Bootstrap-MAE Coding Report

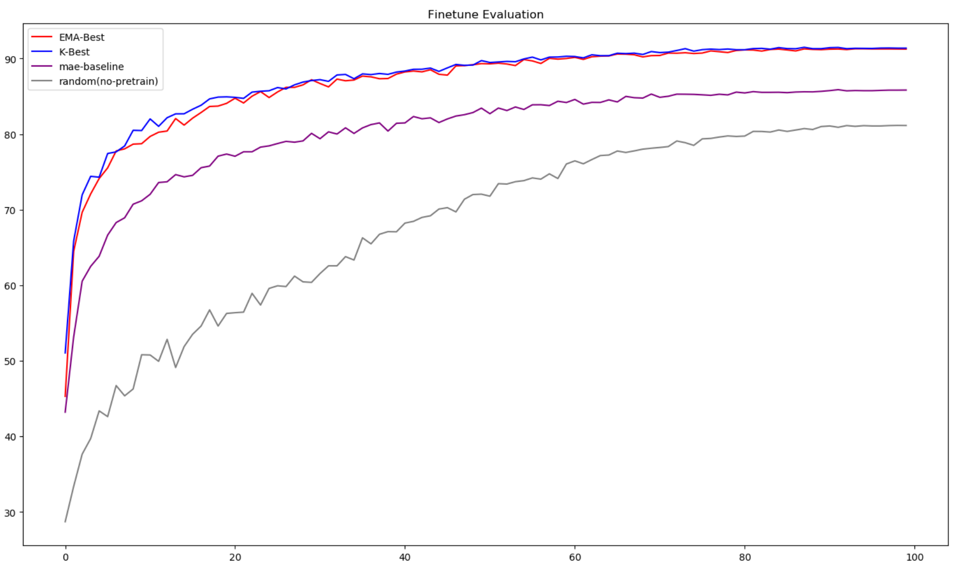
Guanghe Li

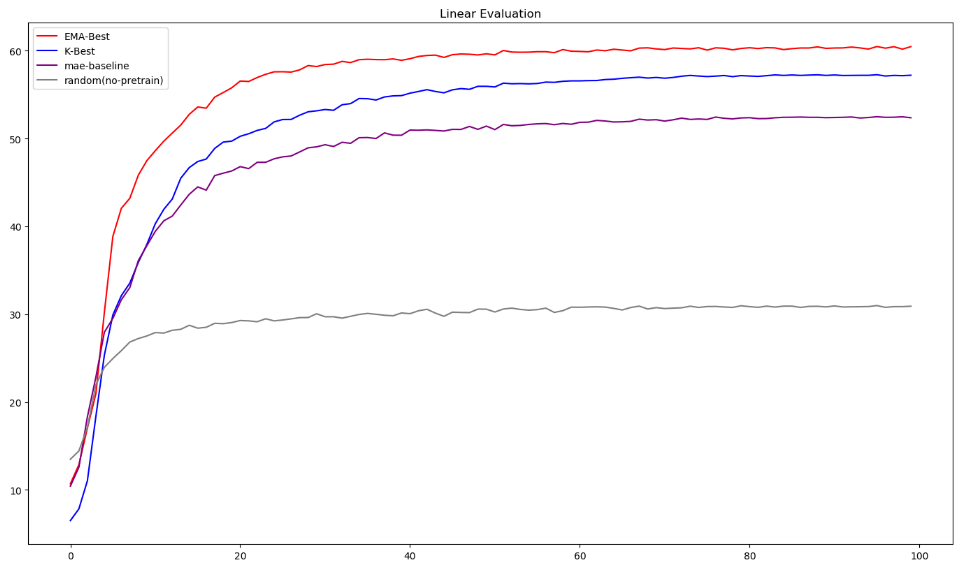
Abstract

In this project, we aim to improve the performance of MAE on downstream tasks(classification tasks) by changing the reconstruction target from original pixels to features extracted by a pretrained MAE encoder. The main contributions and findings are outlined as follows:

