Semantic Information in Contrastive Learning

Shengjiang Quan\textsuperscript{1,*}, Masahiro Hirano\textsuperscript{2} and Yuji Yamakawa\textsuperscript{3}
\textsuperscript{1}Graduate School of Engineering, The University of Tokyo, Japan
\textsuperscript{2}Institute of Industrial Science, The University of Tokyo, Japan
\textsuperscript{3}Interfaculty Initiative in Information Studies, The University of Tokyo, Japan
quan@g.ecc.u-tokyo.ac.jp, \{mhirano, y-ymkw\}@iis.u-tokyo.ac.jp

Abstract

This work investigates the functionality of Semantic information in Contrastive Learning (SemCL). An advanced pretext task is designed: a contrast is performed between each object and its environment, taken from a scene. This allows the SemCL pretrained model to extract objects from their environment in an image, significantly improving the spatial understanding of the pretrained models. Downstream tasks of semantic/instance segmentation, object detection and depth estimation are implemented on PASCAL VOC, Cityscapes, COCO, KITTI, etc. SemCL pretrained models substantially outperform ImageNet pretrained counterparts and are competitive with well-known works on downstream tasks. The results suggest that a dedicated pretext task leveraging semantic information can be powerful in benchmarks related to spatial understanding. The code is available at https://github.com/sjiang95/semcl.

1. Introduction

Within the field of visual representation learning, contrastive learning attracts special attention for its inspiring performance in transfer learning [57, 49, 31, 22, 8, 9]. The concept of contrastive learning can literally be explained as discovering the difference between positive and negative samples. The definition of positive-negative samples is one of the main subjects of contrastive learning, which directly determines the pretext task and the corresponding loss function.

We humans can recognize an object from its even complicated environment because we have learned and bound the typical geometric feature and the corresponding semantic concept of the object class. For example, we can recognize a cat in a scene thanks to our knowledge of its appearance (typical geometric feature) and the concept of cat (semantic information). Similarly, the target of prevalent visual tasks, such as semantic/instance segmentation, object detection, etc., can be summarized as distinguishing and localizing subjects from surroundings. Inspired by and to extend the spatial recognition mechanism to CV, we propose our method SemCL which directly teaches models to extract a subject from its environment. Utilizing publicly off-the-shelf datasets providing semantic labels, SemCL can be considered as a supervised contrastive learning approach whose samples consist of subjects and their corresponding surroundings (semantically not the subject). The pretext task of SemCL is to tell the difference between one subject and its surroundings. From a representation learning perspective, the pretext task is to maximize the distance ($L^2$ norm) between a subject and its environment in the embedding space (see Figure 1). The pretext task mimics the spatial recognition mechanism to discriminate objects and surroundings, which significantly improves the spatial information understanding of pretrained models.

In SemCL, contrast is performed pairwise between positive-negative pairs from one image/scene. Inter-scene contrast is considered inappropriate: only the contrasts

---

\*Corresponding author.
among subject-surrounding pairs from one scene are considered valid. Therefore, paired InfoNCE loss is adopted. With the mechanism of contrasting only paired samples, SemCL is characterized by decoupling the number of negative samples from the batch size. Compared to MoCo [22], whose performance is positively correlated with batch size and thus increases hardware cost (e.g. \textit{batchsize} = 4096 on 128 GPUs for ViT-B in MoCo v3 [9]) for satisfactory results, SemCL can achieve consistent yet competitive results on downstream tasks with relatively small batch size (e.g., \textit{batchsize} = 64 @ 224 × 224). We adopt the MoCo v3 [9] as the pretraining framework.

The primary motivation of representation learning is to pretrain general representations that can be fine-tuned and transferred to downstream tasks. In this work, the SemCL pretrained models are benchmarked on semantic segmentation, object detection and instance segmentation, and depth estimation tasks. SemCL models substantially outperform their ImageNet pretrained counterparts due to the improved ability of spacial information understanding. In a system-level comparison, SemCL models also show gains over previous well-known works (e.g. Deeplabv3+ [7] in semantic segmentation, Mask2Former [10] in instance segmentation and Binsformer [35] in depth estimation).

