Revitalizing CNN Attention via Transformers in Self-Supervised Visual Representation Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Studies on self-supervised visual representation learning (SSL) improve encoder backbones to discriminate training samples without labels. While CNN encoders via SSL achieve comparable recognition performance to those via supervised learning, their network attention is under-explored for further improvement. Motivated by the transformers that explore visual attention effectively in recognition scenarios, we propose a CNN Attention REvitalization (CARE) framework to train attentive CNN encoders guided by transformers in SSL. The proposed CARE framework consists of a CNN stream (C-stream) and a transformer stream (T-stream), where each stream contains two branches. C-stream follows an existing SSL framework with two CNN encoders, two projectors, and a predictor. T-stream contains two transformers, two projectors, and a predictor. T-stream connects to CNN encoders and is in parallel to the remaining C-Stream. During training, we perform SSL in both streams simultaneously and use the T-stream output to supervise C-stream. The features from CNN encoders are modulated in T-stream for visual attention enhancement and become suitable for the SSL scenario. We use these modulated features to supervise C-stream for learning attentive CNN encoders. To this end, we revitalize CNN attention by using transformers as guidance. Experiments on several standard visual recognition benchmarks, including image classification, object detection, and semantic segmentation, show that the proposed CARE framework improves CNN encoder backbones to the state-of-the-art performance. Code will be made publicly available.

1 Introduction

Learning visual features effectively has a profound influence on the recognition performance [5, 50]. Upon handling large-scale natural images, self-supervised visual representation learning benefits downstream recognition tasks via pretext feature training. Existing SSL methods typically leverage two branches to measure the similarity between different view representations derived from the same input image. By maximizing the similarity between the correlated views within one image (e.g., BYOL [24] and SimSiam [14]), or minimizing the similarity between views from different images (e.g., MoCo [27], SimCLR [12] and Barlow Twins [65]), these methods are shown to be effective towards learning self-supervised visual representations.

SSL evolves concurrently with the transformer. Debuting at natural language processing [58, 17], transformers have shown their advantages to process large-scale visual data since ViT [19]. The encoder-decoder architecture in this vision transformer consistently explores global attention without convolution. This architecture is shown effective for visual recognition with [7, 68] or without CNN integration [36, 21]. Inspired by these achievements via supervised learning, studies [15, 10, 4, 63] arise recently to train transformers in a self-supervised manner. These methods maintain most of the

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

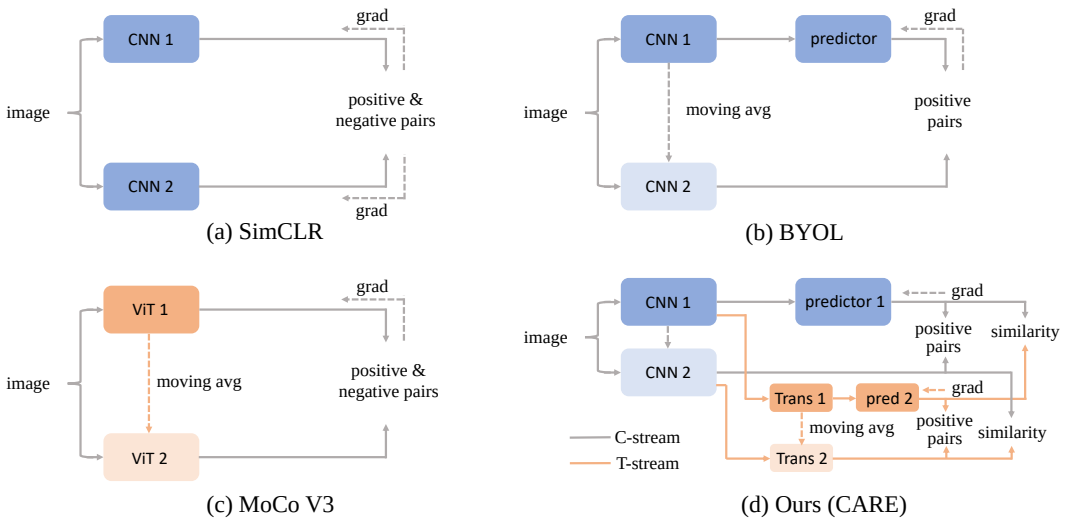

Figure 1: SSL framework overview. The solid lines indicate network pipeline, and the dash lines indicate network updates. MoCo V3 [15] explores visual attention by explicitly taking a vision transformer as encoder, while SimCLR [12] and BYOL [24] do not learn an attentive CNN encoder. Our CARE framework consists of a C-stream and a T-stream to explore visual attention in CNN encoders with transformer supervision. Note that only target CNN encoder (i.e., CNN_1) is preserved after pre-training. We do not show projectors in (b) and (d) for simplicity.

SSL pipeline (i.e., encoder, projector, and predictor) utilized for training CNN encoders. Without significant framework alteration, original SSL methods for CNN encoders can be adapted to train transformer encoders and achieve favorable performance.

The success of using transformer encoders indicates that visual attention benefits encoder backbones in SSL. On the other hand, in supervised learning, CNN attention is developed via network supervision [44]. However, we observe that existing SSL methods do not incorporate visual attention within CNN encoders. This motivates us to explore CNN attention in SSL. We expect CNN encoders to maintain similar visual attention to transformers for recognition performance improvement with lower computational complexity and less memory consumption.

