Paper Review

Emerging Properties in Self-Supervised Vision Transformers

YeongHyeon Park

Department of Electrical and Computer Engineering

SungKyunKwan University

2021 ICCV OCTOBER 11-17 VIRTUAL

facebook Inria SORBONNE UNIVERSITÉ

Emerging Properties in Self-Supervised Vision Transformers

Mathilde Caron[1,2] Hugo Touvron[1,3] Ishan Misra[1] Hervé Jegou[1]
Julien Mairal[2] Piotr Bojanowski[1] Armand Joulin[1]

[1] Facebook AI Research [2] Inria* [3] Sorbonne University

Sorbonne University

Outline

- **Summary**
- **DINO**
- **Experiments**
- **Appendix A.** How does DINO work?
- **Appendix B.** DINO Use case
- **Appendix C.** DINO Example (w/ MNIST)
- **Appendix D.** Easy way for negative sampling (from SimCLR)

Summary

Summary

Proposal

- **Self-distillation with no labels:** enable training process with few or no label
- **Softmax Centering:** reduce score dominance on a specific dimension
- **Softmax Sharpening:** reduce uniform distributed score

Contributions

- Pre-trained DINO ViT can be applied to various tasks.
  1. Image classification
  2. Image retrieval
  3. Copy detection
  4. Object segmentation
- DINO framework does not need full labeling due to self-distillation technique.
- The authors present a comparison across the architectures.
  1. Different **backbone** architecture within the **same DINO framework** (ResNet vs ViT).
  2. Different **patch sizes** within the **same ViT architecture** (16 vs 8).
  3. Different **ViT** while using the **same patch size** (ViT-B vs ViT-S).

Limitations

- Readers should already understand prior studies.
  - But, we are ready!

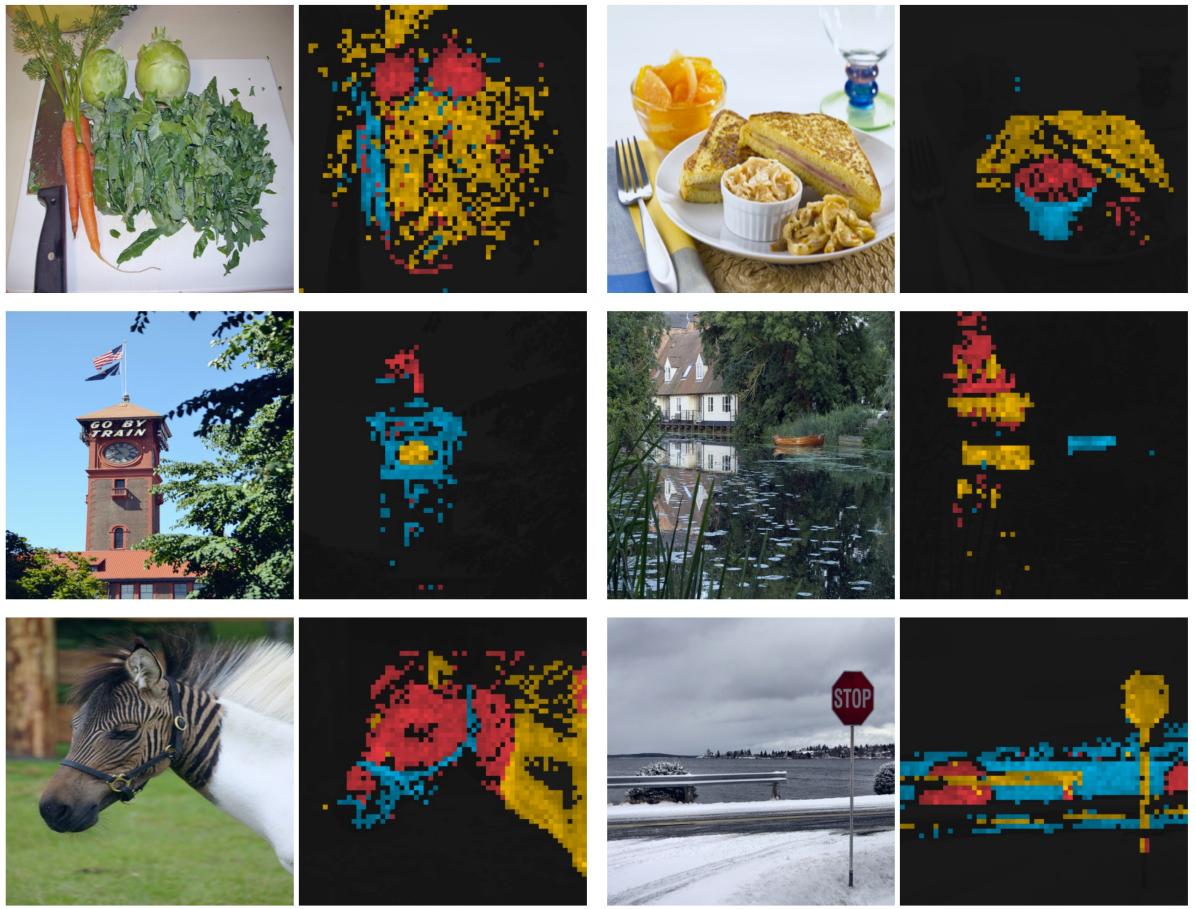
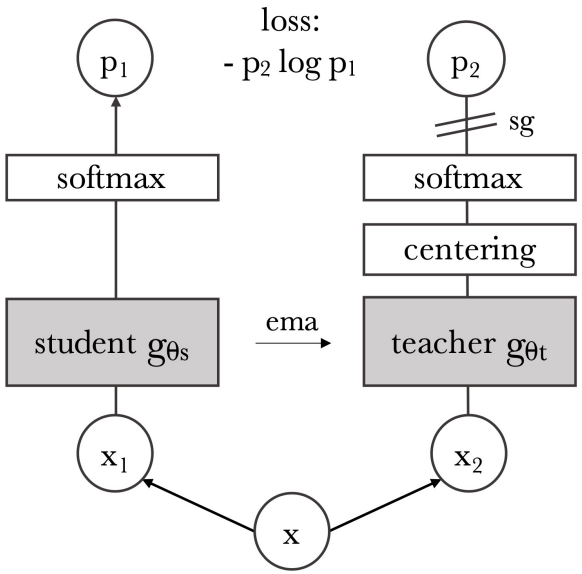


Figure 3: **Attention maps from multiple heads.** We consider the heads from the last layer of a ViT-S/8 trained with DINO and display the self-attention for `[CLS]` token query. Different heads, materialized by different colors, focus on different locations that represents different objects or parts (more examples in Appendix).

DINO

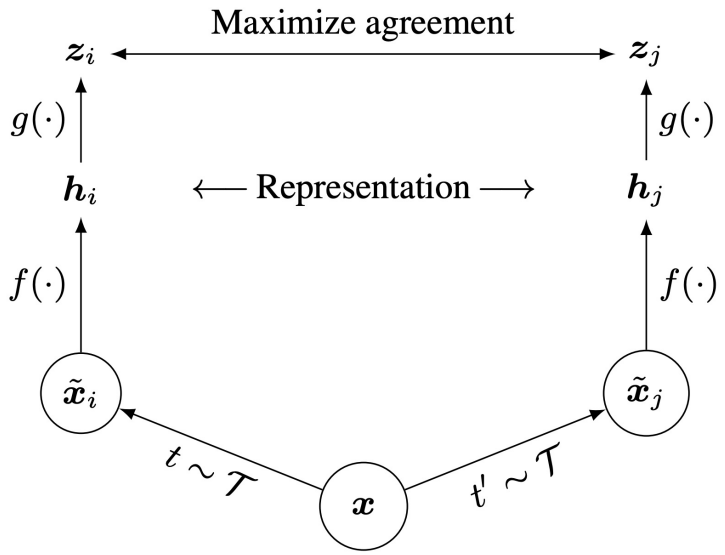
Self-Distillation with No Labels

Self-Distillation



DINO

Figure 2: **Self-distillation with no labels.** We illustrate DINO in the case of one single pair of views (x_1, x_2) for simplicity. The model passes two different random transformations of an input image to the student and teacher networks. Both networks have the same architecture but different parameters. The output of the teacher network is centered with a mean computed over the batch. Each networks outputs a K dimensional feature that is normalized with a temperature softmax over the feature dimension. Their similarity is then measured with a cross-entropy loss. We apply a stop-gradient (sg) operator on the teacher to propagate gradients only through the student. The teacher parameters are updated with an exponential moving average (ema) of the student parameters.



SimCLR

Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder network $f(\cdot)$ and a projection head $g(\cdot)$ are trained to maximize agreement using a contrastive loss. After training is completed, we throw away the projection head $g(\cdot)$ and use encoder $f(\cdot)$ and representation $\boldsymbol{h}$ for downstream tasks.