2. Related Work

2.1. Contrastive Learning

The seminal idea of contrastive learning dates back at least to the 1990s [2, 3, 31], but it has become prominent in recent years thanks to the large pretrained models in the fields of NLP and CV [31]. As the name suggests, contrastive learning mines abstract representations by comparing between “similar” and “dissimilar” samples: minimizing the representation of “similar” samples and maximizing that of “dissimilar” samples in the embedding space. Formally, the “similar” inputs are called positive samples, and the “dissimilar” inputs are called negative samples.

Defining “similar” and “dissimilar” samples is an important topic in contrastive learning. In instance discrimination [57], which is a ubiquitous approach to define positive-negative samples in CV, each sample in a mini-batch $x \in \{x_1, \ldots, x_n\}$ is considered mutually exclusive to the rest. The encoded output $\theta (\cdot)$ of each sample is pushed away from the others in the embedding space, denoted as $\theta (x_1) \iff \{\theta (x_2), \ldots, \theta (x_n)\}$ (\iff for push). Contrastive Predictive Coding (CPC) [49] defines positive-negative samples in a generative way: Given a sequential input $\{\ldots, x_t-3, x_t-2, x_t-1, x_t\}$, a context latent representation $C_t$ is computed to predict the encoded outputs of future inputs $\{\theta (X_{t+1}), \theta (X_{t+2}), \theta (X_{t+3}), \ldots\}$. Those predicted future embeddings are considered as positive samples to real future embeddings, and exclusive to some other irrelevant inputs.

2.2. Momentum Contrast

He et al. [22] proposed a momentum contrast framework, MoCo, for contrastive representation learning. MoCo summed unsupervised visual representation learning [57, 49, 26, 69, 25, 48, 1] up as dictionary look-up: the input data are represented and sampled by an encoder network $\theta_k$ as “keys” $k = \theta_k (x^k)$. The goal of contrastive learning is to train encoders $\theta_q$ by which a “query” $q = \theta_q (x^q)$ should be similar to its matching key and dissimilar to others. Since dictionary consistency is considered crucial for unsupervised learning with contrastive loss [22], MoCo introduces momentum updating. In the MoCo framework, the key coder $\theta_k$ is progressively updated by the weighted average of $\theta_k$ and the query coder $\theta_q$,

$$\theta_k \leftarrow m \theta_k + (1 - m) \theta_q,$$

where the momentum coefficient $m \in [0,1)$ is set quite high ($m = 0.999$ in [22]) for slower key encoder update. The MoCo series has proven the power of the Momentum Contrast framework on feature transferring and is therefore employed as the basis for this work.

2.3. Labeled Data In Unsupervised Framework

Large networks are quite thirsty for training data, while labelling the large amount of collected data is a human resource consuming challenge. Apart from turning to unsupervised learning, which inherently avoids using labelled data, using unlabeled data in supervised learning is also a solution. Pseudo-Label [44, 38, 56] is a trick that uses (usually supervised) pretrained networks to generate features that can play the role of labels. And the generated pseudo-labels participate in the iterations of the target network just like annotations.

In contrast, Khosla et al. [28] also investigated the function of labelled data in unsupervised learning. As a contrastive learning research, Khosla et al. [28] intuitively uses the classification label to define samples within the same class are mutually positive, and those from other categories are negative. To exploit the label information, clusters of points belonging to the same class are pulled together in the embedding space, while clusters of samples from different classes are pushed apart [28]: $\{\theta (x^0_1) \iff \theta (x^0_2) \iff \theta (x^0_3) \iff \theta (x^0_n)\} \iff \{\theta (x^1_1) \iff \theta (x^1_2) \iff \theta (x^1_3) \iff \theta (x^1_n)\}$ (\iff for pull). The experimental result shows that supervised contrastive learning can improve both the accuracy and robustness of classifiers – [28] achieves a top-1 accuracy of 81.4% on the ImageNet (IN) dataset, which is 0.8% above the best result reported by ResNet-200 architecture [34]. Likewise, our work investigates the effectiveness of semantic information in unsupervised framework contrastive learning.
2.4. Semantic Information in Contrastive Learning

Other than [28], there are works that use small amounts of labelled data in unsupervised representation learning. Wang et al. [54] proposed a pixel-wise contrastive algorithm for semantic segmentation: for a pixel $i$, it is pulled towards pixels belonging to the same category and pushed away from pixels in other categories in the embedding space. This contrast is performed across images from the same dataset. In the Cityscapes test benchmark, [54] R101 achieves 79.2 mIoU, which is 1.1 points better than DeepLabv3 [6].

