# Localizing Objects with Self-Supervised Transformers and Improvements AI6103 Project Report

**Xi Zhen, Ju Xilai, Yu Yue, Wang Yanghao, Luo Hao**

Nanyang Technological University
{G2202373F, G2202544B, G2202151A, G2202518G, G2202279H}
{xizh0002, zu0001ai, yyu025, ywang150, luoh0013}@e.ntu.edu.sg

**Abstract**

Unsupervised learning methods has recently attracted great attention, as annotating takes enormous manpower in the traditional object detection field. In our project, we reproduced the implementation of Localizing Objects with Self-Supervised Transformers (LOST) algorithm to achieve self-supervised object detection without any labels, which proposed by Siméoni et al. in BMVC 2021(Siméoni et al. 2021). Besides, based on the LOST model, we proposed our approach called Similarity-orientated LOST (SoLOST) aiming to improve the performance through reconstructed seed expansion algorithm. We outperform LOST by up to 7.3 CorLoc points on PASCAL VOC 2012, and also achieved comparable results by training a class-agnostic detector (CAD) with the pseudo-label generated. The reproduction code to can be found at https://github.com/YuYue525/SoLOST.

## Introduction to LOST

Object detection has been developing rapidly in recent years and has been been applied to some critical areas like autonomous driving. However, most object detection methods require massive annotated data for training, which usually require very high manpower cost to produce, and thus restrict the wide-scale application of object detection technologies. Facing that, methods like semi-supervised learning, active learning, or self-supervised learning has been raised up to ease the the demand to annotation.

In 2021, Siméoni et al. proposed the LOST framework, which utilizes a well-designed algorithm to detect objects from the feature extracted by another self-supervised visual attention network, DINO(Caron et al. 2021a).

### DINO

DINO, proposed by Caron et al., is the base and inspiration of LOST. DINO is a self-supervised visual transformers model that extract attention over an specified image, inspired by the success of ViTs (Dosovitskiy et al. 2020), it learns the correct way of distributing attention by learning the correspondence of local and global views of images with a knowledge-distillation-like framework.

Specifically, the DINO network consists of two networks of same structure but different parameters, called teacher and student respectively. The global and local views of the same image, which are produced by random transforms, are fed to them respectively, the gradient of the teacher would filter into the student by a common loss function. In each iteration, the student would be optimized from the gradients, and the teacher would be updated with a certain policy, the moving average of the parameters of the student, for instance. Listing 1 shows a pseudo code of DINO network, "C" and "tpt" are for controlling output distribution.

Listing 1: Example DINO pseudo code w/o multi-crop

```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, m: network and center momentum rates
gt.params = gs.params
for x in loader:
    x1, x2 = augment(x), augment(x) # random views, global and local
    s1, s2 = gs(x1), gs(x2)
    t1, t2 = ft(x1), gt(x2)

    loss = H(t1, s2)/2 + H(t2, s1)/2
    loss.backward()
    update(gs) # SGD optimize student
    gt.params = l*gt.params + (1-l)*gs.params # moving average
    C = m*C + (1-m) * mean(concatenate([t1,t2]), dim=0)

def H(t, s):
    # cross-entropy
    t = t.detach() # stop gradient
    s = softmax(s / tps, dim=1)
    t = softmax((t - C) / tpt, dim=1)

    return -mean(sum((t * log(s)), dim=1))
```

One thing to be noted is that the "teacher" network in DINO merely means its role in the learning, unlike the teachers in knowledge distillation, the teacher networks in DINO are not pre-trained networks containing prior knowledge, but initialized together with the student network and learns global knowledge.