In this paper, we propose a CNN Attention REvitalization framework (CARE) to make CNN encoder attentive via transformer guidance. Fig. 1 (d) shows an intuitive illustration of CARE and compares it with other state-of-the-art SSL frameworks. There are two streams (i.e., C-stream and T-stream) in CARE where each stream contains two branches. C-stream is similar to existing SSL frameworks with two encoders, two projectors, and one predictor. T-stream consists of two transformers, two projectors, and one predictor. T-stream takes CNN encoder features as input and improves feature attention via transformers. During the training process, we perform SSL in both streams simultaneously and use the T-stream output to supervise C-stream. The self-supervised learning in T-stream ensures attentive features produced by transformers are suitable for this SSL scenario. Meanwhile, we use the attention supervision on C-stream. This supervision enables both C-stream and T-stream to produce similar features. The feature representation of CNN encoders is improved by visual attention from transformers. As a result, the pre-trained CNN encoder produces attentive features, which benefits downstream recognition scenarios. Experiments on standard image classification, object detection, and semantic segmentation benchmarks show that the proposed CARE framework improves prevalent CNN encoder backbones to the state-of-the-art performance.

2 Related works

In the proposed CARE framework, we introduce transformers into self-supervised visual representation learning. In this section, we perform a literature survey on related works from the perspectives of visual representation learning as well as vision transformers.

2.1 Visual representation learning

There is an increasing need to learn good feature representations with unlabeled images. The general feature representation benefits downstream visual recognition scenarios. Existing visual representation learning methods can be mainly categorized as generative and discriminative methods. The generative methods typically use an auto-encoder for image reconstruction [59, 46], or model data and representation in a joint embedding space [18, 6]. The generative methods focus on image pixel-level details and are computationally intensive. Further adaption is required for downstream visual recognition scenarios.

The discriminative methods formulate visual representation learning as sample comparisons. Recently, contrastive learning is heavily investigated since its efficiency and superior performance. By creating different views from images, positive and negative sample pairs constitute the learning process. Examples include memory bank [62], multi-view coding [54, 57], predictive coding [30, 56], pretext invariance [40], knowledge distillation [22, 10] and information maximization [31]. While negative pairs are introduced in MoCo [27] and SimCLR [12], studies (e.g., BYOL [24] and SimSiam [14]) show that using only positive pairs are effective. Also, clustering methods [8, 9] construct clusters for representation learning. The negative pairs are not introduced in these methods. Besides these discriminative methods focusing on image data, there are similar methods learning representations from either video data [26, 32] or multi-modality data [1, 2, 3]. Different from these SSL methods, we use transformer to improve CNN encoder attention.

2.2 Vision transformers

Transformer is proposed in [58] where self-attention is shown effective for natural language processing. BERT [17] further boosts its performance via self-supervised training. The sequential modeling of transformer has activated a wide range of researches in natural language processing [53], speech processing [52], and computer vision [25]. In this section, we only survey transformer related works from the computer vision perspective.

There are heavy researches on transformers in both visual recognition and generation. ViT [19] has shown that CNN is not a must in image classification. DETR [7], Deformable DETR [70] and RelationNet++ [16] indicate that transformers are able to detect objects with high precisions. SETR [68] brings transformers into semantic segmentation while VisTR [61] has shown transformers are able to perform video object segmentation. TrackFormer [38] introduces transformers into multiple object tracking. A general form of transformer is formulated in NLM [60] for video classification. Furthermore, transformers have been show effective in image generation [43] and image processing [11] scenarios. Examples include image super-resolution [64], video inpainting [66], and video captioning [69]. There are several emerging studies [15, 10, 4, 63] on how to use self-supervised learning to improve a transformer backbone. The learning paradigm for CNN encoders is adapted to the transformer without significant alteration. Different from existing methods that focus on learning a transformer encoder backbone with supervised or self-supervised learning. We explore how to use the transformer as guidance to enhance CNN visual attention. The pretrained CNN encoder benefits downstream recognition scenarios.

3 Proposed method

Our CARE framework consists of C-stream and T-stream. Fig. 2 shows an overview of the pipeline. We first illustrate the detailed structure of these streams. Then, we illustrate the network training process. The CNN encoder features are visualized as well for attention display.

3.1 CNN-stream (C-stream)

Our C-stream is similar to the existing SSL framework [24] where there are two CNN encoders, two projectors, and one predictor. The structures of the two encoders are the same, and the structures of the two projectors are the same. Given a training image x, we process it with a set of random augmentations to create two *different* augmented views. We feed these two views to C-stream and obtain corresponding outputs $f_1(x)$ and $f_2(x)$, respectively. Then, we compute a loss $\mathcal{L}_c$ to penalize the dissimilarity of the outputs. This loss term is the mean squared error of the normalized feature

