POA: Pre-training Once for Models of All Sizes

Yingying Zhang, Xin Guo, Jiangwei Lao, Lei Yu, Lixiang Ru, Jian Wang, Guo Ye, Huimei He, Jingdong Chen, and Ming Yang

Ant Group.
qichu.zyy@antgroup.com

A Design of Elastic Student

A.1 Elastic Swin Transformer

The Swin Transformer's basic block [20] closely resembles that of the ViT, and we apply the same parameter extraction method to its Shifted Window based Self-Attention (SW-MSA) and MLP modules. In the Swin Transformer architecture, the final linear layer of the MLP in the last block of each stage quadruples the dimension of the tokens. Subsequently, a Patch Merging (PM) layer is employed to halve the token dimension. We adjust the parameters of these linear layers in the MLP and PM to match their respective expansion and reduction ratios as follows:

$$w_i^{mlp} = w^{mlp}[:D_i^s \cdot 4, :D_i^s] \cdot \alpha_i, \qquad w_i^{pm} = w^{pm}[:D_i^s \cdot 2, :D_i^s \cdot 4] \cdot \alpha_i, \tag{1}$$

where D_i^s represents the i-th elastic width of stage s, w^{mlp} are the weight of the last linear layer of the MLP, and w^{pm} are the weight of the PM module. For the Swin Transformer, we apply elastic depth exclusively to the third stage, where the number of blocks is larger. The approach for selecting the activated block IDs follows the same method as used for the ViT, ensuring consistency across different transformer architectures in the process of creating sub-networks with varying depths.

A.2 Elastic ResNet

ResNet is composed of many basic units known as bottleneck building blocks [14]. In ResNet, we primarily focus on constructing sub-networks with varying numbers of blocks. We make the feature dimension(channel) of the middle layer in each building block elastic, while keeping the output dimension from each block unchanged. The weights of the three convolutional layers within a block are extracted as follows:

$$w_i^1 = w^1[:D_i^s, :, :, :],\ w_i^2 = w^2[:D_i^s, :D_i^s, :, :] \cdot \alpha_i,\ w_i^3 = w^3[:, :D_i^s, :, :] \cdot \alpha_i, \tag{2}$$

where D_i^s denotes the number of mid-layer channel for stage s. We implement elastic depth in both the second and third stages of ResNet. The method for selecting block IDs in each stage is consistent with that used for the ViT and Swin Transformer.

B Pseudocode

Algorithm 1: POA PyTorch-like Pseudocode without multi-crop augmentation and MPH.

Input : g_{IS}, g_T // intact student and teacher network
M, N // number of elastic width and depths
D_{max}, L_{max} // max width, max depth
D_h // head dimension
τ_s, τ_t, μ // temperatures and momentum for EMA
λ, γ // loss weight

Initialize: g_T.params = g_{IS}.params, cand_ids = [$[i, j]$ for i in range(M + 1) for j in range(N + 1)], idx=0

for x in dataloader **do**
  x_a, x_b=augment(x), augment(x) // random views
  g_{ES}, idx = `ExtractElastic`(cand_ids, idx, D_{max}, D_h, L_{max}, g_{IS})
  $f_a = g_T(x_a)$, $f_{b1} = g_{IS}(x_b)$, $f_{b2} = g_{ES}(x_b)$
  $\mathcal{L}_{IS}$ = `H`($f_{b1}, f_a, \tau_s, \tau_t$)
  $\mathcal{L}_{ES} = \mathcal{L}_{ES1} + \mathcal{L}_{ES2}$ = `H`($f_{b2}, f_a, \tau_s, \tau_t$) + `H`($f_{b2}, f_{b1}, \tau_s, \tau_s$, False)
  $\mathcal{L} = \lambda\mathcal{L}_{IS} + (1-\lambda)\mathcal{L}_{ES} + \gamma\mathcal{L}_{koleo}$ // total loss with Koleo regularization
  $\mathcal{L}$.backward(), update(g_{IS}) // Note that g_{ES} and g_{IS} share parameters, and the gradient from g_{ES} is accumulated onto the gradient of g_{IS}.
  g_T.params = $\mu \cdot g_T$.params + $(1-\mu)\cdot g_{IS}$.params // update teacher with momentum EMA
end

def `ExtractElastic`(cand_ids, idx, D_{max}, D_h, L_{max}, g_{IS}):
  if idx == len(cand_ids) − 1 **then**
    random.shuffle(cand_ids)
    idx = 0
  i, j = cand_ids[idx]
  $D_i = D_{max} - i \cdot D_h$
  $L_j = L_{max} - j$
  g_{ES} = Net(g_{IS}, D_i, L_j) // sub-network extracted from g_{IS} with width D_i and depth L_j
  idx+ = 1
  return g_{ES}, idx

def `H`(s, t, τ_s, τ_t, centering = True):
  $t = t$.detach() // stop gradient
  s = softmax(s/τ_s, dim = 1)
  if centering **then**
    t = SK(t) // SK centering
  t = softmax(t/τ_t, dim = 1)
  return $-(t \cdot \log(s))$.sum(dim = −1)

C Implementation Details

C.1 Multiple Projection Heads

Following [6, 21, 35], we employ 3-layer MLP with L2-normalized bottlenecks to serve as projection heads. To ensure effective training of each network within these elastic frameworks, we introduce multiple projection heads, each with a varying number of prototypes, positioned subsequent to the backbone network. Our MPH design is distinct from the multiple heads utilized in the ENT [24]. In the ENT, each head contains an identical number of prototypes and employs an averaging of cross-entropy loss which is weighted by the predictive entropies of each head, to ensemble the learning of each head. In contrast, our MPH design features heads with varying numbers of prototypes, which acts as multiple semantic spaces for representation. This design is intended to improve the distillation process between the intact network and the elastic network. In our experiments, we designate the output dimensions for these projection heads as: $K_1 = 8192, K_2 = 16384, K_3 = 32768, K_4 = 65536$.

C.2 KoLeo Regularizer

To mitigate the issue of feature collapse, we incorporate the KoLeo regularizer into our training process for the global views, as described in [21]. The regularizer's loss function is expressed as:

$$\begin{aligned} &\text{KoLeo}(z) = -\frac{1}{B}\sum_{i=1}^{B}\log(d_{B,i}), \quad d_{B,i} = \min_{j\neq i}||z_i - z_j||, \\ &\mathcal{L}_{koleo} = \text{KoLeo}(\frac{Z_{b1}}{||Z_{b1}||}) + \text{KoLeo}(\frac{Z_{b2}}{||Z_{b2}||}), \end{aligned} \tag{3}$$

where B is the batch size, and $d_{B,i}$ represents the Euclidean distance between the i-th feature z_i and its nearest feature z_j within the batch. The KoLeo loss is scaled with a modest loss weight γ set to 0.1.

C.3 Masked Patch Tokens Prediction

iBOT [35] has demonstrated the effectiveness of predicting the masked patch tokens of student networks according to the tokens of teacher network. We also incorporate this mechanism into the training of our POA SSL. In our experiments, we observed that early implementation of masked patch token prediction can result in unstable training. To address this issue, we delay the activation of the masked patch token prediction until the model has completed 30 epochs of training.

