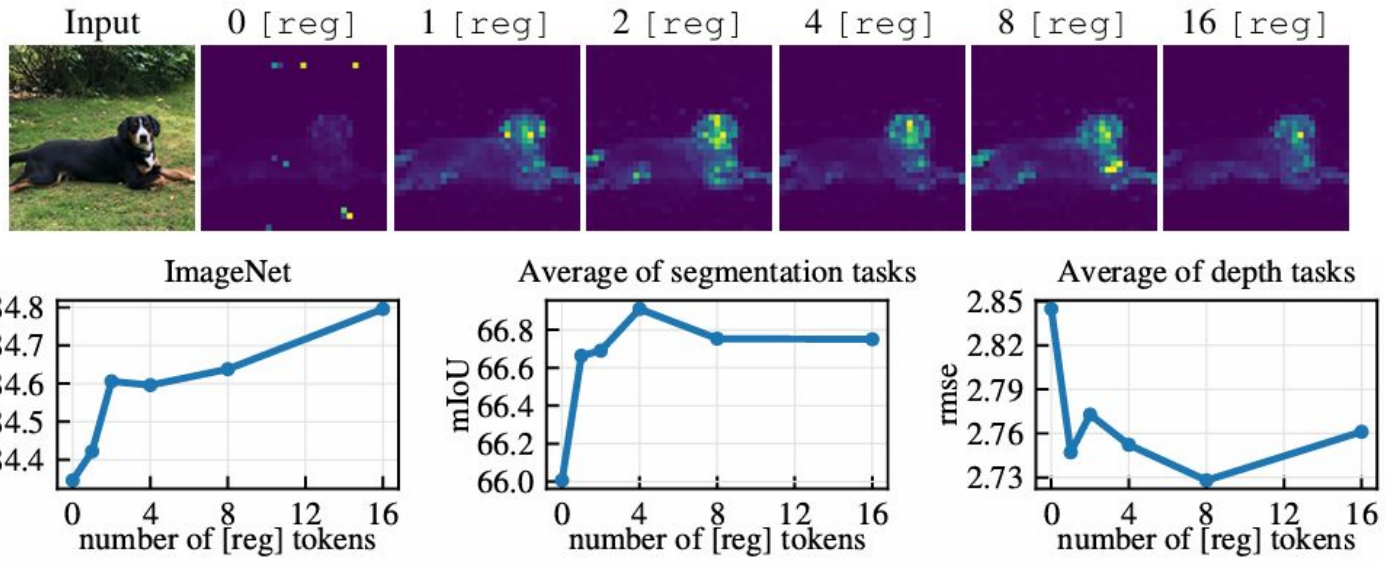


Figure 8: Ablation of the the number of register tokens used with a DINOv2 model. **(top)**: qualitative visualization of artifacts appearing as a function of number of registers. **(bottom)**: performance on three tasks (ImageNet, ADE-20k and NYUd) as a function of number of registers used. While one register is sufficient to remove artefacts, using more leads to improved downstream performance.

