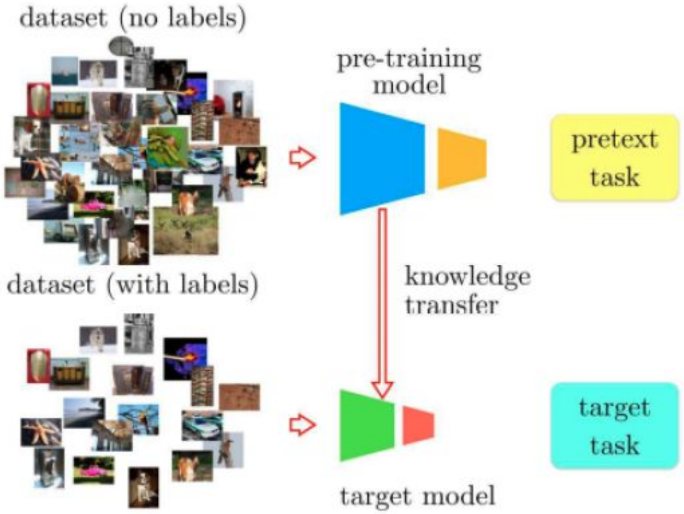
Emerging Properties in Self-Supervised Vision Transformers

Presenters: Nicholas Boyer, Levi Harris, Soumitri Chattopadhyay

Date: 04/01/2024

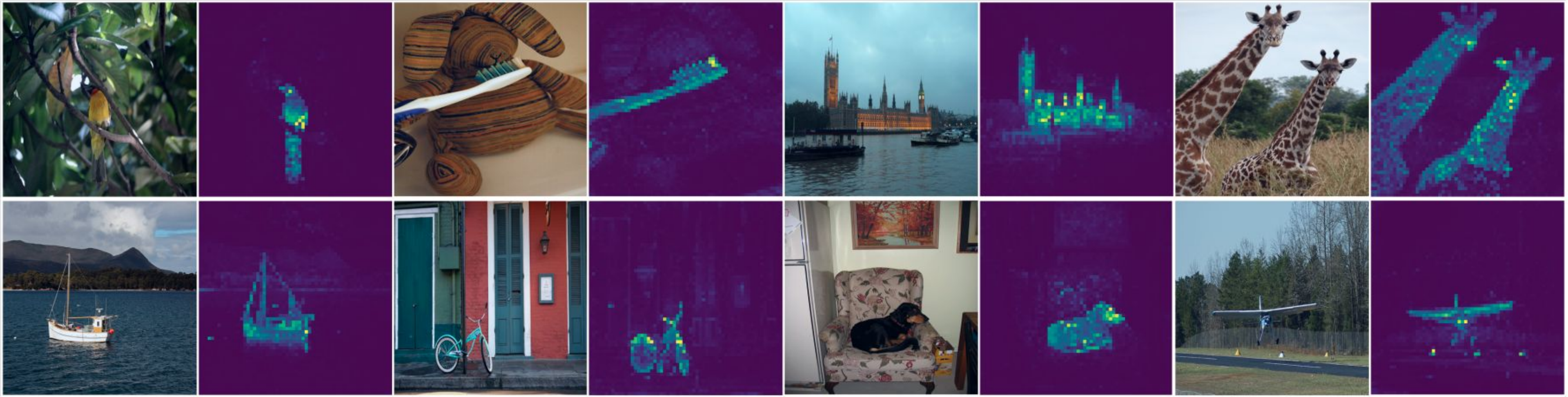
Background

- Previous works pre-train ViT using labeled data
- Popular NLP models (e.g., BERT, GPT, etc.) use *self-supervised* pre-training
  - No labels needed
  - End-to-end language modeling
- **The big question**: can self-supervised pre-training extend from NLP to Vision?



Motivation

- **Unique properties** emerge from representations learned by ViTs when trained using self-supervision, such as **scene layout** and **salient object knowledge**.



Self-attention map visualisations from a self-supervised ViT.

Motivation

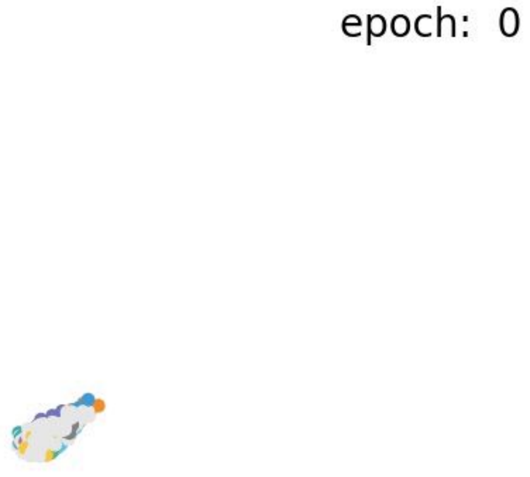
- These unique properties are *exclusive to self-supervised ViTs*; not shown by CNNs or supervised models.