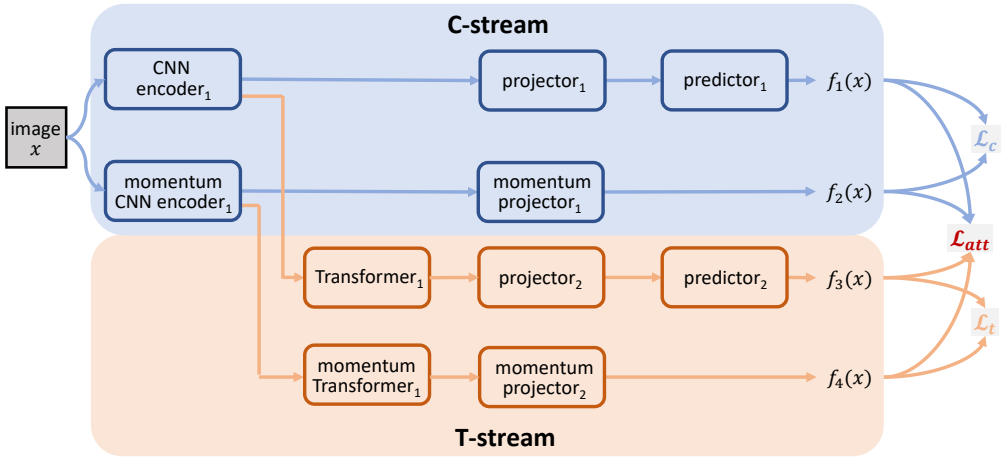

Figure 2: The pipeline of CARE. It consists of C-stream and T-stream. C-stream is similar to the existing SSL framework, and we involve transformers in T-stream. During training, we perform SSL in each stream (i.e., $\mathcal{L}_c$ and $\mathcal{L}_t$), and use T-stream output to supervise C-stream (i.e., $\mathcal{L}_{att}$). The CNN encoder becomes attentive via T-stream attention supervision.

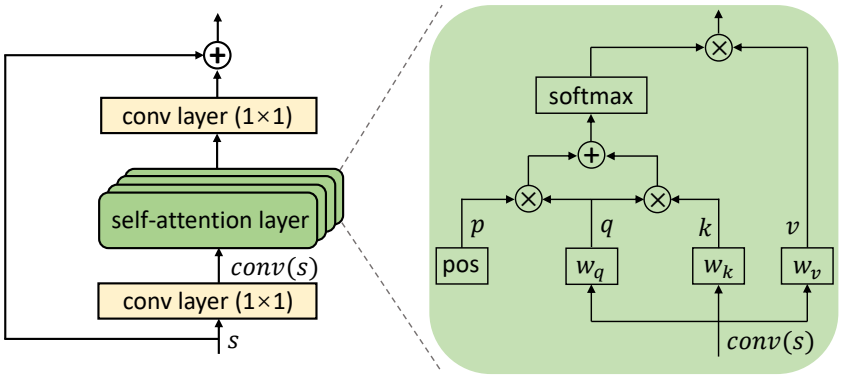

Figure 3: Transformer framework. The architectures of the two transformers in T-stream are the same. Each transformer consists of n attention blocks. We show one attention block on the left, where the structure of one self-attention layer is shown on the right.

vectors and can be written as what follows:

$$\mathcal{L}_c = 2 - 2 \cdot \frac{\langle f_1(x), f_2(x) \rangle}{\|f_1(x)\|_2 \cdot \|f_2(x)\|_2} \tag{1}$$

where $|| \cdot ||_2$ is the ℓ_2 normalization. As the inputs of C-stream are from one image, the outputs of C-stream are set to become similar during the training process.

3.2 Transformer-stream (T-stream)

The T-stream takes the output feature maps of the CNN encoders as its inputs, which are set in parallel to the C-stream. It consists of two transformers, two projectors, and one predictor. The structures of the projectors and the predictor are the same as those in the C-stream. The structures of two transformers share the same architecture, which consists of n consecutive attention blocks where each block contains two 1×1 convolutional layers and multi-head self-attention (MHSA) layers. Fig. 3 shows the structure of an attention block in the transformer. The input feature map (denoted as s) of an attention block is first processed by the a convolutional layer, and then passes the self-attention layer and finally the another convolutional layer. MHSA consists of multiple attention heads that process the input in parallel. In one attention head, as illustrated on the right of Fig. 3, the input feature map is mapped to the query feature (q), the key feature (k), and the value feature (k) via

3 different 1×1 convolutional layers w_q, w_k, and w_v, respectively. As detailed in Eq. (2), the query q and key k are multiplied to form the content-based attention, and q and the position encoding p are multiplied to form the position-based attention.

$$\text{Attention}(q, k, v) = \text{softmax}(\frac{qp^{\text{T}} + qk^{\text{T}}}{\sqrt{d_k}})v \tag{2}$$

where d_k is the dimension of the query and the key, there are learnable parameters in the positional encoding module [43] to output p that follows the shape of s. In Eq. (2), we perform matrix multiplication between q and p^{T}, q and k^{T}, and the softmax output and v by treating each pixel as a token [58] (i.e., for a feature map with c channels and spatial dimension of $h \times w$, it forms $h \cdot w$ c-dimensional tokens and thus a matrix of size $h \cdot w \times c$). Besides, we perform matrix addition between qp^{T} and qk^{T}. The output of the second convolutional layer is added to the original input feature map s via a residual connection [29], and finally passes a ReLU activation layer.