- We have successfully developed and implemented the MAE-K algorithm. In the MAE-K algorithm, the target encoder undergoes adjustments every K epochs to align with the training parameters of MAE, and the training MAE uses the output of the target encoder as its reconstruction target. Evaluation results demonstrate that MAE-K surpasses the original MAE model in performance on downstream classification tasks.
- We have also successfully implemented the EMA-MAE algorithm. In the EMA-MAE algorithm, the target network is updated using the exponential moving average of the training MAE. Experimental results show that EMA-MAE also outperforms the MAE baseline on downstream classification tasks. More than that, EMA-MAE achieves comparable results with MAE-K on fine-tune evaluation and outperforms MAE-K on linear evaluation.

Overall experimental results are shown below:





Implementation details, experiments and further analysis are in the following sections.

1 Environment Setup

Following the requirements, the codebase is modified directly from MAE's official codebase. The running environment remains consistent with MAE's official setup, with only two minor adjustments.

- Firstly, the version of the `timm` package in the original codebase, 0.3.2, was slightly outdated and conflicted with our PyTorch version. To resolve this, we installed `timm==0.4.12` and removed version checks from the original codebase.
- Secondly, we encountered a minor issue when utilizing the SummaryWriter from TensorBoard. Thus, we did a modification in file `lib\site-packages\torch\utils\tensorboard__init__.py` and removed seven lines of code. The removed codes are shown below:

```
# file: lib/site-packages/torch/utils/tensorboard__init__.py
import tensorboard
from setuptools import distutils

#LooseVersion = distutils.version.LooseVersion
#if not hasattr(tensorboard, '__version__') or LooseVersion(tensorboard.__version__) < LooseVersion('1.15'):
#     raise ImportError('TensorBoard logging requires TensorBoard version 1.15 or above')
#del distutils
#del LooseVersion
#del tensorboard

from .writer import FileWriter, SummaryWriter  # noqa: F401
from tensorboard.summary.writer.record_writer import RecordWriter  # noqa: F401
```

2 Baseline Parameter Configuration

The official MAE codebase is well-implemented, so we largely maintained the default parameters, making only a few modifications.

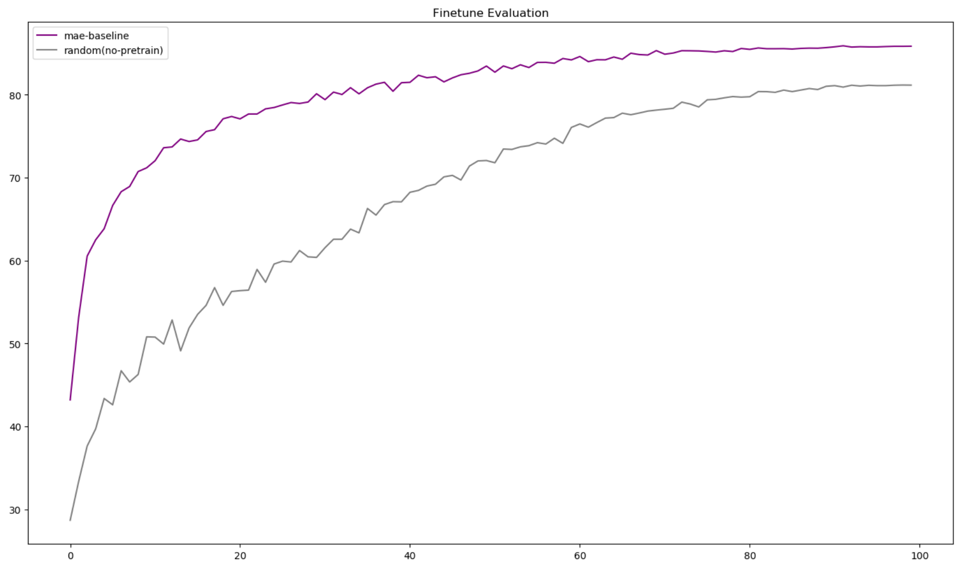
- Number of GPUs: All experiments were conducted using a single RTX Titan GPU.
- Batch size: We opted for a batch size of 512, as we observed that larger batch sizes caused burden to the CPU.
- Training epochs: In accordance with specifications, pretraining takes 200 epochs, while finetuning takes 100 epochs.
- Network architecture: In accordance with specifications, we employed the architecture configurations of Deit-Tiny.
- Dataset: We use CIFAR10 dataset. Input images are standarlized by the dataset mean and std values.