Original video | Supervised segmentation model | DINO attention maps

Motivation

- Self-supervised learned representations implicitly contain visual **concept categorizations**.

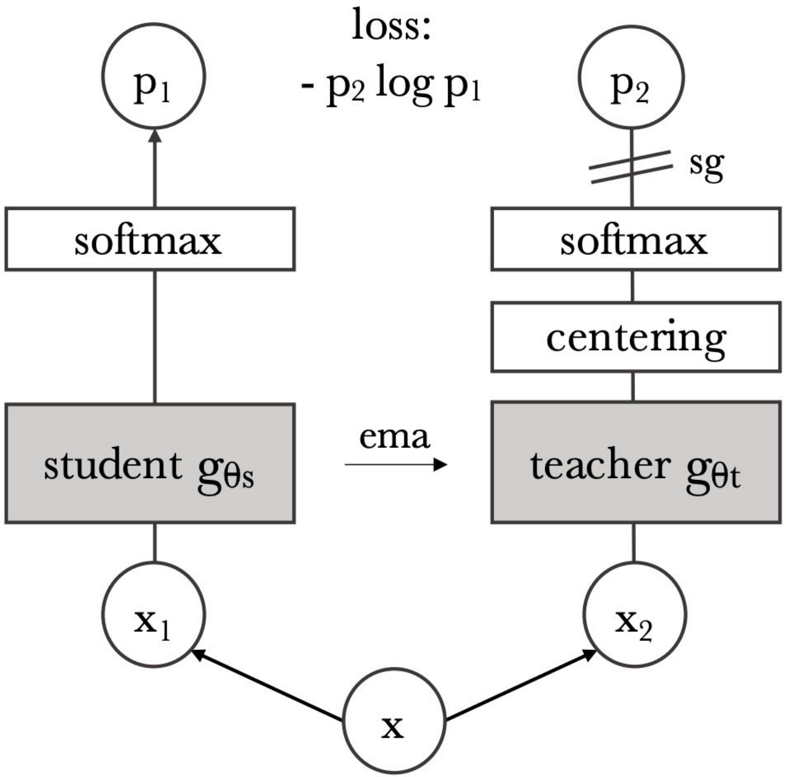
- Plotting *t*-SNE of features show *clustering of similar categories*, even without explicit class labels.



Methodology

<u>At a glance:</u>

- DINO: self-**di**stillation with **no** labels
- *Same model* for student & teacher
- Negative samples **not** required i.e. *non-contrastive* (different from VATT!)
- Learning via knowledge distillation paradigm