Self-Supervised Learning

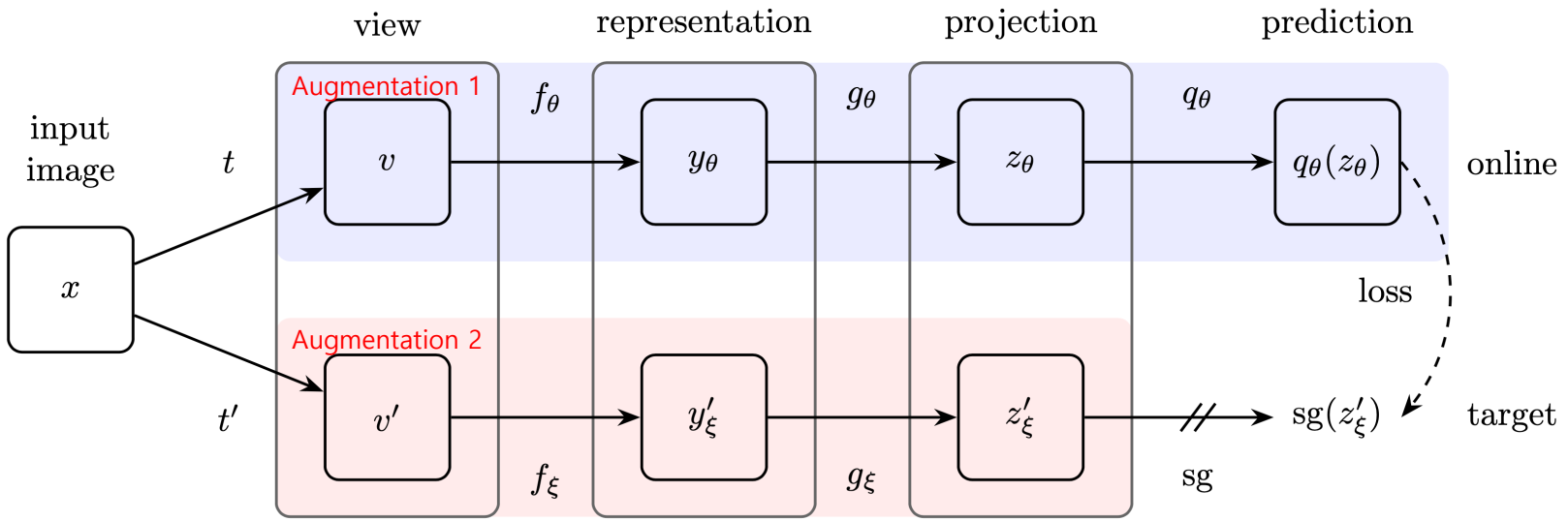


Figure 2: BYOL's architecture. BYOL minimizes a similarity loss between $q_\theta(z_\theta)$ and $\mathrm{sg}(z'_\xi)$, where θ are the trained weights, ξ are an exponential moving average of θ and sg means stop-gradient. At the end of training, everything but f_θ is discarded, and y_θ is used as the image representation.

- Train online network to predict latent representation of target network
- Update target network by copying parameter of the online network.
- After whole training process, only the online network will be used for inference.

Why EMA?

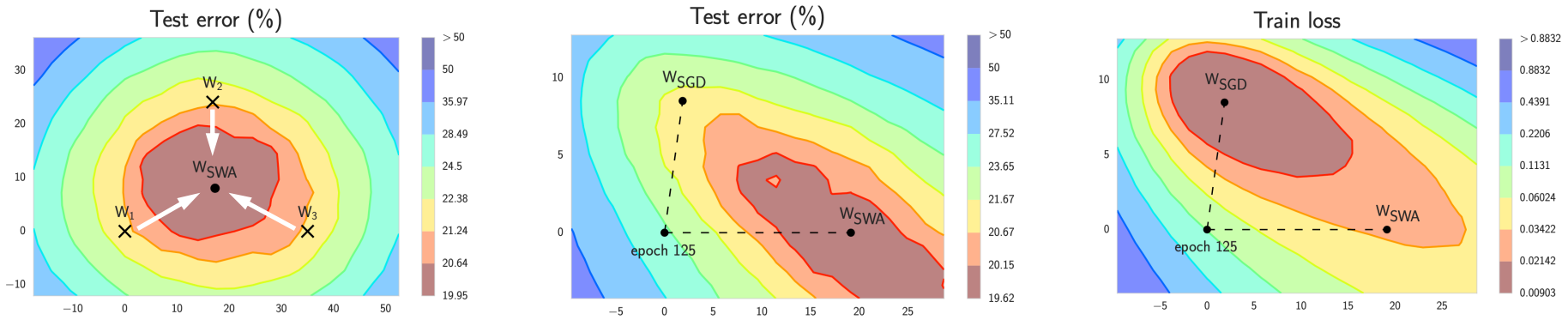
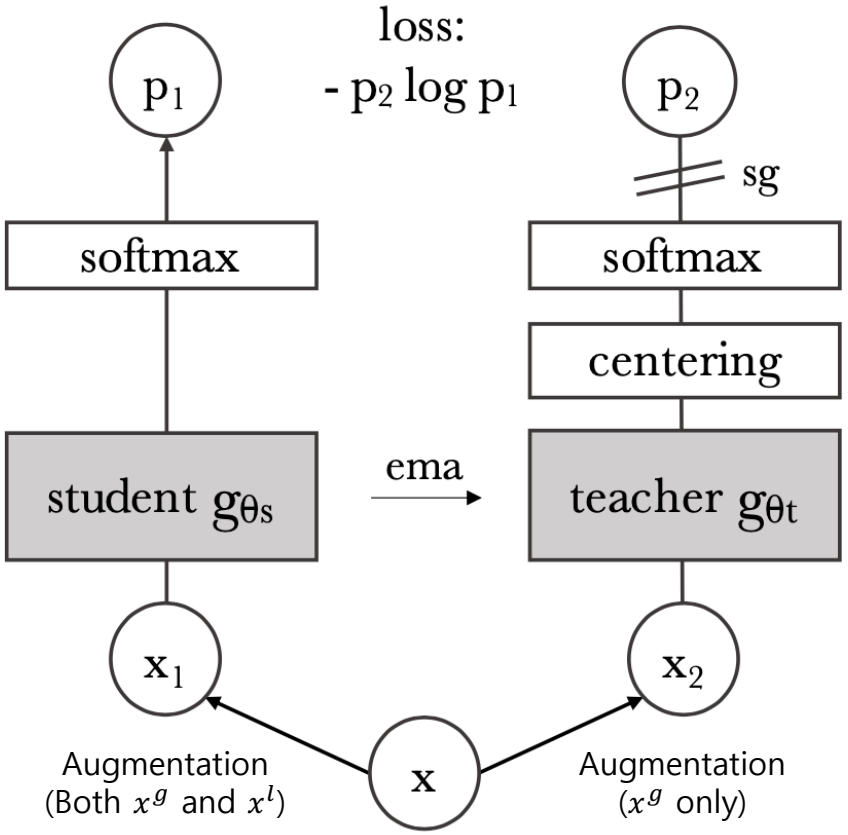


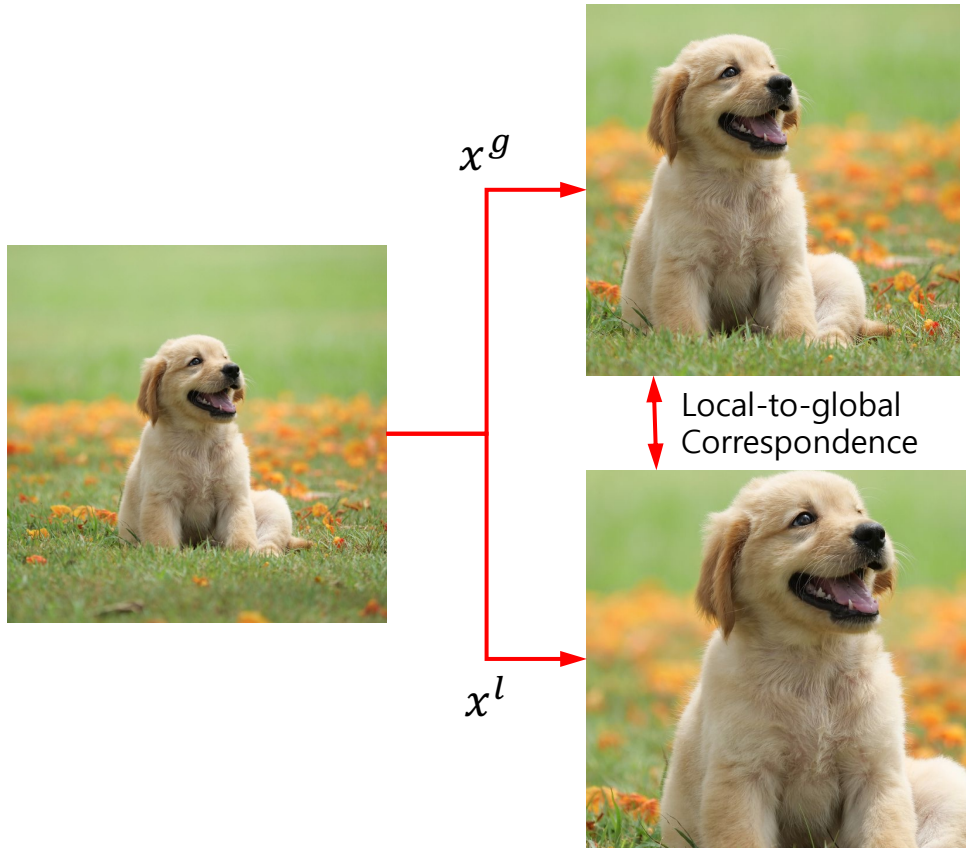
Figure 1: Illustrations of SWA and SGD with a Preactivation ResNet-164 on CIFAR-100[1]. **Left**: test error surface for three FGE samples and the corresponding SWA solution (averaging in weight space). **Middle** and **Right**: test error and train loss surfaces showing the weights proposed by SGD (at convergence) and SWA, starting from the same initialization of SGD after 125 training epochs.

Weight averaging leads model into optimal solution.