In T-stream, the outputs of the two transformers are feature maps with spatial dimensions, which are then average-pooled and sent to the projectors and the predictor. We denote the outputs of T-stream as $f_3(x)$ and $f_4(x)$. Following the dissimilarity penalty in Sec. 3.1, we compute $\mathcal{L}_t$ as follows:

$$\mathcal{L}_t = 2 - 2 \cdot \frac{\langle f_3(x), f_4(x) \rangle}{\|f_3(x)\|_2 \cdot \|f_4(x)\|_2}. \tag{3}$$

Besides introducing SSL loss terms in both streams, we use the T-stream output to supervise the C-stream. This attention supervision loss term can be written as:

$$\mathcal{L}_{\text{att}} = ||f_1(x) - f_3(x)||_2 + ||f_2(x) - f_4(x)||_2 \tag{4}$$

where the C-stream outputs are required to resemble the T-stream outputs during the training process.

3.3 Network training

The proposed CARE consists of two streams and the loss terms have been illustrated above. The final objective function for network training can be written as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_c + \mathcal{L}_t + \lambda \cdot \mathcal{L}_{\text{att}} \tag{5}$$

where λ is a constant value controlling the influence of the attention supervision loss. After computing $\mathcal{L}_{\text{total}}$, we perform back-propagation only on the upper branches of the C-stream and T-stream. Specifically in Fig. 2, the CNN encoder$_1$, the projector$_1$, and the predictor$_1$ are updated via the computed gradients in C-stream. Meanwhile, the Transformer$_1$, the projector$_2$, and the predictor$_2$ are updated via computed gradients in T-stream. Afterwards, we perform a moving average update [34, 24, 27] on the momentum CNN encoder$_2$ based on the CNN encoder$_1$, on the momentum projector$_1$ based on the projector$_1$, on the momentum transformer$_1$ based on the transformer$_1$, and on the momentum projector$_2$ based on the projector$_2$. We only use positive samples when training the network. As analyzed in [24], using projectors and a predictor with a moving average update is effective to prevent model collapse. In our network, we adopt a similar design in both streams to facilitate network training and obverse that using only positive samples does not cause model collapse. After pretext training, we only keep the CNN encoder$_1$ in CARE. This encoder is then utilized for downstream recognition scenarios. We provide our pseudo-code in the supplementary materials.

3.4 Visualizations

Our CARE framework improves CNN encoder attention via transformer guidance. We show how encoders attend to the input images by visualizing their attention maps. The ResNet-50 encoder backbone is used for visualization. We train this encoder for 200 epochs using only C-stream and the whole CARE framework, respectively. For input images, we use [48] to visualize encoder responses. The visualization maps are scaled equally for comparison.

Fig. 4 shows the visualization results. The input images are presented in (a), while the attention maps from the encoders trained with C-stream and CARE are shown in (b) and (c), respectively. Overall, the attention of the encoder trained by CARE is more intense than that by C-stream, which indicates that T-stream in CARE provides effective supervision for CNN encoders to learn to attend to object

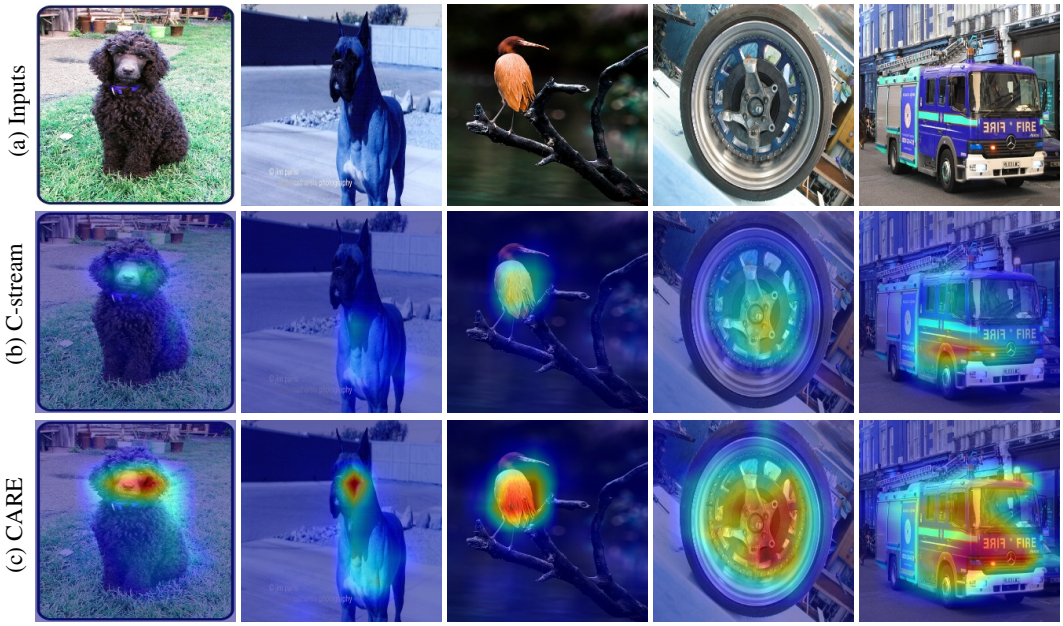

Figure 4: Attention visualization of CNN encoders. We train two ResNet-50 encoders by using only C-stream and the whole method CARE, respectively. By taking the same image in (a) as inputs, the attention maps of these two encoders are shown in (b) and (c). The attention learned via CARE is more intense around the object regions shown in (c). In the attention visualization maps, pixels marked as red indicate the network pays more attention to the current regions.

parts. The T-steam helps CNN encoders adaptively choose to focus on local regions or global regions. For example, when global information is needed for classification, the CNN encoder learned by CARE will pay more attention to the whole object, as in the last column in (c), rather than a limited region, as shown in (b). On the other hand, when local information is sufficient for classification, the CNN encoder learned via CARE will pay more intense attention to the specific regions (e.g., the animals' heads in (c) on the first and second columns). The attention maps shown in the visualization indicate that the CNN encoder becomes attentive on the object region via transformer guidance in our CARE framework.