C.4 Probabilistic Sampling for Elastic Student

In POA, there are an array of candidate elastic networks that vary in width and depth, each offering a different level of diversity when compared to the intact student model. For example, considering the intact student structure as ViT-L characterized by a width of 1024 and depth of 24, an elastic network such as ViT-S, with a width of 384 and depth of 12, exhibits a higher degree of diversity relative to the intact student than an elastic network with dimensions closer to the intact student, such as a width of 960 and depth of 23. Intuitively, the elastic networks with greater diversity should be sampled more frequently to ensure sufficient training. To facilitate this, we implement a probabilistic sampling method influenced by the width and depth of the elastic networks in our experiments. For $N+1$ available widths and $M+1$ available depths, we calculate the sampling probability for the i-th width and j-th depth elastic network as follows:

$$p_{i,j} = \frac{((S_w - 1) \cdot \frac{N-i}{N} + 1) \cdot ((S_h - 1) \cdot \frac{M-j}{M} + 1)}{\sum_{k=0,l=0}^{N,M}((S_w - 1) \cdot \frac{N-k}{N} + 1) \cdot ((S_h - 1) \cdot \frac{M-l}{M} + 1)}. \tag{4}$$

It is important to note that when $i = 0$ and $j = 0$, the elastic network is at its smallest width and depth, and the sampling probability $p_{i,j}$ achieves the largest value. In our experiment, we set: $S_w = S_d = 3$.

C.5 Data Augmentation Setting

We adopt the same data augmentation techniques as DINOv2 [21], which include color jittering, Gaussian blur, solarization, flipping, and multi-crop strategies as described in [5]. The specific parameters for these augmentations are detailed in Table 1.

Table 1: Hyper-parameters of different data augmentations. The parameters prob_g1, prob_g2, and prob_l refer to the activation probabilities for the first global crop, the second global crop, and the local crops, respectively. The parameter min_gcs represents the minimum global crop scale, while max_gcs indicates the maximum global crop scale. Similarly, min_lcs and max_lcs specifies the minimum and maximum local crop scale, respectively.

Color jittering	Gaussian blur	Solarization	Multi-crop	Flipping
brightness: 0.4 contrast: 0.4 saturation: 0.2 hue: 0.1 prob: 0.8	radius_min: 0.1 radius_max: 2.0 prob_g1: 1.0 prob_g2: 0.1 prob_l: 0.5	thresh: 128 prob_g1: 0.0 prob_g2: 0.2 prob_l: 0.0	global crops: 2 local crops: 8 global size: 224 local size: 96 min_gcs: 0.32 max_gcs: 1.0 min_lcs: 0.05 max_lcs: 0.32	direction: horizontal prob: 0.5

C.6 Hyper-parameters of POA

We provide following hyper-parameters setting in our method:

- **projection heads:** bottleneck dim: 256 hidden layer dim: 2048
- **drop path rate:** ViT: 0.2 Swin: 0.2 ResNet: 0.0
- **loss weights:** $\lambda = 0.8$ $\gamma = 0.1$

C.7 Optimizing Setting

In our training, we utilize the AdamW optimizer with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The total training batch sizes for Vit, Swin, and ResNet are 1600, 2048, and 1280 respectively. We apply a learning rate decay from top to bottom across the network blocks, scaling down by a factor of 0.9. For transformer-based backbones, the patch embedding module's learning rate is further reduced by a factor of 0.2. Within each projection head, we keep the parameters of the final layer fixed during the initial epoch of training. Additionally, to maintain stable training, the gradient is clipped at an L2 norm of 1.5 for all parameters. The momentum in EMA updating for teacher network is initialized as 0.992 and decay to 0.9999 with cosine schedule.

D Experiments

D.1 k-NN and Linear Probing Evaluation on ImageNet-1K Dataset

We provide a more detailed comparison of k-NN and linear probing evaluations against existing methods in Table 2.

Table 2: Comprehensive comparison of k-NN and linear probing (LP) accuracy (%) on the ImageNet-1K dataset. "Param." indicates the quantity of parameters within the backbone network, measured in megabytes. "Epoch" refers to the adjusted number of effective training epochs, corrected for the number of views processed by the models as described in [35]. "$*$" denotes our implementation based on official code. "†" denotes results reproduced using the official code.

Method	Publication	Arch.	Param.	Epoch	k-NN	LP
SwAV [5]	NeurIPS 20	RN-50	23	2400	65.7	75.3
		RN-200	250	2000	73.9	79.6
BYOL [12]	NeurIPS 20	RN-50	23	2000	64.8	74.4
MoCov3 [8]	ICCV 21	RN-50	23	1600	-	74.6
DINO [6]	ICCV 21	RN-50	23	3200	67.5	75.3
UniGrad [29]	CVPR 22	RN-50	23	2400	-	75.5
ReLICv2 [32]	arXiv 22	RN-50	23	4000	-	**77.1**
		RN-101	41	4000	-	78.7
		RN-152	56	4000	-	79.3

Univip [19]	CVPR 22	RN-50	23	1200	-	74.2
Caco [33]	arXiv 22	RN-50	23	3200	-	75.7
SMoG [22]	ECCV 22	RN-50	23	1200	-	76.4
VICReg [4]	ICLR 22	RN-50	23	2000	-	73.2
HCSC [13]	CVPR 22	RN-50	23	800	66.6	73.3
SDMP-MoCov3 [23]	CVPR 22	RN-50	23	600	-	73.5
SimSiam+GSG [16]	NeurIPS 23	RN-50	23	400	58.4	69.4
BYOL+GSG [16]	NeurIPS 23	RN-50	23	400	62.2	71.1
GroCo [26]	ICCV 23	RN-50	23	400	64.8	71.3
MOKD [27]	CVPR 23	RN-50	23	400	70.6	75.6
SCFS [28]	CVPR 23	RN-50	23	3200	68.5	75.7
BYOL+LDReg [15]	ICLR 24	RN-50	23	400	-	68.5
AUC-CL [25]	ICLR 24	RN-50	23	1400	-	73.5
SimCLR+WNW [18]	ICLR 24	RN-50	23	1600	-	66.3
SimSiam+WNW [18]	ICLR 24	RN-50	23	1600	-	71.3
POA(Ours)		RN-50	23	0	**73.4**	76.9
		RN-101	41	0	**75.7**	**79.1**
		RN-152	56	2400	**76.4**	**79.9**
MoBY [34]	arXiv 21	Swin-T/W7	28	600	-	75.0
iBOT [35]	ICLR 22	Swin-T/W7	28	1200	75.3	78.6
		Swin-T/W14	28	1200	76.2	79.3
SMoG [22]	ECCV 22	Swin-T/W7	28	1200	-	77.7
EsViT [17]	ICLR 22	Swin-T/W7	28	1200	75.7	78.1
		Swin-S/W7	49	1200	77.7	79.5
		Swin-B/W7	87	1200	78.9	80.4
		Swin-T/W14	28	1200	77.0	78.7
		Swin-S/W14	49	1200	79.1	80.8
		Swin-B/W14	87	1200	79.3	81.3
DINOv2* [21]	TMLR 24	Swin-T/W7	28	1200	75.4	78.0
		Swin-S/W7	49	1200	76.1	79.8
		Swin-B/W7	87	1200	76.9	80.9
POA(Ours)		Swin-T/W7	28	0	**77.5**	**78.9**
		Swin-S/W7	49	0	**79.3**	**81.3**
		Swin-B/W7	87	1200	**79.6**	**82.0**
SwAV [5]	NeurIPS 20	ViT-S/16	21	2400	66.3	73.5
MoCov3 [8]	ICCV 21	ViT-S/16	21	1200	-	73.4
		ViT-B/16	85	1200	-	76.7
		ViT-L/16	307	1200	-	77.6
DINO [6]	ICCV 21	ViT-S/16	21	3200	74.5	77.0
		ViT-B/16	85	1600	76.1	78.2
iBOT [35]	ICLR 22	ViT-S/16	21	3200	75.2	77.9
		ViT-B/16	85	1600	77.1	79.5
		ViT-L/16	307	1200	78.0	81.0
SDMP-MoCov3 [23]	CVPR 22	ViT-S/16	21	600	-	73.8
		ViT-B/16	85	600	-	77.2