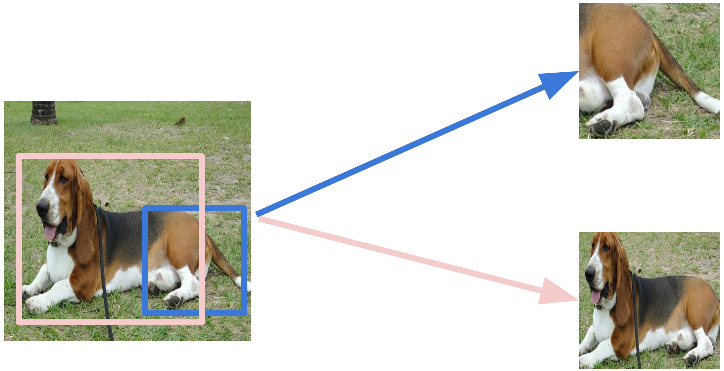
Methodology

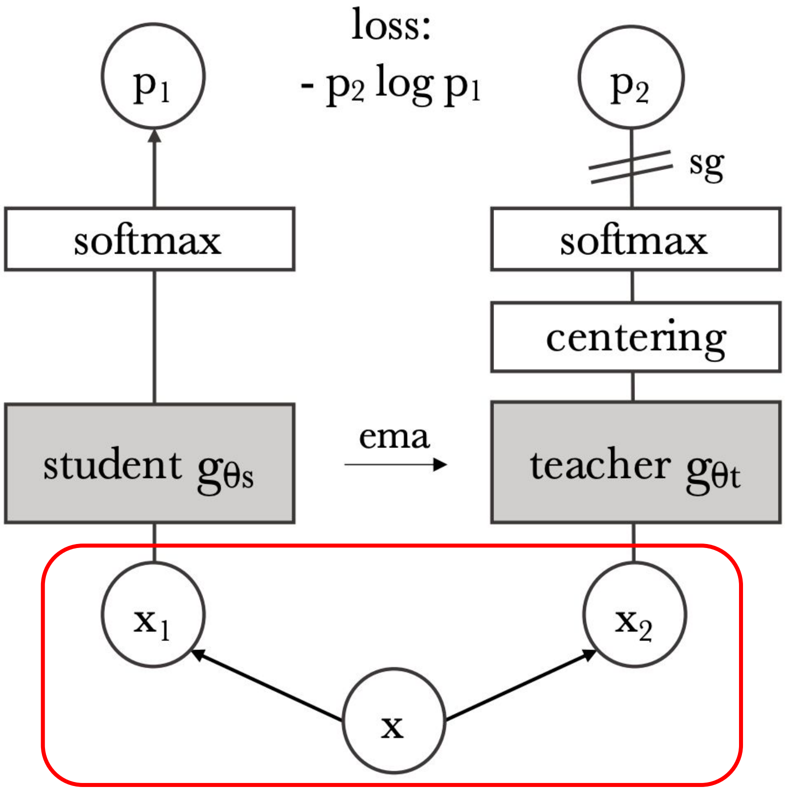
Augmentation

- Color jitter, solarization, rotations
- **Multi-cropping**

Global crop: covers > 50% of image

Local crop: covers < 50% of image



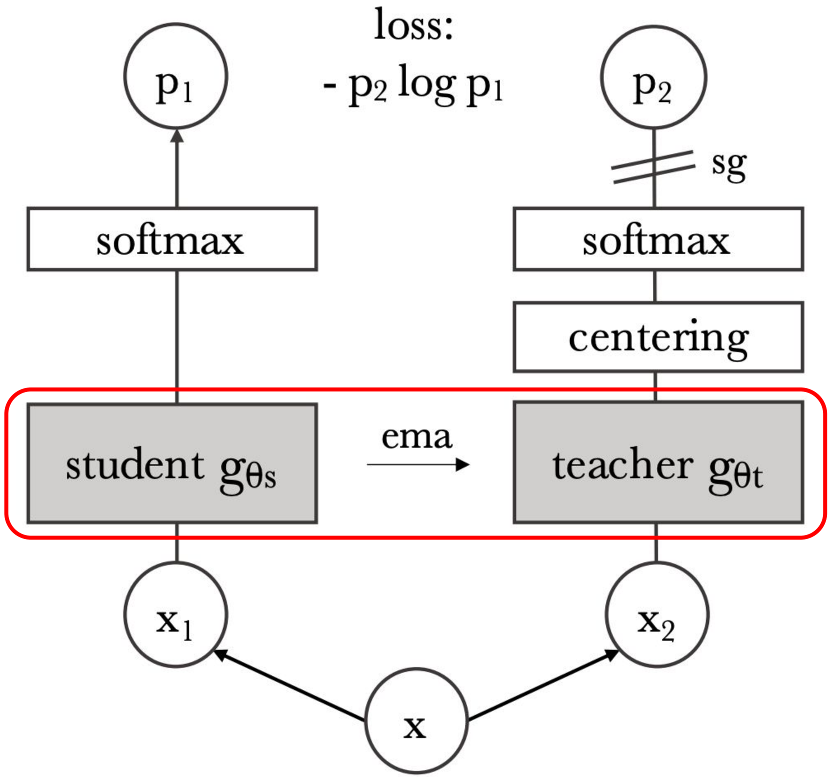


Methodology

Momentum encoder

- Teacher model is not trained, only student is trained
- Weights of teacher constructed via EMA of student weights
- Has a stabilising effect over training

$$\theta_t \leftarrow \lambda\theta_t + (1-\lambda)\theta_s$$



Methodology

Knowledge Distillation

- **Align** student network output with teacher network
- Softmax over outputs ⇒ probability distributions

