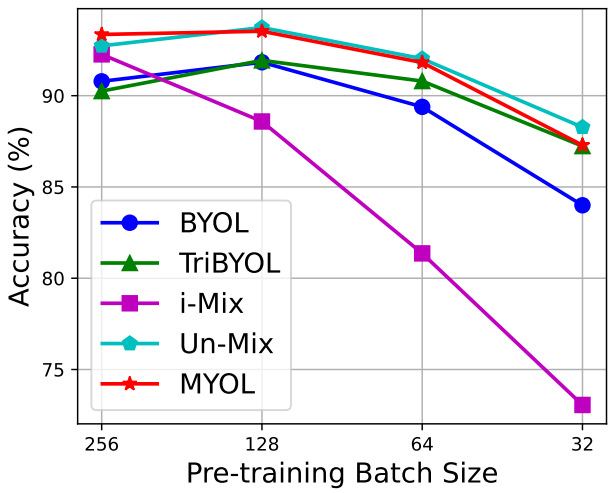
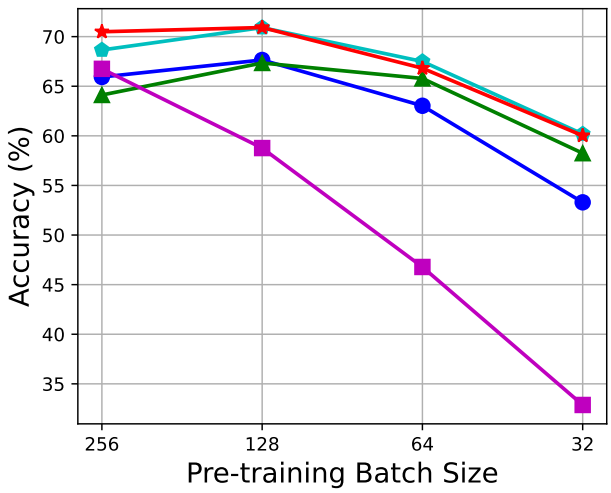
A Plots for the Entire Datasets

In this section, we show the plots for the entire datasets that were used in the experiments, as well as the plots for the Tiny-ImageNet dataset.