SDMP-DINO [23]	CVPR 22	ViT-S/16	21	1200	-	76.4
		ViT-S/16	21	3200	75.6	**78.9**
Mugs [36]	arXiv 22	ViT-B/16	85	1600	78.0	80.6
		ViT-L/16	307	1000	80.3	82.1
		ViT-S/16	21	800	73.3	76.6
MSN [2]	ECCV 22	ViT-B/16	85	400	74.7	78.1
		ViT-B/8	85	300	75.7	80.3
MOKD [27]	CVPR 23	ViT-S/16	21	800	73.1	76.3
		ViT-B/16	85	400	76.0	78.4
I-JEPA [27]	CVPR 23	ViT-B/16	85	600	-	72.9
		ViT-L/16	307	600	-	77.5
SiameseIM [30]	CVPR 23	ViT-B/16	85	1600	-	78.0
ENT-DINO [24]	ICLR 23	ViT-S/16	21	3200	75.2	77.4
		ViT-B/16	85	1600	77.1	79.1
		ViT-S/16	21	800	75.2	77.4
ENT-MSN [24]	ICLR 23	ViT-B/16	85	400	77.2	78.9
		ViT-B/8	85	300	78.9	80.8
SimCLR+LDReg [15]	ICLR 24	ViT-B/16	85	800	-	73.0
		ViT-S/16	21	1200	72.2	73.1
DINOv2† [21]	TMLR 24	ViT-B/16	85	1200	77.4	78.5
		ViT-L/16	307	1200	82.0	83.3
AUC-CL [25]	ICLR 24	ViT-S/16	21	1400	70.7	73.7
		ViT-S/16	21	0	**76.8**	77.6
POA(Ours)		ViT-B/16	85	0	**80.9**	**81.7**
		ViT-L/16	307	1200	**82.3**	**83.6**

D.2 Fine-Tuning Evaluation

Due to the fine-tuning process adjusting the pretrained parameters of the backbone network, the differences between pretrained features are diminished. This may result in comparisons that may not fully reflect the distinct qualities of leaned representation in each method. Consequently, only a handful of studies report this metric. In our experiments, we conduct fine-tuning on the ImageNet-1K dataset and draw comparisons to self-supervised methods utilizing ViT backbone. We adhered to the fine-tuning methodology delineated in [3, 35], which incorporates layer-wise learning rate decay, weight decay, and the AdamW optimizer. The training durations for ViT and Swin variants are set at 200, 100, and 50 epochs for the large, base, and small models, respectively. Due to differences in convergence between convalutional network and transformer, all ResNet variants (ResNet-152, ResNet-101, and ResNet-50) undergo a uniform training period of 100 epochs. We apply a layer-wise decay rate of 0.55 for ViT-S, Swin-T, and ResNet-50; a decay rate of 0.4 for ViT-B, Swin-S, and ResNet-101; and a

decay rate of 0.6 for ViT-L, Swin-B, and ResNet-152. The initial learning rates for fine-tuning are configured as follows: 0.002 for ViT-S, Swin-T, and ResNet-50; 0.0007 for ViT-B, Swin-S, and ResNet-101; and 0.0018 for ViT-L, Swin-B, and ResNet-152.

As is shown in Table. 3, our POA achieves a SOTA accuracy of **85.3%** on the ViT-L/16 backbone, and it demonstrates comparable accuracy when utilizing ViT-S/16 and ViT-B/16 backbones. We also report the fine-tuning results for the Swin and ResNet backbones in Table 5.

Table 3: Fine-tuning results on ImageNet-1K dataset.

Method	Arch.	Epo.	Acc.
DINO	ViT-S/16	3200	82.0
iBOT	ViT-S/16	3200	**82.3**
POA	ViT-S/16	0	82.1
BEiT	ViT-B/16	800	83.4
DINO	ViT-B/16	1600	83.6
iBOT	ViT-B/16	1600	**84.0**
POA	ViT-B/16	0	83.9
iBOT	ViT-L/16	1200	84.8
BEiT	ViT-L/16	800	85.2
POA	ViT-L/16	1200	**85.3**

Table 4: Results of semi-supervised learning on ImageNet-1K. The 1% and 10% indicate the fractions of labeled data used. SD denotes self-distillation.

Method	Arch.	1%.	10%.
SimCLRv2	RN50	57.9	68.1
BYOL	RN50	53.2	68.8
SwAV	RN50	53.9	70.2
SimCLRv2 +SD	RN50	60.0	70.5
POA	RN50	**61.8**	**73.1**
DINO	ViT-S/16	60.3	74.3
iBOT	ViT-S/16	61.9	75.1
POA	ViT-S/16	**68.2**	**75.9**

Table 5: Fine-tuning results of Swin and ResNet backbone on ImageNet-1K dataset.

Method	Arac.	Epo.	Acc
POA	ResNet-50	0	77.8
POA	ResNet-101	0	80.0
POA	ResNet-152	2400	81.1

(a) ResNet backbone.

Method	Arac.	Epo.	Acc
POA	Swin-T	0	81.0
POA	Swin-S	0	82.9
POA	Swin-B	1200	83.7

(b) Swin backbone.

D.3 Semi-Supervised Learning Evaluation

For semi-supervised learning, we concentrate our comparison on methods that adopt the unsupervised pre-training followed by supervised fine-tuning paradigm with partial labeled data. As shown in Table 4, our method significantly outperforms iBOT when using only 1% of labeled data, with an improvement of **6.3%**. These results demonstrate our method's superior label efficiency. We attribute this performance enhancement primarily to the distillation loss $\mathcal{L}_{ES2}$, which facilitates knowledge transfer from the intact model to its elastic counterpart. This effect mirrors the improvement observed in SimCLRv2, where self-distillation from a larger model contributes to performance gains.

D.4 Unsupervised Learning Evaluation

For evaluating the pre-trained model on unsupervised learning, we employ standard metrics such as accuracy (ACC), adjusted rand index (ARI), normalized mutual information (NMI), and the Fowlkes-Mallows index (FMI), following [35]. We benchmark our POA with a ResNet-50 backbone against established methods like SimCLRv2 [7], Self-label [1], InfoMin [31], and SCAN [11]. Additionally, we compare POA with a ViT-S/16 backbone to DINO and iBOT. According to the results presented in Table. 6, our POA method attains accuracies of **61.8%** with ViT-S/16 and **55.7%** with ResNet-50, respectively. These results indicate that the POA approach in self-supervised learning enables models to learn stronger visual semantical representation.