DINO Framework

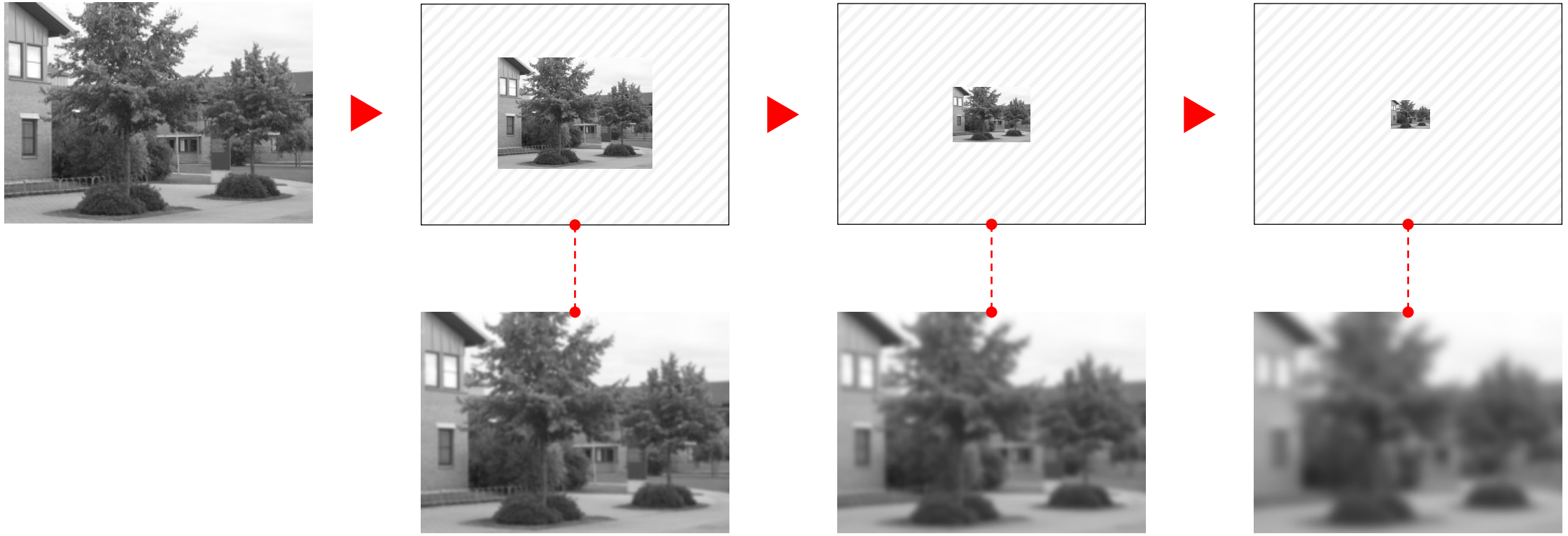


- x^g: global augmentation
- x^l: local augmentation (same as global but crop tightly)



https://en.wikipedia.org/wiki/Scale_space

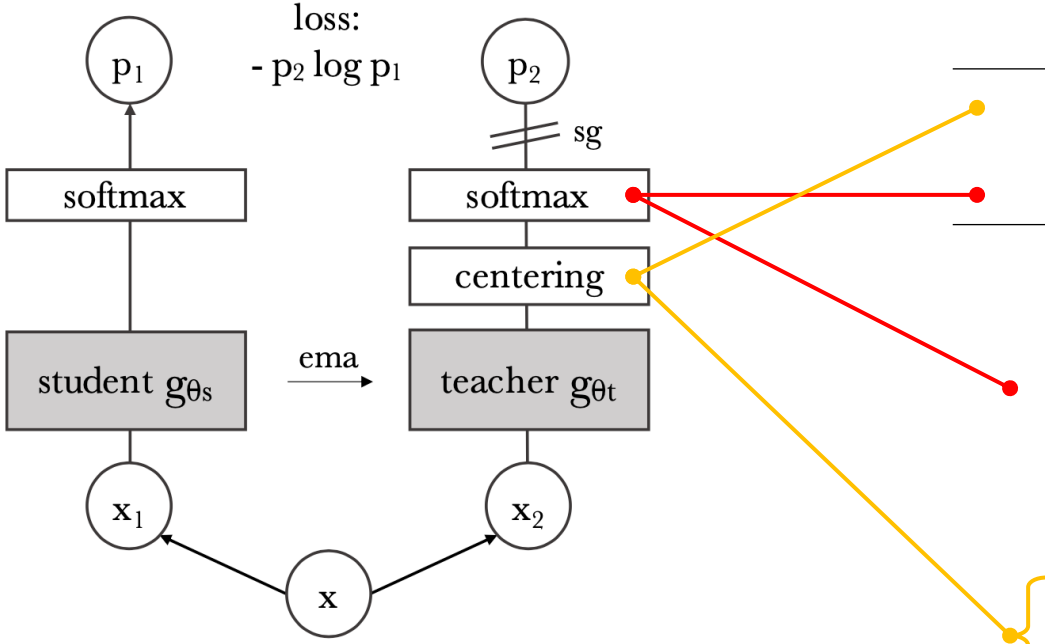
Scale Space



Avoiding Collapse

- Two known collapse problems (due to the same architecture of student and teacher)
  - Uniform distributed scoring
  - Score dominance on a specific dimension

- Centering and sharpening have opposite effects on avoiding collapse.
  - Thus, both should be employed for effect balancing.



	Uniform scoring	Dominant scoring
Centering	Worse	Ease
Sharpening	Ease	Worse

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

$$c \leftarrow mc + (1-m)\frac{1}{B}\sum_{i=1}^{B} g_{\theta_t}(x_i), \qquad (4)$$

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

Sharpening
Revisit Gumbel Softmax

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(\pi_j) + g_j)/\tau)}$$

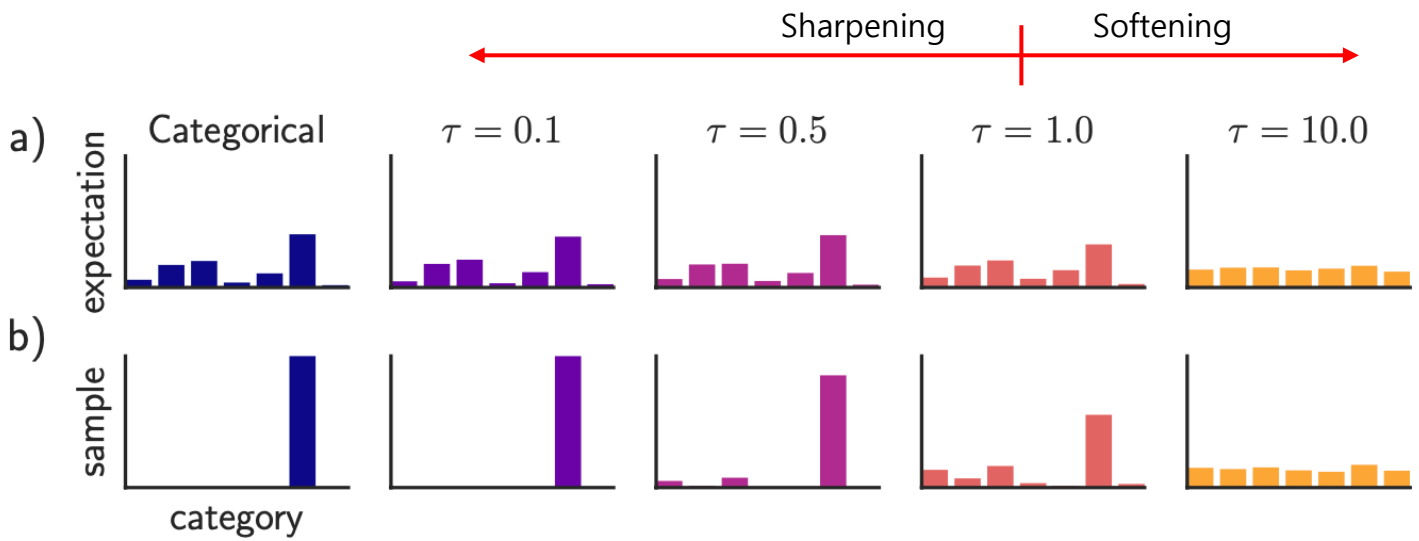
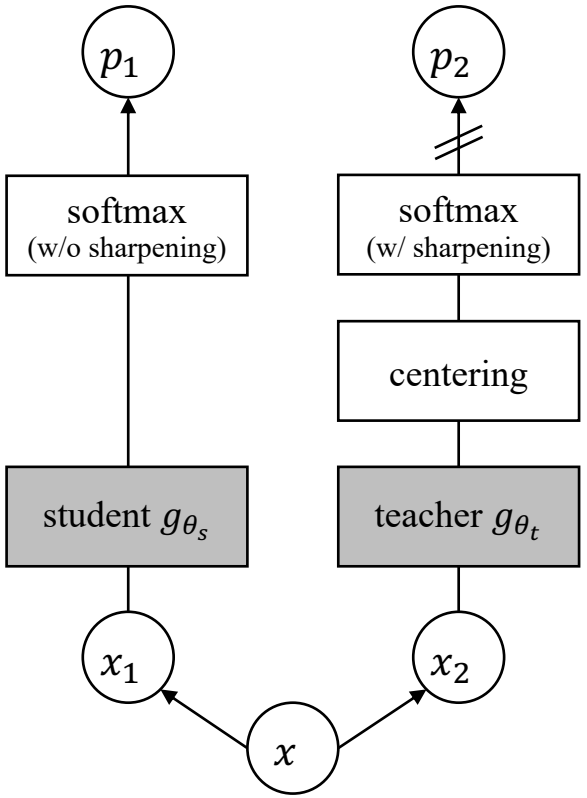
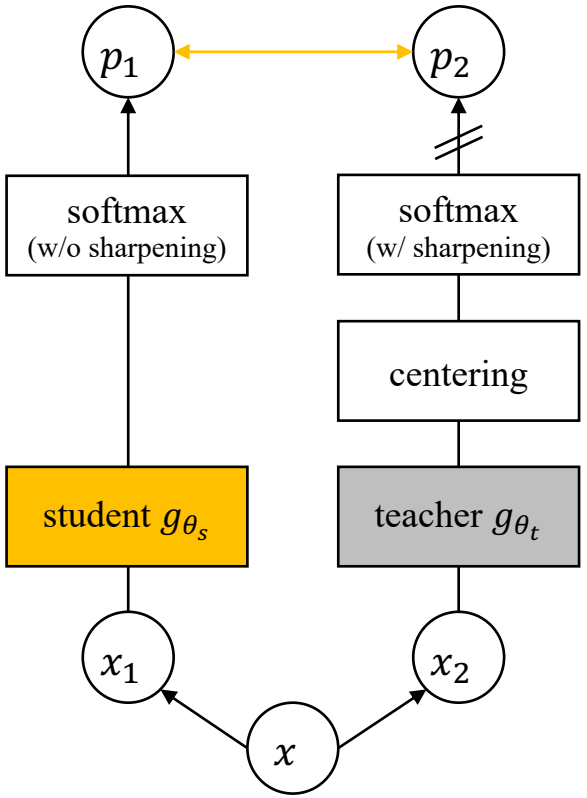


Figure 1: The Gumbel-Softmax distribution interpolates between discrete one-hot-encoded categorical distributions and continuous categorical densities. (a) For low temperatures ($\tau = 0.1, \tau = 0.5$), the expected value of a Gumbel-Softmax random variable approaches the expected value of a categorical random variable with the same logits. As the temperature increases ($\tau = 1.0, \tau = 10.0$), the expected value converges to a uniform distribution over the categories. (b) Samples from Gumbel-Softmax distributions are identical to samples from a categorical distribution as $\tau \to 0$. At higher temperatures, Gumbel-Softmax samples are no longer one-hot, and become uniform as $\tau \to \infty$.