4 Experiments

In this section, we perform experimental validation on our CARE method. First, we introduce the implementation details. Then, we compare our CARE method to state-of-the-art SSL methods on standard benchmarks, including image classification, object detection, and semantic segmentation. Furthermore, we conduct an ablation study to analyze each component of the proposed CARE method. More results are provided in the supplementary files.

4.1 Implementation details

Image pre-processing. The training images we use during pretext training are from the ImageNet-1k [47] dataset. We follow [24] to augment image data before sending them to the encoders. Specifically, we randomly crop patches from one image and resize them to a fixed resolution of 224×224. Then, we perform random horizontal flip and random color distortions on these patches. The Gaussian blur, the decolorization, and the solarization operations are also adopted to preprocess these patches.

Network architectures. We use ResNet encoder backbones [29] (i.e., ResNet-50, ResNet-101, and ResNet-152) in our experiments. The architectures of the projectors and the predictors are the same and follow [24]. Each projector and predictor consist of a fully-connected layer with a

Table 1: Linear evaluations on ImageNet with top-1 accuracy (in %). The evaluation results of CNN encoders that are trained by using more epochs are provided in the supplementary files.

(a) Classification accuracy by using the ResNet-50 encoder.

Method	100ep	200ep	400ep
CMC [54]	-	66.2	-
PCL v2 [33]	-	67.6	-
SimCLR [12]	66.5	68.3	69.8
MoCo v2 [13]	67.4	69.9	71.0
SwAV [9]	66.5	69.1	70.7
SimSiam [14]	68.1	70.0	70.8
InfoMin Aug. [55]	-	70.1	-
BYOL [24]	66.5	70.6	73.2
Barlow Twins [65]	-	-	72.5
CARE (ours)	**72.0**	**73.8**	**74.7**

(b) Classification accuracy via CNN and Transformer encoders.

Method	Arch.	Param.	Epoch	GFlops	Top-1
CMC [54]	ResNet-50(2×)	69M	-	11.4	70.6
BYOL [24]	ResNet-50(2×)	69M	100	11.4	71.9
BYOL [24]	ResNet-101	45M	100	7.8	72.3
BYOL [24]	ResNet-152	60M	100	11.6	73.3
BYOL [24]	ViT-S	22M	300	4.6	71.0
BYOL [24]	ViT-B	86M	300	17.7	73.9
MoCo v3 [15]	ViT-S	22M	300	4.6	72.5
CARE (ours)	ResNet-50	25M	200	4.1	**73.8**
CARE (ours)	ResNet-50(2×)	69M	100	11.4	**73.5**
CARE (ours)	ResNet-50(2×)	69M	200	11.4	**75.0**
CARE (ours)	ResNet-101	45M	100	7.8	**73.5**
CARE (ours)	ResNet-152	60M	100	11.6	**74.9**

batch normalization and a ReLU [42] activation, followed by another fully-connected layer. The transformer in the T-stream contains n attention blocks as shown in Fig. 3.

Network training process. We use the SGD optimizer with a momemtum of 0.9 during pretext training. The base learning rate is set as 0.05 and scaled linearly with respect to the batch size [23] (i.e., $\mathrm{lr}_{\mathrm{base}} = 0.05 \times \mathrm{BatchSize}/256$). We start the pretext training with a warm-up of 10 epochs where the learning rate rises linearly from 10^{-6} to $\mathrm{lr}_{\mathrm{base}}$. Then, we use a cosine decay schedule for the learning rate without restarting it [37, 24] to train the network. The momentum update coefficient of network parameters (denoted as τ) is increased from 0.99 to 1 via a cosine design (i.e., $\tau = 1 - (1 - \tau_{\mathrm{base}}) \cdot (\cos(\pi t/T) + 1)/2$, where t is the current training step and T is the total number of training steps). We train CARE using 8 Tesla V100 GPUs with a batch size of 1024. The automatic mixed precision training strategy [39] is adopted for training speedup.

4.2 Comparison to state-of-the-art approaches

We compare feature representations of CNN encoders learned by our method and state-of-the-art SSL methods. The comparison resides on recognition scenarios, including image classification (self-supervised and semi-supervised learning configurations), object detection, and semantic segmentation.

Self-supervised learning on image classifications. We follow [27] to use standard linear classification protocol where the parameters of the encoder backbone is fixed and an additional linear classifier is added to the backbone. We train this classifier using SGD for 80 epochs with a learning rate of 0.2, a momentum of 0.9, and a batch size of 256. The ImageNet training set is used for the training and the ImageNet validation set is used for evaluation.