Table 6: Unsupervised learning on ImageNet-1K dataset. "†" denotes k-means clustering on frozen features extracted by backbones.

Method	Arch.	ACC	ARI	NMI	FMI
Self-label†	ResNet-50	30.5	16.2	75.4	-
InfoMin†	ResNet-50	33.2	14.7	68.8	-
SCAN	ResNet-50	39.9	27.5	72.0	-
POA†	ResNet-50	**55.7**	**38.2**	**79.9**	**38.9**
DINO	ViT-S/16	41.4	29.8	76.8	32.8
iBOT	ViT-S/16	43.4	32.8	78.6	35.6
POA†	ViT-S/16	**61.8**	**47.7**	**82.5**	**47.9**

Table 7: Transfer learning experiments by fine-tuning models pre-trained on various datasets. The Top-1 accuracy for the ViT-S/16 is presented on the left, and for the ViT-B/16 on the right.

Method	Cif_{10}	Cif_{100}	iNa_{18}	iNa_{19}	Flwrs	Cars
BEiT	98.6	87.4	68.5	76.5	96.4	92.1
DINO	99.0	90.5	72.0	78.2	98.5	93.0
iBOT	99.1	90.7	73.7	78.5	**98.6**	94.0
POA	**99.1**	**90.7**	**74.2**	**79.1**	98.4	**94.2**

Method	Cif_{10}	Cif_{100}	iNa_{18}	iNa_{19}	Flwrs	Cars
BEiT	99.0	90.1	72.3	79.2	98.0	94.2
DINO	99.1	91.7	72.6	78.6	98.8	93.0
iBOT	99.2	92.2	74.6	79.6	**98.9**	94.3
POA	**99.4**	**92.6**	**76.2**	**81.7**	98.8	**94.6**

D.5 Transfer Learning

We evaluate transfer learning by pre-training models on ImageNet-1K and subsequently fine-tuning them on a variety of smaller datasets, adhering to the protocol established in [9]. The results are detailed in Table 7. Our method achieves SOTA transfer performance compared to other self-supervised learning (SSL) approaches, with the exception of the Flowers dataset. Notably, we observe a more pronounced performance improvement over iBOT on larger datasets such as iNaturalist18 and iNaturalist19. This suggests that the results have not yet reached their peak, thereby providing a more effective measure for evaluating the quality of pre-trained features.

D.6 k-NN Accuracies of Elastic Networks

Elastic Networks of ViT We present the k-NN evaluation accuracy of each elastic network derived from the pre-trained ViT trained by POA, as detailed in Table 8, here L_i and D_i are the depths and widths of each elastic network, repectively.

Table 8: k-NN accuracy of elastic networks derived from pretrained ViT-L/16.

L_i/D_i	384	448	512	576	640	704	768	832	896	960	1024
12	76.78	78.42	79.17	79.78	80.27	80.68	80.86	81.02	81.10	81.14	81.09
13	76.84	78.37	79.17	79.85	80.83	80.66	80.93	81.23	81.24	81.12	81.15
14	77.59	78.95	79.69	80.26	80.69	81.04	81.25	81.36	81.52	81.60	81.59
15	77.89	79.16	79.86	80.35	80.75	81.16	81.40	81.62	81.62	81.63	81.58
16	78.15	79.30	80.13	80.75	81.06	81.36	81.55	81.78	81.92	81.77	81.85
17	78.34	79.65	80.26	80.76	81.11	81.54	81.63	81.76	82.02	81.92	81.88
17	78.58	79.75	80.43	80.92	81.28	81.69	81.72	81.99	82.04	82.02	82.05
19	78.62	79.90	80.52	80.98	81.32	81.57	81.83	82.13	82.09	82.14	82.12
20	78.93	79.97	80.61	81.17	81.45	81.79	81.98	82.17	82.22	82.15	82.17
21	78.99	80.21	80.73	81.31	81.49	81.86	82.05	82.18	82.32	82.22	82.20
22	79.25	80.25	80.89	81.36	81.63	81.87	82.10	82.31	82.34	82.36	82.41
23	79.37	80.30	80.89	81.35	81.73	81.93	82.19	82.37	82.41	82.41	82.39
24	79.33	80.28	80.90	81.33	81.65	81.90	82.15	82.35	82.42	82.28	82.27

Elastic Networks of Swin For the Swin Transformer architecture, we designate Swin-T as the smallest elastic network and Swin-B as the largest. We explore elastic widths of 96, 112, and 128, with elastic depths varying from 12 to 24. The k-NN accuracies of total 39 elastic networks configuration are presented in Table. 9.

Table 9: k-NN accuracy of elastic networks derived from pretrained Swin-B.

D_i/L_i	12	13	14	15	16	17	18	19	20	21	22	23	24
96	77.48	77.80	78.07	78.18	78.17	78.67	78.81	78.83	79.05	79.12	79.20	79.34	79.31
112	77.84	78.16	78.33	78.47	78.47	78.94	79.03	79.09	79.22	79.22	79.32	79.43	79.43
128	77.90	78.23	78.47	78.52	78.52	79.00	79.07	79.19	79.45	79.45	79.53	79.52	79.63

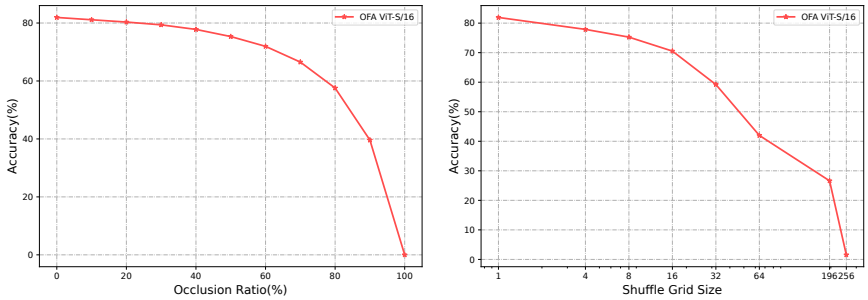
Elastic Network of ResNet In the case of the ResNet architecture, we designate ResNet-50 as the smallest and ResNet-152 with wider middle layer in each building block as the largest elastic network configurations. It yields a total number of 465 distinct ResNet sub-networks with the combination of 3 widths and 155 depths configurations. For the sake of clarity, we present a subset of the k-NN accuracies of these elastic networks in Table 10. Here, N_2 and N_3 refer to the count of bottleneck building blocks in the second and third stages, respectively, while W denotes the bottleneck dimension of middle layer in building block at the first stage.

Table 10: k-NN accuracy of elastic networks derived from pretrained ResNet-152.

$W/N_2 - N_3$	4-6	4-8	4-10	4-12	4-14	4-16	4-16	4-18
64	73.44	74.11	74.32	74.52	74.92	75.17	75.19	75.49
96	74.41	75.00	75.18	75.63	75.95	76.01	76.11	76.42
128	74.73	75.43	75.43	75.81	75.91	76.20	76.44	76.47
$W/N_2 - N_3$	4-20	4-24	8-26	8-28	8-30	8-32	8-34	8-36
64	75.49	75.77	76.05	76.20	76.20	76.33	76.31	76.38
96	76.34	76.63	76.91	76.95	77.07	77.13	77.30	77.14
128	76.53	76.90	77.13	77.34	77.31	77.54	77.59	77.72

D.7 Robustness Evaluation.