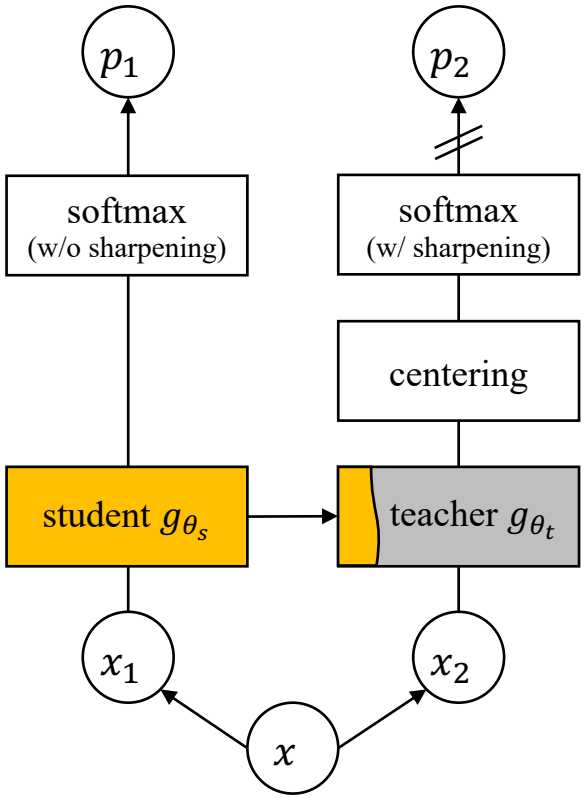
Training on DINO Framework



Initial state



$Step_i$
(update student)



$Step_{i+1}$
(update teacher)

$\theta_t \leftarrow \lambda\theta_t + (1-\lambda)\theta_s,\ \lambda = [0.996, 1]$

Experiments

Classification

Method	Arch.	Param.	im/s	Linear	k-NN
Supervised	RN50	23	1237	79.3	79.3
SCLR [11]	RN50	23	1237	69.1	60.7
MoCov2 [13]	RN50	23	1237	71.1	61.9
InfoMin [54]	RN50	23	1237	73.0	65.3
BarlowT [66]	RN50	23	1237	73.2	66.0
OBoW [21]	RN50	23	1237	73.8	61.9
BYOL [23]	RN50	23	1237	74.4	64.8
DCv2 [9]	RN50	23	1237	75.2	67.1
SwAV [9]	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* [23]	ViT-S	21	1007	71.4	66.6
MoCov2* [13]	ViT-S	21	1007	72.7	64.4
SwAV* [9]	ViT-S	21	1007	73.5	66.3
DINO	ViT-S	21	1007	**77.0**	**74.5**

- **Linear**: classification after training extra linear classifier
- **k-NN**: classification via 20 NNs
  - without training classification model

(1) Different backbone architecture within the **same DINO framework**
- ViT has an almost infinite receptive field than ResNet

(2) Different patch sizes within the **same ViT architecture**
- Small patch → much tokens → slow down

(3) Different ViT while using the **same patch size**
- Small model → fewer params → fast inference

Method	Arch.	Param.	im/s	Linear	k-NN
Comparison across architectures					
SCLR [11]	RN50w4	375	117	76.8	69.3
SwAV [9]	RN50w2	93	384	77.3	67.3
BYOL [23]	RN50w2	93	384	77.4	–
DINO	ViT-B/16	85	312	78.2	76.1
SwAV [9]	RN50w5	586	76	78.5	67.1
BYOL [23]	RN50w4	375	117	78.6	–
BYOL [23]	RN200w2	250	123	79.6	73.9
DINO	ViT-S/8	21	180	79.7	**78.3**
SCLRv2 [12]	RN152w3+SK	794	46		
DINO	ViT-B/8	85	63	**80.1**	77.4

(1) (2) (3) Best!

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.

Image Retrieval

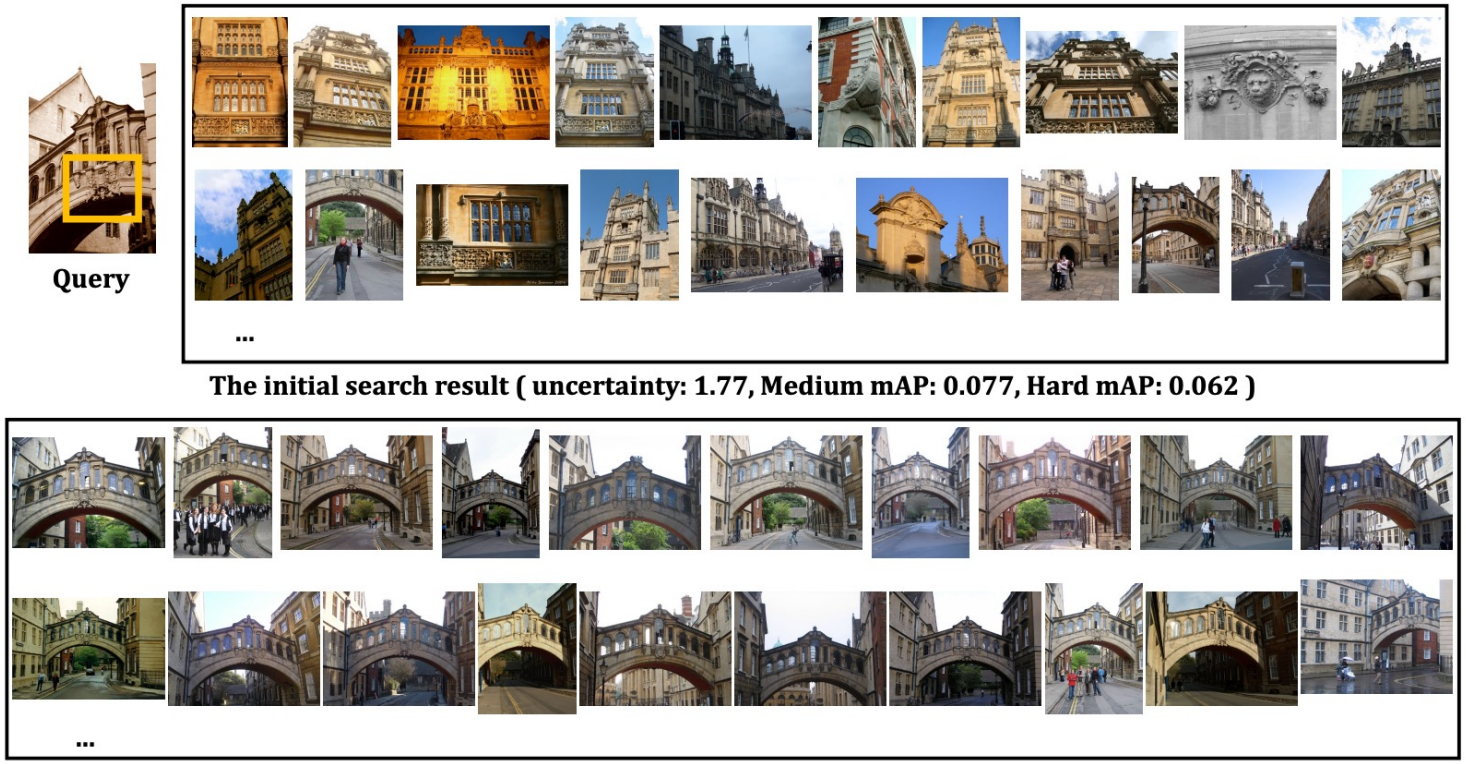


Figure 4: In this example, the top 11 images and the initial dominant community is unrelated to the query. Its uncertainty index is 1.77, which means the search engine is very uncertain about the accuracy of initial dominant community. So CS finds a new dominant community using spatial verification. After hypergraph propagation in this new dominant community, the retrieval performance is significantly improved.

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 [46].

			$\mathcal{R}$Ox		$\mathcal{R}$Par	
Pretrain	Arch.	Pretrain	M	H	M	H
Sup. [46]	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