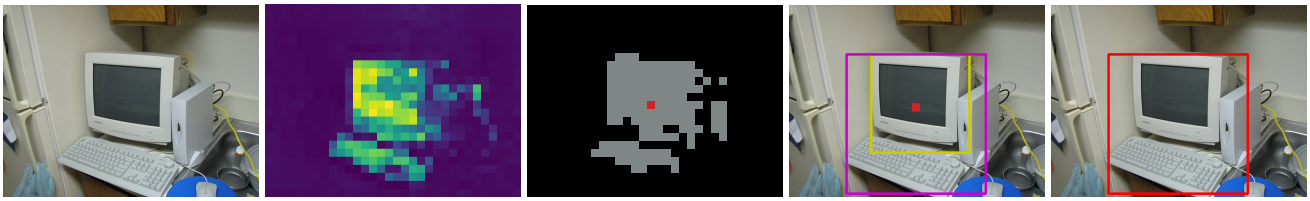

Figure 1: LOST process, including correlation maps, initial seed and similarity maps, seed expansion and pseudo box

### Object Detection Over DINO

Upon the result of DINO, Siméoni et al. proposed a method of object detection over the activation map.

The authors have made 2 basic assumptions: 1) in a same activation map, two regions of a same object would have a higher similarity than regions of object and background, and 2) single object covers smaller area than background. The assumption 1) was made by their empirical observation to the transformer-extracted features that the similarities between blocks in one object are usually positive and that between blocks of object and background are usually negative.

With above assumptions, the authors then raised the idea of using similarities between blocks or regions in the transformer-extracted feature map or activation map to conduct object detection.

In practice, like the transformers in natural language processing, the visual transformers would divide the images to $N$ patches or blocks and generate a $d$-dimensional feature $\mathbf{f}_p$ for $N+1$ blocks, in which the first feature corresponds to the "[CLS]" token, or the global information in vision field and others each corresponds to one block.

Their practice will be explained in the following content. There are multiple choices for the specific vision transformer or backbone of their net, all of which are self-supervised trained with the method of DINO. Fig.2 shows results of LOST with ViT-S as backbone and patch size 16, see the experiments section for more details.

1. The features produced by vision transformer corresponding to each block of the original image are kept as the feature for the blocks, and the global feature is discarded.
2. Calculate the similarity between each pair of the blocks.
3. **Initial seed selection**: for each block, count the blocks whose similarity with it is greater than 0. This step is to find how many blocks are likely to belong to the same class as the block.
4. Take the block with least similar blocks as the seed block.
5. **Seed expansion**: take $k$ blocks with the least number of similar blocks as patch blocks.
6. Calculate similarities between each patch blocks and seed block, take those with similarities greater than 0 as blocks belong to the same object as the seed block.
7. **Box extraction**: calculate mean value of the blocks taken in step 6 as well as the seed block, calculate similarities of other blocks in the feature map, mark the blocks with similarities greater than 0 as blocks belong to the same object.
8. With steps 3-7, a target region that likely to be an object has been selected out, generate a boundary box that just fit all the selected blocks.
9. **Generate class label**: after step 8, the algorithm has extracted one object from the image.

For further utilizing LOST on more tasks, the authors have also raised that the results produced by LOST, the pseudo-boxes, can be used to train existing supervised object detectors. They have discussed 2 scenarios: class-agnostic detection (CAD) and class-aware detection (OD).

For CAD, the object detector are required to find foreground objects in the images. The result of LOST can be used to train such detector by assigning "foreground" label to pseudo-box and "background" label to other parts of the image.

For OD, it is a bit more complicated because training such detectors need class label. To keep the framework fully-unsupervised, the authors proposed that they can cut the pseudo-boxes out and use DINO pre-trained transformer extract a feature, then use K-means algorithm to cluster the features to specific classes, then use Hungarian algorithm to align clustered labels to ground truth labels.

## Similarity-orientated LOST

When the LOST algorithm is localizing the objects by exploiting image representations extracted by a vision transformer, it always follows the procedure that selecting the seed $p^*$ with the smallest number of positive correlations with other patches, then applying seed expansion to obtain $\mathcal{S}$ and finally executing box extraction with the help of $\mathcal{S}$.