The attempt and results of works using labels in unsupervised representation learning, such as [28, 54], indicate that label information can bring gains to a fully unsupervised representation. It is promising to explore advanced pretext tasks to improve the performance of pretrained models in downstream tasks.

3. Method

3.1. SemCL Dataset

The core of this work is to define semantically contrastive pairs by utilizing off-the-shelf semantic labels, which are mappings between categories of objects and their coordinates on the raw image at the pixel level. As decided by the authors of a dataset, semantic labels can be color or grayscale images. Each predefined color or grayscale denotes a category, and the distribution of label pixels reflects the location of pixels belonging to a particular category in the raw image.

The SemCL dataset is generated by the mechanism shown in Figure 2, where the raw image and its semantic label are selected from the PASCAL VOC2012 dataset [14]. From the semantic label annotating two objects - a motorcycle and a rider - two binary masks BM0 and BM1 can be extracted to separately represent two annotated objects. Their inverted counterparts ¬BM0 and ¬BM1 represent pixels other than the target object. These binary masks are then bitwise applied to the raw image to separate an object (Anchor) from its environment (¬Anchor), forming a semantically contrastive pair. Each element in a raw Anchor-¬Anchor pair is considered to be semantically opposite to the other.

In practice, the SemCL-1M dataset is produced from training sets of augmented PASCAL VOC2012 [14] (also known as SBD [20]), Cityscapes [11], ADE20K [67, 68] and COCO [36] (stuff+thing 2017 [4]) datasets. The components of the SemCL dataset are illustrated in Table 1. When generating the dataset, a threshold $t = 0.01$ is set to filter out too small (less than $\sqrt{t \times \text{height}} \times \sqrt{t \times \text{width}}$) objects that are considered semantically unsaturated.

### Table 1: Components of the SemCL dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># training samples</th>
<th># SemCL pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>VOC2012</td>
<td>10,582</td>
<td>14,203</td>
</tr>
<tr>
<td>Cityscapes</td>
<td>2,975</td>
<td>20,280</td>
</tr>
<tr>
<td>ADE20K</td>
<td>25,574</td>
<td>270,218</td>
</tr>
<tr>
<td>COCO</td>
<td>118,287</td>
<td>712,212</td>
</tr>
<tr>
<td>Total</td>
<td>157,418</td>
<td>1,016,913</td>
</tr>
</tbody>
</table>

For one anchor as query $q_0$ ($t = 0$), the paired InfoNCE loss tries to classify it as $k_j^+$, $j \in [1, n]$. The next step is to go forward: classify the first positive $q_1$ ($t = 1$) of the anchor as $k_j^-$, $j \in [2, n]$, and so on, as shown in the algorithm 1. Appendix A.1. To contrast with the paired InfoNCE, we refer to the original InfoNCE loss as unpaired InfoNCE hereafter. The behavior difference between unpaired and paired InfoNCE losses is compared in Figure 3. The paired InfoNCE only maximizes the distance between anchors and ¬anchors in the embedding space, neglecting irrelevant samples. The implementation of the paired InfoNCE loss is introduced in detail in Appendix A.2.

3.3. Pretext Task

Since the aim of this work is to enable networks to strip targets (anchors) from their surroundings (¬anchors), we use a restricted instance discrimination task [57]: a query and a key are positive pairs if they are different views (by key $k_i$ of the dictionary [22]). The value of the objective function should be low if $q$ is similar to its positive key $k_+$ and dissimilar to all negative keys $\{k_0, k_1, k_2, \ldots\} \setminus \{k_+\}$. In MoCo [22], the form of a contrastive loss function InfoNCE [49] is adopted

$$L_q = -\log \frac{\exp(q \cdot k_+ / \tau)}{\sum_{i=0}^{K} \exp(q \cdot k_i / \tau)} \tag{2}$$

