Object Detection with Semi-supervised Learning

Tanran Zheng
New York University
tz408@nyu.edu

Guojin Tang
New York University
gt1188@nyu.edu

Deep Mehta
New York University
dnm7500@nyu.edu

Abstract

In this paper, we focus on the semi-supervised learning techniques for Object Detection task. We have 514k unlabelled data together with 30k labelled data for our task and explore the potential gains of utilizing the self-supervised learning technique. We use SwAV for the model backbone pre-training and Faster R-CNN for the downstream object detection. Our final semi-supervised learning model achieves an AP@0.5 of 44.4% on validation data, outperforming the AP@0.5 of 22 using the baseline Faster R-CNN with no SwAV pre-training.

1 Literature Review

In this report, we reviewed three major contrastive approaches MoCo, SimCLR and SwAV for the pre-training part of semi-supervised learning. Firstly, MoCo [1], Momentum Contrast for Unsupervised Visual Representation Learning is a dictionary look-up approach to generate a visual representation for the task of unsupervised learning. In MoCo, we look at contrastive learning as an encoder for dictionary lookup tasks. Dictionary keys are built using the train data and used to look up for encoding of the new data. MoCo benefits from large models and achieves 68.6% accuracy with 4x width R50. Secondly, SimCLR [2] uses stochastic data augmentation to transform the train data and then use a neural encoder to extract the feature representations. It uses a projection head to map the representations and uses contrastive loss to do the optimization. SimCLR achieves 76.5% accuracy on ImageNet using a 4x width ResNet-50 for Image Classification tasks.

While these methods offer a great advantage over their previous work, our method, SwAV [3] outperforms both of them achieving 78.5% with 4x width ResNet50 for Image Classification. Therefore, given its excellent performance on object recognition, we use SwAV on our object detection task.

2 Methodology

2.1 SwAV: Unsupervised backbone

In SwAV [3], we follow an online self-supervised learning method based on clustering. Here, instead of comparing the features of the images, we contrast the dataset by comparing their cluster assignments. For a given image, we perform multiple data augmentations and extract their feature representations through ResNet50 followed by a projection head. Further, we perform swap assignments in calculating the loss of 2 different views of the same image such that through optimization the model will be able to give different views of the same object with similar representations. We will see how SwAV transfer on object detection pre-training task. Additionally, SwAV does not require a memory bank for training which makes it efficient for training with large data.

2.2 Faster RCNN: Supervised downstream

Given a well-trained backbone using unsupervised training, it is crucial to transfer the features learned from unlabelled data to a downstream task: object detection. We choose Faster RCNN as our downstream object detection task model. It utilizes Feature Pyramid Networks (FPN) and Region Proposal Network (RPN) on top of the backbone, which shows promising performance for the object

detection task. We also make modifications to the architecture of the vanilla Faster RCNN model to address some issues during the transfer learning of the downstream task.

Normalization One of the biggest challenges is that the features produced by unsupervised pre-training have different distributions compared with the labelled dataset. This will disrupt the features learned of the backbone from the pre-training, especially for the normalization layers in the backbone. For other downstream tasks like image classification, this might not be a big problem as one can always fine-tune most or all the layers of the backbone. But because the Region Proposal Networks in Faster RCNN can only be trained by feeding one image at a time, we have to train it with very small batch size. M. Caron et al. [3] uses `SyncBatchNorm` with batch size equals 2 and train the model on 64 GPUs for the downstream task, which involves the resources that we don't have access to. Using a small batch size makes the batch statistics very poor and degrades performance. Therefore, we make modifications to the normalization layers in the model, including replacing the `BatchNorm` layers in the backbone and adding a normalization layer in the vanilla MLP RoI head of Faster RCNN.

Box-head of RoI Head In the original paper of SwAV, M. Caron et al. [3] uses a different type of RoI head in the object detection model for the downstream task. It is a block similar to the ResNet bottleneck block (refer as `res5` layer in [3]) with an additional `BatchNorm` layer on the output side. We also investigate this method and compare its results with the traditional MLP heads.

3 Experiments and Results

Backbone	$AP_{0.5}$[all]	AP[small]	AP[medium]	AP[large]
res18_ep70_p1000	0.329	0.037	0.072	0.213
res34_ep100_p3000	0.391	0.038	0.095	0.265
res50_ep100_p2000	0.399	0.043	0.102	0.265
res50_ep100_p3000	0.427	0.038	0.111	0.263
res50_ep200_p3000	**0.444**	**0.055**	**0.113**	**0.280**

Table 1: The ep in backbone refers to the epoch number and the p refers to the prototype number in SwAV training. AP[small/medium/large] refers to AP IoU@0.50:0.95 for different area.

