TECHNICAL UNIVERSITY OF MUNICH

Guided Research

Learning Scene Representation with Knowledge Distillation from Sequential Data

Authors Theodor Stoican
Supervisor M.Sc. Shun-Cheng Wu

Nov 2022

Abstract

In this work, we question if the newest self-supervised approaches are able to learn a semantic representation of the images on which they are trained. In particular, we focus on datasets which contain scenes with multiple objects. The model is expected to learn a meaningful representation of the object (including contours, shapes, etc.), in a way which resembles a semantic segmentation of the scene. It has been proven that such a representation occurs as a side-effect in visual transformers ([1]), by examining the attention map of such a model. Furthermore, the class of models that we explore perform **knowledge distillation** of some kind, by training a teacher and a student in tandem. We take a well-known architecture that makes use of visual transformers and we explore it in order to identify various ways in which the models can learn the representation of an arbitrary scene.

Contents

1 Introduction

1.1 Self-supervised Learning

Representation learning has been a hugely recurrent theme in virtually the entire area of deep learning. From word2vec ([9]) in natural language processing and convolutional neural networks ([8]) in computer vision to transformers in both areas, representation learning has yielded a tremendous advance in the field, facilitating impressive perfomance on various downstream tasks. Furthermore, efficient representation learning has been more recently obtained via self-supervised learning ([2]), an approach which leverages huge amounts of non-annotated data in order to learn a fundamental structure of the data and represent it properly (in NLP, this has been done by learning to model the language, by predicting missing words and consecutive sentences). With the rise of transformers in NLP, Computer Vision has also tried to borrow from the newly developed paradigm, by incorporating vision transformers ([3]) in various tasks, as well as, in a more general way, self-supervised approaches. Various architectures and training objectives have been designed lately in order to tackle the general task of self-supervised learning.

1.2 Contrastive Learning

One line of research of performing self-supervised learning in computer vision is centered on contrastive learning ([7]). Contrastive learning, in its most intuitive form, refers to teaching a model to recognize similar and different features in an image.

On an implementation level, an image is augmented in multiple ways and the network must identify than an augmented version of an image is, in fact, very similar to the image. In this way, one obtains **positive** samples. Similarly, one can obtain **negative** samples by creating pairs between an image and all the other images from the batch (Figure 1).

Figure 1: Positive and negative samples for contrastive learning. [1]

Both positive and negative samples are then fed into the model that is trained by maximizing similarity for positive samples and maximizing dissimilarity for negative samples. The model is typically split into two (i.e. the network is duplicated, into one network the sample is fed and into the other one the augmented sample is fed). For positive samples,

[1] ©https://towardsdatascience.com/understanding-contrastive-learning-d5b19fd96607

the similarity between the outputs of the 2 models must be high.

Typically, each of the two networks are based on the same backbone:

- an encoder (e.g. CNN, transformer, etc.)
- a projector, that takes the encoding and projects it in a lower-dimensional space

The necessity of the projector has been proven empirically to improve performance, by forcing the model to learn the high-level features of the image ([2]). An overview of an entire such architecture can be seen in Figure 2.

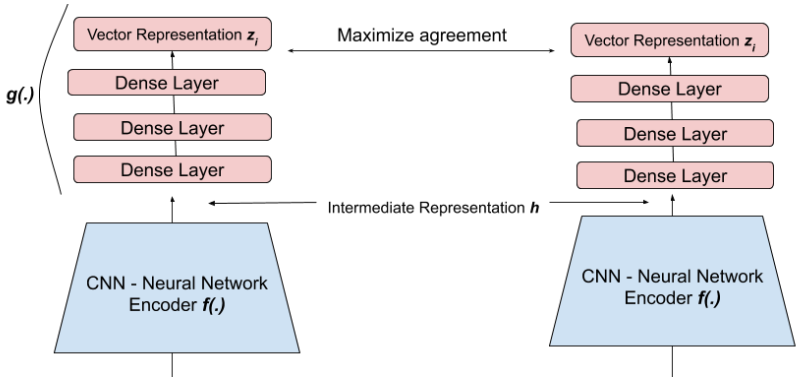

Figure 2: An example of contrastive learning architecture, using a CNN encoder as its base model. [2]