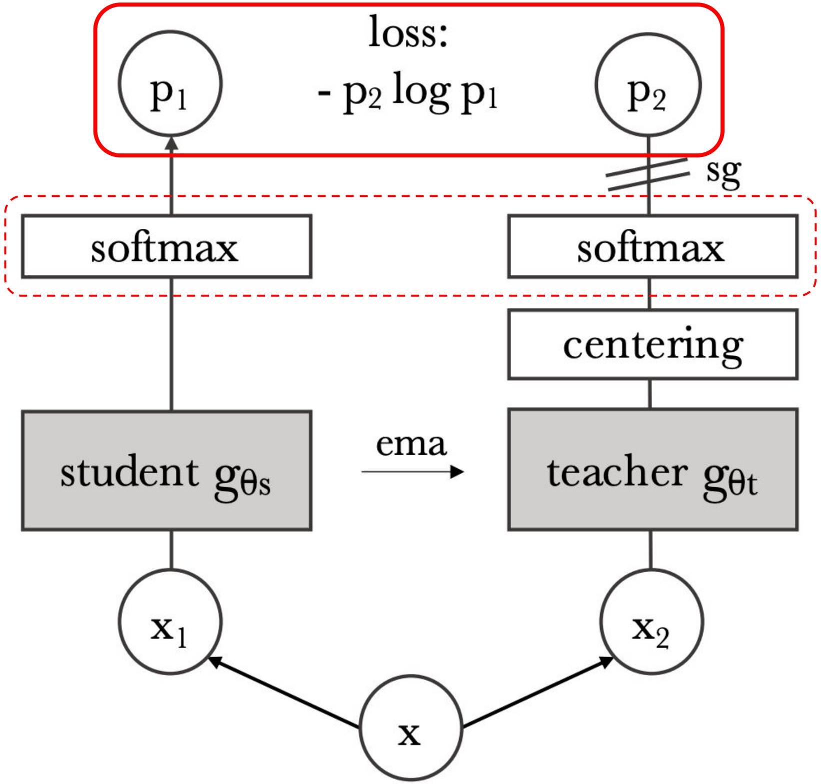
$$P_s(x)^{(i)} = \frac{\exp(g_{\theta_s}(x)^{(i)}/\tau_s)}{\sum_{k=1}^{K}\exp(g_{\theta_s}(x)^{(k)}/\tau_s)}$$

- Teacher predictions act as **soft labels** for student predictions ⇒ optimize with *cross-entropy loss*

$$\min_{\theta_s} H(P_t(x), P_s(x)) \quad \text{where } H(a,b) = -a\log b$$

```
s1, s2 = gs(x1), gs(x2) # student output n-by-K
t1, t2 = gt(x1), gt(x2) # teacher output n-by-K

loss = H(t1, s2)/2 + H(t2, s1)/2
```



Methodology

Stop-gradient

- Gradients flow *only* through student branch
- Stop-gradient induces *asymmetry* in the model
- Helps preventing model collapse

Algorithm 1 DINO PyTorch pseudocode w/o multi-crop.

```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, m: network and center momentum rates
gt.params = gs.params
for x in loader: # load a minibatch x with n samples
    x1, x2 = augment(x), augment(x) # random views

    s1, s2 = gs(x1), gs(x2) # student output n-by-K
    t1, t2 = gt(x1), gt(x2) # teacher output n-by-K

    loss = H(t1, s2)/2 + H(t2, s1)/2
    loss.backward() # back-propagate

    # student, teacher and center updates
    update(gs) # SGD
    gt.params = l*gt.params + (1-l)*gs.params
    C = m*C + (1-m)*cat([t1, t2]).mean(dim=0)

def H(t, s):
    t = t.detach() # stop gradient
    s = softmax(s / tps, dim=1)
    t = softmax((t - C) / tpt, dim=1) # center + sharpen
    return - (t * log(s)).sum(dim=1).mean()
```

Methodology

Stop-gradient

- Gradients flow *only* through student branch
- Stop-gradient induces *asymmetry* in the model
- Helps preventing model collapse

What is Model collapse?

$f_\theta(I) = f_\theta(\mathrm{augment}(I)) \longrightarrow f_\theta(I) = \mathrm{constant}$

Satisfies the invariance property, but not useful

Algorithm 1 DINO PyTorch pseudocode w/o multi-crop.

```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, m: network and center momentum rates
gt.params = gs.params
for x in loader: # load a minibatch x with n samples
    x1, x2 = augment(x), augment(x) # random views

    s1, s2 = gs(x1), gs(x2) # student output n-by-K
    t1, t2 = gt(x1), gt(x2) # teacher output n-by-K

    loss = H(t1, s2)/2 + H(t2, s1)/2
    loss.backward() # back-propagate

    # student, teacher and center updates
    update(gs) # SGD
    gt.params = l*gt.params + (1-l)*gs.params
    C = m*C + (1-m)*cat([t1, t2]).mean(dim=0)

def H(t, s):
    t = t.detach() # stop gradient
    s = softmax(s / tps, dim=1)
    t = softmax((t - C) / tpt, dim=1) # center + sharpen
    return - (t * log(s)).sum(dim=1).mean()
```

