Self-supervised Pre-training for Object Detection

Haoming(Hammond) Liu
NYU Shanghai
hl3797@nyu.edu

Wenbin(Jim) Qi
NYU Shanghai
wq372@nyu.edu

Haorui(Harry) Lee
NYU Shanghai
hl3794@nyu.edu

Abstract

This work pre-trains a Vision Transformer (ViT) in a Masked Autoencoder (MAE) manner and uses it for object detection. In the fine-tuning stage, we adopt the plain feature pyramid network (FPN) from ViTDet and a standard Faster R-CNN detection head. With strong pre-trained features and effective fine-tuning tricks, our model finally achieves 30.3% mAP on the test set (team rank 3).

1 Introduction

Along with the boom of deep learning, new model architectures and training paradigms are constantly being proposed and updated. With the help of powerful hardware devices, we are now able to pre-train our models with a huge amount of unlabeled data to boost the performance on various downstream tasks. Following the competition guidelines, the goal of this work is to boost the performance of object detection via self-supervised pre-training.

Specifically, 512,000 unlabeled images of size 224×224 are provided for pre-training; whereas 50,000 labeled images of different sizes are used for the adaption of object detection (`train:val=3:2`), where the bounding boxes consist of objects from 100 classes. The final model will be evaluated on a hidden test set with 10,000 labeled images (drawn from the same data distribution).

Convolutional Neural Network [5] has achieved great success in the past decade. As one of the most popular CNN-based architectures, ResNets [2] were almost the default backbone while evaluating pre-training methods. Until recently, Vision Transformers (ViT) [1] start to demonstrate its power. Accordingly, ViT-based pre-training methods like Masked Autoencoder (MAE) [3] were proposed to explore the potential of this new architecture. Following the recent work by Li et al. [6], we aim to investigate the performance of ViT models as a backbone network for object detection.

2 Related Works

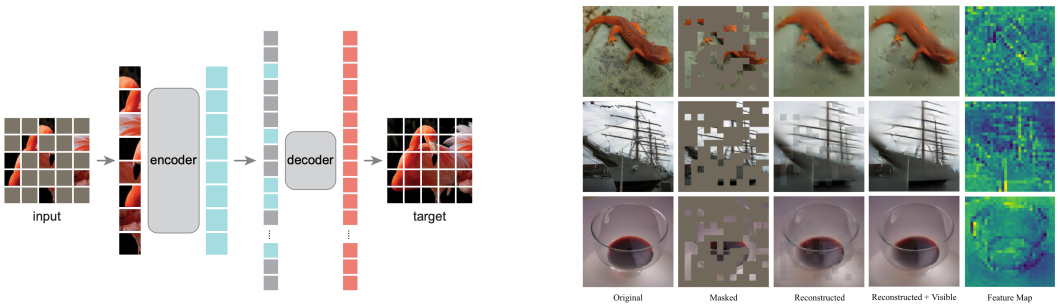
Autoencoding. Autoencoding [4] is a classical method for learning representations. The encoder maps an input to some latent representation, whereas the decoder is used to reconstruct the input. For example, Denoising Autoencoders (DAE) [8] is a class of methods that corrupt the input signal and tries to reconstruct the uncorrupted signal. The Masked Autoencoder (MAE) [3] method we used in this work is a special form of DAE, where the random masking is introduced as the noise.

Object Detection. Object detection aims to find the bounding boxes for the objects in a given image. Faster R-CNN [7] is a classical 2-stage detection method, which is generally used as a default detection head while evaluating the performance of different pre-training methods.

3 Method

Pre-training. We adopt the Masked Autoencoder (MAE) proposed by He et al. [3] to perform the pre-training on the unlabeled data. We use a standard ViT-Base [1] model with a patch size of 16.

GitHub Repository: `https://github.com/hmdliu/DL-SP22-Team2`



(a) MAE architecture (figure from He et al. [3]).

(b) Reconstruction results on unlabeled data.

Figure 1: Illustrations of the Masked Autoencoder (MAE).

During pre-training, we mask off most input patches (around 75%) and feed the remaining ones into the encoder, which produces some latent representations; then, we use a lightweight decoder to reconstruct the original image. Visual illustrations of MAE are provided in Figure 1a. After the pre-training stage, we can drop the decoder and use the encoder as a backbone network.