1.3 Problem Definition and Goals

In this work, we attempt to leverage models based on contrastive learning and self-supervised learning in order to obtain meaningful representations of scenes. Various ways will be explored, by which the representation can be extracted from the model. Furthermore, the models that will be explored perform some form of knowledge distillation ([4]), due to the existence of two models (an **online** network and a **target** network) that learn from one another. However, in a typical setup, knowledge distillation refers to having already an available teacher from which the student can learn, whereas in our setup, the student learns simultaneously with the teacher. This is more of a codistillation setup, although, unlike common models, our models are updated differently (the online network is updated using gradient descent-based optimizers, whereas the target network is updated using an EMA optimizer).

Last but not least, an evaluation of the models will be performed using multiple datasets. In particular, the IoU (intersection over union) metric is used in order to evaluate the quality of the feature maps (that contain the objects in the image) produced.

[2] ©https://towardsdatascience.com/understanding-contrastive-learning-d5b19fd96607

2 Prior Work

In the current section, a collection of models that have been significant in the recent past and that are very representative of the contrastive learning approach is presented.

2.1 BYOL

One of the models which have yielded the best performance in recent years and is paramount 9 the contrastive learning approach is BYOL (Bootstrap Your Own Latent [5]). BYOL uses two backbone networks, an online and a target one, feeds 2 augmented versions of the input into them, and forces the representation produced by both to match. An overview of the architecture of the model can be seen in Figure 3.

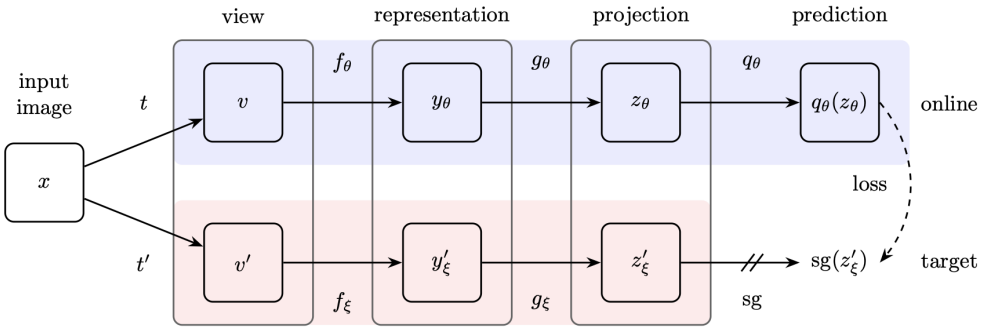

Figure 3: The architecture of BYOL[3]

.

The model works only with positive samples and, unlike a standard contrastive learning architecture, it also applies a predictor on top of the output yielded by the online network. Furthermore, it uses gradient descent to update only the online network, while the target network is updated using a slow-moving average of the online network. This is hypothesized to help avoid collapsed solutions.

2.2 DINO

Caron et al. (2021) have introduced in [1] a method of self-**di**stillation with **no** labels (DINO), which makes use of visual transformers as a backbone and of a similar (to BYOL) architecture based on contrastive learning. The main insight that the authors have discovered is that ViTs (trained in self-supervised fashion) contain features that yield explicit information about the semantic segmentation of an image (Figure 4). Furthermore, these feature yield excellent results on downstream tasks as well.

The architecture that DINO uses and the features maps extracted seem to work on arbitrarily complex scenes (with one or multiple objects).

[3]©Figure extracted from [5].

Figure 4: The self-attention heads corresponding to the [CLS] token from the last ViT layer.[4]

.

2.3 ODIN

Recent research has shown the advantage of manually including information about the structure of images in a self-supervised method. In [6], Hénaff et al. (2022) take this idea to the next level, by forcefully designing the model to learn such information in fully self-supervised fashion. ODIN couples **o**bject **di**scovery and representation **n**etworks and design a new training strategy that facilitates the learning of an arbitrary image structure.