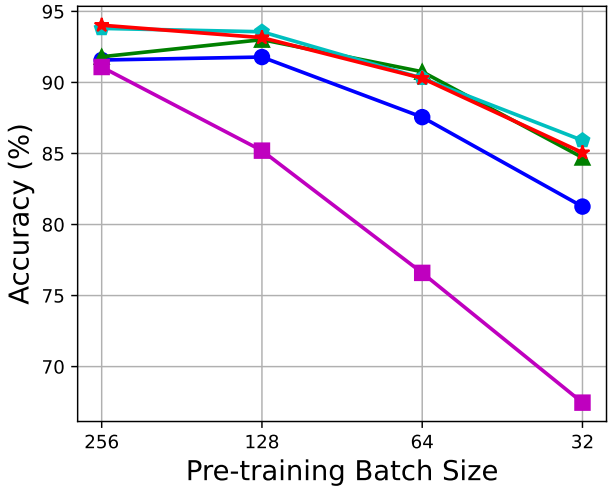
A.1 Pre-training Batch Sizes Comparison



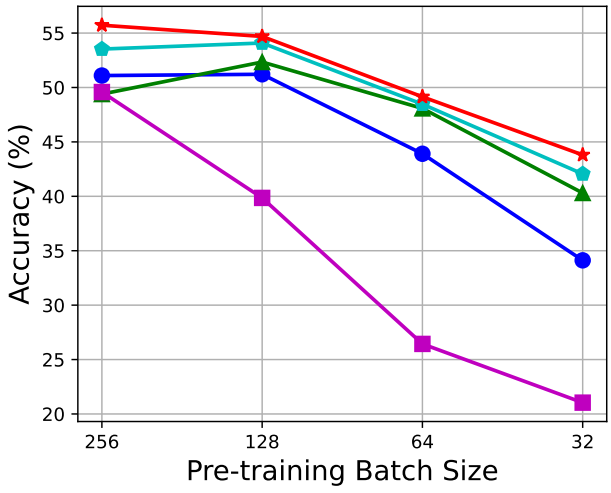
(a) CIFAR-10



(b) CIFAR-100



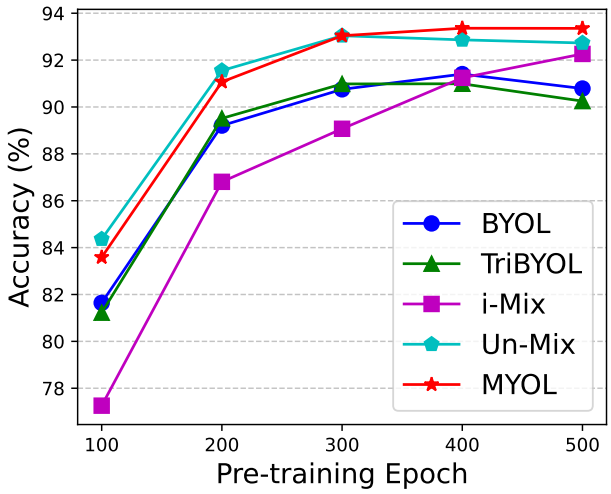
(c) STL-10



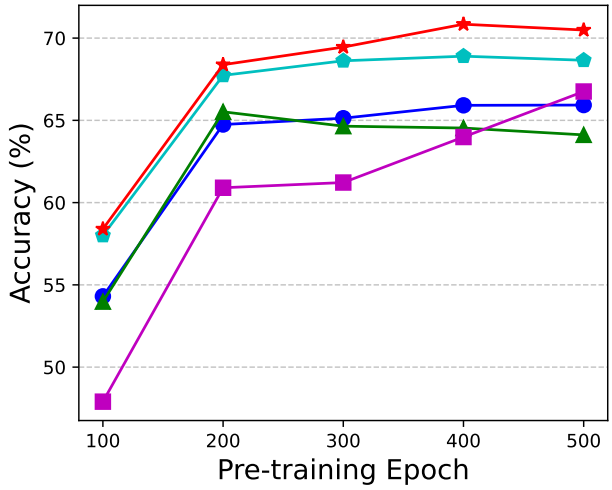
(d) Tiny-ImageNet

Figure 1: Comparison of linear evaluation performance according to the pre-training batch sizes in terms of classification accuracy (%).