Robustness to Occlusion and Shuffling. We evaluate the pre-trained model's robustness to occlusion and alterations in spatial structure by applying masking and shuffling to the input image patches. Detailed results for various occlusion ratios are depicted in Figure 1a. Additionally, we present the effects of different shuffling grid sizes in Figure 1b.



(a) Robustness to occlusion with different ratio.

(b) Robustness to shuffling with varying grid sizes.

Fig. 1: Robustness to Occlusion and Shuffling.

D.8 Additional Ablation Studies

Influence of Input Dimension Scaling Factor α_i**.** To adapt the reduction of the input dimension, we apply a scaling factor α_i to weight parameters. We conduct comparative experiments to assess the impact of employing this scaling factor, with results presented in Table 11. The results indicate that the scaling factor α_i enhances the performance of an elastic network, particularly in cases where there is a significant reduction in width compared to the intact network, such as with ViT-S.

Influence of Loss Weight λ**.** Within our POA framework, the parameter λ regulates the balance between the loss contributions from the intact student and the elastic student. We assessed the performance impact of varying λ during

Table 11: Importance of Scaling Factor in POA SSL pre-training.

Scaling Factor	k-NN			LP		
	ViT-S	ViT-B	ViT-L	ViT-S	ViT-B	ViT-L
✓	76.8	80.9	82.3	77.6	81.7	83.6
	75.3	80.8	82.3	75.9	81.7	83.6

pre-training. The results presented in Table 12 suggest that a larger value of λ, representing a greater loss weight for intact branch, enhances the performance of larger models like ViT-L. However, this same increase in λ adversely affects the performance of the extracted smaller sub-networks such as ViT-S and ViT-B. Conversely, a smaller λ value improves the performance of these smaller sub-networks while potentially diminishing the effectiveness of larger models. To achieve a more balanced outcome, we have chosen $\lambda = 0.8$ for our OPA approach.

Table 12: Influence of Loss Weight λ in POA SSL pre-training.

λ	k-NN			LP		
	ViT-S	ViT-B	ViT-L	ViT-S	ViT-B	ViT-L
0.6	77.4	81.0	82.0	78.3	81.9	83.1
0.7	77.0	80.9	82.1	77.9	81.8	83.4
0.8	76.8	80.9	82.3	77.6	81.7	83.6
0.9	75.3	80.0	82.6	76.5	81.0	84.0

Influence of Probabilistic Sampling for Elastic Student. We provide the ablation study about probabilistic sampling for elastic student in Table. 13. The result confirms our intuitive assumption that elastic networks with greater diversity should be sampled more frequently.

Table 13: Influence of sampling for elastic student in POA SSL pre-training.

$S_w = S_d$	k-NN			LP		
	ViT-S	ViT-B	ViT-L	ViT-S	ViT-B	ViT-L
1	75.8	80.6	82.3	76.5	81.4	83.6
2	76.5	80.7	82.3	77.4	81.6	83.6
3	76.8	80.9	82.3	77.6	81.7	83.6

Influence of Number of Elastic Students. We investigate the impact of varying the number of candidate elastic networks in our POA. We manipulate the number of candidates by adjusting the sampling intervals of network widths and depths. Except for the number of candidate elastic networks, all hyper-parameters and training settings remain constant. The comparative results are presented in Table 14. From these results, we observe that the increase of the sampling interval, which effectively decreases the number of candidate networks, improves the k-NN accuracy for the derived ViT-S model. The primary reason is that with a constant number of iterations, a reduction in the total count of networks results in a higher proportion of smaller networks being sampled. This

leads to more training iteration for these networks. However, this adjustment appears to have a negligible effect on the performance of ViT-L models, due to the existing of the intact student branch which is trained at each iteration.

Table 14: Influence of number of candidate elastic students in POA SSL pre-training.

Number of Candidates (#Widths × #Depths)	Interval of Elastic Sampling (Width/Depth)	k-NN ViT-S	ViT-B	ViT-L
$143(11 \times 13)$	64/1	76.8	80.9	82.3
$42(6 \times 7)$	128/2	77.3	80.9	82.3
$20(4 \times 5)$	192/3	77.7	80.8	82.4
$16(4 \times 4)$	192/4	77.8	81.1	82.4

D.9 Alternative Designs to Elastic Pre-training

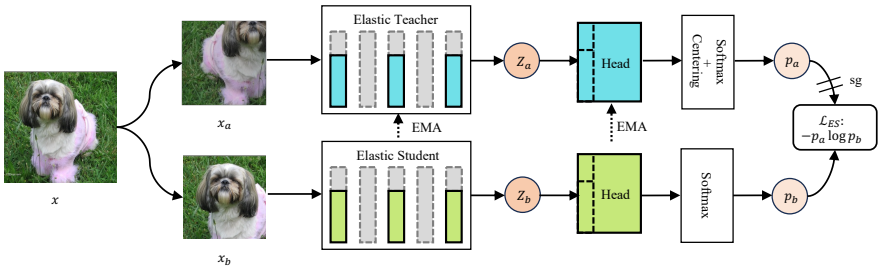
We provide a detailed illustration of the three variants mentioned in Sec.4.4 of our paper. In the first variant, illustrated in Figure 2a, the intact student is removed from the POA framework and the teacher is adapted to be elastic, aligning with the architecture of the elastic student. The second variant, which also discards the intact student from POA, is depicted in Figure 2b. The third variant introduces an additional elastic teacher branch that shares the architecture of the elastic student, facilitating cross-view distillation, shown in Figure 2c.

D.10 Convergence Comparison with Single Pre-training Method

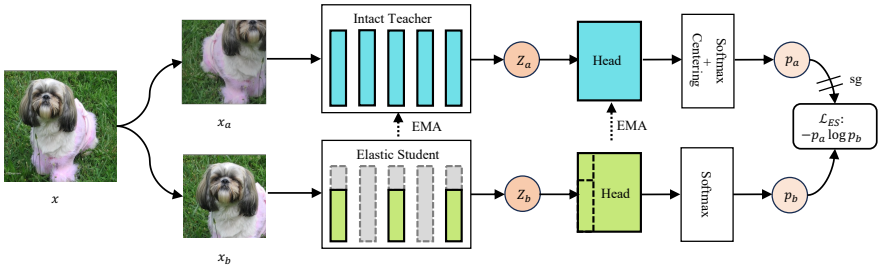
By integrating same-view and cross-view distillation, POA achieves faster convergence speed, particularly for smaller-sized sub-networks. For instance, in the case of ViT-S, many existing self-supervised learning methods [6,24,35,36] necessitate a substantial number of effective training epochs (3200 epochs) reach good performance. In contrast, the ViT-S model extracted from our POA framework, pre-trained for just 1200 epochs, outperforms those methods trained separately for 3200 epochs. Figure 3 illustrates the k-NN evaluation accuracy progression throughout the 1200 effective epochs of training. It is evident that POA consistently delivers superior performance at each stage of the training process.