However, when carefully exploring the process of the seed expansion, we found some effects unwanted may occur. Specifically, when applying seed expansion, the original LOST method firstly finds $\mathcal{D}_k$ in which the $k$ patches with the lowest degree ($k$ is set to 100), and then selects those patches similar to the original seed $p^*$ as the seed set $\mathcal{S}$ from $\mathcal{D}_k$. We call this original seed expansion as correlation-orientated because the number of correlations is firstly considered and the similarity to the seed is secondary. The main problem is that sometimes the candidate seeds which have higher positive correlations but still very similar to the seed $p^*$ may not be considered if correlation-orientated. Therefore, the similarity-orientated LOST (SoLOST) is proposed.

Different from the original LOST, the seed expansion algorithm of SoLOST is similarity-orientated. After finding the seed $p^*$, we firstly find the potential seed set $\mathcal{P} = \{q \mid \mathbf{f}_q^T \mathbf{f}_{p^*} \geq 0\}$, and then we select $m\%$ of seeds with the lowest number of positive correlations from $P$ as $\mathcal{S}$. Here $m\%$ is the

parameter denotes the volume of $\mathcal{S}$ and it is set around $50\%$. In summary, the whole procedure of SoLOST is as follows:

1. **Initial seed selection** a binary symmetric adjacency matrix $A = (a_{pq})_{p,q\in\{1,...,N\}} \in \{0,1\}^{N\times N}$ is built to represent the patch similarity graph $\mathcal{G}$ for each image, in which

$$a_{pq} = \begin{cases} 1, \mathbf{f}_q^T \mathbf{f}_{p^*} \geq 0 \\ 0, otherwise \end{cases}$$

   Then the initial seed $p^*$ is selected as which has the lowest positive correlations with others:

$$p^* = \underset{p\in\{1,...,N\}}{\arg\min} \sum_{q=1}^{N} a_{pq}$$

2. **Seed expansion** Find the potential seeds that are similar to the initial seed $p^*$:

$$\mathcal{P} = \{q \mid \mathbf{f}_q^T \mathbf{f}_{p^*} \geq 0\}$$

   Then we sort $\mathcal{P}$ according to the degree in ascending order:

$$\mathcal{P}_{sorted} =< q_1, q_2, ..., q_{|\mathcal{P}|} >$$

   , where $deg(q_i) \leq deg(q_{i+1}), i \in 1, 2, ..., |\mathcal{P}| - 1$. Then the seed set is constructed as:

$$\mathcal{S} = \{q_i \mid q_i \in \mathcal{P}_{sorted}, i \leq m\%|\mathcal{P}|\}$$

3. **Box extraction** We finally compute the mask $\mathbf{m} \in \{0,1\}^N$ such that

$$m_q = \begin{cases} 1, \sum_{s\in\mathcal{S}} \mathbf{f}_q^T \mathbf{f}_s \geq 0 \\ 0, otherwise \end{cases}$$

   Which means the patch $q$ is regard as part of the object if its feature positively correlates with the features of the patches in $\mathcal{S}$.

Compared to the original LOST algorithm, SoLOST is able to handle some situations better theoretically. Consider the seed expansion process of LOST, it tends to find those "special" patches which have small number of positive correlations with others as the potential seed set. However, among those "special" patches, there may be very few which is similar to the seed because seed and those potentials are all "special". Different from LOST, SoLOST find those similar patches as the potentials, and then select a certain percentage of the special patches from those potentials as $\mathcal{S}$, so that the capacity of $\mathcal{S}$ is guaranteed to be considerable. Besides, in our perspective, LOST aims to localize only one object in the image, so we just need a seed and a considerable mount of those patches similar to the seed as a judging criterion. we don't really need much "special" patches as our potential seeds because it might introduce more noises if some patches happen to be slightly similar to the seed but with very small degree. In our experiments, the obtained results proves our idea to some extent.