ODIN uses 3 networks (each with the same backbone). One of them, the object discovery network, identifies the objects within an image, by generating relevant features on which K-Means is applied to generate segmentation masks. The other two (the online and the target networks) work similarly to BYOL, in the sense that their output features should be similar for different augmented versions of the same image. The online network is updated via gradient descent, whereas the other two networks are updated via an exponential moving average mechanism. An overview of the entire architecture can be seen in Figure 5.

2.4 How Do Our Approaches Relate to the Prior Work?

As part of our approaches to learn a good scene representation, we attempt to use/implement the models that we identified in literature and presented above in order to evaluate their behaviour on this task. Wherever it is appropriate, we attempt to use the pre-trained versions of the models, although, in multiple cases, we end up fine-tuning the models on custom datasets. Additionally, we take the freedom to adjust the models as we see fit in

[4]©Figure extracted from [1].

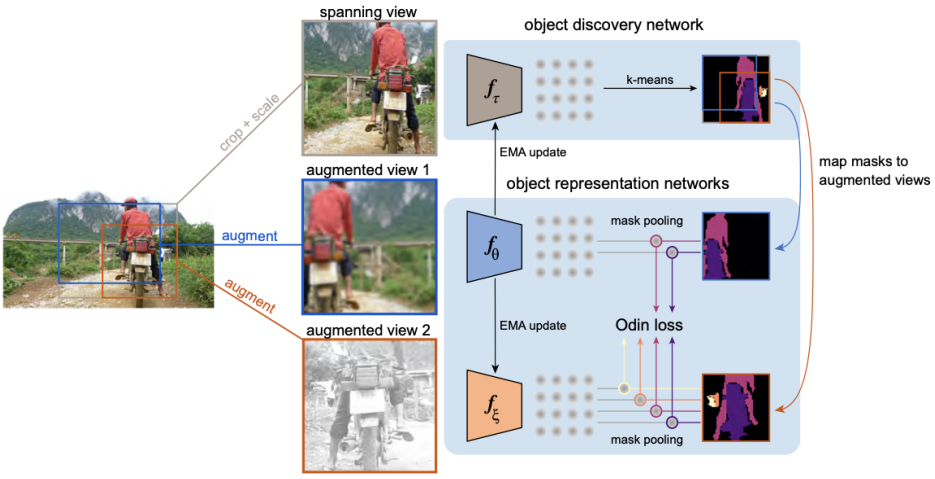

Figure 5: An overview of the architecture of ODIN[5]

.

order to improve their performance on our specific task.

In particular, we start off with a less challenging dataset, that contains scenes with singular objects. Then, we move on to a more complex dataset, which contains multiple objects in a singular scene. The datasets that we choose for our task are made of multiple consecutive frames, which are part of the same video.

3 Scenes With a Single Object

The starting point of the experiments has been a phenomenon that occurs when one trains DINO with visual transformers. It turns out that visual transformers, trained in a self-supervised manner (e.g. as part of a DINO-like pipeline), learn a semantic segmentation of the image. This is visible by examining the self-attention map of the last layer corresponding to the CLS token. Each of the 12 attention heads are designed to focus on a certain part of an image, so, in our experiments, in order to get a full representation of an image, we average them in order to get a high level representation of an image.

3.1 Dataset

We have made use of a dataset that contains consecutive frames embedded in a video. Furthermore, each frame contains a single object in the scene. The CO3D dataset([11]) issued by Meta has proved to fulfil all of our expectations, containing 50 object categories and 1.5 million frames from nearly 19000 videos. In practice, however, we have selected manually around 1000 samples from a large variety of objects and videos in order to construct our fine-tuning dataset and we have never really trained our models on the full CO3D dataset.

[5]©Figure extracted from [6].