D.11 Computational Resources Required

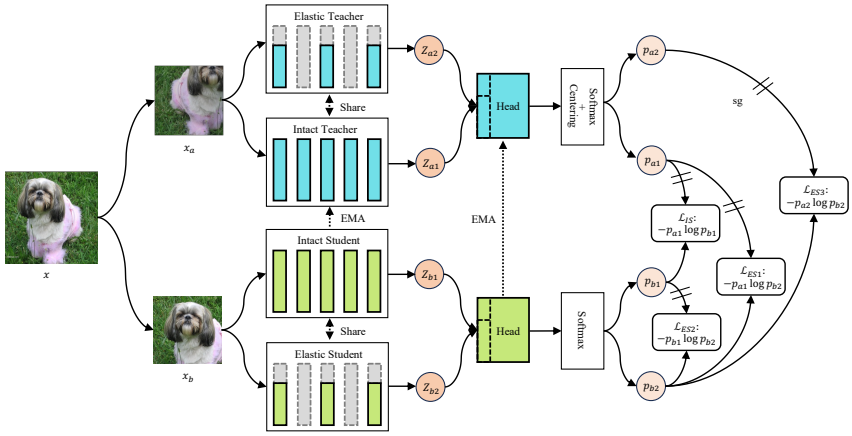
Table. 15 provides a detailed account of the computational resources required for training with a ViT backbone on 4 machines, each equipped with 8 A100 GPUs. We compare the time and GPU memory demands of our method to those of DINOv2, which incorporates self-supervised knowledge distillation [10] and yields superior performance compared to training the models independently. Notably, our approach can generate numerous elastic networks beyond the three primary structures: small, base, and large.



(a) POA-V1: This variant of POA features both an elastic teacher and an elastic student, streamlining the architecture by ensuring both components are adaptable in size.



(b) POA-V2: In this variant, POA includes an intact, intact teacher paired with an elastic student, allowing the smaller student model to learn from the larger, fully-trained teacher.



(c) POA-V3: This variant of POA introduces an extra elastic teacher alongside the standard configuration, providing another potential for cross-view distillation within the framework.

Fig. 2: Three alternative variants of POA.

E Visualization

E.1 Visualization of Self-attention Map

We visualize the self-attention maps generated by the ViT-S/16 model, which is pre-trained using DINOv2 and our POA. For the visualizations, we select the class token as the query and represent attention maps from different heads of the final layer using distinct colors, as depicted in Figure 4. The results indicate

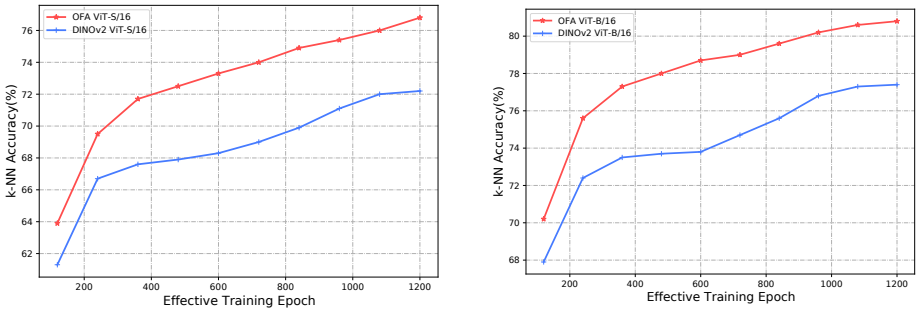
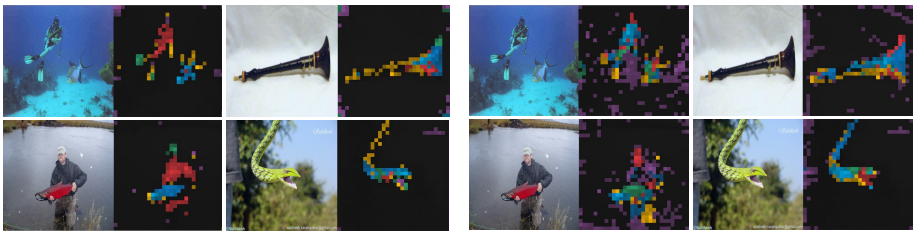


Fig. 3: Comparison of k-NN accuracy during training.

Table 15: Comparison of computational resources required.

Method	Arch.	Epoch	Mem.	Batch Size	k-NN	GPU hours
DINOv2	ViT-L	1200	41G	2048	82.0	1152
DINOv2	ViT-L→ViT-B	1200	46G	4096	79.7	1024
DINOv2	ViT-L→ViT-S	1200	35G	4096	75.5	928
Total						3104
POA	ViT-L	1200	77G	1600	82.3	2752
POA	ViT-B	0	-	-	80.9	0
POA	ViT-S	0	-	-	76.8	0

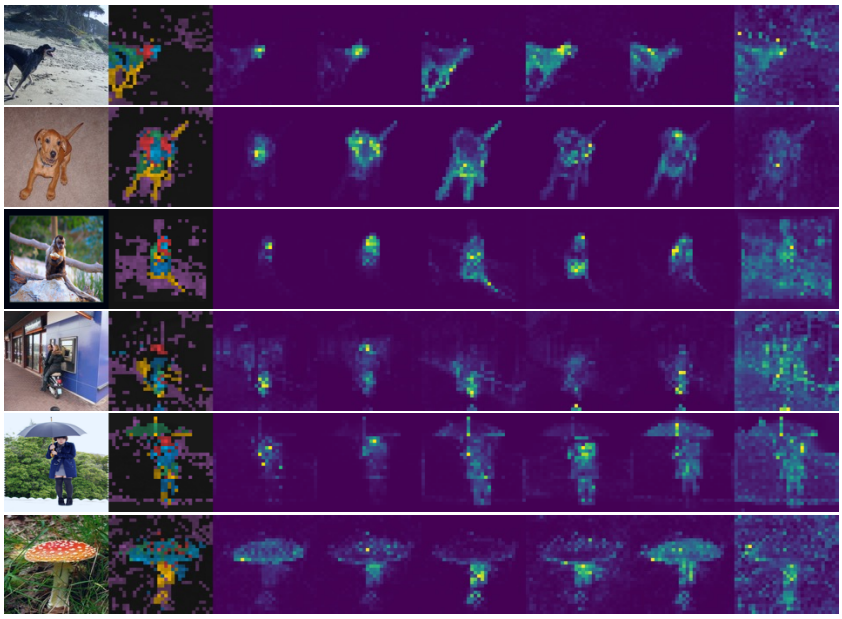
that POA's self-attention focuses more concentratedly on the foreground objects compared to DINOv2. For instance, in Figure 4a, POA distinctly highlights the regions of interest associated with foreground elements (such as the human, fish, trumpet, and snake). In contrast, the DINOv2 generates attention maps exhibit a more dispersed focus, often including areas of the background. In Figure 5, we showcase more self-attention map visualizations, comparing the outputs from multiple heads in the final layer of our method with those from DINOv2.