## Class-agnostic Detection

To measure the performance, the single-object localization labels generated by LOST and SoLOST are exploited to train other object detection models without any human supervision. Here we explored another application called class-agnostic detection (CAD) to detect multiple objects in one image. The class-agnostic detectors ignore the semantic categories of the detected objects, and just localize those salient objects. In our implementation, we built a class-agnostic detector trained on the images with pseudo-boxes generated by LOST or SoLOST which breaks the limit of only being able to localize a single object. The trained detector not only can recognize more than one objects in an image, but also reflects the performance of LOST and SoLOST according to its performance.

In our experiments, we use the pseudo-boxes generated by LOST and SoLOST to train a Faster-R-CNN model to implement class-agnostic detection, and Average Precision at IoU 0.5 metric (AP@0.5) is measured to compare the performance. The results shows the effectiveness of the class-agnostic detector trained on pseudo-boxes generated by LOST and SoLOST, and indirectly prove that the SoLOST performance is more robust than LOST according to the class-agnostic detector's performance. To explore more, please refer to the experiment part.

## Experiments

In this part we introduce our experiments details and show the results for comparison. Similar to the experiments conducted in the original paper, we explored several variants of the object localization scenarios including single object detection by LOST or SoLOST and multiple objects detection without category labels trained on pseudo-boxes.

### Backbone networks

To obtain the activation maps for LOST and SoLOST, the ViT-S model proposed by Caron et al.(Caron et al. 2021b) follows the architecture of DEiT-S (Touvron et al. 2020) is trained by utilizing DINO (Caron et al. 2021a). We follow the parameters configuration proposed in the original paper to set the patch size $P = 16$, and extract the keys $K$ of the last layer as the activation map $F$ for better comparison, and the author also points out that the best performance can be achieved with these configurations.

Also, for more comparison among different versions of ViT, we conduct the comparing experiments on ViT-B and ViT-S with patch size $P = 8$ as well. For other different models, we also extract the last layer of the features as the activation map from the ResNet-50 (He et al. 2015) and VGG16 (Simonyan and Zisserman 2014) pre-trained by using DINO or on Imagenet (Deng et al. 2009) in supervised way, to compare the performance.

### Datasets

In the implementation, the performance of each model is mainly evaluated on three open-source datasets accessible: VOC2007 trainval and test set (Everingham and Winn 2010), VOC2012 trainval set (Everingham and Winn

| Backbone | pre-training | VOC07_traival | VOC12_trainval | COCO_20k |
|---|---|---|---|---|
| VGG16 | Imagenet | 41.4 | 47.2 | 30.2 |
| ResNet50 | Imagenet | 33.8 | 39.1 | 25.5 |
| ResNet50 | DINO | 36.8 | 42.7 | 26.5 |
| ViT-B/16 | DINO | 60.0 | 63.3 | 50.0 |
| ViT-S/8 | DINO | 55.3 | 57.0 | 49.8 |
| ViT-S/16 | DINO | **61.5** | **64.1** | **50.7** |

Table 1: CorLoc scores of different models based on LOST. By utilizing LOST on different features generated by different backbones: ViT-S with patch size $P = 8$ or 16, ViT-B, ResNet50 pre-trained following DINO, and VGG16 and ResNet50 trained on Imagenet.

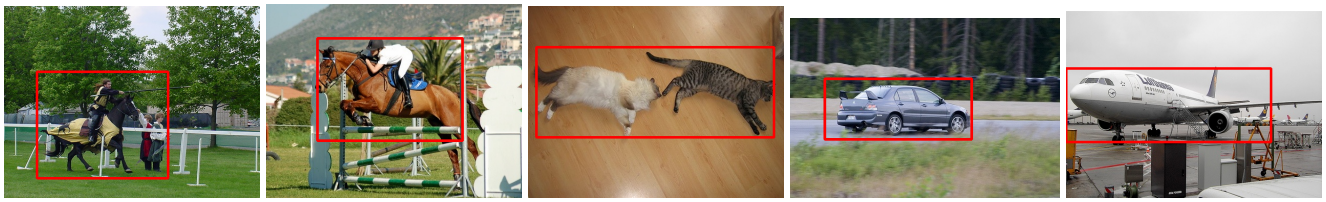

Figure 2: Example images show the object boxes generated by LOST exploiting the activation map from ViT-S with patch size $P = 16$.