We largely adopt the code and training configurations from the official GitHub repository[1]. Our ViT-B/16 model uses a mask ratio of 75% while pre-training on unlabeled data. We use the AdamW optimizer (`betas=0.9,0.95`) with a weight decay of 0.05. Our base learning rate is `1.5e-4` with a cosine scheduler. We set the per device batch size to 64 and train on 4 Tesla V100 GPUs. Our effective batch size is 512 (accumulation iteration = 2), which mimics a single node in the original training setting [3]. We adopt a standard data augmentation (random resized crop & random horizontal flip) and warm up the learning rate in the first 20 epochs. Figure 1b shows the reconstruction results after pre-training for 80 epochs. Overall, our MAE model gives reasonable reconstructions for the masked inputs. Besides, we also plot the feature maps for the uncorrupted inputs, where some patterns of the objects, though a bit coarse, are still shown in the single-scaled ViT feature maps (stride = 16).

Fine-tuning. To adapt the pre-trained backbone network to the detection task, we interpolate the positional embeddings and transform all inputs to 512×512 to ensure the consistency of input sizes. For a fair evaluation, we only use a standard Faster R-CNN [7] detection head with a slight modification on the Feature Pyramid Network (FPN).

Different from CNN-based architectures, ViT-based models have single-scaled feature maps, where the stride is in proportion to the patch size. In a recent paper, Li et al. [6] empirically found that it's already sufficient to generate the feature pyramid with the last feature map (instead of an aggregation of all stages), where each layer is produced by up-sampling (transpose convolution) or down-sampling (max pooling). A possible reason is that ViT models can reply on positional embeddings to encode spatial information, and moreover, its feature extraction procedure won't lead to a information loss in later stages (*e.g.,* shrinking size via max pooling). In our case, the output feature map is of size $32 \times 32 \times 768$, then we introduce four extra layers and construct the feature pyramid of sizes [128, 64, 32, 16, 8]. We adopt the default anchor generator for Faster R-CNN (sizes = [32, 64, 128, 512], aspect ratios = [0.5, 1.0, 2.0]) for detection. The feature pyramid is visualized in Figure 2a, where the object regions activate nicely after up- or down-sampling the raw output.

We fine-tune the pre-trained backbone network on a single GPU with a batch size of 2. We use AdamW optimizer (`betas=0.9,0.999`) with a weight decay of 0.01. The base learning rate is set to `1e-4` with 8000 steps of warm-up and it decays by a factor of 0.1 every 10 epochs. We apply random horizontal flip as the standard data augmentation. More experimental attempts are discussed and evaluated in Section 4, including color jitter, backbone freezing, and per-layer decayed learning rate.

4 Experiment

In this section, we investigate the performance of different configurations of learning rate and data augmentation and perform the experiments in an iterative manner (*i.e.,* resuming from the best checkpoints in the previous stage). We report the mAP, AP50, and AP75 metrics in Tabel 1.

[1] `https://github.com/facebookresearch/mae`

Method	mAP	AP50	AP75
Stage 1 (15 epochs)			
(A) No Pre-training	6.5	14.3	4.9
(B) Freeze	16.3	31.0	15.3
(C) Freeze + Color Jitter	13.4	25.7	12.4
(D) Unified lr	23.2	37.5	24.7
(E) Decay lr	**25.9**	**41.9**	**27.6**
(F) Decay lr + Color Jitter	24.0	39.0	25.9

Method	mAP	AP50	AP75
Stage 2 (15 epochs)			
(B) → (E)	28.8	46.5	30.8
(B) → (F)	29.4	**46.6**	31.7
(C) → (F)	28.7	45.9	30.9
(E) → (E)	29.0	44.9	31.1
(E) → (F)	**30.1**	**46.6**	**32.9**
(F) → (F)	29.5	45.7	32.1

Table 1: Average Precision metrics (%). The best method is in **bold**, top 3 methods are underlined. "(A) → (B)" means resuming from (A)'s best checkpoint in Stage 1 and continue training using (B).

To start with, we investigate the effect of MAE pre-trained features. As shown in the left sub-table, there is a clear gap between the non-pretraining one and any of the others, which demonstrates that the pre-training on unlabeled data has successfully boosted the performance on downstream tasks.

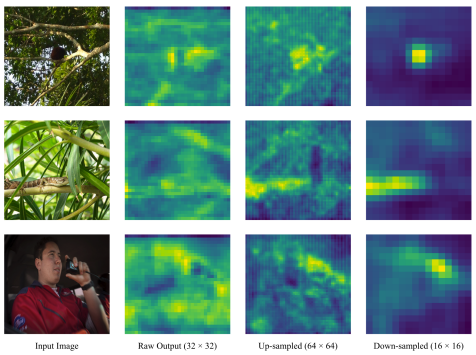
We divide fine-tuning into two stages to allow more flexibility of comparisons. Overall, the results of our attempts can be summarized as follows:

Per-layer decayed learning rate (backbone). As shown in the left sub-table, applying a per-layer decayed learning rate in the backbone gives a 2.7% mAP increment. Such a trick stabilizes the pre-trained representations in the backbone and thus ensures a better performance.