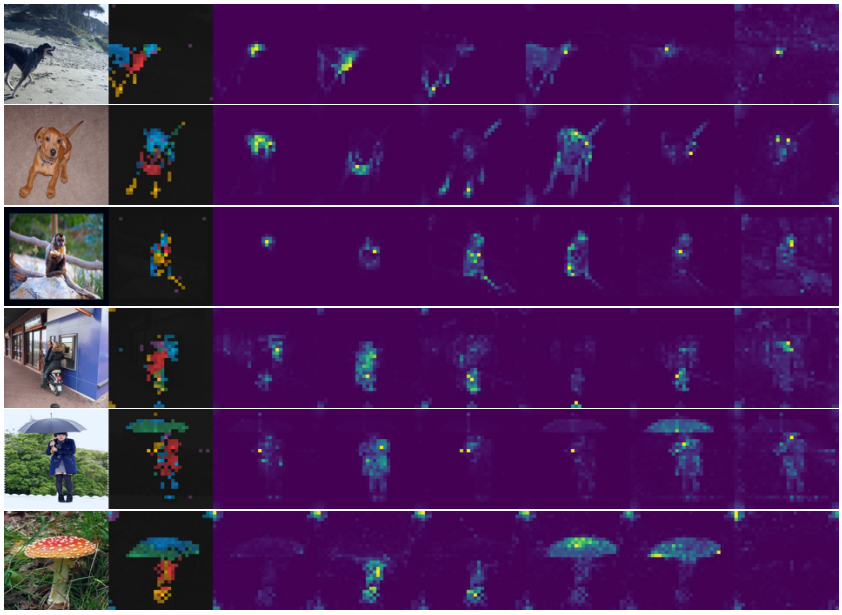
(a) Self-attention map of POA.

(b) Self-attention map of DINOv2.

Fig. 4: Visualization of Self-Attention Maps: we display the self-attention maps from multiple heads using distinct colors for differentiation. For both POA and DINOv2, we set the visualization threshold to 0.6, retaining top 60% of the attention mass.



(a) DIONv2



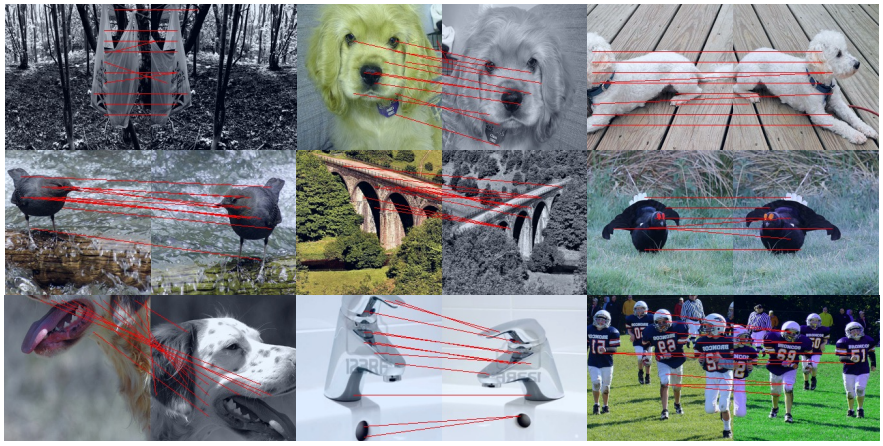
(b) POA

Fig. 5: Visualization for self-attention map from multiple heads of the last layer in ViT-S/16. The results indicate that POA concentrates its attention more accurately on foreground objects than DINOv2 does.

E.2 Visualization of Correspondence

We conduct a correspondence task that involves matching overlapping patches from two different augmentations of the same image or patches from two distinct images labeled as the same class. We present visualizations of the these patches with the highest self-attention scores obtained from a ViT-S/16 model pre-trained by POA, averaging the scores across multiple heads in the final layer. Figure 6 illustrates a selection of these image pair samples. The results indicate that POA excels in identifying correspondences both within varied views of a single image and across different segments of separate instances within the same class.

(a) Correspondence between two different images of the same category



(b) Correspondence between two different views of the same image

Fig. 6: Visualization of Correspondences: The top panel displays pairs of images sampled from two different views of a single image. The bottom panel shows pairs of images taken from two distinct images belonging to the same class.

E.3 Visualization of Pattern Layout for Class Token

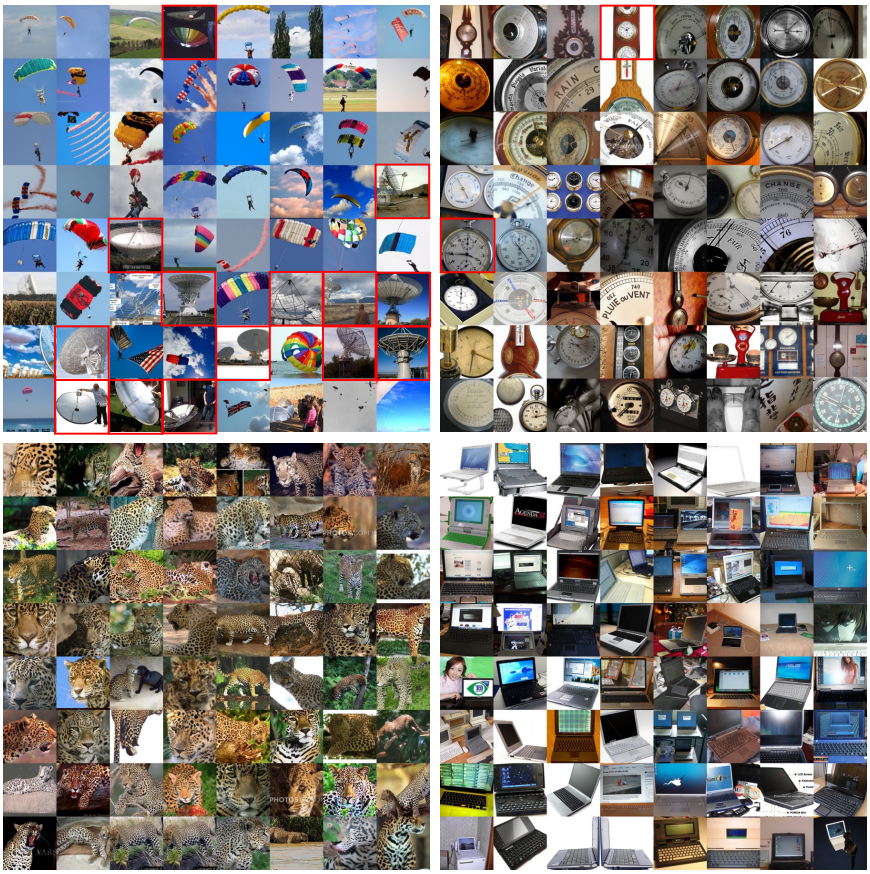


Fig. 7: Visualization for pattern layout of class token of POA. We indicate that the prototypes effectively cluster images based on similar semantic features, even when they may span different categories.

Figure 7 presents a visualization of the pattern layout associated with the class token in ViT-S/16 trained by POA. We display images that have the top-64 similarity scores with each prototype in ImageNet validation set, arranged in an 8×8 grid. The results indicates the high-quality semantic structure achieved through the self-distillation process applied to cross-view images within our POA framework. Furthermore, we observe that the prototypes effectively cluster images based on similar semantic features, even when they may span different categories. For instance, in the top-left image of the grid, while the primary category is 'parachute', there are also images of related but distinct categories such as 'radio reflector' and 'umbrella', which are outlined in red boxes within these prototypes. Similarly, in the top-right image, the main category featured is 'odometer', but it also includes images of semantically similar objects like 'clock'.