Results

Exp 3: Ablating number of register tokens

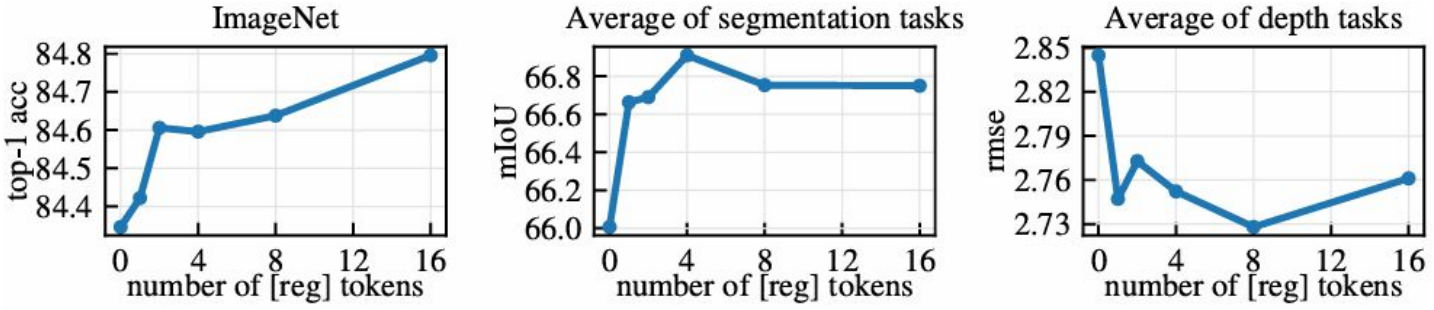


Figure 8: Ablation of the the number of register tokens used with a DINOv2 model. **(top)**: qualitative visualization of artifacts appearing as a function of number of registers. **(bottom)**: performance on three tasks (ImageNet, ADE-20k and NYUd) as a function of number of registers used. While one register is sufficient to remove artefacts, using more leads to improved downstream performance.

- Just ONE register removes artifacts
- Classification improves with more registers
- More registers sometimes improve dense tasks

Results

Exp 3: Ablating number of register tokens

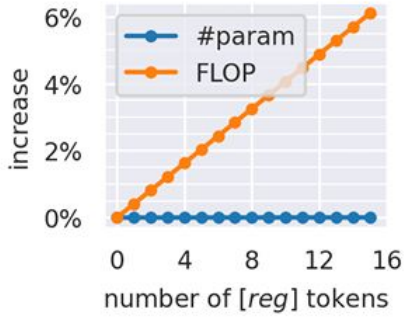


Figure 12: Increase in model parameter and FLOP count when adding different numbers of registers. Adding registers can increase model FLOP count by up to 6% for 16 registers. However, in the more common case of using 4 registers, that we use in most of our experiments, this increase is below 2%. In all cases, the increase in model parameters is negligible.

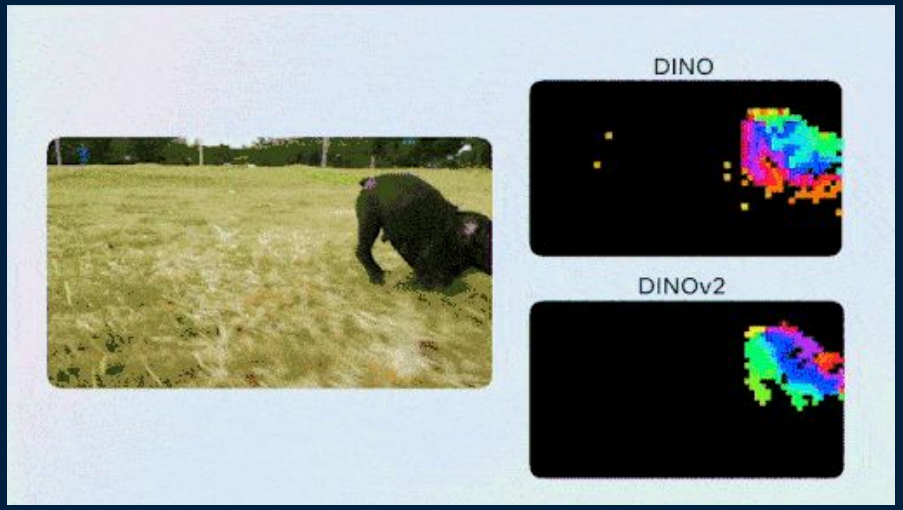
- parameter increase ≈ negligible.
- FLOPs increase small.

Results

Exp 4: Do they help in object discovery?

What is object discovery?

finding objects in images without labels



Results

Exp 4: Do they help in object discovery?