| datasets statistics | | |
|---|---|---|
| class name | set type | image num |
| VOC2007 | trainval | 5,011 |
| VOC2007 | test | 4,953 |
| VOC2014 | trainval | 11,540 |
| COCO20k | train | 19,817 |

Table 2: Datasets used and the statistics.

2012) and COCO20k, in which COCO20k is the subset of COCO2014 train set (Lin et al. 2014). The table 2 shows some statistic information of the datasets for reference.

We conduct LOST and SoLOST, and then train the class-agnostic detectors on the VOC2007 trainval, VOC2012 trainval and COCO20k train set. Then we also test on these datasets plus VOC2007 test because LOST and SoLOST are both unsupervised, the ground-truth labels are only utilized when computing CorLoc and AP50.

## Single Object Detection by LOST

In the first experiment, we reproduce the results of LOST on the datasets VOC2007 trainval, VOC2012 trainval and COCO20k train set. To compare the different models' performance, we conduct the same process of LOST algorithm on different activation maps generated by different models discussed above. The table 1 shows the CorLoc scores by different backbone results. As we can see from the table, different feature maps really have a great impact on the localization results. When we exploit the key $K$ generated by ViT-S with patch size $P = 16$, then LOST achieves the best results.

In the original paper, the similarities between patch pairs $p$ and $q$ are computed directly by multiple the feature vectors of the two patches $\mathbf{f}_p^T \mathbf{f}_q$. However, after exploring, we found that the feature vectors are not normalized, which means we can use other measurement such as cosine similarity or Pearson product-moment correlation coefficient (PCCs). The results in the table 3 shows the difference. Actually, initially, because we compare the similarity computed by $\mathbf{f}_p^T \mathbf{f}_q$ with 0, the cosine similarity can get the same results as the original implementation theoretically. However, we found by utilizing cosine similarity, the results are improved slightly, especially for the model ViT-S and ViT-B, but the results obtained by Pearson product-moment correlation coefficient is relatively worse than before. In this experiments, we improved the original implementation, and then in SoLOST we will also use cosine similarity to measure the similarities among patches.

We also shows some examples to visualize the algorithmic effect of LOST in the figure 2. The figures are selected from VOC2007 trainval.

## Single Object Detection by SoLOST

Then in the second experiment, we implement SoLOST algorithm on the datasets VOC2007 trainval, VOC2012 trainval and COCO20k train set. Similarly, we conduct the same process of SoLOST algorithm on different activation maps as well. The table 4 shows the CorLoc scores.

From the table, compared to the results generated by LOST, SoLOST can shot higher CorLoc scores by ViT-B, ViT-S and VGG16 models, which reflects SoLOST's better performance. At the same time, the results obtained by ResNet50 is consistent with the experiments when we change the similarity measurement: by exploiting the feature map extracted from the last layer of ResNet, LOST