3.2 Visualizing ViT's Self-attention Map

In order to understand what kind of information the ViT learns, we explored the self-attention map of a pretrained ViT on ImageNet on some samples from CO3D. More specifically, the ViT receives an additional input corresponding to the [CLS] token in addition to the usual input, that is made up of an aggregation of all the input tokens. The output of this token is meant to be typically used for classification (hence, [CLS]). We are interested in exploring the last self-attention layer of the ViT, and, in particular, the 12 attention heads that focus on this part of the input. We investigate only 4 of the attention heads, for now, in order to have an idea about where the focus is on the image.

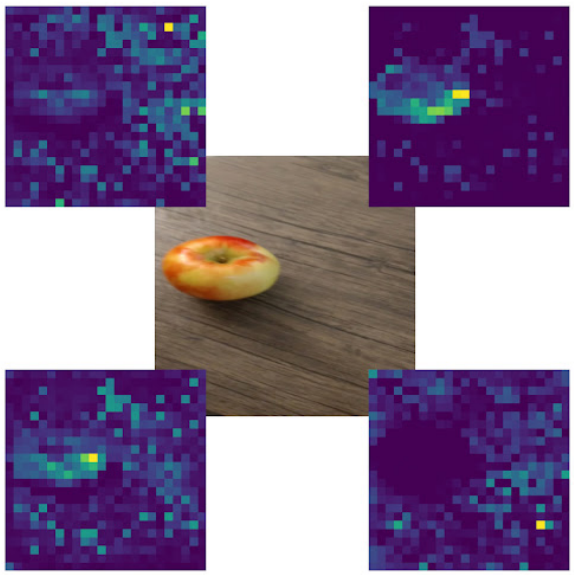

Figure 6: 4 attention heads of the last self-attention layer from a ViT based on a sample from CO3D.

.

In Figure 6, one can see that in two of the heads the attention is vaguely on the apple, due to prominent existence of noise, whereas in the other 2 the background is slightly resembled.

3.3 DINO

Given that even a simple pretrained ViT can retain information about various objects within a scene on a novel dataset, we naturally ask what a more powerful model can do. Therefore, we have switched to a pretrained version of DINO (also on ImageNet, with a resolution of 224 x 224 and a patch size of 16 x 16), publicly available using HuggingFace, and applied it on a couple of samples from CO3D. DINO uses 2 networks (ViTs) in parallel, but for inference, we use only one of the two networks (the online network) and extract the 4 attention heads in the same way as in the previous section.

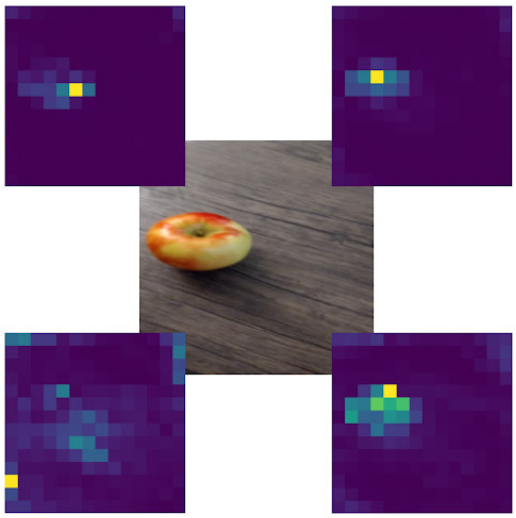

Figure 7: 4 attention heads of the last self-attention layer from DINO based on a sample from CO3D.

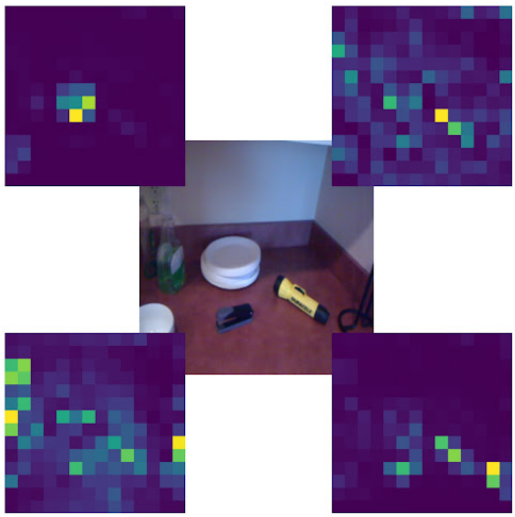

Figure 8: 4 attention heads of the last self-attention layer from DINO based on a sample from a multi-object dataset.