Table 1 shows the linear evaluation results with the top-1 accuracy. We show the classification results by using the ResNet-50 encoder learned via different SSL methods in Table 1a. In this table, our CARE method consistently outperforms other methods under different training epochs. Specifically, our method achieves a 74.7% top-1 accuracy under 400 training epochs, which is 1.5% higher than the second-best method BYOL. Meanwhile, we compare our method to other methods that use CNN and transformer (i.e., ViT [19]) encoders in Table 1b. The results show that, under similar number of parameters of CNN and transformer encoders (i.e., ResNet-50 and ViT-S), CARE achieves higher accuracy than other SSL methods. This indicates that CARE improves CNN encoders to outperform transformer encoders by utilizing visual attention. The linear classification results with more training time are provided in the supplementary files, where our CARE method also consistently prevails.

Semi-supervised learning on image classifications. We evaluate our CARE method by using a semi-supervised training configuration on the ImageNet dataset. After pretext training, we finetune the encoder by using a small subset of ImageNet's training set. We follow the semi-supervised learning protocol [24, 12] to use 1% and 10% training data (the same data splits as in [12]) during finetuning. Table 2 shows the top-1 and top-5 accuracy on the ImageNet validation set. The results

Table 2: Image classification by using semi-supervised training on ImageNet with Top-1 and Top-5 accuracy (in %). We report our method with more training epochs in the supplementary files.

(a) Classification accuracy by using the ResNet-50 encoder.

Method	Epoch	Top-1		Top-5	
		1%	10%	1%	10%
Supervised [67]	-	25.4	56.4	48.4	80.4
PIRL [41]	-	-	-	57.2	83.8
SimCLR [12]	800	48.3	65.6	75.5	87.8
BYOL [24]	800	53.2	68.8	78.4	89.0
CARE (Ours)	400	**60.0**	**69.6**	**81.3**	**89.3**

(b) Classification accuracy by using other CNN encoders.

Method	Arch.	Epoch	Top-1		Top-5	
			1%	10%	1%	10%
BYOL [24]	ResNet-50(2×)	100	55.6	66.7	77.5	87.7
BYOL [24]	ResNet-101	100	55.8	65.8	79.5	87.4
BYOL [24]	ResNet-152	100	56.8	67.2	79.3	88.1
CARE (ours)	ResNet-50(2×)	100	57.4	67.5	79.8	88.3
CARE (ours)	ResNet-101	100	57.1	67.1	80.8	88.2
CARE (ours)	ResNet-152	100	59.4	69.0	82.3	89.0

Table 3: Transfer learning to object detection and instance segmentation. The best two results in each column are in bold. Our method achieves favorable detection and segmentation performance by using limited training epochs.

Method	Epoch	COCO det			COCO instance seg			VOC07+12 det		
		AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}	AP	AP_{50}	AP_{75}
Rand Init	-	26.4	44.0	27.8	29.3	46.9	30.8	33.8	60.2	33.1
Supervised	90	38.2	58.2	41.2	33.3	54.7	35.2	53.5	81.3	58.8
PIRL[41]	200	37.4	56.5	40.2	32.7	53.4	34.7	55.5	81.0	61.3
MoCo[27]	200	38.5	58.3	41.6	33.6	54.8	35.6	55.9	81.5	62.6
MoCo-v2[13]	200	38.9	58.4	42.0	34.2	55.2	36.5	57.0	82.4	63.6
MoCo-v2[13]	800	39.3	58.9	42.5	34.4	55.8	36.5	57.4	82.5	64.0
SwAV[9]	200	32.9	54.3	34.5	29.5	50.4	30.4	-	-	-
SwAV[9]	800	38.4	58.6	41.3	33.8	55.2	35.9	56.1	82.6	62.7
Barlow Twins[65]	800	39.2	59.0	42.5	34.3	56.0	36.5	56.8	82.6	63.4
BYOL [24]	200	39.2	58.9	42.4	34.3	55.6	36.7	57.0	82.3	63.6
CARE (Ours)	200	**39.4**	**59.2**	**42.6**	**34.6**	**56.1**	**36.8**	**57.7**	**83.0**	**64.5**
CARE (Ours)	400	**39.6**	**59.4**	**42.9**	**34.7**	**56.1**	**36.9**	**57.9**	**83.0**	**64.7**

indicate that our CARE method achieves higher classification accuracy than other SSL methods under different encoder backbones and training epochs.

Transfer learning to object detection and semantic segmentation. We evaluate CARE's representations on the downstream object detection and semantic segmentation scenarios. We use the standard VOC-07, VOC-12, and COCO datasets [20, 35]. We follow the standard protocol [27] to integrate the pretext trained CNN encoder into Faster-RCNN [45] when evaluating object detection results on VOC-07 and VOC-12 datasets. On the other hand, we integrate this encoder into Mask-RCNN [28] when evaluating object detection and semantic segmentation in COCO dataset. The ResNet-50 encoder is used in all the methods. All detectors are finetuned for 24k iterations using VOC-07 and VOC-12 training sets and are evaluated on the VOC-07 test set. On the COCO dataset, all models are finetuned via the $1\times$ schedule. The results are averaged over five independent trials.