Methodology

Centering

- *Prevents* any *one dimension* from *dominating* the softmax output
- Interpretation: adding a bias term 'c' to the teacher; updated via EMA
- *Encourages collapse to a uniform distribution*

$$c \leftarrow mc + (1-m)\frac{1}{B}\sum_{i=1}^{B} g_{\theta_t}(x_i)$$

$$g_t(x) \leftarrow g_t(x) + c$$

Algorithm 1 DINO PyTorch pseudocode w/o multi-crop.

```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, m: network and center momentum rates
gt.params = gs.params
for x in loader: # load a minibatch x with n samples
    x1, x2 = augment(x), augment(x) # random views

    s1, s2 = gs(x1), gs(x2) # student output n-by-K
    t1, t2 = gt(x1), gt(x2) # teacher output n-by-K

    loss = H(t1, s2)/2 + H(t2, s1)/2
    loss.backward() # back-propagate

    # student, teacher and center updates
    update(gs) # SGD
    gt.params = l*gt.params + (1-l)*gs.params
    C = m*C + (1-m)*cat([t1, t2]).mean(dim=0)

def H(t, s):
    t = t.detach() # stop gradient
    s = softmax(s / tps, dim=1)
    t = softmax((t - C) / tpt, dim=1) # center + sharpen
    return - (t * log(s)).sum(dim=1).mean()
```

Methodology

Softmax sharpening

- *Low temperature* value for *teacher* in its softmax
- **"Sharpens"** the softmax (i.e. *encourages* one dimension to dominate)
- *Prevents* collapse to uniform distribution

$$P_s(x)^{(i)} = \frac{\exp(g_{\theta_s}(x)^{(i)}/\tau_s)}{\sum_{k=1}^{K} \exp(g_{\theta_s}(x)^{(k)}/\tau_s)}$$

Centering & softmax sharpening **counteract** *each other!*

Algorithm 1 DINO PyTorch pseudocode w/o multi-crop.

```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, m: network and center momentum rates
gt.params = gs.params
for x in loader: # load a minibatch x with n samples
    x1, x2 = augment(x), augment(x) # random views

    s1, s2 = gs(x1), gs(x2) # student output n-by-K
    t1, t2 = gt(x1), gt(x2) # teacher output n-by-K

    loss = H(t1, s2)/2 + H(t2, s1)/2
    loss.backward() # back-propagate

    # student, teacher and center updates
    update(gs) # SGD
    gt.params = l*gt.params + (1-l)*gs.params
    C = m*C + (1-m)*cat([t1, t2]).mean(dim=0)

def H(t, s):
    t = t.detach() # stop gradient
    s = softmax(s / tps, dim=1)
    t = softmax((t - C) / tpt, dim=1) # center + sharpen
    return - (t * log(s)).sum(dim=1).mean()
```

Experimental setup

Model:

- Standard ViT architecture, with 8x8 or 16x16 resolution patches as input
- Features obtained from [CLS] token

Implementation:

- ImageNet pre-training, self-supervised, batch size = 1024; 300 epochs
- AdamW optimizer, learning rate = 0.0005 * BS / 256; cosine schedule decay
- temperature = 0.1; linear warm-up from 0.04 → 0.07
- Augmentations: color jitter, blurring, solarization, multi-crop

Evaluation:

- (a) Linear evaluation (b) fine-tuning evaluation (c) k-NN classification (zero-shot)

Quantitative Evaluation

Method	Arch.	Param.	im/s	Linear	k-NN
Supervised	RN50	23	1237	79.3	79.3
SCLR [12]	RN50	23	1237	69.1	60.7
MoCov2 [15]	RN50	23	1237	71.1	61.9
InfoMin [67]	RN50	23	1237	73.0	65.3
BarlowT [81]	RN50	23	1237	73.2	66.0
OBoW [27]	RN50	23	1237	73.8	61.9
BYOL [30]	RN50	23	1237	74.4	64.8
DCv2 [10]	RN50	23	1237	75.2	67.1
SwAV [10]	RN50	23	1237	**75.3**	65.7
DINO	RN50	23	1237	**75.3**	**67.5**
Supervised	ViT-S	21	1007	79.8	79.8
BYOL* [30]	ViT-S	21	1007	71.4	66.6
MoCov2* [15]	ViT-S	21	1007	72.7	64.4
SwAV* [10]	ViT-S	21	1007	73.5	66.3
DINO	ViT-S	21	1007	**77.0**	**74.5**
Comparison across architectures					
SCLR [12]	RN50w4	375	117	76.8	69.3
SwAV [10]	RN50w2	93	384	77.3	67.3
BYOL [30]	RN50w2	93	384	77.4	–
DINO	ViT-B/16	85	312	78.2	76.1
SwAV [10]	RN50w5	586	76	78.5	67.1
BYOL [30]	RN50w4	375	117	78.6	–
BYOL [30]	RN200w2	250	123	79.6	73.9
DINO	ViT-S/8	21	180	79.7	**78.3**
SCLRv2 [13]	RN152w3+SK	794	46	79.8	73.1
DINO	ViT-B/8	85	63	**80.1**	77.4