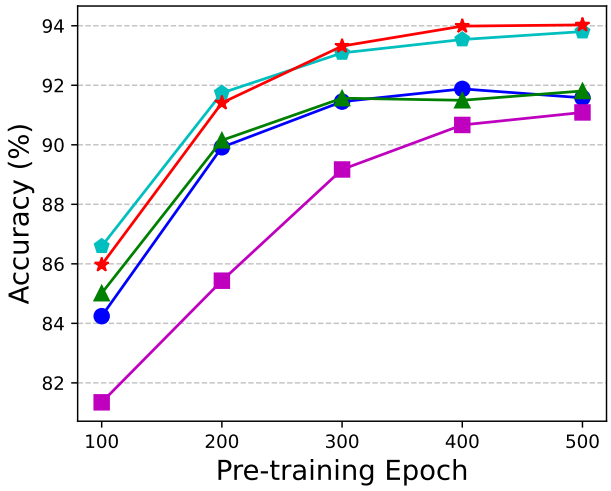
A.2 Pre-training Epochs Comparison



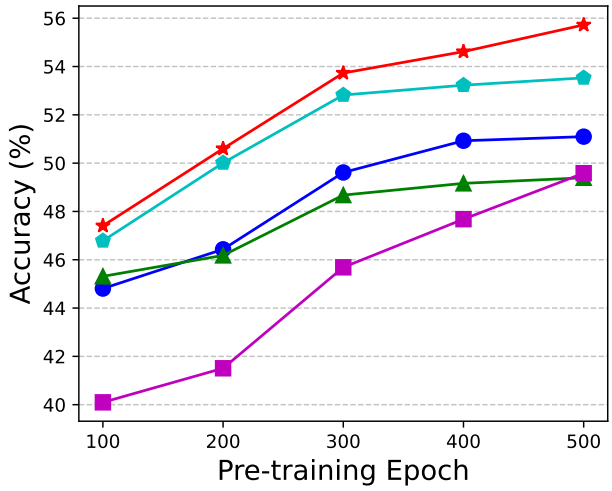
(a) CIFAR-10



(b) CIFAR-100



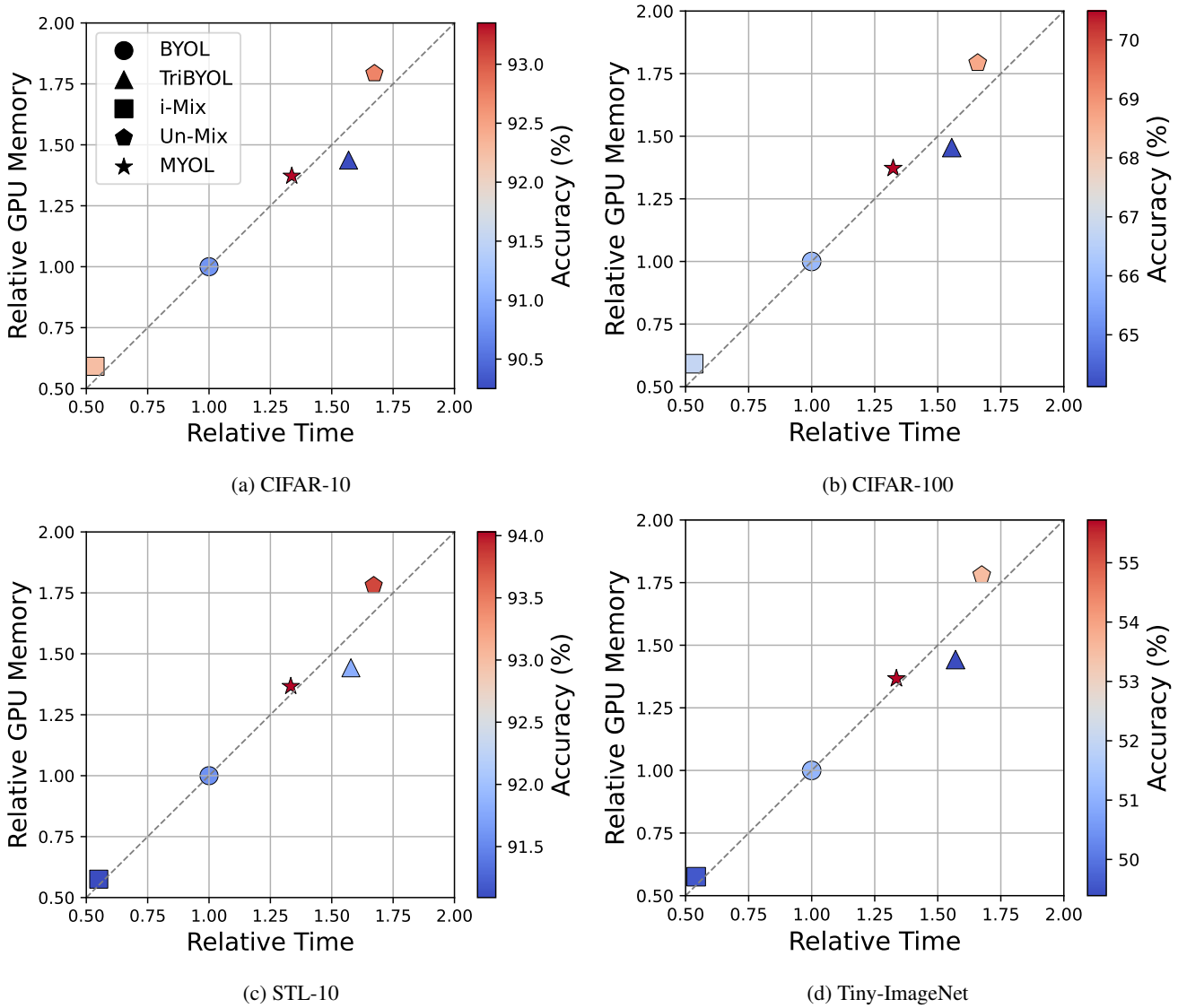
(c) STL-10



(d) Tiny-ImageNet

Figure 2: Comparison of linear evaluation performance according to the number of pre-training epochs in terms of classification accuracy (%).

A.3 Time and GPU Memory Comparison



(a) CIFAR-10

(b) CIFAR-100

(c) STL-10

(d) Tiny-ImageNet

Figure 3: Comparison of time and GPU memory required for pre-training on a relative scale, with marker color indicating classification accuracy.

B PyTorch-like Pseudocode

Algorithm 1: PyTorch-like pseudocode for MYOL loss computation

Input: Input views `v1, v2`

```
p1, p2 = pred(encoder(v1)), pred(encoder(v2)) # online network
z1, z2 = encoder_t(v1), encoder_t(v2) # target network
# input mixup
lam = Beta(alpha, alpha).sample()
randidx = randperm(len(v1))
mixed_v = lam * v1 + (1-lam) * v2[randidx]

# norm_mse: normalized mean squared error
loss_byol = (norm_mse(p1, z2) + norm_mse(p2, z1)) / 2

# projection mixup
mixed_z = lam * z1 + (1-lam) * z2[randidx]
loss_myol = norm_mse(pred(encoder(mixed_v)), mized_z)

loss = loss_byol + loss_myol
```