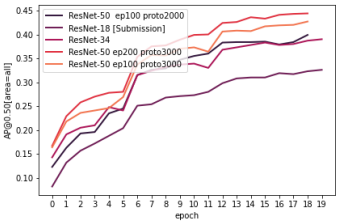
3.1 Model Architecture

SwAV: backbone model For the unsupervised part, we tried ResNet18, ResNet34 and ResNet50 as our backbone. As we can see from the final R-CNN results in Table 1, the larger architecture ResNet50 had better overall object detection results than the smaller ones. Though in large area detection, ResNet50 had similar results to that of ResNet34, in small and medium area detection, ResNet50 managed to make a further improvement than ResNet34. Larger architecture can capture more information about the image, resulting in not only a better overall detection ability but also an increasing capability to detect smaller and more complicated objects in images. As in the error analysis image in Figure 1a, the background and the objects in this picture have relatively complicated colors and contours, and ResNet50 managed to detect the object carts while ResNet18 failed.

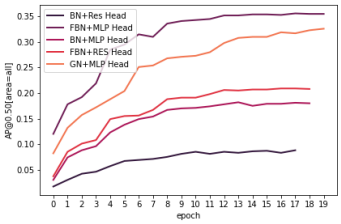
SwAV: num of prototypes Prototypes in SwAV can be thought of as the clusters to which the images are assigned. In actual neural network computing, a prototype matrix is the weights of the dense layer with a softargmax activation that maps the encoded features of the image to its probability distribution over K prototypes or clusters. Therefore, the number of prototypes acts similar to the hidden dimensions of the neural network. As we can see from the final R-CNN results of ResNet50 in Table 1, 3000 prototypes did give better results than 2000 prototypes. But for APs of more strict IoU standard, the model with 3000 prototypes required more epochs to converge.

RCNN: Normalization Layer After pre-training the backbone with SwAV, we replace the original backbone in Faster RCNN with the new pre-trained backbone. Then we replace the `BatchNorm` (BN) layers in the backbone with `FrozenBatchNorm` (FBN), `GroupNorm` (GN), and compare their result with the result of using `BatchNorm`. In the experiment of BN and FBN, the backbone is pre-trained with `BatchNorm` layers as normalization layers; in the experiment of GN, the backbone is pre-trained

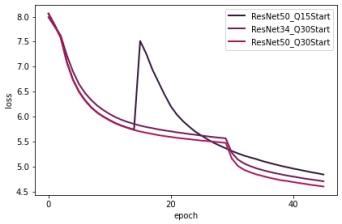
(a) Error analysis example



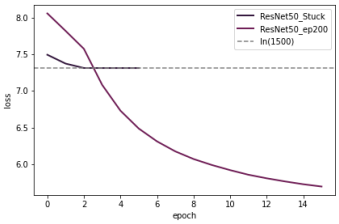
(b) RCNN: Different Backbones



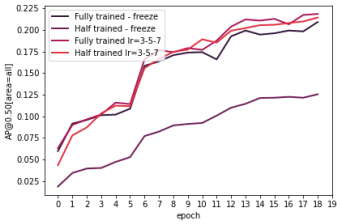
(c) RCNN: Norm layers, RoI heads



(d) SwAV loss: #epoch queue starts



(e) SwAV loss stuck



(f) RCNN: Different learning rate

Figure 1: Results of SwAv and RCNN with model architecture

with `GroupNorm` instead. In this way, we can keep the pre-trained weights in these layers. Figure 1c shows that FBN outperforms all other types of normalization layers. Because FBN will freeze its weights during the downstream training, but keep updating the running variance and running mean so that it can retain the features learnt from pre-training while keep adapting to the new data set. The BN layers perform the worst because its parameters are trained using the unlabelled data set and we are not able to train Faster RCNN with a large batch size as mentioned in section 3.2. Thus, the inaccuracy of batch statistics degrades the performance of the model. The GN performs worse than FBN but gradually catches up as the training progresses. The margin is still obvious because although GN is suitable for the downstream training, it is not suitable for the pre-training using SwAV. Thus, the backbone trained by GN has less ability to perform feature extraction in the downstream task, and this gap cannot be closed by the downstream task training with a small labelled data set.