The effect is that DINO reduces the noise produced by a vanilla ViT (Figure 7). However, when applied in a multi-object setting, even DINO struggles (Figure 8). The model can barely identify the objects.

Therefore, this has served as a guidance to what we would like to do next. One of the ideas to improve performance has been to force the attention heads of the transformer to focus on each object on the scene. Of course, although the number of objects within the scene is variable and this mechanism does not generalize to a variable number of objects, we just wanted to show the functionality of the principle. In practice, some form of supervised learning has been applied on top of the attention heads, in order to force them to learn the representation of each individual object. In the particular case of CO3D, the segmentation masks (2 masks per frame, since there are 2 objects in each frame) of each frame have been used as ground truth for the task. The loss function for one sample has been:

$$L = \frac{1}{N} \cdot ||mask \cdot first\_head - (1 - mask) \cdot second\_head||_F^2$$

, where $first\_head$ and $second\_head$ are the first two heads of the last self attention layer, N is the number of elements in each feature map of the two heads, and $mask$ is the ground truth segmentation of one of the 2 objects.

After fine-tuning on the subset of the CO3D dataset, we managed to make the first 2 heads of the last self-attention layer of DINO to learn to focus on the 2 different objects from within each frame (Figure 9), which validated the motivation behind this approach.

Nevertheless, this method is not ideal, due to the incorporation of a supervised learning mechanism. Ideally, one would like this method to learn by itself a semantic representation

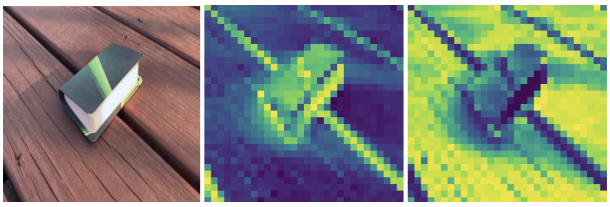

Figure 9: Fine-tuned DINO on a sample from CO3D with the visualization of the first two attention heads from the last self-attention layer.

.

of an image, instead of being forced into it. Similarly, DINO is not exactly computationally cheap, requiring large amounts of training resources to train two visual transformers in parallel.

3.4 ODIN

With ODIN ([6]), the learning of the semantic representation of an image is implicit, due to the presence of K-Means within it. We have implemented ODIN from scratch using PyTorch Lightning and Scikit ([10]) - for K-Means -. Ideally, ODIN contains visual transformers as part of its backbone (the three neural networks within the architecture of ODIN are based on ViTs), but, due to computational constraints, ResNet-18 is used, such that training time is feasible.

3.4.1 Extracting the Feature Maps

Unlike DINO, which is based on transformers and where the focus is on the last self-attention layer, in ODIN we focus on the last layer before the classification layer of the backbone (ResNet18 in our case). The output is a feature map of size 8 x 8, with 512 channels, on top of which we apply K-Means to visualize the semantic segmentation of the image.

3.4.2 Training Mechanism

The backbone (ResNet-18) was pretrained on ImageNet, while we fine-tuned the whole architecture on the CO3D dataset. Due to the nature of our dataset (a singular object on a certain background), we have implemented the mask computation mechanism by generating only 2 masks per image (one for the background and one for the singular object, which are inverted versions of one another).

Additionally, the input images have been resized to a resolution of 256 x 256, due to the capacity (in this case) of networks to learn meaningful representations. We have tried to make the network learn with 512 x 512 images, as well as with 1024 x 1024 images. In both cases, the network detects even the slightest details in the image (edges, screws, and so forth), which is undesirable in our setup. As a consequence, a resolution of 256 x 256 has proved to be a decent trade-off (Figure 10).