where $\tau$ is a scalar temperature hyperparameter controlling the strength of penalties on hard negative samples [50], e.g. in the instance discrimination task [49, 22], there may be potential positive samples in the randomly chosen negative samples. Summing over one positive and $K$ negative samples, Equation (2) is the log loss of a (K+1)-way softmax-based classifier trying to classify $q$ as $k_+$ [22]. While in our case an anchor instance $q$ compares only with its own $n$ augmentations (positive) $K^+ := \{k_1^+, k_2^+, k_3^+, \ldots, k_n^+\}$ and paired $n + 1$ negatives $K^- := \{k_0^-, k_1^-, k_2^-, k_3^-, \ldots, k_n^-\}$ using paired InfoNCE

$$L_q = -\sum_{j=1}^{n} \log \frac{\exp(q \cdot k_j^+ / \tau)}{\sum_{i=0}^{K} \exp(q \cdot k_i^- / \tau) + \exp(q \cdot k_j^+ / \tau)} \tag{3}$$
Figure 2: SemCL dataset. The semantic label of a sample is utilized to generate binary masks. Each binary mask depicts the pixel-level information of one object (anchor) in the raw image. Those binary masks, together with their inverts, are applied on the raw image to extract objects (anchors) and corresponding surroundings (~anchors).

Figure 3: Comparison between unpaired (left) and paired InfoNCE losses (right). Left: a sample (anchor) is positive only to its own augmentations, while negative to others in the same mini-batch. Right: A sample (anchor) is positive only to its own augmentations, while negative only to its paired counterparts.

data augmentation) of the same anchor, and are negative pairs if they originate from an anchor and its corresponding surroundings (~anchors). A PyTorch-style pseudocode is given in the Appendix A.2. The inputs are encoded by base encoder \( f_b \) and momentum encoder \( f_m \), producing queries and keys. Theoretically, the backbone of \( f_b \) and \( f_m \) can be any architecture. In practice, we use ResNet [23] and Swin Transformer [37] because they are the most representative architectures in recent years. All keys are stacked along dimension 1 for paired InfoNCE loss.

4. Experiments

Considering the enormous scale difference between ImageNet and SemCL dataset, all SemCL models are initialized by corresponding IN pretrained weights, which are compared as counterparts in downstream tasks. The total number of pretraining iterations is set to 30k to minimize the significant
variation in training time caused by scale differences among SemCL sub-datasets. We adopt AdamW [40] with $wd = 0.1$, and the OneCycle [46] learning rate schedule consisting of warmup [17] for the initial 3,750 (warmup ratio 0.125) steps and cosine annealing [9, 39]. Following [9], the base $lr$ is set to $1.5 \times 10^{-4}$ and scaled linearly by $lr \times \text{batchsize}/256$ and temperature $T = 0.2$. The initial MoCo momentum is 0.99 and is gradually increased to 1 with a half cycle cosine schedule. The data augmentation strategy includes $224 \times 224$ random crops after random rescaling the original image with an area ratio in the range $[0.08, 1.0]$, followed by random color jitter, random grayscale, random Gaussian blur, and random horizontal flip. All batch norm (BN) layers are synchronized across GPUs (SyncBN [41]). Limited by the two dual-GPU nodes (RTX3090×2+Titan RTX×2) we use, a native batch size of 64 can be applied to all ResNets and up to Swin-B for Swin Transformers with crop size $224 \times 224$. For Swin-L in particular, we compromise by using a batch size of 32 with gradient accumulation over 2 iterations to achieve an equivalent batch size of 64.

SemCL aims to improve the understanding of spatial information of models for which downstream tasks of semantic segmentation, object detection and depth estimation are conducted. Unless otherwise stated, all SemCL backbones are pretrained on the corresponding SemCL sub-datasets. The implementation details of downstream tasks are listed Appendix A.3.

### 4.1. Semantic Segmentation

**Backbone comparison.** In the semantic segmentation benchmark on validation sets of VOC2012, Cityscapes, ADE20K and COCO 2017, SemCL models pretrained on the corresponding SemCL sub dataset are compared with ImageNet pretrained counterparts in Table 2. For VOC2012 and COCO, SemCL outperforms its IN counterparts by a maximum of 0.83 and 0.66 points respectively. SemCL can comprehensively outperform its IN counterparts on all backbones in the Cityscapes and ADE20k benchmarks.