Box-head of RoI Head We also compare the result of using MLP box-head and the result of using a ResNet block (`res5`) as box-head in the RoI head of RCNN. We also modified the MLP head by inserting an extra `LayerNorm` layer in between the FC layers to further mitigate the normalization problem. We also add an extra `BatchNorm` layer after the `res5` layer. From Figure 1c we can see that the improved version of MLP box-head outperforms the `res5` head in a different setting by a big margin. The reason that it performs well in [3] but shows a different conclusion here is because we have less labelled data for the downstream task training, as a result, it cannot be fully train to support the class classification and bbox predictions.

3.2 Training

SwAV: distributed data parallel training Single GPU training cannot train large architectures with decent batch size, therefore distributed data parallel training is crucial for our unsupervised training part that has 512k images as data set. We trained ResNet34 and ResNet50 on one node with 4 GPUs on GCP. The training speed is approximately 45 epochs per day. The ResNet50 can have a batch size of 256 and 2 data loader workers for 4 16G GPUs without causing Cuda memory issues.

SwAV: Queue As we can see from the Figure (1d), the SwAV losses had a sharp decrease at epoch 30 where we started to employ the queue technique. The queue is composed of feature representations from the previous batches. In each epoch, the queue discards the oldest feature representations from the queue and save the newest feature representations just learned in current batch. In this way, we can assign queue_length + batch_size images to K prototypes instead of just batch_size images to K prototypes, which is very helpful since we have relatively small batch size. However, we found if

we started the queue at epoch 15 as the original paper did, the SwAV loss would increase back to ln(prototype_number) and then restarted to decrease. We think this is because we have much less data than the original paper, therefore if we start the queue at epoch15, the feature representations haven't learned enough information and will disturb the assignment procedure.

SwAV: Mixed precision Other than improvements in training efficiency, mixed precision technique also helped us solve the problem of gradient collapsing using loss scaling. As we can see from Figure (1e), we have run into situations where the SwAV loss stuck at $ln(prototype\ number)$. The gradient collapsed and the examples were decoded into same feature representation. Then, to satisfy the equi-partition constraint, the uniform assignment was applied and would cause the loss to be stuck at $ln(prototype\ number)$. However, in mixed precision training, the loss will be scaled larger before backward propagation which can simultaneously avoid the vanishing gradient problem. The gradients will be scaled back when updating the gradients. Though we had couple times of gradient overflow, the mix_precision technique in the Apex module managed to skip these updates and gave us better overall results without the collapsing problem.

SwAV: Data augmentation Since most available self-supervised learning frameworks for images including SwAV are designed for object recognition tasks, the data augmentation techniques can cause the images to lose important information about the position and size of the objects which is very crucial in object detection tasks. The default multi-crop strategy in SwAV creates 2 large views and 6 small views for each image. And the random crop area for large views is 14% ~100% of the original image area and that for small views is 5% ~14%. Therefore, small objects in images could be completely cropped off in some of the views, and this could mislead the model in the process of swapping prediction within different views of same image. As we can see from the AP results for different sized areas in Table 1, the models generally performed worse on smaller objects. We tried to increase the crop scale to 40% ~100% for large views and 20% ~40% for small views, but the results proved to be worse. For future experiments, we want to investigate further into this problem of data augmentation for object detection.

RCNN: learning rate Learning rate is a crucial hyper-parameter for the task of downstream fine-tuning. We experiment with different types of settings: freezing the backbone, "3-5-7" rules, and training the backbone using the same learning rate as the other part of the model. The "3-5-7" rules here refers to assigning $base\_lr \times 10^{-3}$ for the RCNN, $base\_lr \times 10^{-5}$ for later part of backbone, and $base\_lr \times 10^{-7}$ for the first a few layers and the first residual block of the backbone. We can see from Figure 1f that if the backbone is well pre-trained (a ResNet-50 trained for 200 epoch), there is not much difference between these two methods. But if the backbone is not fully pre-trained (a ResNet-50 trained for 100 epoch), the difference between performances is huge.

4 Conclusion

In this project, we implemented the SwAV and Faster RCNN model to solve the object detection task and performed extensive experiments for the architectural and training details of these models.

Though our model achieved decent results for our object detection task, there is still a performance gap between Object Detection and Image Recognition tasks. Most of the self-supervised learning frameworks are designed to solve for Image Recognition tasks, and the contrastive training techniques can cause the images to lose important information of the position and size of the objects which is very crucial in object detection tasks. Thus, in the future we would like to explore more about applying self-supervised learning frameworks to solve for object recognition tasks.

References

[1] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. *CoRR*, abs/1911.05722, 2019.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. 2020.

[3] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *CoRR*, abs/2006.09882, 2020.