| Backbone | pre-training | VOC07_traival | | | VOC12_trainval | | | COCO_20k | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $\mathbf{f}_p^T\mathbf{f}_q$ | cos sim | PCCs | $\mathbf{f}_p^T\mathbf{f}_q$ | cos sim | PCCs | $\mathbf{f}_p^T\mathbf{f}_q$ | cos sim | PCCs |
| VGG16 | Imagenet | 41.4 | **41.6** | - | 47.2 | 47.0 | - | 30.2 | 30.1 | - |
| ResNet50 | Imagenet | 33.8 | 33.6 | 31.1 | 39.1 | 39.0 | 36.2 | 25.5 | 25.4 | 25.5 |
| ResNet50 | DINO | 36.8 | 36.5 | 30.8 | 42.7 | 42.5 | 35.9 | 26.5 | 26.4 | 26.5 |
| ViT-B/16 | DINO | 60.0 | **60.1** | **60.3** | 63.3 | **63.4** | **63.4** | 50.0 | 50.0 | **50.3** |
| ViT-S/8 | DINO | 55.3 | 55.3 | 55.0 | 57.0 | **57.2** | **57.1** | 49.8 | **49.9** | 49.8 |
| ViT-S/16 | DINO | 61.5 | **61.7** | **61.6** | 64.1 | **64.3** | 64.1 | 50.7 | 50.7 | 50.6 |

Table 3: CorLoc scores of different models by using different similarity measurement based on LOST. By utilizing LOST with different similarity measurement to measure the patch similarity. In the table, "-" means the results have no reference value due to the seed $p*$ is wrongly generated in the background by the corresponding algorithm.

| Backbone | pre-training | VOC07_traival | | VOC12_trainval | | COCO_20k | |
|---|---|---|---|---|---|---|---|
| | | LOST | SoLOST | LOST | SoLOST | LOST | SoLOST |
| VGG16 | Imagenet | 41.4 | **41.9** | 47.2 | **48.9** | 30.2 | **30.7** |
| ResNet50 | Imagenet | 33.8 | 32.6 | 39.1 | 37.9 | 25.5 | 24.7 |
| ResNet50 | DINO | 36.8 | 34.6 | 42.7 | 39.6 | 26.5 | 24.8 |
| ViT-B/16 | DINO | 60.0 | **61.8** | 63.3 | **64.4** | 50.0 | **52.0** |
| ViT-S/8 | DINO | 55.3 | **59.8** | 57.0 | **64.3** | 49.8 | **51.1** |
| ViT-S/16 | DINO | 61.5 | **62.2** | 64.1 | **64.8** | 50.7 | **52.2** |

Table 4: CorLoc scores of SoLOST compared with LOST. SoLOST gets higher CorLoc scores on the whole.

and SoLOST both cannot generate reliable object boxes, and also, the improved algorithm didn't show good scores in ResNet50. Therefore, we can conclude that the feature map extracted from the last layer of ResNet50 is not suitable for LOST/SoLOST.

Also, we shows some examples to visualize the results generated by SoLOST in the figure 3. From the object boxes generated by LOST and SoLOST on the same images, we can see that obviously SoLOST is much more accurate than LOST. It is not only able to recognize the edge of the car to expand the box, but also accurately localize the goat from grass to shrink the object box.

## Class-agnostic Unsupervised Object Detection

We use the R50-C4 model of Detectron2 with ResNet50 pre-trained with DINO self-supervision model to construct the class-agnostic detector. Some training details are list in the following:

- mini-batches of size 2 across 1 GPU using BatchNorm
- extra BatchNorm layer for the RoI head after conv5, i.e., Res5ROIHeadsExtraNorm layer in Detectron2
- frozen first two convolutional blocks of ResNet-50, i.e., conv1 and conv2 in Detectron2
- learning rate is first warmed-up for 100 steps to 0.02 and then reduced by a factor of 10 after 18K and 22K training steps
- we use in total 24K training steps for all the experiments, except when training class-agnostic detectors on the pseudo-boxes of the VOC07 trainval set, in which case we use 10K steps.

For more training details, please refer to our github. The table 5 shows the CorLoc and AP50 scores of CAD trained on the pseudo-boxes generated by LOST and SoLOST, respectively. As we can see from the results, to detect multiple objects in an image, the CAD trained on the pseudo-boxes from SoLOST is better than LOST, which is consistent to the previous results show that SoLOST is more accurate at detecting single object. The figure 4 shows the comparison between LOST and SoLOST.