**System-level comparison.** SemCL pretrained backbones are compared with other methods on semantic segmentation tasks in Table 3. On VOC2012, SemCL R50/R101 outperform Deeplabv3+ [7] by 2.37/3.12 points. The Deeplabv3+ X-71 is 0.55 points behind SemCL R101. For CP² [51] ViT-S/16, SemCL Swin-S significantly outperforms it on all benchmarks by 4.97, 4.53 and 6.71 points. Compared to MAE [21] ViT-B, SemCL Swin-B achieves a gain of 0.58 points. A qualitative comparison between MAE [21] ViT-B and SemCL Swin-B is shown in Figure 4.

### 4.2. Object Detection and Instance Segmentation

**Backbone comparison.** In the benchmark for object detection and instance segmentation on the test set of VOC07 and validation sets of Cityscapes and COCO 2017, SemCL models pretrained on the corresponding SemCL sub-datasets are compared with benchmark on validation sets of VOC2012, Cityscapes, and COCO 2017, SemCL detection and instance segmentation on the test set of VOC07 [21] ViT-B and SemCL Swin-B is shown in Figure 4.

For CP² [7] X-71 is

<table>
<thead>
<tr>
<th>Method</th>
<th>VOC2007 AP50</th>
<th>Cityscapes1 APbox</th>
<th>APmask</th>
<th>COCO APbox</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoCo v3[9] IN-1k 81.88</td>
<td>42.7</td>
<td>37.5</td>
<td>42.6</td>
<td>37.9</td>
</tr>
<tr>
<td>SemCL</td>
<td>81.57 (−0.31) 81.99 (0.2) 42.8 (0.5) 42.8 (0.2) 38.0 (0.1)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a) R50

<table>
<thead>
<tr>
<th>Method</th>
<th>VOC2007 AP50</th>
<th>Cityscapes1 APbox</th>
<th>APmask</th>
<th>COCO APbox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swin[37] IN-22k 86.58</td>
<td>45.9</td>
<td>39.8</td>
<td>49.2</td>
<td>43.0</td>
</tr>
<tr>
<td>SemCL</td>
<td>86.53 (−0.05) 46.0 (0.1) 40.2 (0.2) 49.0 (−0.2) 43.3 (0.3)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Swin-T

Table 4: Results of object detection and instance segmentation on various backbones w/ Cascade Mask R-CNN. All results are from our implementation. †: Sync BN is disabled due to gradient accumulation.

**System-level comparison.** Table 5 compares the object detection and instance segmentation results of SemCL backbones with other models. On Cityscapes and COCO, SemCL backbones comprehensively outperform their twin-born MoCo Specifically, SemCL ResNet50 backbones achieve +1.7 over MoCo IG-1B [22] in COCO detection and...
<table>
<thead>
<tr>
<th>Backbone</th>
<th>Method</th>
<th>VOC2012</th>
<th>Cityscapes</th>
<th>ADE20K</th>
<th>COCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>R50</td>
<td>MoCo v3[9] IN-1k†</td>
<td>79.65</td>
<td>79.01</td>
<td>42.71</td>
<td>40.59</td>
</tr>
<tr>
<td></td>
<td>SemCL</td>
<td>79.57(−0.08)</td>
<td>79.13(+0.12)</td>
<td>42.96(+0.25)</td>
<td>40.80(+0.21)</td>
</tr>
<tr>
<td>R101</td>
<td>timm[55] IN-1k</td>
<td>81.82</td>
<td>79.87</td>
<td>44.43</td>
<td>41.43</td>
</tr>
<tr>
<td></td>
<td>SemCL</td>
<td>81.97(+0.15)</td>
<td>80.10(+0.23)</td>
<td>44.66(+0.23)</td>
<td>41.58(+0.15)</td>
</tr>
<tr>
<td>Swin-T</td>
<td>Swin[37] IN-22k</td>
<td>80.90</td>
<td>78.67</td>
<td>44.60</td>
<td>42.39</td>
</tr>
<tr>
<td></td>
<td>SemCL</td>
<td>81.73(+0.83)</td>
<td>78.91(+0.26)</td>
<td>44.72(+0.23)</td>
<td>42.53(+0.14)</td>
</tr>
<tr>
<td>Swin-S</td>
<td>Swin[37] IN-22k</td>
<td>83.71</td>
<td>81.25</td>
<td>47.62</td>
<td>44.75</td>
</tr>
<tr>
<td></td>
<td>SemCL</td>
<td>84.47(+0.83)</td>
<td>81.53(+0.26)</td>
<td>47.91(+0.29)</td>
<td>44.57(−0.18)</td>
</tr>
<tr>
<td>Swin-B</td>
<td>Swin[37] IN-22k</td>
<td>84.64</td>
<td>81.61</td>
<td>48.21</td>
<td>44.81</td>
</tr>
<tr>
<td></td>
<td>SemCL</td>
<td>84.67(+0.03)</td>
<td>81.94(+0.33)</td>
<td>48.68(+0.47)</td>
<td>45.47(+0.66)</td>
</tr>
</tbody>
</table>