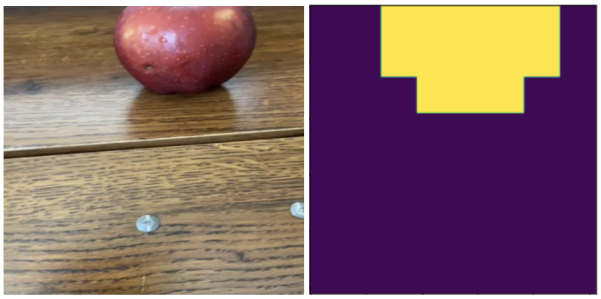

Figure 10: Sample from ODIN (at inference), by applying K-Means on the representation of the online network.

3.5 BYOL

An ancestor of both DINO and ODIN is another model, BYOL, which also learns an implicit representation of the network. BYOL, unlike ODIN, uses two networks for representation.

3.5.1 Extracting the Feature Maps

Since the backbone of BYOL is identical to the backbone of ODIN, we use the output of the backbone in the same way in order to extract the feature map. On top of it, as before, we then apply K-Means to obtain the segmentation.

3.5.2 Training Mechanism

Similarly with the case of ODIN, we have used ResNet-18 as the backbone of the networks, pre-trained on ImageNet, and fine-tuned the entire architecture on the CO3D dataset. We have obtained the best results with a resolution of 512 x 512, however, unlike the case with ODIN (Figure 11).

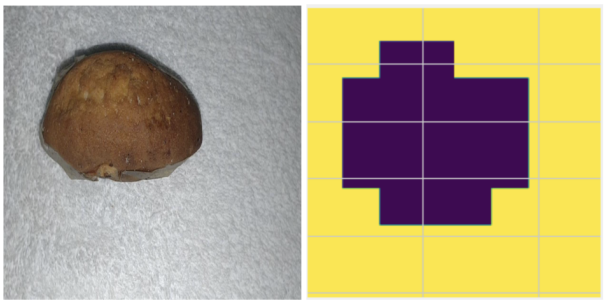

Figure 11: Sample from BYOL (at inference), by applying K-Means on the representation output by the online network.

For some images, the performance of BYOL is deficitary. In particular, in the case of CO3D, we have initially applied K-Means with a cluster size of 2. This is not optimal, the representation is not perfect and some objects may be missed (depending on the input). Therefore, we identify the cluster size empirically by applying a knee locator type of

algorithm, that identifies the maximum curvature of a function (which abstracts a distortion score among all clusters and which depends on k). In this way, we find the optimal amount of clusters and identify missing objects, that would normally not be identified with $k = 2$ (Figure 12).

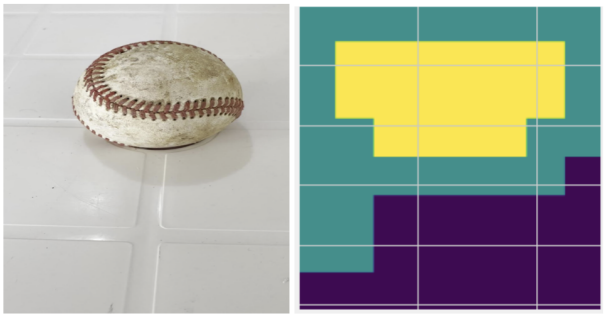

Figure 12: Sample from BYOL (at inference), by applying K-Means (with knee locator) on the representation output by the online network. The ball was initially not detected with $k = 2$.

Furthermore, in order to improve performance even more, we have tried to apply dimensionality reduction (PCA) on top of the representation produced by the network. The main motivation behind this is that within the feature map (of 8 x 8 with 512 channels), a lot of redundant information may be contained that may actually not be of value (such as contours or irrelevant objects). However, the experiments have shown that the information is already contained within a 3-dimensional subspace (i.e. we can reduce 512 channels to 3 channels and still preserve all the information) and that reducing it further damages the performance.