## Conclusion

In this project, based on the LOST proposed by Siméoni et al. in BMVC 2021(Siméoni et al. 2021), we reproduce the experiments of LOST part based on the open-source code provided. Based on their work, we proposed our own algorithm SoLOST which aims to modify the seed expansion process to achieve better accuracy. Our experiments shows the efficiency of SoLOST over LOST and achieves higher CorLoc scores. Besides, based on the pseudo-boxes generated by LOST and SoLOST, we also trained a class-agnostic detector to localize multiple objects, and shows the higher CorLoc and AP50 scores of SoLOST over LOST.

In summary, there is no label utilized in the whole pipeline of SoLOST by unsupervised approach, and the algorithm is very faster than other training models and also achieves better performance over LOST. Hopefully, SoLOST will pro-

Figure 3: Example images compare the object boxes generated by LOST and SoLOST both exploiting the activation map from ViT-S with patch size $P = 16$. The images above shows the results of LOST and below shows the results of SoLOST. As we can see, SoLOST is more accurate.

| datasets | | VOC07 | | VOC12 | COCO_20k |
|---|---|---|---|---|---|
| training set | | trainval | | trainval | trainval |
| evaluation set | eval value | trainval | test | trainval | trainval |
| LOST+CAD | CorLoc | 60.7 | - | 67.8 | 53.3 |
| | AP50 | 23.7 | 23.4 | 30.7 | 8.8 |
| SoLOST+CAD | CorLoc | **61.2** | - | 67.1 | **54.8** |
| | AP50 | **25.7** | **25.7** | **31.0** | **9.1** |

Table 5: CorLoc and AP50 scores for CAD. The results show that CAD trained with the help of SoLOST is better than LOST on the whole.

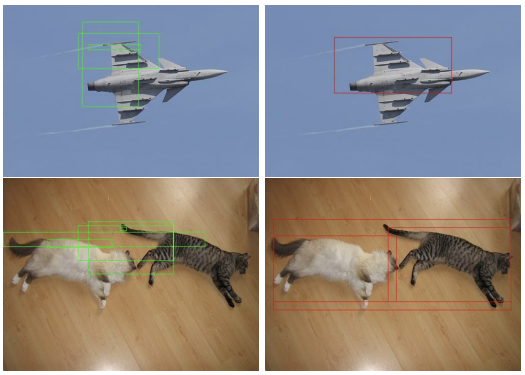

Figure 4: CAD trained on pseudo-boxes generated by LOST (left) and SoLOST (right).

vide some inspiration for future research.

## References


Caron, M.; Touvron, H.; Misra, I.; Jégou, H.; Mairal, J.; Bojanowski, P.; and Joulin, A. 2021a. Emerging Properties in Self-Supervised Vision Transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

Caron, M.; Touvron, H.; Misra, I.; Jégou, H.; Mairal, J.; Bojanowski, P.; and Joulin, A. 2021b. Emerging Properties in Self-Supervised Vision Transformers.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *CoRR*, abs/2010.11929.

Everingham, M.; and Winn, J. 2010. The PASCAL visual object classes challenge 2007 (VOC2007) development kit. *Int. J. Comput. Vis*, 88(2): 303–338.

Everingham, M.; and Winn, J. 2012. The PASCAL visual object classes challenge 2012 (VOC2012) development kit. *Pattern Anal. Stat. Model. Comput. Learn., Tech. Rep*, 2007: 1–45.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, 740–755. Springer.

Siméoni, O.; Puy, G.; Vo, H. V.; Roburin, S.; Gidaris, S.; Bursuc, A.; Pérez, P.; Marlet, R.; and Ponce, J. 2021. Localizing Objects with Self-Supervised Transformers and no Labels. *CoRR*, abs/2109.14279.

Simonyan, K.; and Zisserman, A. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition.

Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; and Jégou, H. 2020. Training data-efficient image transformers amp; distillation through attention.