† Although MoCo v3 [9] is originally based on ViT, R50 pretrained models are provided in their repo [https://github.com/facebookresearch/moco-v3].

Table 2: SemCL vs. ImageNet pretrained backbones on VOC2012, Cityscapes, ADE20K and COCO semantic segmentation (mIoU). All results are from our implementation.

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>VOC2012</th>
<th>Cityscapes</th>
<th>ADE20K</th>
<th>COCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoCo IN-1M[22]</td>
<td>R50</td>
<td>72.5</td>
<td>74.4</td>
<td>75.5</td>
<td>74.6</td>
</tr>
<tr>
<td>Supervised-IN[22]</td>
<td>R50</td>
<td>74.4</td>
<td>75.5</td>
<td>76.3</td>
<td>76.5</td>
</tr>
<tr>
<td>MoCo IG-1B[22]</td>
<td>R50</td>
<td>75.5</td>
<td>76.3</td>
<td>77.8</td>
<td>78.0</td>
</tr>
<tr>
<td>BYOL[18]</td>
<td>R50</td>
<td>76.3</td>
<td>77.8</td>
<td>79.1</td>
<td>79.3</td>
</tr>
<tr>
<td>SemCL</td>
<td>R50</td>
<td>79.57</td>
<td>79.13</td>
<td>80.10</td>
<td>80.4</td>
</tr>
<tr>
<td>Deeplabv3+[7]</td>
<td>R101</td>
<td>78.85</td>
<td>77.5</td>
<td>79.5</td>
<td>79.5</td>
</tr>
<tr>
<td>Re2Net[5]</td>
<td>R101</td>
<td>80.2</td>
<td>79.5</td>
<td>80.2</td>
<td>80.2</td>
</tr>
<tr>
<td>DFN[61]</td>
<td>R101</td>
<td>80.60</td>
<td>79.5</td>
<td>80.60</td>
<td>80.60</td>
</tr>
<tr>
<td>SemCL</td>
<td>R101</td>
<td>81.97</td>
<td>80.10</td>
<td>81.97</td>
<td>81.97</td>
</tr>
<tr>
<td>CP*[51]</td>
<td>ViT-S/16</td>
<td>79.5</td>
<td>80.10</td>
<td>80.10</td>
<td>80.10</td>
</tr>
<tr>
<td>SemCL</td>
<td>Swin-S</td>
<td>84.47</td>
<td>81.97</td>
<td>81.97</td>
<td>81.97</td>
</tr>
<tr>
<td>Leopart[70]</td>
<td>ViT-B/8</td>
<td>76.3</td>
<td>79.5</td>
<td>80.2</td>
<td>80.2</td>
</tr>
<tr>
<td>SemCL</td>
<td>Swin-B</td>
<td>84.67</td>
<td>84.47</td>
<td>84.47</td>
<td>84.47</td>
</tr>
</tbody>
</table>

Table 3: Comparison with previous best results on validation sets of VOC2012, Cityscapes, and ADE20K semantic segmentation (mIoU). Best results are bolded.

+5.1/+0.6 in Cityscapes/COCO instance segmentation tasks. For SemCL Swin-T backbones, SemCL achieves +1.2/+0.1 over Focal [60] on COCO detection/instance segmentation. Figure 5 demonstrates a qualitative comparison between Mask2Former [10] and SemCL Swin-T on Cityscapes validation set. With the improved spatial information understanding, SemCL model clearly discriminates (even overlapped) instances in Figures 5a and 5b and successfully classify part of an instance to a defined category as in Figure 5c.