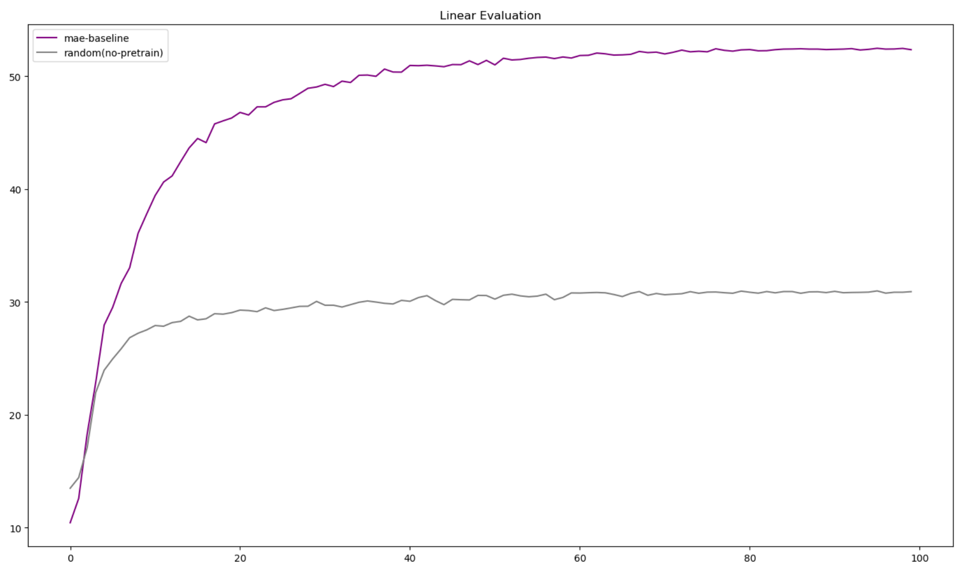
3 MAE Baseline

Following the parameter configurations above, we pretrain an MAE baseline model on CIFAR10 dataset for 200 epochs, and then finetune it for 100 epochs to classify images.

We compare the MAE baseline with a randomly initialized network to show that unsupervised pretraining benefits downstream classification task.

The finetune and linear evaluations are shown below:



