A. Experimental Details

We follow the implementation details of the official MAE [28] for all pre-training and fine-tuning. Meantime, we follow the implementation of bootMAE [20] for settings of segmentation tasks. While MAE uses only ViT-B and ViT-L, we also use ViT-S [60].

For pre-training, we use the same effective batch size of 4096 as MAE with accum_iter, *e.g.*, 256 (batch_size_per_gpu) × 8 (GPUS) × 2 (accum_iter). The detail configurations is listed in Table 9. For end-to-end fine-tuning, we provide the configures in Table 10 for different backbones.

Table 9. Pre-training settings.

Config	Value
optimizer	AdamW
optimizer momentum	0.9, 0.95
weight decay	0.05
base learning rate	1.4e-4
learning rate schedule	cosine
warmup epochs	40
augumentation	RandomResizeCrop

Table 10. Fine-tuning settings.

Config	Value
optimizer	AdamW
optimizer momentum	0.9, 0.999
weight decay	0.05
layer-wise lr decay	0.75 (S), 0.65 (B, L)
base learning rate	5e-4
drop path	0.1 (S,B), 0.2(L)
batch size	1024
learning rate schedule	cosine
warmup epochs	5
training epochs	300(S), 100 (B,L)
label smoothing	0.1

B. Pytorch Code for Adversarial Training

We also provide the simplified pytorch implementation for adversarial training in GAN-MAE model, to illustrate the optimization process more expressly.

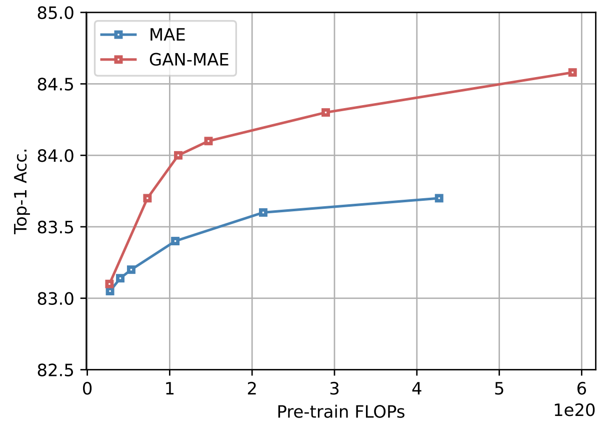

Figure 4. **Performance comparison given the same compute budget with ViT-B structure**. With the advancing of discriminator capacity, the performance gains consistently increase.

Listing 1: Example Pytorch Code

```
model = mae_gan_vit_base_patch16()

optimizer_gen = AdamW(generator_module.
    parameters(), lr=lr)
optimizer_disc = AdamW(discriminator_module.
    parameters(), lr=lr)

for raw_img in dataloader_pretrain:
    # generator training
    optimizer_gen.zero_grad()
    gen_loss, corrupt_img, mask = model.
        forward_with_generator(raw_img)
    gen_loss.backward()
    optimizer_gen.step()

    # discriminator training
    corrupt_img, mask = corrupt_img.detach(),
        mask.detach()
    optimizer_disc.zero_grad()
    disc_loss = model.
        forward_with_discriminator(
        corrupt_img, mask)
    disc_loss.backward()
    optimizer_disc.step()
```

C. Extra Experimental Results

Compute-matched Comparison. We also chose to measure compute usage in terms of floating point operations (FLOPs) because it is a measure agnostic to the particular hardware, low-level optimizations, etc. We plot the computation-performance curve in Figure 4, and we can observe that GAN-MAE outperforms the MAE persistently under the same computation budget in the downstream classification task.

Linear Probing Evaluation. We conduct the linear probing experiment to evaluate the semantic level of a representation, and the results are listed in Table 11. As can be seen, our method pre-trained on 800 epochs achieves a considerable gain (+4.0%) compared with MAE baselines, even on 1600 epochs.

Table 11. **Linear probing evaluation on ImageNet-1K.** We report the top-1 accuracy for classification with different ViT structures.

Method	Pre-train epochs	ViT-B	ViT-L
MAE	800	64.4	73.5
MAE	1600	67.8	75.1
GAN-MAE	800	**69.3**	**77.5**