4.3. Depth Estimation

**Backbone comparison.** SemCL pretrained backbones are compared with corresponding IN pretrained ones in Table 6. For cityscapes, SemCL comprehensively outperforms its IN pretrained counterparts. In particular, the ResNet50 backbones improve the RMSE by 0.138 points. For KITTI and NTUv2, all SemCL backbones show improvements over the IN pretrained ones. The SemCL pretrained Swin-L model achieves improvements of 0.001/0.002 points on KITTI AbsRel/RMSE, and the SemCL pretrained ResNet50 model achieves improvements of 0.003/0.005 points on NYUv2.

**System-level comparison.** We report the comparison of SemCL pretrained backbones with the previous state-of-the-art method Binsformer[35] and other works in Table 7. As for the Cityscapes results shown in Table 7a, SemCL pretrained ResNet50 significantly outperforms SDC [53] by −0.089 and −2.218 points on AbsRel and RMSE respectively. And SD-SSMDE [43] lags behind SemCL by 0.005/3.742 points. Meanwhile, the test on KITTI (Eigen split [13]) is shown in Table 7b, SemCL pretrained ResNet50 and Swin-T comprehensively outperform the previous SOTA.
Table 5: System-level comparison on object detection and instance segmentation. The best results are bolded. * claims their backbones are CNN-based, but are considered and compared as transformer-based networks, since transformer blocks are massively used.

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>AP&lt;sub&gt;50&lt;/sub&gt;</th>
<th>Method</th>
<th>Backbone</th>
<th>AP&lt;sup&gt;mask&lt;/sup&gt;</th>
<th>Method</th>
<th>Backbone</th>
<th>AP&lt;sup&gt;mask&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised-IN[22]</td>
<td>R50</td>
<td>81.3</td>
<td>MoCo IN-1M[22]</td>
<td>R50</td>
<td>32.3</td>
<td>Supervised-IN[22]</td>
<td>R50</td>
<td>40.6</td>
</tr>
<tr>
<td>MoCo IN-1M[22]</td>
<td>R50</td>
<td>81.5</td>
<td>MoCo IG-1B[22]</td>
<td>R50</td>
<td>32.9</td>
<td>MoCo IN-1M[22]</td>
<td>R50</td>
<td>40.8</td>
</tr>
<tr>
<td>MoCo v2 800ep[8]</td>
<td>R50</td>
<td>82.5</td>
<td>Supervised-IN[22]</td>
<td>R50</td>
<td>32.9</td>
<td>MoCo IG-1B[22]</td>
<td>R50</td>
<td>41.1</td>
</tr>
<tr>
<td>SemCL</td>
<td>R50</td>
<td>81.57</td>
<td>Mask2Former[10]</td>
<td>R50</td>
<td>37.4</td>
<td>SemCL</td>
<td>R50</td>
<td>42.8</td>
</tr>
<tr>
<td>UP-DETR/300[12]</td>
<td></td>
<td>80.1</td>
<td>BoundaryFormer[30]</td>
<td>*</td>
<td>38.3</td>
<td>MST[34]</td>
<td>Swin-T</td>
<td>42.7</td>
</tr>
<tr>
<td>SemCL</td>
<td>Swin-T</td>
<td>86.55</td>
<td>SemCL</td>
<td>Swin-T</td>
<td>40.2</td>
<td>Focal[60]</td>
<td>Focal-Base</td>
<td>47.8</td>
</tr>
</tbody>
</table>

(a) Object det. on VOC2007 test.  
(b) Instance seg. on Cityscapes val.  
(c) Object det. and instance seg. on COCO val2017.

Table 6: SemCL vs. ImageNet pretraining on Cityscapes, KITTI and NYUv2 depth estimation (lower is better). All results are from our implementation.

