Masked AutoDecoder is Effective Multi-Task Vision Generalist

Supplementary Material

A. Implementation Details

Tab. 1 shows the default settings of our experiments. Most of the hyper-parameter setting and training strategies follow DETR [1]. We adopt the AdamW optimizer with a learning rate of 1e-4 for all the experiments. The backbone is fine-tuned with a smaller learning rate of 1e-5. We use scale augmentation for ResNet-50 and Swin-Base models. Specifically, the shortest side of the input image is resized to between 480 and 800 pixels and randomly cropped with a probability of 0.5. For the ViT-Base model, we use Large-Scale Jittering (LSJ) [5] with a fixed image size of 1024 following ViTDet [7] and Pix2SeqV2 [3]. The resizing range is set to [0.3, 2.0]. We essentially follow the architecture configurations of ViT-Base from ViTDet and EVA-02 [4] with the alternated windowed attention and global attention mechanisms and extract the feature map with a stride of 32 from its simple feature pyramid network. For training time, it takes about 48 hours to train MAD-Resnet50 on 4 A100 GPUs.

B. Autoregressive Decoding

We convert MAD into an autoregressive variant for comparison. It follows the same architecture as well as training settings as MAD with a few modifications on tokenization of task sequences, attention mechanism, and decoding process.

For tokenization, we adopt similar approaches as in MAD to construct task sequences for instance segmentation, keypoint detection, and image captioning, while following pix2seq [2] to build object detection sequences that the ground-truth objects are placed at the beginning of sequences. For all tasks, we add $< start >$ tokens at the start of the input sequences. During training, we add the $< end >$ tokens at the end of the original target sequences.

For the attention mechanism, the self-attention layer in the decoder is applied with a triangular causal mask for unidirectional attention.

At inference time, the task sequences are recursively generated, starting from the $< start >$ token, and generating up to the maximum length corresponding to each task (instead of stopping at the $< end >$ token). We adopt the $argmax$ sampling strategy and cache the KV features of previous generation steps in the self-attention layers for acceleration. Although some other complex sampling strategies, i.e., beam searching or nucleus sampling [6] may improve performance, these strategies would also further slow down the inference speed of autoregressive decoding.

Table 1. Experimental Settings.

(a) Model with ResNet-50.

config	value
epoch	50
optimizer	AdamW
learning rate	1e-4
learning rate scheduler	multi-step scheduler
learning rate drop epoch	40
weight decay	1e-4
batch size	16
image size	800 × 1333
image augmentation	MultiScaleResize

(b) Model with Swin-Base.

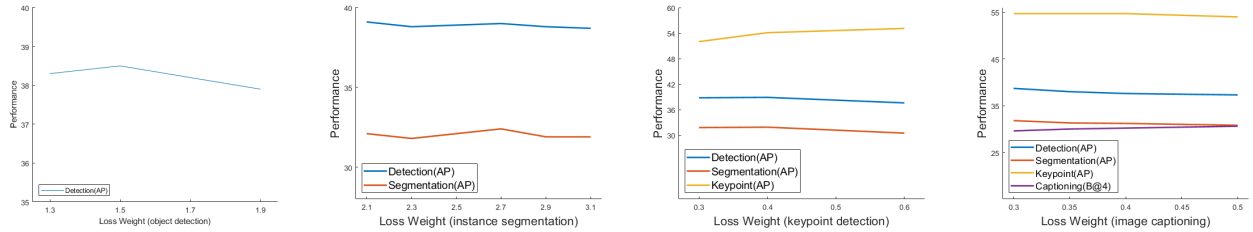
config	value
epoch	300
optimizer	AdamW
learning rate	1e-4
learning rate scheduler	multi-step scheduler
learning rate drop epoch	240
weight decay	1e-4
batch size	32
image size	800 × 1333
image augmentation	MultiScaleResize

(c) Model with ViT-Base.

config	value
epoch	100
optimizer	AdamW
learning rate	1e-4
learning rate scheduler	multi-step scheduler
learning rate drop epoch	80
weight decay	1e-4
batch size	32
image size	1024 × 1024
image augmentation	LargeScaleJitter

C. Task Weighting

In Fig. 1, we search for the appropriate loss weight for each task. We first evaluate object detection performance and obtain the optimal loss weight of 1.5. Then we introduce the instance segmentation. As Fig. 1b shows, both tasks perform well over a wide range of weights, with only small fluctuations. We thus simply take a weight of 2.7 for instance segmentation. For keypoint detection, it seems to conflict with the existing tasks, and increasing its weight would hinder the performance of object detection and instance segmentation. According to the trade-off of performance, the keypoint detection task is weighted by a factor of 0.5. Finally, we add the image captioning task, where we



(a) Instance segmentation Weights. (b) Instance segmentation Weights. (c) Keypoint detection weights. (d) Image captioning weights.

Figure 1. Performance with different loss weights by gradually adding new tasks to the existing tasks.

find that a weight of 0.3 is appropriate for preserving the performance of existing vision tasks.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with Transformers. In *ECCV*, 2020. 1

[2] Ting Chen, Saurabh Saxena, Lala Li, David J Fleet, and Geoffrey Hinton. Pix2seq: A language modeling framework for object detection. *arXiv preprint arXiv:2109.10852*, 2021. 1

[3] Ting Chen, Saurabh Saxena, Lala Li, Tsung-Yi Lin, David J Fleet, and Geoffrey E Hinton. A unified sequence interface for vision tasks. *Advances in Neural Information Processing Systems*, 35:31333–31346, 2022. 1

[4] Yuxin Fang, Quan Sun, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. Eva-02: A visual representation for neon genesis. *arXiv preprint arXiv:2303.11331*, 2023. 1

[5] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D Cubuk, Quoc V Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2918–2928, 2021. 1

[6] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019. 1

[7] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part IX*, pages 280–296. Springer, 2022. 1