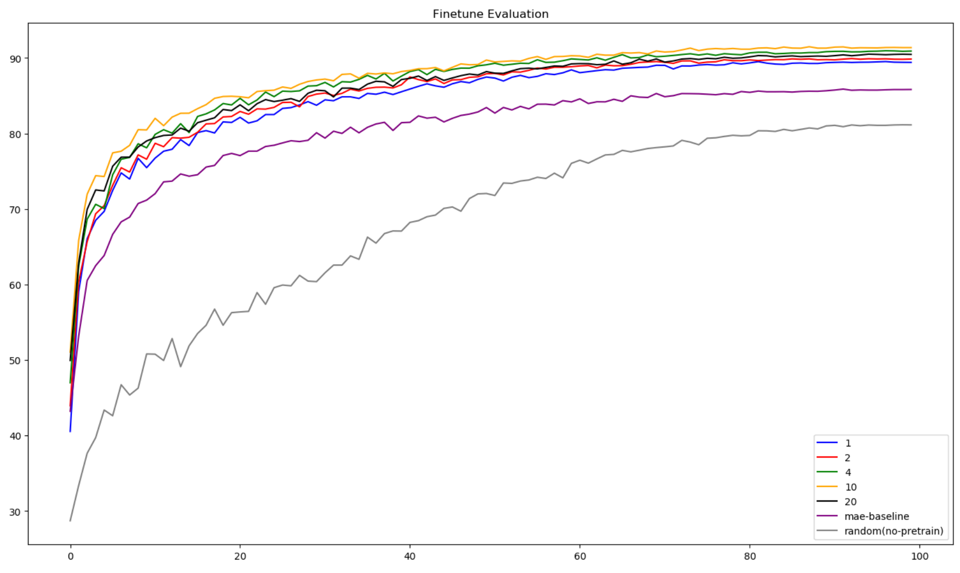


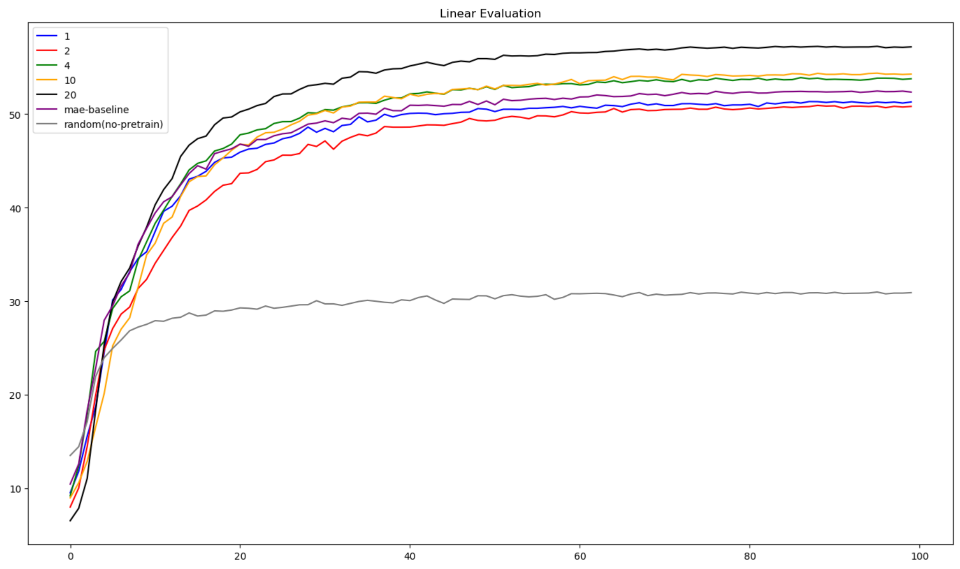
As shown above, pretraining an MAE model is able to boost downstream classification tasks by a huge margin, which shows the effectiveness of MAE baseline.

4 MAE-K

In MAE-K, we update the target network every K epochs in alignment with the parameter of the training network. As shown below, The selection of the hyperparameter K significantly impacts the final performance.

In fine-tuning evaluation, MAE-K achieves optimal results when K is set to 10, whereas during linear evaluation, the optimal performance is observed when K is set to 20.





Across fine-tuning evaluations, all versions of MAE-K outperform the MAE baseline. However, during linear evaluation, only MAE-K configurations with larger K values (4, 10, 20) surpass the baseline, while those with smaller K values (1, 2) underperform compared to the baseline algorithm. We hypothesize that during linear evaluation, since the weights of the pretrained encoder remain fixed and unaltered, they need to converge in order to provide valuable features for the classification head. Consequently, a larger value of K would facilitate the trained model's convergence towards a target model, whereas a smaller K could lead to oscillations in the reconstruction objective, hindering the convergence of the trained network