- DINO outperforms prior SSL on *same architectures*
- ViT-DINO is superior to RN50-SSL methods; requires *fewer parameters*
- DINO with 8x8 patches fare better than 16x16 patches
- k-NN evaluation *(zero-shot, no further training)* shows how powerful & robust features are learned by DINO
- DINO with ViT comes closest to supervised setting

Table 2: **Linear and k-NN classification on ImageNet.** We report top-1 accuracy for linear and k-NN evaluations on the validation set of ImageNet for different self-supervised methods. We focus on ResNet-50 and ViT-small architectures, but also report the best results obtained across architectures. * are run by us. We run the k-NN evaluation for models with official released weights. The throughput (im/s) is calculated on a NVIDIA V100 GPU with 128 samples per forward. Parameters (M) are of the feature extractor.

Downstream tasks: Image Retrieval & Copy Detection

- Use **raw features** from DINO w/o any fine-tuning → kNN for **image retrieval** and cosine similarity for **copy detection**.
- Results show these **raw features** are sufficient to yield very competitive performance / outperform prior SOTA.

Table 3: **Image retrieval.** We compare the performance in retrieval of off-the-shelf features pretrained with supervision or with DINO on ImageNet and Google Landmarks v2 (GLDv2) dataset. We report mAP on revisited Oxford and Paris. Pretraining with DINO on a landmark dataset performs particularly well. For reference, we also report the best retrieval method with off-the-shelf features [57].

			$\mathcal{R}$Ox		$\mathcal{R}$Par	
Pretrain	Arch.	Pretrain	M	H	M	H
Sup. [57]	RN101+R-MAC	ImNet	49.8	18.5	74.0	**52.1**
Sup.	ViT-S/16	ImNet	33.5	8.9	63.0	37.2
DINO	ResNet-50	ImNet	35.4	11.1	55.9	27.5
DINO	ViT-S/16	ImNet	41.8	13.7	63.1	34.4
DINO	ViT-S/16	GLDv2	**51.5**	**24.3**	**75.3**	51.6

Table 4: **Copy detection.** We report the mAP performance in copy detection on Copydays "strong" subset [21]. For reference, we also report the performance of the multigrain model [5], trained specifically for particular object retrieval.

Method	Arch.	Dim.	Resolution	mAP
Multigrain [5]	ResNet-50	2048	224^2	75.1
Multigrain [5]	ResNet-50	2048	largest side 800	82.5
Supervised [69]	ViT-B/16	1536	224^2	76.4
DINO	ViT-B/16	1536	224^2	81.7
DINO	ViT-B/8	1536	320^2	**85.5**

Downstream task: Video Instance Segmentation

- No fine-tuning, use of raw features
- Achieves highest accuracy on INet
- Visualising attention maps (next slide) across video frames show precise segmentation

Method	Data	Arch.	$(\mathcal{J}\&\mathcal{F})_m$	$\mathcal{J}_m$	$\mathcal{F}_m$
Supervised					
ImageNet	INet	ViT-S/8	66.0	63.9	68.1
STM [48]	I/D/Y	RN50	81.8	79.2	84.3
Self-supervised					
CT [71]	VLOG	RN50	48.7	46.4	50.0
MAST [40]	YT-VOS	RN18	65.5	63.3	67.6
STC [37]	Kinetics	RN18	67.6	64.8	70.2
DINO	INet	ViT-S/16	61.8	60.2	63.4
DINO	INet	ViT-B/16	62.3	60.7	63.9
DINO	INet	ViT-S/8	**69.9**	**66.6**	**73.1**
DINO	INet	ViT-B/8	**71.4**	**67.9**	**74.9**

Table 5: **DAVIS 2017 Video object segmentation.** We evaluate the quality of frozen features on video instance tracking. We report mean region similarity $\mathcal{J}_m$ and mean contour-based accuracy $\mathcal{F}_m$. We compare with existing self-supervised methods and a supervised ViT-S/8 trained on ImageNet. Image resolution is 480p.

All these results obtained from mere raw DINO features showcases its powerful feature encoding abilities!