4 Scenes with Multiple Objects

In this section, we showcase the results that we obtained by trying to apply some of the techniques explored above to scenarios in which we have multiple objects.

4.1 Dataset

In particular, for this case, we have used the TUM RGB-D Slam Dataset ([13]). The dataset contains consecutive frames of multi-object scenes, filmed by a Kinect sensor at full frame rate.

4.2 BYOL

The only feasible model for this tasks has been BYOL. ODIN's standard K-Means mechanism doesn't allow for a variable number of objects within the scene and hence, a more

complex approach would be required. On the other hand, BYOL's input signal is represented by mere images and, thus, can learn effectively regardless of the dataset used.

Thus, we have used a pre-trained backbone (ResNet) on ImageNet, we have fine-tuned BYOL on the RGB-D dataset, and lastly we have applied K-Means with a variable number of clusters (identified via the knee locator algorithm) on the representation produced by BYOL (Figure 13).

Figure 13: Sample from BYOL (at inference), by applying K-Means (with knee locator) on the representation output by the online network on a sample from the RGB-D dataset.

5 Evaluation

We have compared various approaches that we have implemented against each other. One good metric often used in practice for evaluating the accuracy of semantic segmentation models is IoU (Intersection over Union - [12]) . For evaluation, we have used the ground truth segmentation masks from the CO3D dataset.

The way we apply IoU goes as follows. For an object within the image, we count the number of pixels that were identified as part of it and that are also part of the object in the ground truth. Similarly, we count the total number of pixels that are either in the ground truth mask or in the detected object in the image. Lastly, we divide the two numbers and we obtain the IoU score.

	IoU score on CO3D
ODIN 256	78.6
ODIN 512	59.08
BYOL 256	66.58
BYOL 256 + PCA	66.58
BYOL 512	78.01
BYOL 512 + PCA	78.01

Table 1: A comparison between all models that we have evaluated based on IoU score.

As one can see in Table 1, the model with the highest score has been ODIN trained on 256 x 256 images. Furthermore, as we discussed above, PCA applied on the feature map doesn't yield any benefits and, hence, the IoU score has remained intact after reducing the dimensionality to 3 channels.

6 Conclusion

In this work, we have tried out different state-of-the-art existing models in order to obtain an implicit semantic representation of the scene. We have taken various architectures with pre-trained backbones on generic datasets (e.g. ImageNet) and fine-tuned them on different custom datasets. On one of these datasets (CO3D) with singular objects on a background, we have shown the promise of the models.

In addition, we have also shown that one of these approaches (BYOL) can be directly applied to datasets that contain multiple objects, just by fine-tuning alone, with decent results. One the other hand, adapting other models (such as ODIN), might require substantial changes. K-Means is far from an ideal solution on multiple objects and, so, changing towards other novel approaches (e.g. based on superpixels) might prove beneficial.

References

[1] Mathilde Caron et al. "Emerging properties in self-supervised vision transformers". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9650–9660.

[2] Ting Chen et al. "Big self-supervised models are strong semi-supervised learners". In: *Advances in neural information processing systems* 33 (2020), pp. 22243–22255.

[3] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[4] Jianping Gou et al. "Knowledge distillation: A survey". In: *International Journal of Computer Vision* 129.6 (2021), pp. 1789–1819.

[5] Jean-Bastien Grill et al. "Bootstrap your own latent-a new approach to self-supervised learning". In: *Advances in neural information processing systems* 33 (2020), pp. 21271–21284.

[6] Olivier J Hénaff et al. "Object discovery and representation networks". In: *arXiv preprint arXiv:2203.08777* (2022).

[7] Prannay Khosla et al. "Supervised contrastive learning". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 18661–18673.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[9] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[10] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[11] Jeremy Reizenstein et al. "Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10901–10911.

[12] Hamid Rezatofighi et al. "Generalized intersection over union: A metric and a loss for bounding box regression". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 658–666.

[13] Jürgen Sturm et al. "A benchmark for the evaluation of RGB-D SLAM systems". In: *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2012, pp. 573–580.