Table 3 shows the evaluation results. Compared to the supervised method, our CARE improves detection (i.e., 1.4% on COCO and 4.4% on VOC) and segmentation (i.e., 1.4% on COCO) performance. On the other hand, our CARE method compares favorably against other SSL methods. Note that the results of our method are reported under 200 and 400 epochs, which are still higher than other methods under 800 epochs. This indicates the effectiveness of our CARE method on learning the CNN encoder backbone. The comparisons on COCO datasets are similar to those on VOC. Specifically, our CARE method achieves a 0.5% AP^{bb} increase and a 0.4% AP^{mk} increase upon MoCo v2 under 200 epochs. The performance improvement is mainly brought by the visual attention integration from transformer guidance.

Table 4: Analysis on λ.

λ	Top-1
0	70.52
1	70.88
10	70.96
100	72.06
250	72.00

Table 5: Analysis on n.

n	Top-1
2	71.08
3	71.60
4	71.79
5	72.06
6	69.37

Table 6: Analysis on the positional encoding.

Position encoding	Top-1
none	69.49
sin-cos absolute [58]	66.68
learnable absolute [51]	72.01
learnable relative [49]	72.06

4.3 Ablation studies

In our CARE method, visual attention is explored via transformers to supervise C-stream. We analyze the influence of attention supervision by using different values of λ in Eq. (5). Also, we analyze how the number of attention blocks and the positional encoding integration affect feature representations. We use the ResNet-50 as encoder backbone and the number of training epochs is set to 100. The top-1 image classification accuracy on ImageNet via SSL is reported to indicate feature representation effectiveness.

Supervision influence λ. We study how the attention supervision term $\mathcal{L}_{\text{att}}$ in Eq. (5) affects feature representation by using different values of λ. Table 4 shows the evaluation results. When we set λ as 0, the CNN encoder is learned without attention supervision. In this case, the performance decreases significantly. When we increase the value of λ, the attention supervision increases as well. We observe that $\lambda = 100$ achieves the best performance and adopt this setting in the experiments.

Number of attention blocks. We analyze how the capacity of transformer in T-stream affects the recognition performance. We set $n = [2, ..., 6]$ to constitute five transformers in T-stream with increasing capacities. Then, we report the recognition performance of the corresponding CNN encoders in Table 5. When n is larger, the transformer capacity increases and stronger visual attention are explored. This attention supervises C-stream to improve the encoder. However, the recognition performance drop when $n = 6$. This may be due to the broken balance between the attention loss $\mathcal{L}_{\text{att}}$ and the original SSL loss $\mathcal{L}_c$. In our experiment, we set $n = 5$ in our CARE method.

Positional encoding. We analyze how positional encoding affects the recognition performance. A series of positional encoding configurations [58, 51, 49] are adopted for our analysis. Table 6 shows the recognition results. Without positional encoding, the performance decreases significantly. Meanwhile, the conventional fixed sine and cosine encoding configuration is not suitable for SSL. We experimentally find that using learnable parameter configuration in positional encoding improves recognition performance and adopt [49] in our CARE method.

5 Concluding remarks

Transformers have advanced visual recognition via attention exploration architectures. In self-supervised visual representation learning, studies emerge to utilize transformer backbones for recognition performance improvement. This indicates that visual attention explored by the transformers benefit SSL methods. Motivated by this success, we in this paper investigate how to explore visual attention effectively to benefit CNN encoders in SSL. We propose CARE to develop a parallel transformer stream together with the CNN stream. The visual attention is thus explored via transformers to supervise the CNN stream during SSL. Although the limitation occurs that more computational cost is spent on the SSL loss terms and attention supervision loss term, the learned CNN encoder becomes attentive with transformer guidance and does not consume more cost in downstream tasks. Experiments on the standard visual recognition benchmarks, including image classification, object detection, and semantic segmentation, indicate that our CARE method improves CNN encoder backbones to a new state-of-the-art performance.

Negative social impact. The SSL methods utilize unlabeled data for neural network training. The data content may not be verified because human labeling is not required during the procedure of data collection. Nevertheless, a careful data verification will avoid this negative impact.

References

[1] Jean-Baptiste Alayrac, Adrià Recasens, Rosalia Schneider, Relja Arandjelović, Jason Ramapuram, Jeffrey De Fauw, Lucas Smaira, Sander Dieleman, and Andrew Zisserman. Self-supervised multimodal versatile networks. In *Advances in Neural Information Processing Systems*, 2020.

[2] Humam Alwassel, Dhruv Mahajan, Bruno Korbar, Lorenzo Torresani, Bernard Ghanem, and Du Tran. Self-supervised learning by cross-modal audio-video clustering. In *Advances in Neural Information Processing Systems*, 2020.

[3] Yuki M Asano, Mandela Patrick, Christian Rupprecht, and Andrea Vedaldi. Labelling unlabelled videos from scratch with multi-modal self-supervision. In *Advances in Neural Information Processing Systems*, 2020.

[4] Sara Atito, Muhammad Awais, and Josef Kittler. Sit: Self-supervised vision transformer. *arXiv preprint arXiv:2104.03602*, 2021.

[5] Y-Lan Boureau, Francis Bach, Yann LeCun, and Jean Ponce. Learning mid-level features for recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2010.

[6] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, 2020.

[8] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision*, 2018.