	VOC 2007	VOC 2012	COCO 20k
DeiT-III	11.7	13.1	10.7
DeiT-III+reg	27.1	32.7	25.1
OpenCLIP	38.8	44.3	31.0
OpenCLIP+reg	37.1	42.0	27.9
DINOv2	35.3	40.2	26.9
DINOv2+reg	55.4	60.0	42.0

Table 3: Unsupervised Object Discovery using LOST (Siméoni et al., 2021) on models with and without registers. We evaluated three types of models trained with various amounts of supervision on VOC 2007, 2012 and COCO. We measure performance using corloc. We observe that adding register tokens makes all models significantly more viable for usage in object discovery.

Results

Exp 4: Do they help in object discovery?

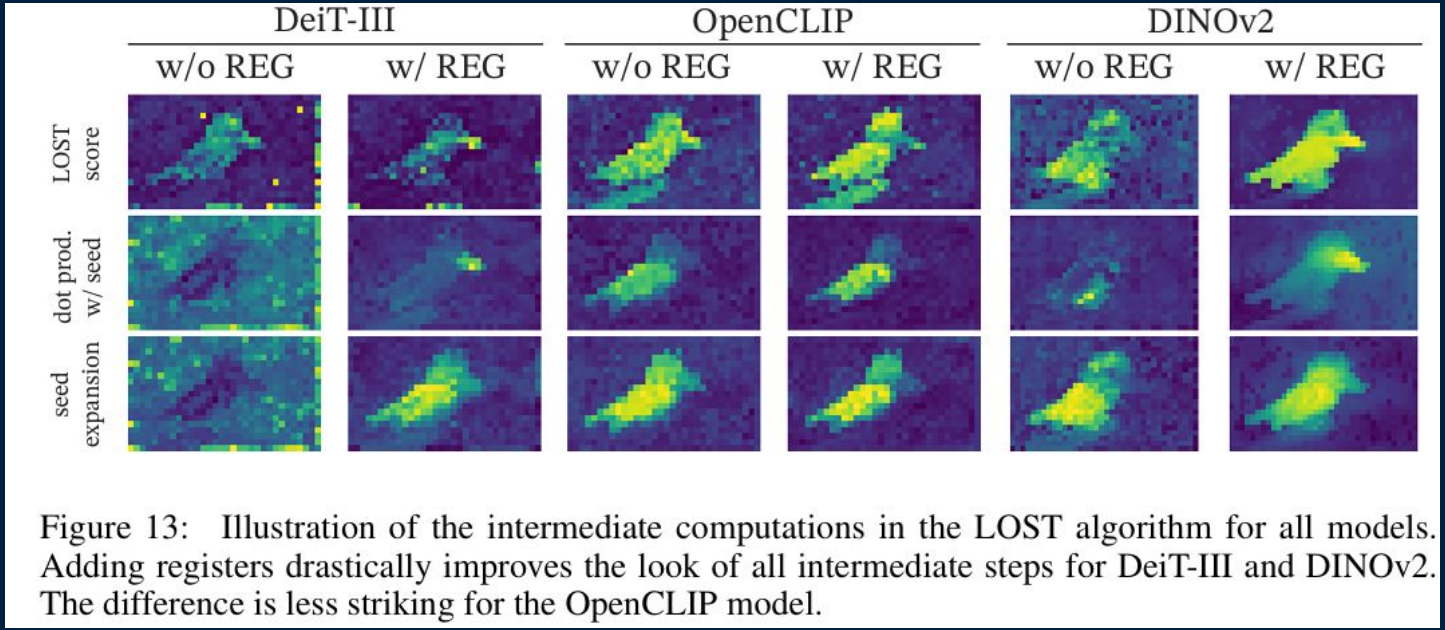


Figure 13: Illustration of the intermediate computations in the LOST algorithm for all models. Adding registers drastically improves the look of all intermediate steps for DeiT-III and DINOv2. The difference is less striking for the OpenCLIP model.

Darcet, Timothée, et al. "Vision transformers need registers."

- **Without registers:**
  - some patch tokens become high-norm outliers which distort similarity computation.
  - So dot products become noisy and seed selection becomes unstable.