Copy Detection

Image Retrieval under Adversarial Attack

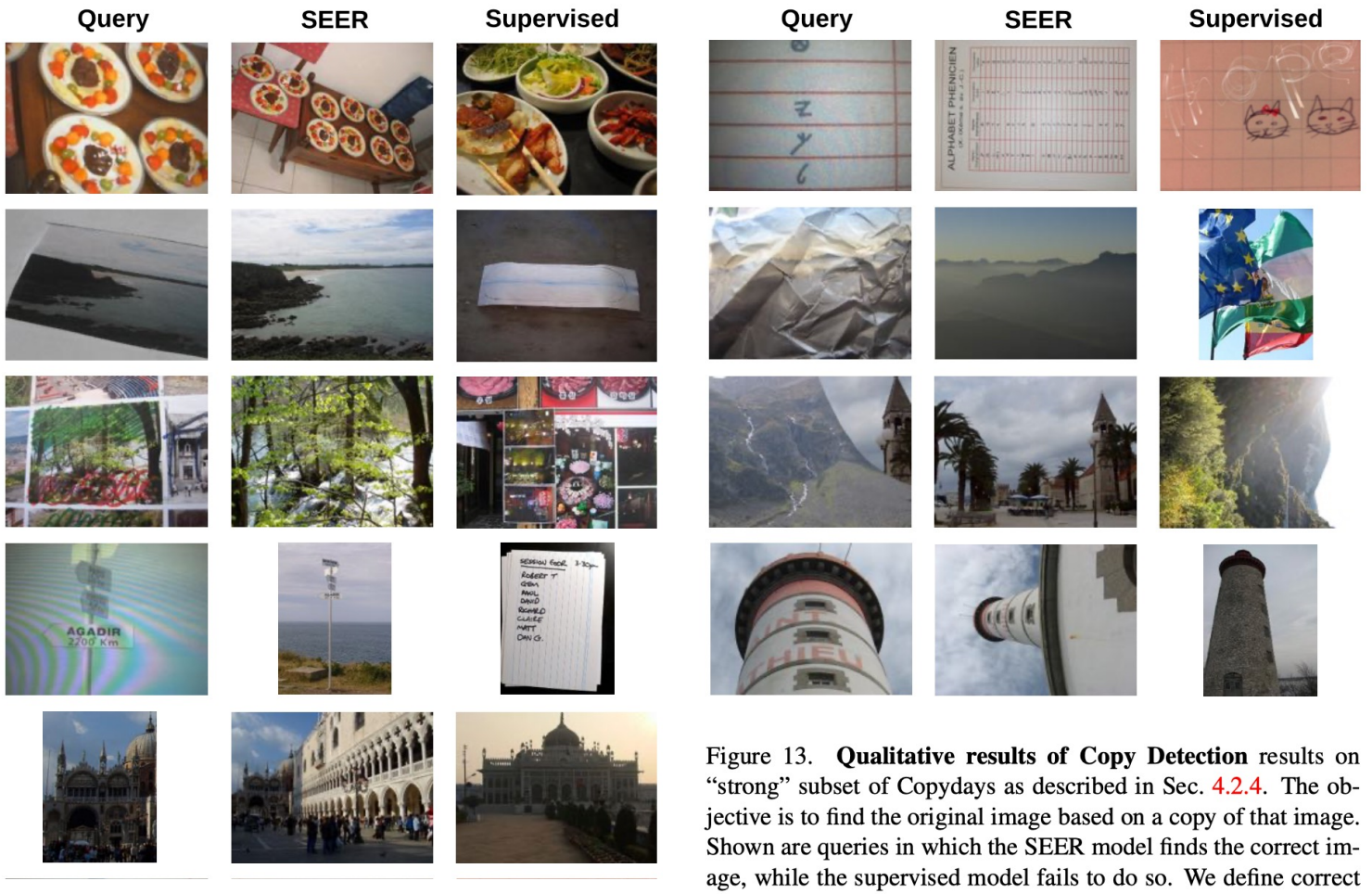


Figure 13. **Qualitative results of Copy Detection** results on "strong" subset of Copydays as described in Sec. 4.2.4. The objective is to find the original image based on a copy of that image. Shown are queries in which the SEER model finds the correct image, while the supervised model fails to do so. We define correct as the original image being ranked first among all 10,157 database and distractor images.

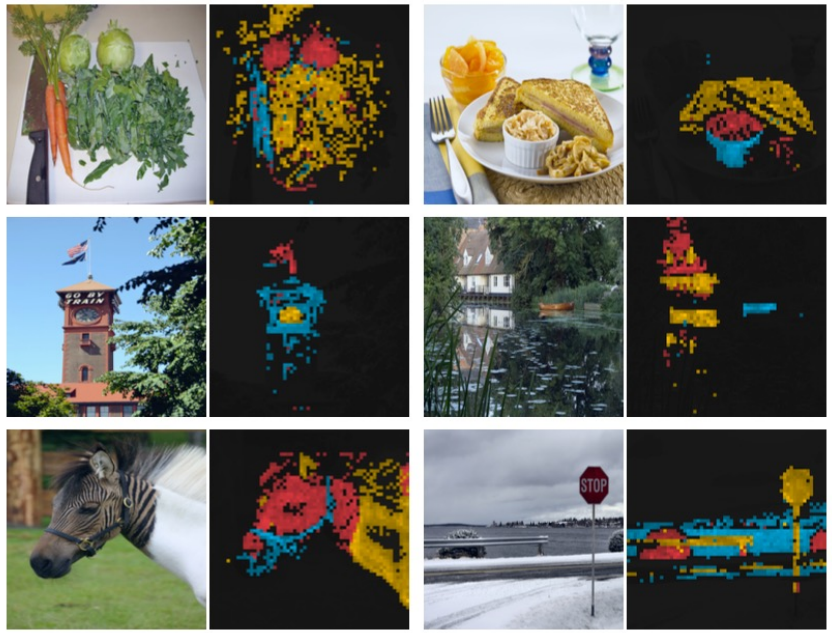
Table 4: **Copy detection.** We report the mAP performance in copy detection on Copydays "strong" subset [18]. For reference, we also report the performance of the multigrain model [4], trained specifically for particular object retrieval.

Method	Arch.	Dim.	Resolution	mAP
Multigrain [4]	ResNet-50	2048	224^2	75.1
Multigrain [4]	ResNet-50	2048	largest side 800	82.5
Supervised [56]	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**

Object Segmentation

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.

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 [39]	I/D/Y	RN50	81.8	79.2	84.3
Self-supervised					
CT [58]	VLOG	RN50	48.7	46.4	50.0
MAST [33]	YT-VOS	RN18	65.5	63.3	67.6
STC [30]	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**



Supervised

DINO

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

Figure 4: **Segmentations from supervised versus DINO.** We visualize masks obtained by thresholding the self-attention maps to keep 60% of the mass. On top, we show the resulting masks for a ViT-S/8 trained with supervision and DINO. We show the best head for both models. The table at the bottom compares the Jaccard similarity between the ground truth and these masks on the validation images of PASCAL VOC12 dataset.

[13] Xinlei Chen, et al. "Improved baselines with momentum contrastive learning." arXiv preprint. 2020.
[9] Mathilde Caron, et al. "Unsupervised learning of visual features by contrasting cluster assignments." Advances in neural information processing systems (NeurIPS). 2020.
[23] Jean-Bastien Grill, et al. "Bootstrap your own latent-a new approach to self-supervised learning." Advances in neural information processing systems (NeurIPS). 2020.

Transfer Learning / Ablation Study

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. [56]	**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. [56]	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**

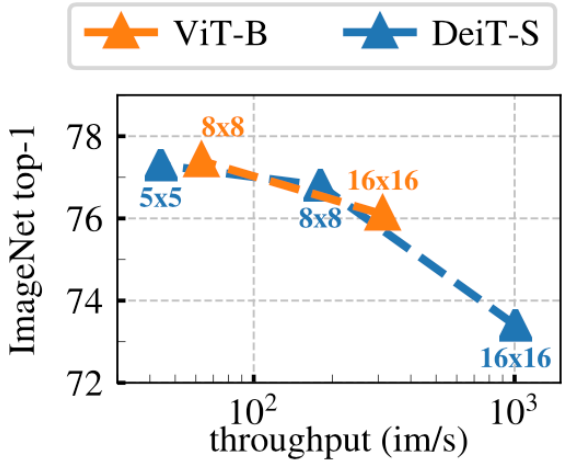


Figure 5: **Effect of Patch Size.** k-NN evaluation as a function of the throughputs for different input patch sizes with ViT-B and ViT-S. Models are trained for 300 epochs.

Table 7: **Important component for self-supervised ViT pretraining.** Models are trained for 300 epochs with ViT-S/16. We study the different components that matter for the k-NN and linear ("Lin.") evaluations. For the different variants, we highlight the differences from the default DINO setting. The best combination is the momentum encoder with the multicrop augmentation and the cross-entropy loss. We also report results with BYOL [23], MoCo-v2 [13] and SwAV [9].

	Method	Mom.	SK	MC	Loss	Pred.	k-NN	Lin.
		MoCov	SwAV			BYOL		
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

Conclusion

Any reason not to use DINO?

Just apply to your work!

Also recommend two techniques, contrastive learning and knowledge distillation.

Appendix A.
How does DINO work?

Attention Map

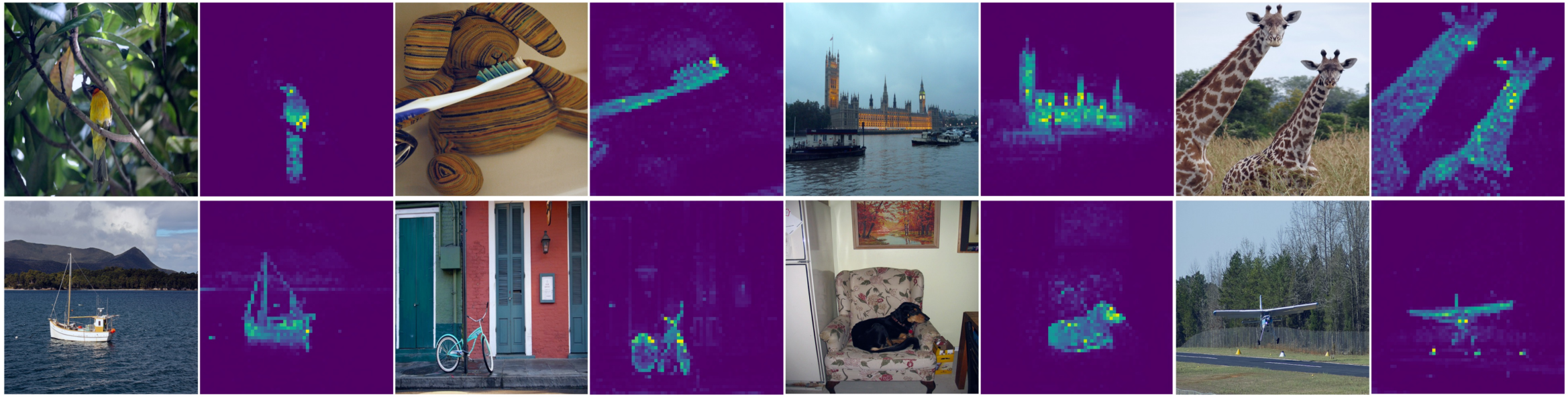


Figure 1: **Self-attention from a Vision Transformer with 8×8 patches trained with no supervision.** We look at the self-attention of the `[CLS]` token on the heads of the last layer. This token is not attached to any label nor supervision. These maps show that the model automatically learns class-specific features leading to unsupervised object segmentations.