<table>
<thead>
<tr>
<th>Backbone</th>
<th>Method</th>
<th>Cityscapes</th>
<th>KITTI</th>
<th>NYUv2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AbsRel</td>
<td>RMSE</td>
<td>AbsRel</td>
<td>RMSE</td>
</tr>
<tr>
<td>R50</td>
<td>MoCo v3[9] IN-1k</td>
<td>0.138</td>
<td>4.837</td>
<td>0.059</td>
</tr>
<tr>
<td></td>
<td>SemCL</td>
<td>0.138</td>
<td>4.699(-0.138)</td>
<td>0.058(-0.001)</td>
</tr>
<tr>
<td>Swin-T</td>
<td>Swin[37] IN-22k</td>
<td>0.134</td>
<td>4.643</td>
<td>0.057</td>
</tr>
<tr>
<td></td>
<td>SemCL</td>
<td>0.133(-0.0001)</td>
<td>4.577(-0.066)</td>
<td>0.057</td>
</tr>
<tr>
<td>Swin-L</td>
<td>Swin[37] IN-22k</td>
<td>0.130</td>
<td>4.536</td>
<td>0.053</td>
</tr>
<tr>
<td></td>
<td>SemCL</td>
<td>0.128(-0.002)</td>
<td>4.426(-0.110)</td>
<td>0.052(-0.001)</td>
</tr>
</tbody>
</table>

(a) Object det. on VOC2007 test.  
(b) Instance seg. on Cityscapes val.  
(c) Object det. and instance seg. on COCO val2017.

Binsformer. In particular, SemCL ResNet50 achieves a gain of $-0.003/-0.089$ over Binsformer, and SemCL Swin-L is on par with Binsformer Swin-L. On NYUv2 (see Table 7c), the gains of SemCL pretrained ResNet50 are also high with $-0.010$ AbsRel and $-0.173$ RMSE. The SemCL

Figure 5: Qualitative comparison of instance segmentation between Mask2Former [10] (left) and SemCL (right) Swin-T on Cityscapes validation set.

Figure 6: Qualitative comparison of depth estimation between Binsformer[35] and SemCL Swin-T on the NYUv2 test set.

Swin-L RMSE is 0.001 points lower than the SOTA Binsformer. Qualitative comparison between Binsformer[35] and SemCL Swin-T is given in Figure 6.
4.4. Attention Map Visualization

To improve the qualitative understanding of the SemCL representation, we visualize attention maps of Swin-T pretrained on SemCL-VOC and ImageNet using PyTorch library for CAM methods [16] with the Score-CAM method [52]. These samples in Figure 7 selected from VOC2012 val and have not been seen by the pretrained models are characterized by simple centered subjects (Figure 7a), complex scene with occlusion (Figure 7b) and multiple subjects (Figures 7c and 7d).

In Figure 7a, the IN pretrained model failed to focus on the subject, while the SemCL model accurately depicted the area of the main subject. Meanwhile, SemCL performs better in complex scenes containing foreground-background relationships, as shown in Figure 7b. The IN pretrained model pays no attention to an occluded object, but the SemCL model recognizes that the occluded part also belongs to the horse, as shown in Figure 7b. In Figure 7c, the IN model directly ignores the distant person, but the SemCL model not only annotates two people but also focuses more on the nearby one, suggesting that the SemCL model can understand the spatial distribution of subjects in an image. Also, in Figure 7d, the SemCL model successfully recognizes one plant and its pot as one object.

Summary. SemCL pretrained backbones can substantially outperform ImageNet pretrained counterparts on semantic segmentation, object detection, instance segmentation and depth estimation tasks. Using semantically contrastive pairs generated from off-the-shelf datasets, SemCL significantly improves the spatial information understanding of pretrained models. Such an improvement is also promising to benefit more challenging tasks including generative ones [32]. Ablation study concerning paired/unpaired InfoNCE loss, pretraining batch size, training length and dataset scale are given in Appendix A.4.

5. Conclusion

We investigate the effectiveness of semantic information in the contrastive learning pretext task on the spatial understanding ability of models. Tests are conducted on semantic/instance segmentation, object detection and depth estimation tasks with both ResNets and Swin Transformers, and SemCL pretrained backbones substantially outperform their ImageNet pretrained counterparts. By supporting small batch sizes and fast pretraining, SemCL is a lightweight yet effective approach. In contrastive representation learning, if dedicated pretext task is designed properly, four ounces can move a thousand pounds. We hope that SemCL will inspire more advanced pretext tasks for contrastive learning.
References


[21] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 16000–16009, 2022. 5, 6