Qualitative Visualisations

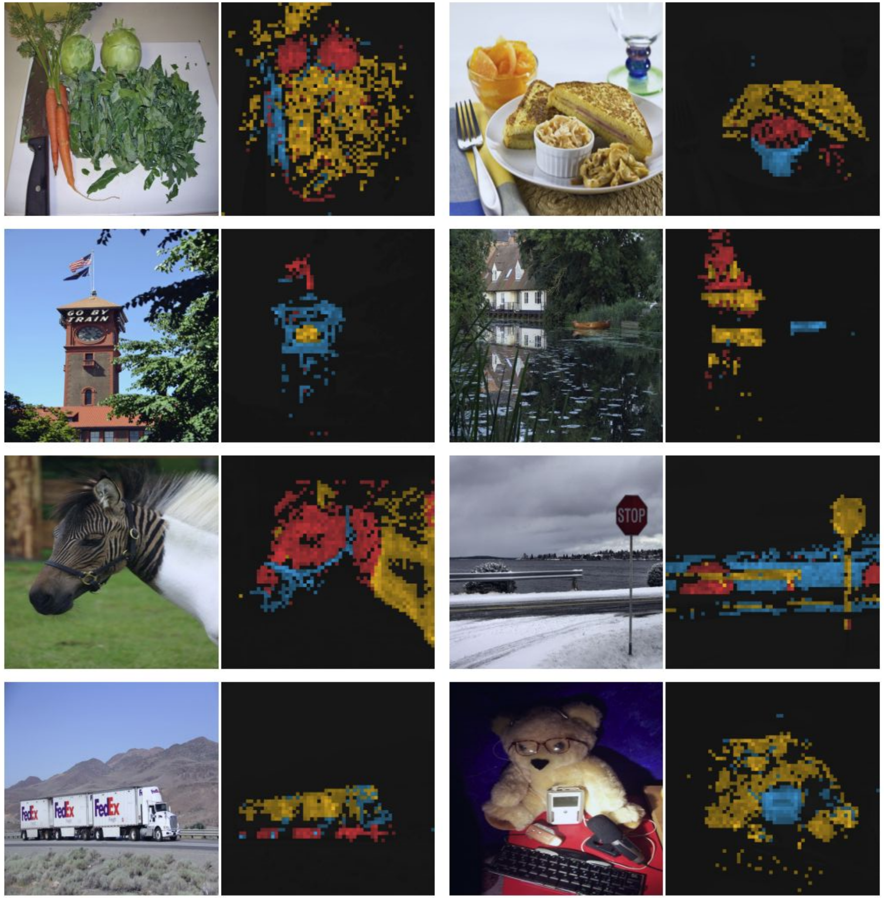
These self-attention maps for selected heads were generated using DINO with videos of a horse, a BMX rider, a puppy, and a fishing boat.

Qualitative Visualisations

DINO focuses on the foreground object even in highly ambiguous situations.

Multiple head Attention maps

- We visualise multiple heads from self-attention maps of DINO
- Different heads denoted by different colors
- Multiple heads learn complementary features of a given image (adjacent figure)



Application: Weakly-supervised segmentation

- We visualise self-attention maps from DINO and compare w/ a supervised ViT model.
- Self-attention maps from DINO are far superior!
- DINO attention maps can be used as weak labels for segmentation models

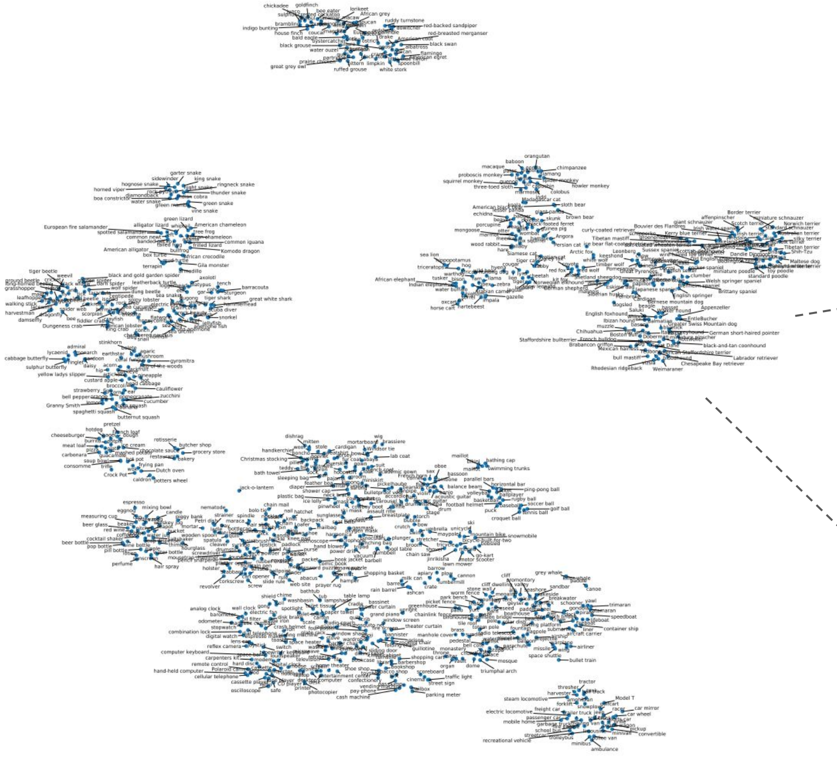
	Random	Supervised	DINO
ViT-S/16	22.0	27.3	45.9
ViT-S/8	21.8	23.7	44.7