References

1. Asano, Y.M., Rupprecht, C., Vedaldi, A.: Self-labelling via simultaneous clustering and representation learning. ArXiv **abs/1911.05371** (2019)
2. Assran, M., Caron, M., Misra, I., Bojanowski, P., Bordes, F., Vincent, P., Joulin, A., Rabbat, M.G., Ballas, N.: Masked siamese networks for label-efficient learning. In: European Conference on Computer Vision (2022)
3. Bao, H., Dong, L., Piao, S., Wei, F.: BEit: BERT pre-training of image transformers. In: International Conference on Learning Representations (ICLR) (2022)
4. Bardes, A., Ponce, J., LeCun, Y.: Vicreg: Variance-invariance-covariance regularization for self-supervised learning. ArXiv **abs/2105.04906** (2021)
5. Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., Joulin, A.: Unsupervised learning of visual features by contrasting cluster assignments. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 9912–9924. Curran Associates, Inc. (2020)
6. Caron, M., Touvron, H., Misra, I., J'egou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. 2021 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 9630–9640 (2021)
7. Chen, T., Kornblith, S., Swersky, K., Norouzi, M., Hinton, G.E.: Big self-supervised models are strong semi-supervised learners. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 22243–22255. Curran Associates, Inc. (2020)
8. Chen, X., Xie, S., He, K.: An empirical study of training self-supervised vision transformers. 2021 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 9620–9629 (2021)
9. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. International Conference on Learning Representations (ICLR) (2021)
10. Fang, Z., Wang, J., Wang, L., Zhang, L., Yang, Y., Liu, Z.: Seed: Self-supervised distillation for visual representation. International Conference on Learning Representations (2021)
11. Gansbeke, W.V., Vandenhende, S., Georgoulis, S., Proesmans, M., Gool, L.V.: Learning to classify images without labels. ArXiv **abs/2005.12320** (2020)
12. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al.: Bootstrap your own latent-a new approach to self-supervised learning. Advances in neural information processing systems **33**, 21271–21284 (2020)
13. Guo, Y., Xu, M., Li, J., Ni, B., Zhu, X., Sun, Z., Xu, Y.: Hcsc: Hierarchical contrastive selective coding. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 9696–9705 (2022)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 770–778 (2015)
15. Huang, H., Campello, R.J.G.B., Erfani, S.M., Ma, X., Houle, M.E., Bailey, J.: LDReg: Local dimensionality regularized self-supervised learning. In: The Twelfth International Conference on Learning Representations (2024), `https://openreview.net/forum?id=oZyAqjAjJW`
16. Lee, B., Lee, S.: Implicit contrastive representation learning with guided stop-gradient. In: Thirty-seventh Conference on Neural Information Processing Systems (2023)

17. Li, C., Yang, J., Zhang, P., Gao, M., Xiao, B., Dai, X., Yuan, L., Gao, J.: Efficient self-supervised vision transformers for representation learning. International Conference on Learning Representations (ICLR) (2022)
18. Li, S., Wu, C., Li, A., Wang, Y., Tang, X., Yuan, G.: Waxing-and-waning: a generic similarity-based framework for efficient self-supervised learning. In: The Twelfth International Conference on Learning Representations (2024), https://openreview.net/forum?id=TilcG5C8bN
19. Li, Z., Zhu, Y., Yang, F., Li, W., Zhao, C., Chen, Y., Chen, Z., Xie, J., Wu, L., Zhao, R., Tang, M., Wang, J.: Univip: A unified framework for self-supervised visual pre-training. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 14607–14616 (2022)
20. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. 2021 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 9992–10002 (2021)
21. Oquab, M., Darcet, T., Moutakanni, T., Vo, H.V., Szafraniec, M., Khalidov, V., Fernandez, P., HAZIZA, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P.Y., Li, S.W., Misra, I., Rabbat, M., Sharma, V., Synnaeve, G., Xu, H., Jegou, H., Mairal, J., Labatut, P., Joulin, A., Bojanowski, P.: DINOv2: Learning robust visual features without supervision. Transactions on Machine Learning Research (2024)
22. Pang, B., Zhang, Y., Li, Y., Cai, J., Lu, C.: Unsupervised visual representation learning by synchronous momentum grouping. In: European Conference on Computer Vision (2022)
23. Ren, S., Wang, H., Gao, Z., He, S., Yuille, A.L., Zhou, Y., Xie, C.: A simple data mixing prior for improving self-supervised learning. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 14575–14584 (2022)
24. Ruan, Y., Singh, S., Morningstar, W.R., Alemi, A.A., Ioffe, S., Fischer, I., Dillon, J.V.: Weighted ensemble self-supervised learning. In: The Eleventh International Conference on Learning Representations (2023)
25. Sharma, R., Ji, K., zhiqiang xu, Chen, C.: AUC-CL: A batchsize-robust framework for self-supervised contrastive representation learning. In: The Twelfth International Conference on Learning Representations (2024), https://openreview.net/forum?id=YgMdDQB09U
26. Shvetsova, N., Petersen, F., Kukleva, A., Schiele, B., Kuehne, H.: Learning by sorting: Self-supervised learning with group ordering constraints. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 16453–16463 (October 2023)
27. Song, K., Xie, J., Zhang, S., Luo, Z.: Multi-mode online knowledge distillation for self-supervised visual representation learning. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 11848–11857 (2023)
28. Song, K., Zhang, S., Luo, Z., Wang, T., Xie, J.: Semantics-consistent feature search for self-supervised visual representation learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 16099–16108 (October 2023)
29. Tao, C., Wang, H., Zhu, X., Dong, J., Song, S., Huang, G., Dai, J.: Exploring the equivalence of siamese self-supervised learning via a unified gradient framework. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 14411–14420 (2021)
30. Tao, C., Zhu, X., Huang, G., Qiao, Y., Wang, X., Dai, J.: Siamese image modeling for self-supervised vision representation learning. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 2132–2141 (2022)

31. Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., Isola, P.: What makes for good views for contrastive learning. ArXiv **abs/2005.10243** (2020)
32. Tomasev, N., Bica, I., McWilliams, B., Buesing, L., Pascanu, R., Blundell, C., Mitrovic, J.: Pushing the limits of self-supervised resnets: Can we outperform supervised learning without labels on imagenet? arXiv preprint arXiv:2201.05119 (2022)
33. Wang, X., Huang, Y., Zeng, D., Qi, G.J.: Caco: Both positive and negative samples are directly learnable via cooperative-adversarial contrastive learning. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**, 10718–10730 (2022)
34. Xie, Z., Lin, Y., Yao, Z., Zhang, Z., Dai, Q., Cao, Y., Hu, H.: Self-supervised learning with swin transformers. ArXiv **abs/2105.04553** (2021)
35. Zhou, J., Wei, C., Wang, H., Shen, W., Xie, C., Yuille, A., Kong, T.: ibot: Image bert pre-training with online tokenizer. International Conference on Learning Representations (ICLR) (2022)
36. Zhou, P., Zhou, Y., Si, C., Yu, W., Ng, T.K., Yan, S.: Mugs: A multi-granular self-supervised learning framework. ArXiv **abs/2203.14415** (2022)