Results

Exp 4: Do they help in object discovery?

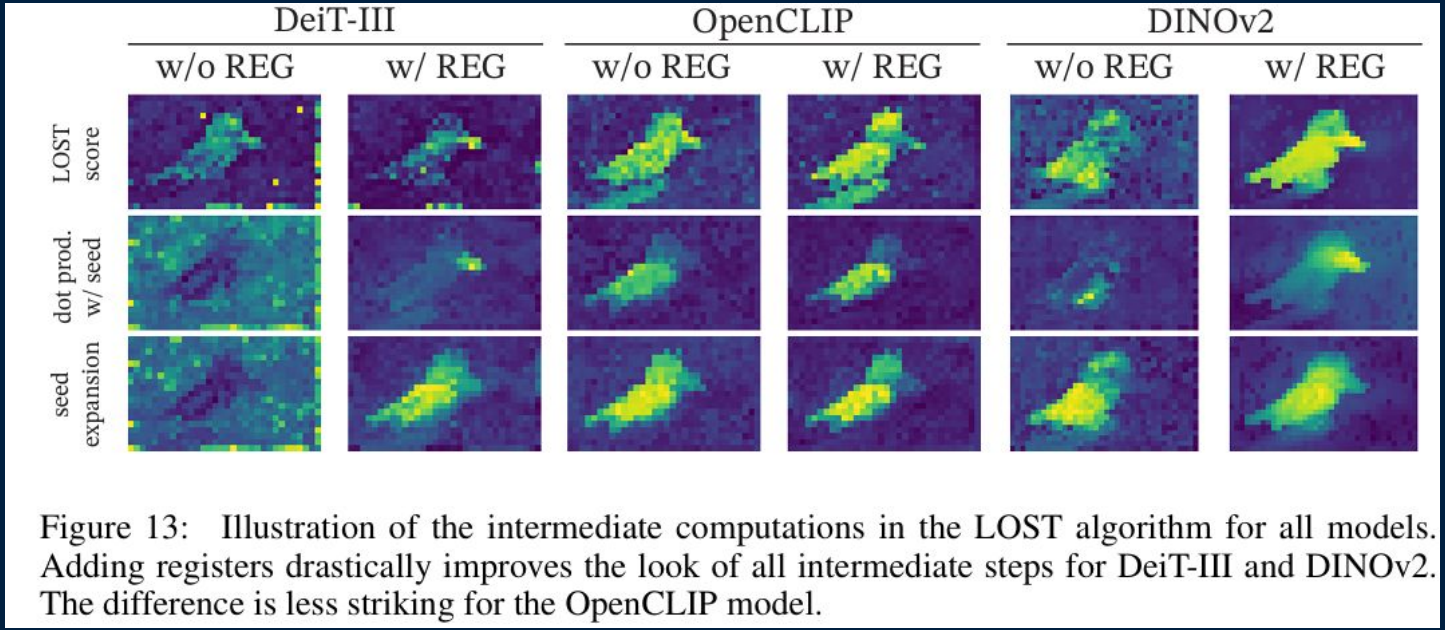


Figure 13: Illustration of the intermediate computations in the LOST algorithm for all models. Adding registers drastically improves the look of all intermediate steps for DeiT-III and DINOv2. The difference is less striking for the OpenCLIP model.

Darcet, Timothée, et al. "Vision transformers need registers."

- Registers fix this by absorbing global information and cleaning patch representations.
- So, LOST dramatically improves.

Results

Exp 4: Do they help in object discovery?