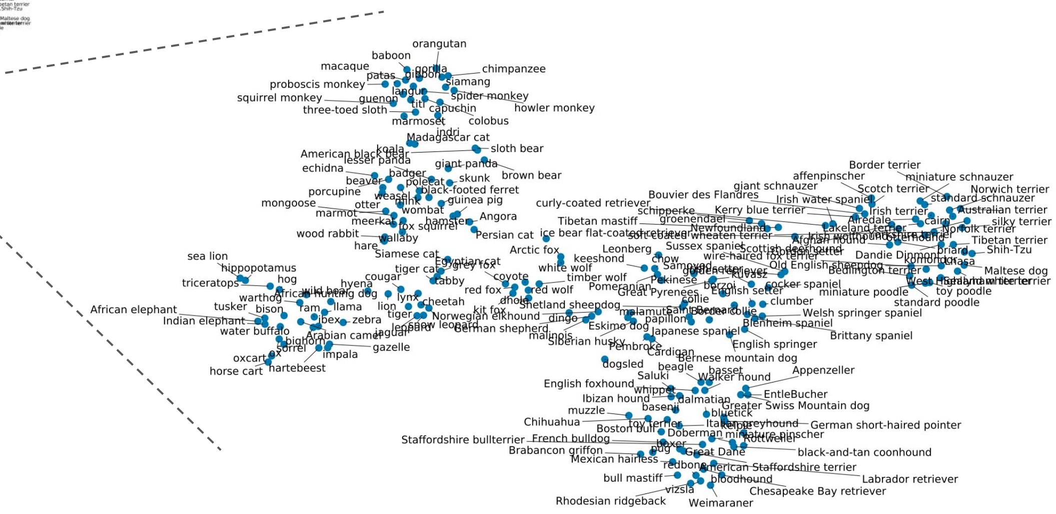
Transfer Learning

Table 6: **Transfer learning by finetuning pretrained models on different datasets.** We report top-1 accuracy. Self-supervised pretraining with DINO transfers better than supervised pretraining.

	Cifar_{10}	Cifar_{100}	INat_{18}	INat_{19}	Flwrs	Cars	INet
ViT-S/16							
Sup. [69]	**99.0**	89.5	70.7	76.6	98.2	92.1	79.9
DINO	**99.0**	**90.5**	**72.0**	**78.2**	**98.5**	**93.0**	**81.5**
ViT-B/16							
Sup. [69]	99.0	90.8	**73.2**	77.7	98.4	92.1	81.8
DINO	**99.1**	**91.7**	72.6	**78.6**	**98.8**	**93.0**	**82.8**

t-SNE visualization





- This suggests that the model managed to connect categories based on visual properties, a bit like humans do.

DINO Component Ablations

- Momentum encoder vital for k-NN classification
- Ablations improve representation quality
- Multi-Crop gives significant boost
- CE for the win!

	Method	Mom.	SK	MC	Loss	Pred.	k-NN	Lin.
1	DINO	✓	✗	✓	CE	✗	72.8	76.1
2		✗	✗	✓	CE	✗	0.1	0.1
3		✓	✓	✓	CE	✗	72.2	76.0
4		✓	✗	✗	CE	✗	67.9	72.5
5		✓	✗	✓	MSE	✗	52.6	62.4
6		✓	✗	✓	CE	✓	71.8	75.6
7	BYOL	✓	✗	✗	MSE	✓	66.6	71.4
8	MoCov2	✓	✗	✗	INCE	✗	62.0	71.6
9	SwAV	✗	✓	✓	CE	✗	64.7	71.8

SK: Sinkhorn-Knopp, MC: Multi-Crop, Pred.: Predictor
CE: Cross-Entropy, MSE: Mean Square Error, INCE: InfoNCE

Summing up…

This paper presents DINO, a new recipe for self-supervised training of ViTs

Alleviates the dependency on a large batch for negative samples (as in contrastive learning) and presents a rather simplified form of self-supervised training

Two unique and very useful properties emerge:

- SSL-trained ViTs *implicitly learn visual scene layout*, evident from its attention maps ⇒ can be used for segmentation/object tracking.
- Raw features learned by DINO are *powerful and highly discriminative*, evident from the (a) t-SNE plots (b) k-NN evaluations for classification/retrieval tasks.

Thank you!

Questions?