[9] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems*, 2020.

[10] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv:2104.14294*, 2021.

[11] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. *arXiv preprint arXiv:2012.00364*, 2020.

[12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, 2020.

[13] Xinlei Chen, Haoqi Fan, Ross B. Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *CoRR*, abs/2003.04297, 2020.

[14] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[15] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised visual transformers. *arXiv preprint arXiv:2104.02057*, 2021.

[16] Cheng Chi, Fangyun Wei, and Han Hu. Relationnet++: Bridging visual representations for object detection via transformer decoder. In *Advances in Neural Information Processing Systems*, 2020.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[18] Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. In *Advances in Neural Information Processing Systems*, 2019.

[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[20] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[21] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. *arXiv preprint arXiv:2104.11227*, 2021.

[22] Zhiyuan Fang, Jianfeng Wang, Lijuan Wang, Lei Zhang, Yezhou Yang, and Zicheng Liu. Seed: Self-supervised distillation for visual representation. In *International Conference on Learning Representations*, 2021.

[23] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[24] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, 2020.

[25] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on visual transformer. *arXiv preprint arXiv:2012.12556*, 2020.

[26] Tengda Han, Weidi Xie, and Andrew Zisserman. Self-supervised co-training for video representation learning. In *Advances in Neural Information Processing Systems*, 2020.

[27] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[28] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *IEEE/CVF International Conference on Computer Vision*, 2017.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016.

[30] Olivier Henaff. Data-efficient image recognition with contrastive predictive coding. In *International Conference on Machine Learning*, 2020.

[31] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019.

[32] Quan Kong, Wenpeng Wei, Ziwei Deng, Tomoaki Yoshinaga, and Tomokazu Murakami. Cycle-contrast for self-supervised video representation learning. In *Advances in Neural Information Processing Systems*, 2020.

[33] Junnan Li, Pan Zhou, Caiming Xiong, Richard Socher, and Steven CH Hoi. Prototypical contrastive learning of unsupervised representations. *arXiv preprint arXiv:2005.04966*, 2020.

[34] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

[35] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, 2014.

[36] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.

[37] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[38] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. *arXiv preprint arXiv:2101.02702*, 2021.

[39] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

[40] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[41] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717, 2020.

[42] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*, 2010.

[43] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, 2018.

[44] Shi Pu, Yibing Song, Chao Ma, Honggang Zhang, and Ming Hsuan Yang. Deep attentive tracking via reciprocative learning. In *Advances in Neural Information Processing Systems*, 2018.

[45] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015.

[46] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and variational inference in deep latent gaussian models. In *International Conference on Machine Learning*, 2014.

[47] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 2015.

[48] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE/CVF International Conference on Computer Vision*, 2017.

[49] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

[50] Ya-Fang Shih, Yang-Ming Yeh, Yen-Yu Lin, Ming-Fang Weng, Yi-Chang Lu, and Yung-Yu Chuang. Deep co-occurrence feature learning for visual object recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017.

[51] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. *arXiv preprint arXiv:2101.11605*, 2021.

[52] Gabriel Synnaeve, Qiantong Xu, Jacob Kahn, Tatiana Likhomanenko, Edouard Grave, Vineel Pratap, Anuroop Sriram, Vitaliy Liptchinsky, and Ronan Collobert. End-to-end asr: from supervised to semi-supervised learning with modern architectures. *arXiv preprint arXiv:1911.08460*, 2019.

[53] Igor V Tetko, Pavel Karpov, Ruud Van Deursen, and Guillaume Godin. State-of-the-art augmented nlp transformer models for direct and single-step retrosynthesis. *Nature communications*, 2020.

[54] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *European Conference on Computer Vision*, 2020.

[55] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning. *arXiv preprint arXiv:2005.10243*, 2020.

[56] Yao-Hung Tsai, Martin Ma, Muqian Yang, Han Zhao, Louis-Philippe Morency, and Ruslan Salakhutdinov. Self-supervised representation learning with relative predictive coding. In *International Conference on Learning Representations*, 2021.

[57] Yao-Hung Tsai, Yue Wu, Ruslan Salakhutdinov, and Louis-Philippe Morency. Self-supervised learning from a multi-view perspective. In *International Conference on Learning Representations*, 2021.

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[59] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning*, 2008.

[60] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[61] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[62] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[63] Zhenda Xie, Yutong Lin, Zhuliang Yao, Zheng Zhang, Qi Dai, Yue Cao, and Han Hu. Self-supervised learning with swin transformers. *arXiv preprint arXiv:2105.04553*, 2021.

[64] Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, and Baining Guo. Learning texture transformer network for image super-resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[65] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, 2021.

[66] Yanhong Zeng, Jianlong Fu, and Hongyang Chao. Learning joint spatial-temporal transformations for video inpainting. In *European Conference on Computer Vision*, 2020.

[67] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1476–1485, 2019.

[68] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[69] Luowei Zhou, Yingbo Zhou, Jason J Corso, Richard Socher, and Caiming Xiong. End-to-end dense video captioning with masked transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[70] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See in Sec. 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See in Sec. 5.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See in the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See in Sec. 4.1.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We show the detection and segmentation results based on an averaged performance from 5 trials.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See in Sec. 4.1.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] See supplementary files.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]