- Actually, we just want to achieve an attention map.
- But, we should understand how it is generated.

ViT Revisit

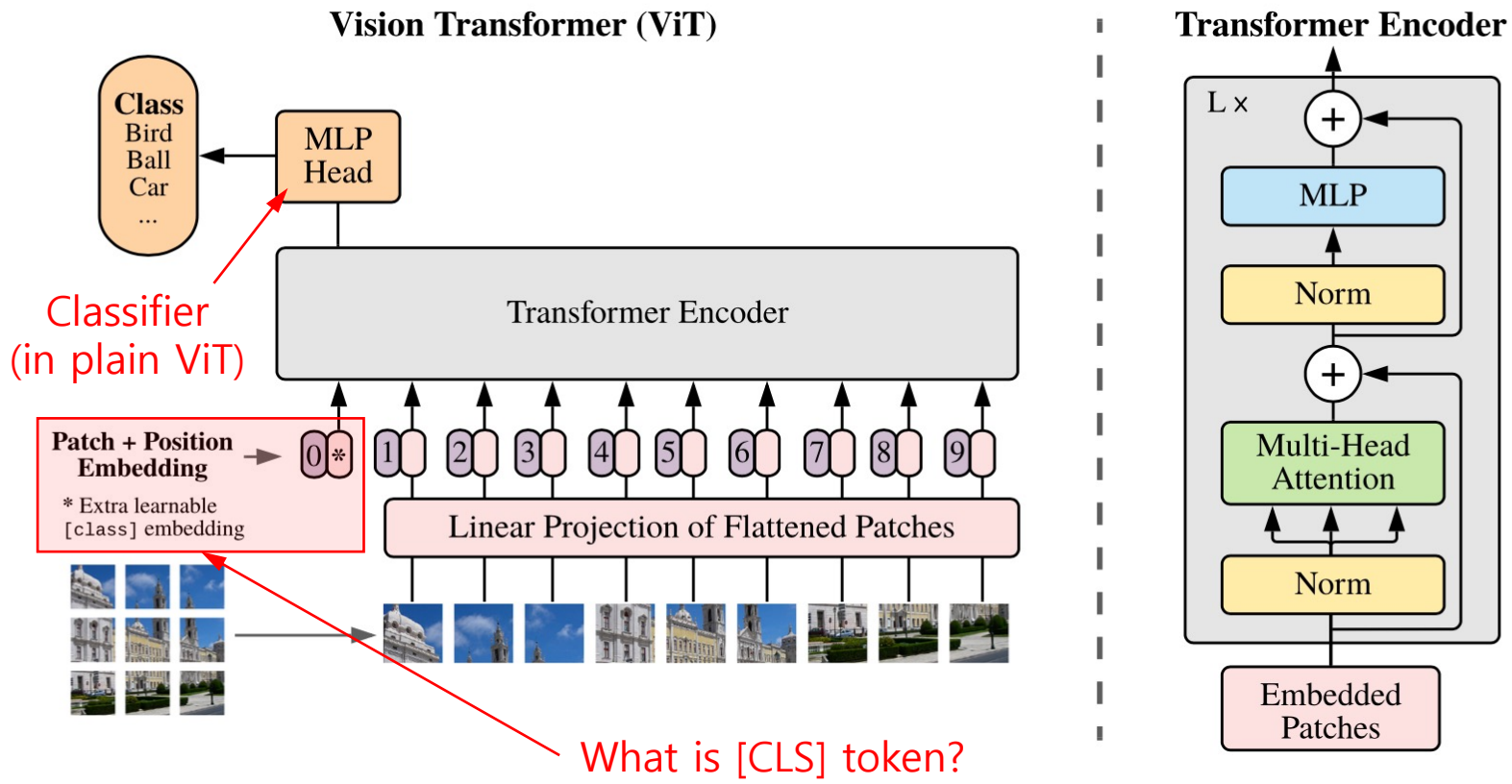


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

BERT Revisit

For CLS Token Understanding

[CLS] token

- Learnable parameter (kind of black box embedding vector)
- Aggregate feature information from other tokens
- Finally, predict next sentence (in BERT) or class score (in ViT)

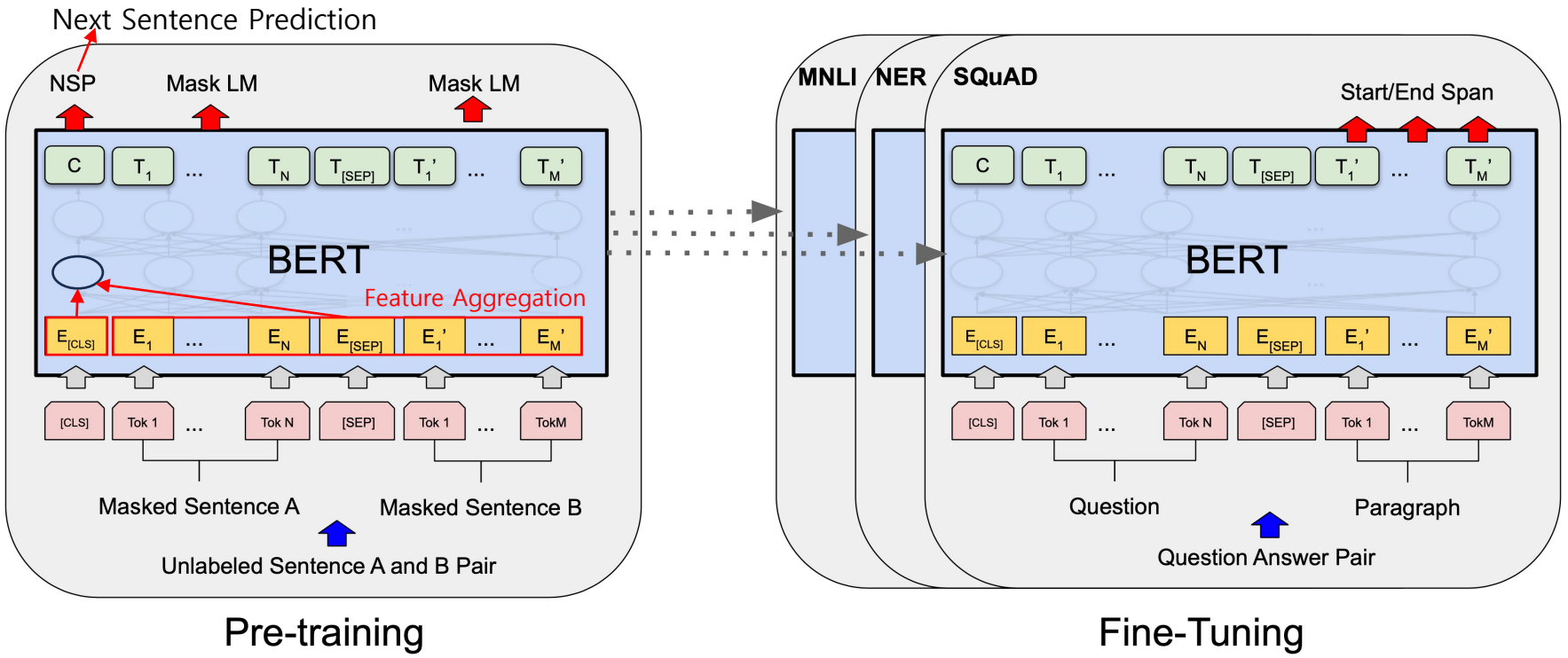
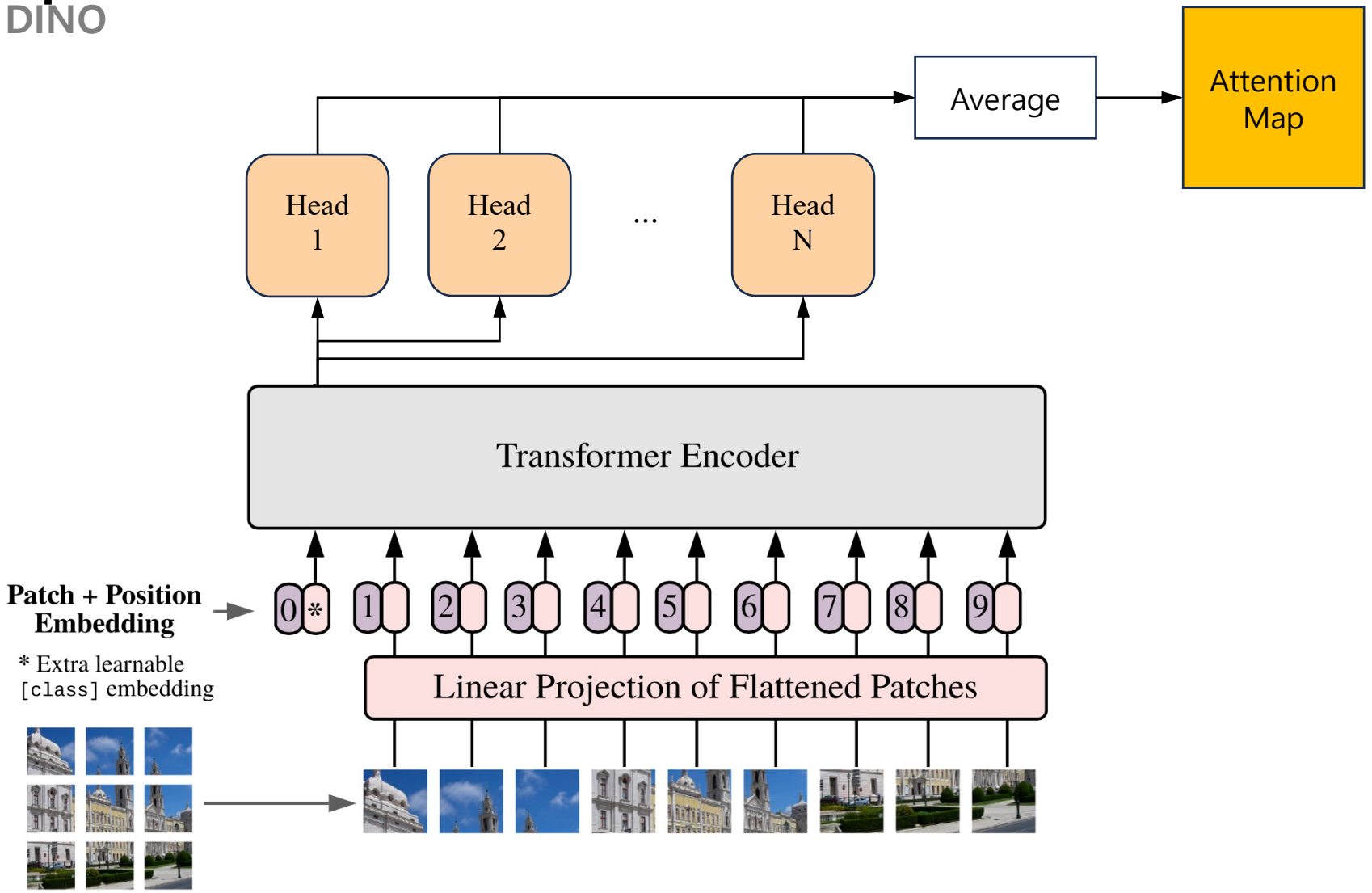


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. `[CLS]` is a special symbol added in front of every input example, and `[SEP]` is a special separator token (e.g. separating questions/answers).