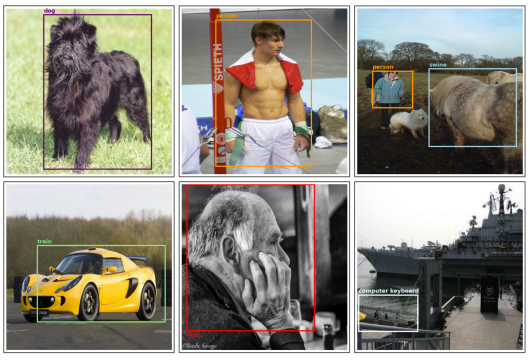
Color Jitter. Following Li et al. [6], we introduce a strong color jitter (brightness, contrast, saturation = [0.1, 2.0]; hue = [-0.1, 0.1]) in some of the experiments, which doesn't change the position of bounding boxes but produces nice variants of the input. According to the results, applying color jitter doesn't improve the performance in the first stage, which suggests it's generally no good to increase the variance of data distribution in the early fine-tuning stage (since the task-specific heads are not "fully working" yet). In comparison, applying color jitter in the second stage usually gives a better performance, and moreover, this potentially allows us to train for more epochs before overfitting.

Backbone Freezing. We also tried to freeze the backbone in the first stage to warm up the detection head (and then use decay lr + color jitter for the second stage), which gives outstanding performance in an early experiment. However, we were unable to reproduce it in later experiments. Apart from this pity, one solid advantage of the freezing trick is its training speed, which is the fastest among all experiment settings (since we don't need to tune the backbone at all).

Summing up all the effective tricks, the best fine-tuning setting according to Table 1 is to repeat method (F) interatively. As we haven't completed all these ablation studies by the deadline of the last leaderboard, our best attempt on the public leaderboard is conducted in "(B) → (F) → (F) → (F)" manner, which achieves 32.0% mAP on the validation set and 30.3% mAP on the test set. Notably, with the help of robust pre-trained features and strong data augmentation, we still didn't observe any sign of convergence even though we've resumed from the previous best checkpoints 3 times.



(a) Feature pyramid outputs.



(b) Detection results.

Figure 2: Object Detection Visualizations.

5 Discussion

Some detection results on the validation set are plotted in Figure 2b, where the success and failure cases are shown in the first and the second row respectively. Overall, our model is able to predict the bounding boxes for most of the salient objects well but sometimes fails in determining the object class (*e.g.,* classify an old man as a bear, treat a fence as a computer keyboard, etc.).

Besides, we adopt the default anchor generator from Faster R-CNN [7], which was primarily designed for images of larger sizes. As most labeled objects are relatively large, we decide to use such a setting to prioritize larger objects, which inevitably degrades the model's ability to detect smaller objects (*e.g.,* failed to detect the dog in the upper right image). However, such a trade-off choice still works as we've achieved the highest AP50 and the third-highest mAP among all the groups.

6 Conclusion

This work validates the effectiveness of MAE [3] pre-training for ViT [1] backbone networks, where the model can reconstruct the corrupted images by the residual semantic information. With the help of ViTDet [6] feature pyramid network and some effective fine-tuning tricks (*e.g.,* per-layer decayed learning rate, color jitter, iterative training, etc.), our model performs reasonably well on the given object detection dataset. As a last note, we may potentially get even better performance if we enlarge the number of epochs for both pre-training and fine-tuning.

Acknowledgements. We thank Prof. Yann LeCun, Prof. Alfredo Canziani, and Jiachen Zhu for the amazing lectures and guidance throughout the course.

References

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners, 2021. URL https://arxiv.org/abs/2111.06377.

[4] G. E. Hinton and R. Zemel. Autoencoders, minimum description length and helmholtz free energy. In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann, 1993. URL https://proceedings.neurips.cc/paper/1993/file/9e3cfc48eccf81a0d57663e129aef3cb-Paper.pdf.

[5] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. doi: 10.1162/neco.1989.1.4.541.

[6] Y. Li, H. Mao, R. Girshick, and K. He. Exploring plain vision transformer backbones for object detection, 2022. URL https://arxiv.org/abs/2203.16527.

[7] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

[8] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.