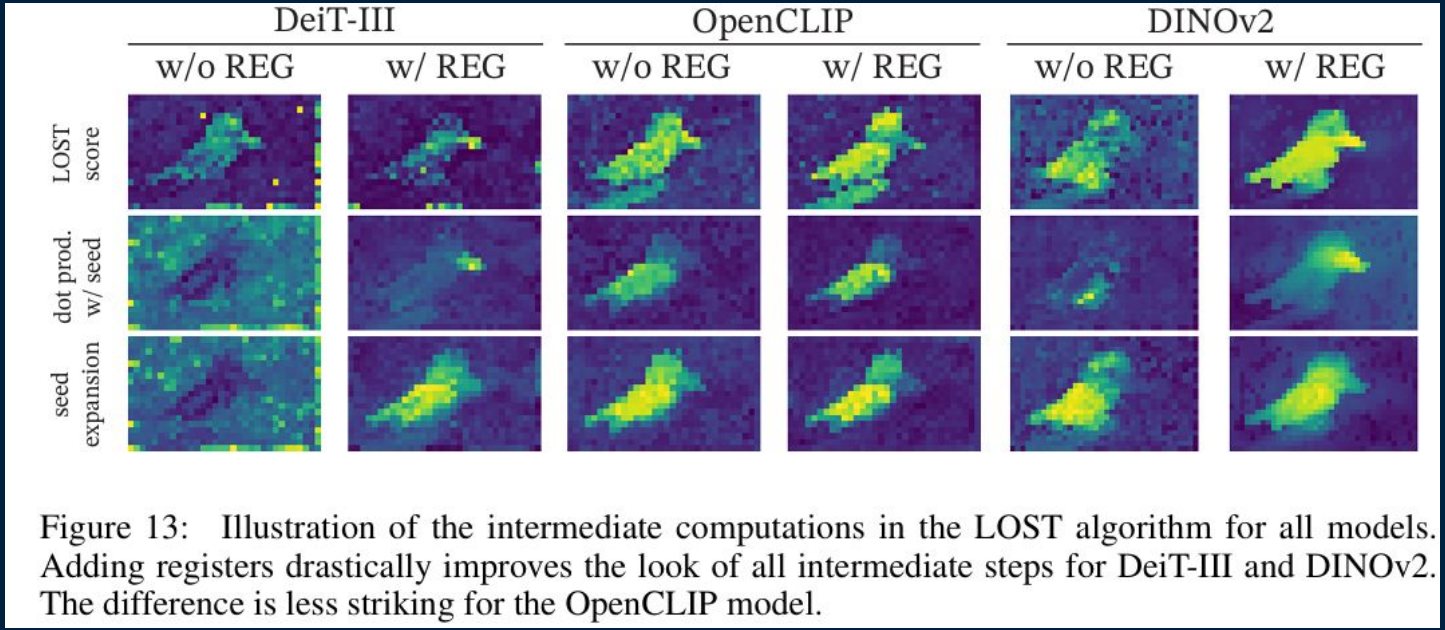


Figure 13: Illustration of the intermediate computations in the LOST algorithm for all models. Adding registers drastically improves the look of all intermediate steps for DeiT-III and DINOv2. The difference is less striking for the OpenCLIP model.

Darcet, Timothée, et al. "Vision transformers need registers."

- In OpenCLIP, value projection filters outliers automatically and local feature geometry already stable.
- So, registers give smaller improvement in LOST score

Results

Exp 5: What Do Registers Learn?

1. Registers behave differently.
   - Some attend to specific objects
   - Some focus on different regions
2. But nothing has been enforced