* BERT: Bidirectional Encoder Representations from Transformers

Attention Map
Via Pre-trained DINO



Appendix B.
DINO Use case

The Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI-23)

Attention-Conditioned Augmentations for Self-Supervised Anomaly Detection and Localization

Behzad Bozorgtabar[1,2], Dwarikanath Mahapatra[3]

[1] École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
[2] Lausanne University Hospital (CHUV), Lausanne, Switzerland
[3] Inception Institute of AI (IIAI), Abu Dhabi, UAE
behzad.bozorgtabar@epfl.ch, dwarikanath.mahapatra@inceptioniai.org

Synthetic Anomalies

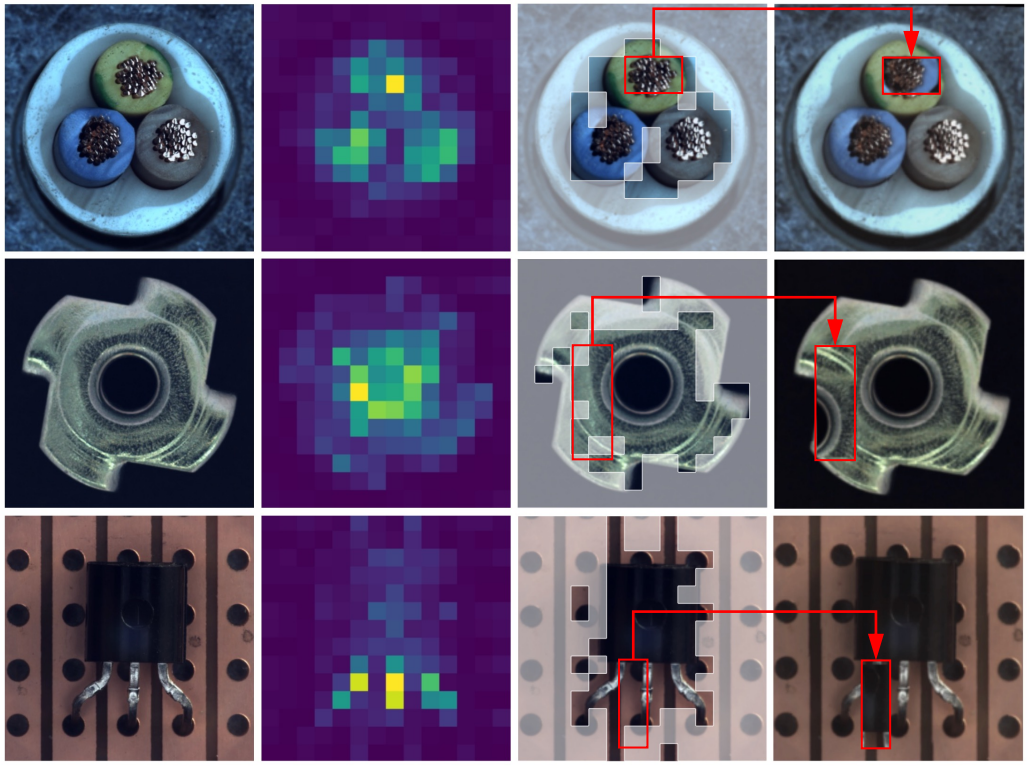


Figure 1: Given (a) the input image, we obtain (b) the average attention map of the transformer's heads, which is then leveraged for (c) attention-conditioned patch masking (APMask) by dropping the least attended patches and (d) attention-conditioned anomaly simulation by cutting and pasting local patch within the salient region.

Fine-Tuning Using Synthetic Anomalies

In the absence of prior knowledge of the types of anomalous features during training, we adopt synthetic anomalies (Bozorgtabar, Mahapatra, and Thiran 2022) and formulate a proxy task to detect and localize simulated anomalies for model fine-tuning (Fig. 3, top).

The central tenet of utilized synthetic anomaly is that most anomalous regions are associated with the salient objects within the image. Similarly, we benefit from the self-attention maps of a pre-trained ViT to delineate salient image regions used for creating synthetic anomalies (Fig. 4 (b)). More precisely, we use the softmax distribution of the $\bar{\mathbf{A}}^{[\mathtt{CLS}]}$ from the normal training image, which is resized to the input image dimensions followed by re-normalization to guide sampling of locations to cut and paste patches within the same image. The sizes for the patches' width r_w and height r_h are sampled from a uniform distribution $\sim \mathcal{U}(0.1W, 0.4W)$ for the image size of $W \times W$. Subsequently, we apply random rotation, resizing and jitter pixel values for the patches, yielding more diverse synthetic anomalies. Finally, unlike (Bozorgtabar, Mahapatra, and Thiran 2022), for the chest X-rays, we blend the pasted patches using Poisson blending (Tan et al. 2021; Pérez, Gangnet, and Blake 2003). This yields creating more close approximation of natural anomalies with fewer artificial discontinuities.

ViT Pre-training Process
(w/ self-attention masking)

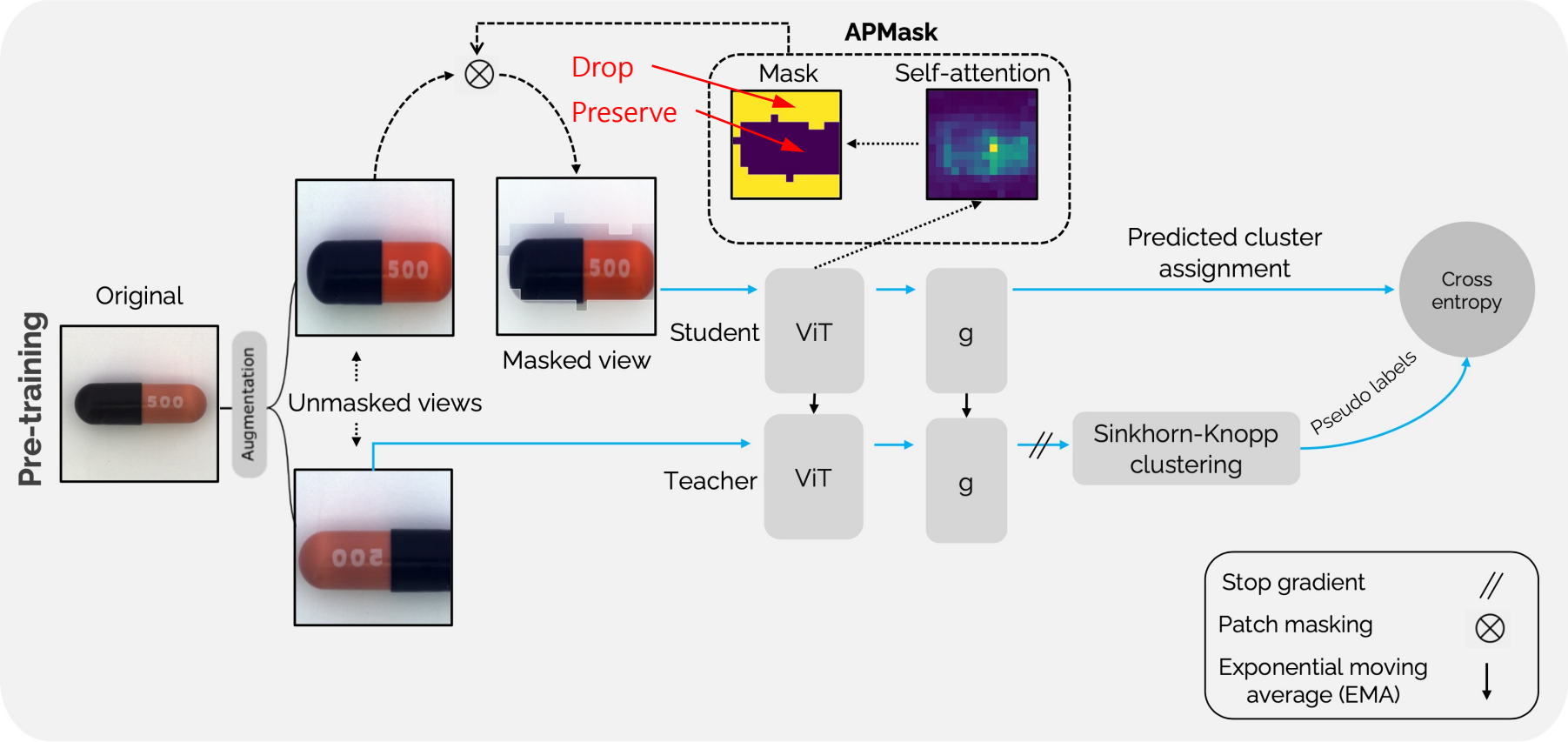


Figure 2: Overview of the proposed MIM pre-training. We set up pre-training from a DINO (Caron et al. 2021) initialization. Given two randomly augmented views, we patchify each augmented view, yielding two sequences of teacher and student views. We feed image views to the student and teacher networks. Subsequently, we apply attention-conditioned patch masking (APMask) to drop the least salient patches from the student view, yielding a masked student view. The networks are optimized using a multi-view consistency objective to enforce the cluster assignments made on the masked student view to match the pseudo labels predicted using the unmasked teacher view. For simplicity, we only illustrate the student's global view.

Appendix C.
DINO Example (w/ MNIST)

https://github.com/YeongHyeon/DINO_MNIST-PyTorch

GitHub

DINO_MNIST-PyTorch Public

Pin | Unwatch 2 | Fork 0 | Starred 2

main | 1 branch | 0 tags | Go to file | Add file | Code

YeongHyeon readme — 214883f 13 hours ago — 8 commits

figures	readme	13 hours ago
neuralnet	source code	16 hours ago
source	source code	16 hours ago
LICENSE	lincense	16 hours ago
README.md	readme	13 hours ago
run.py	source code	16 hours ago

README.md

[PyTorch] DINO: self-DIstillation with NO labels

PyTorch implementation of "Emerging Properties in Self-Supervised Vision Transformers"

Concept

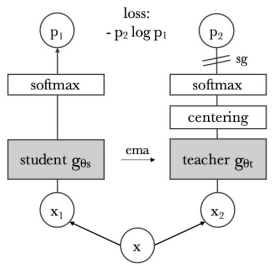


Figure 2: **Self-distillation with no labels.** We illustrate DINO in the case of one single pair of views (x_1, x_2) for simplicity. The model passes two different random transformations of an input image to the student and teacher networks. Both networks have the same architecture but different parameters. The output of the teacher network is centered with a mean computed over the batch. Each networks outputs a K dimensional feature that is normalized with a temperature softmax over the feature dimension. Their similarity is then measured with a cross-entropy loss. We apply a stop-gradient (sg) operator on the teacher to propagate gradients only through the student. The teacher parameters are updated with an exponential moving average (ema) of the student parameters.

Concept ot the DINO [1].

About

Pytorch implementation of "Emerging Properties in Self-Supervised Vision Transformers" (a.k.a. DINO)

torch | pytorch | mnist | mnist-dataset | knowledge-distillation | toy-example | self-supervised-learning | self-distillation

Readme
MIT license
Activity
2 stars
2 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

Python 100.0%

Suggested Workflows

Based on your tech stack

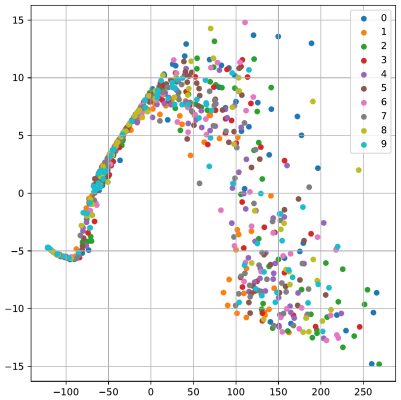
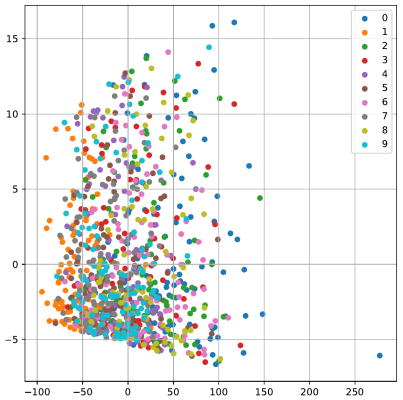
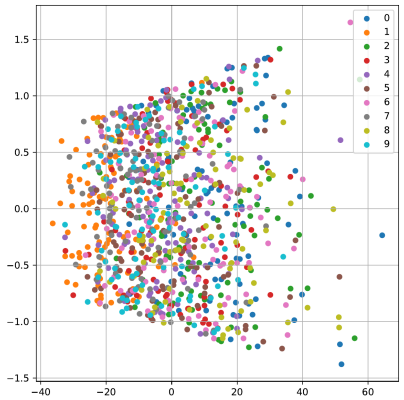
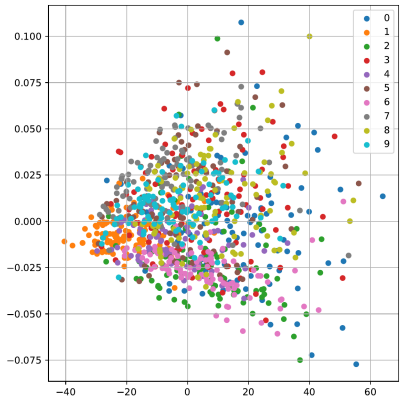
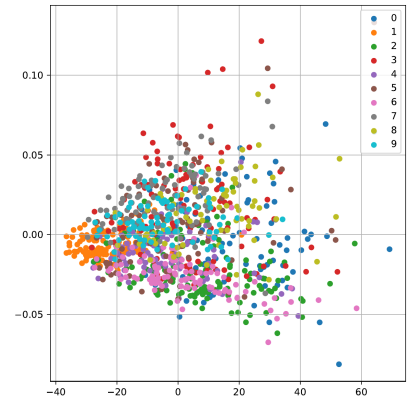
Actions Importer — Set up
Automatically convert CI/CD files to YAML for GitHub Actions.

Pylint — Configure
Lint a Python application with pylint.

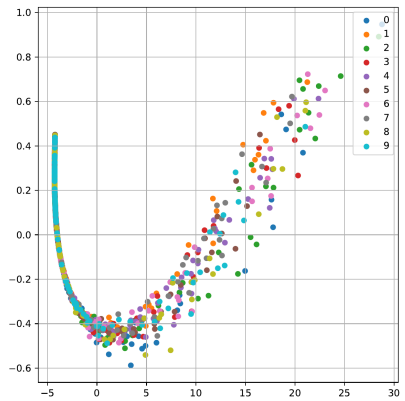
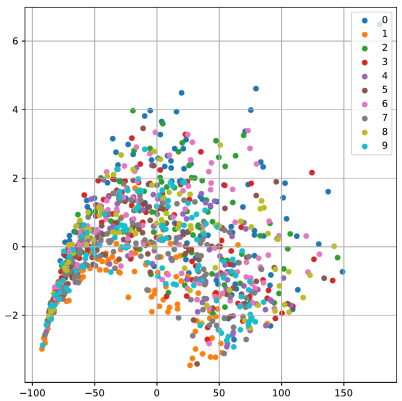
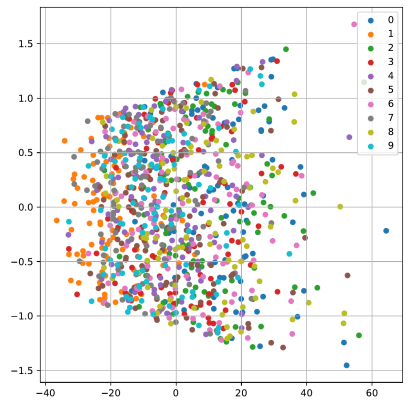
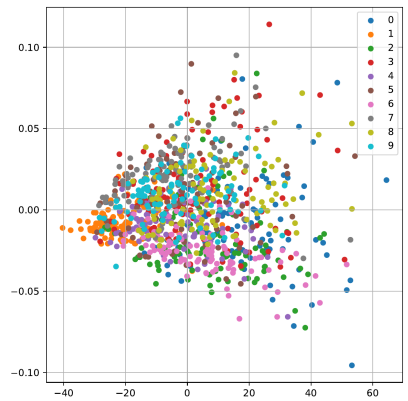
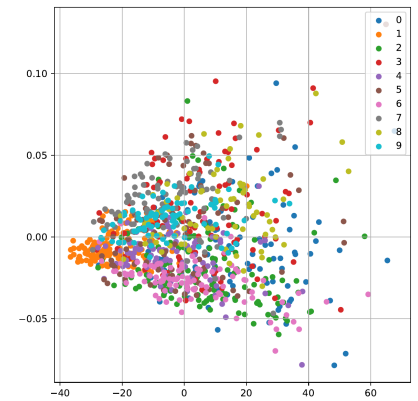
https://github.com/YeongHyeon/DINO_MNIST-PyTorch

Results (feature matching)

Student

Teacher

Appendix D.

Easy way for negative sampling (from SimCLR)

Negative sampling

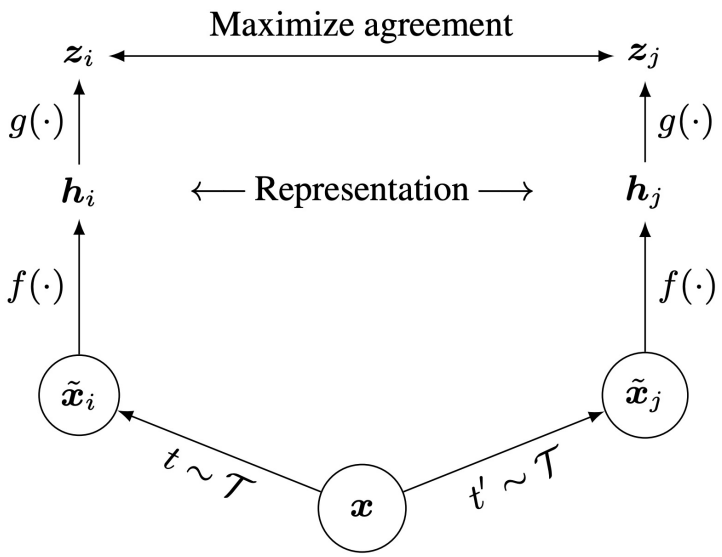


Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder network $f(\cdot)$ and a projection head $g(\cdot)$ are trained to maximize agreement using a contrastive loss. After training is completed, we throw away the projection head $g(\cdot)$ and use encoder $f(\cdot)$ and representation $\boldsymbol{h}$ for downstream tasks.