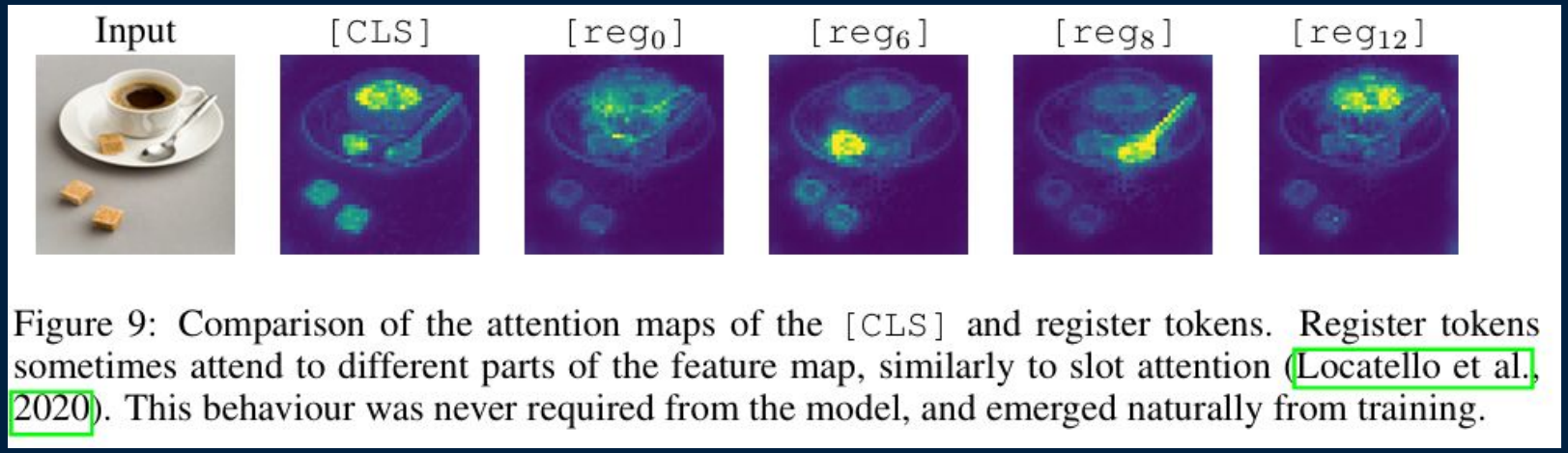


Figure 9: Comparison of the attention maps of the [CLS] and register tokens. Register tokens sometimes attend to different parts of the feature map, similarly to slot attention (Locatello et al., 2020). This behaviour was never required from the model, and emerged naturally from training.

Darcet, Timothée, et al. "Vision transformers need registers."

Take-home: registers “decouple” global scratch space from local patches

- **Artifacts = high-norm outlier** patch tokens (measurable, bimodal).
- **Hypothesis**: large ViTs reuse redundant patches as compute/memory slots.
- **Registers give dedicated slots** → outliers move into registers → patch grid becomes clean.
- **Practical outcome**: smoother local features → better dense/object discovery; minimal compute cost.

Paper Strengths

- **Simple architectural change**, broadly applicable
- **Strong interpretability** story: connects artifacts ↔ norms ↔ behaviour
- **Empirical validation** across multiple training paradigms (supervised, text-supervised, self-supervised)
- **Clear cost/benefit tradeoff** (4 regs: <2% FLOPs)

Limitations

- **Root cause not fully pinned down**: authors note they couldn't fully determine which training aspects trigger artifacts.
- **Not all benefits are equally strong**: OpenCLIP improvements can be smaller because value projection may filter outliers.
- Not equally efficient on models: Effect on OpenCLIP is much less than other models.
- Requires to retrain the model from scratch with additional register tokens.

Questions From Us

- Root cause: What exactly triggers outlier tokens?
  a. Is it model size, training length, augmentation, positional embedding interpolation, optimizer dynamics, or data bias?
  b. The paper shows correlations, but not a causal diagnosis.
- What do registers actually represent?
  a. Are they “global summary,” background, positional anchors, or something like scratchpad memory?
  b. Do different registers specialize (and if yes, why)?
- Why does outliers appears in Q and K spaces, but not in V space?

Suggestion For Future Work

- Adaptive / learned number of registers:
    a. Instead of fixed N, use a mechanism to allocate registers dynamically (e.g., gating, token pruning + register insertion) based on image complexity or outlier score.
- Regularize registers for diversity / interpretability:
    a. Encourage specialization: orthogonality/entropy regularizers, diversity loss, or encourage each register to attend to different regions.
    b. Then show “register i consistently captures X”.