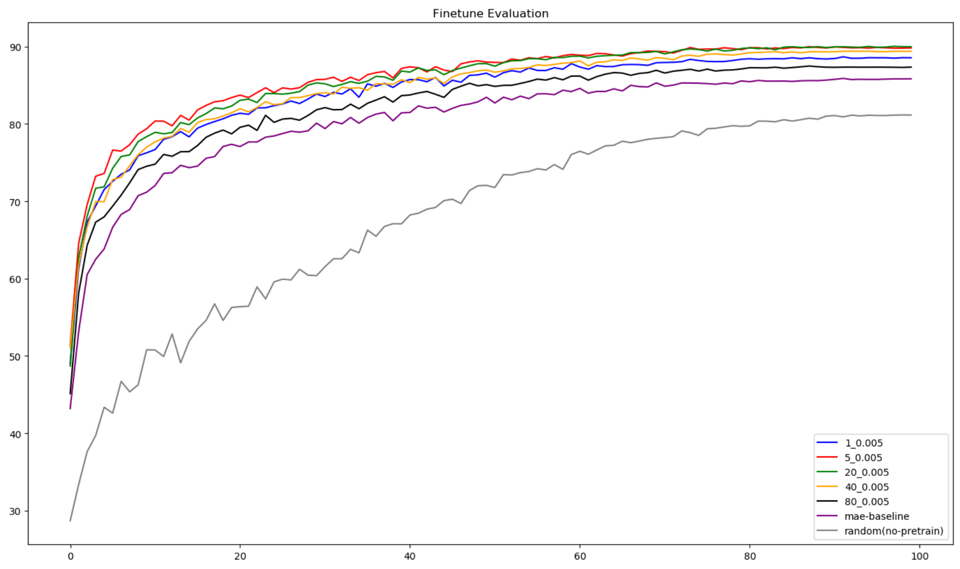
5 EMA-MAE

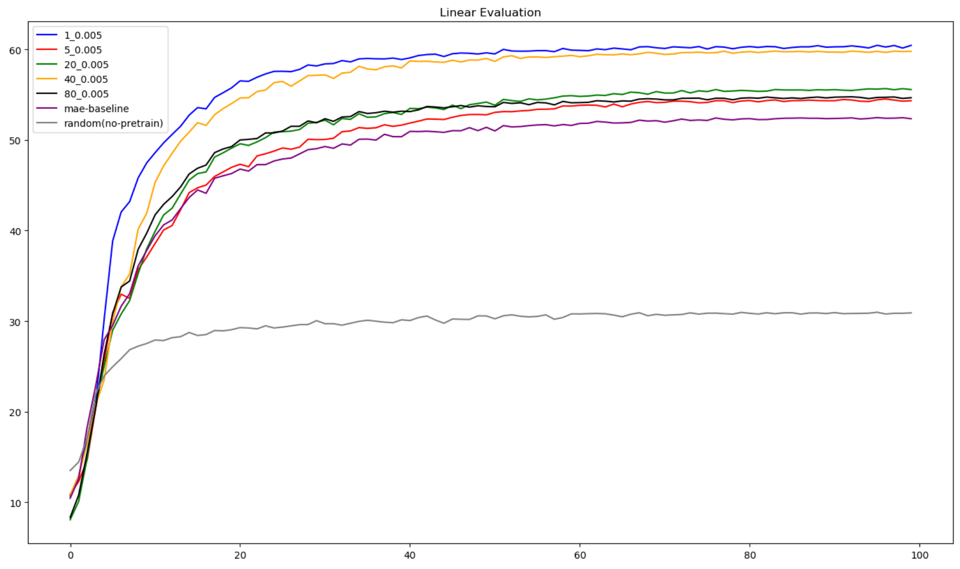
In exponential moving average(EMA) version of MAE, the target encoder is updated every epoch to equal the exponential moving average of trained MAE.

In EMA-MAE, there exists two key hyperparameters: Warmup starting epochs(WSE) and Exponential moving average(ema) coefficient τ. Their detailed analysis are shown below:

Warmup starting epochs (WSE): Initially, the EMA-MAE model is trained to reconstruct pixels for WSE number of epochs, which is consistent with the original training objective of MAE. This initial phase is termed as "warmup-stage". Subsequently, following the warm-up stage, the EMA-MAE model is trained to bootstrap on the output of target encoder(exponential moving average of training EMA-MAE model)

Determining the appropriate value for Warmup Starting Epochs (WSE) is crucial. The features generated by the target encoder becomes valuable only when the target MAE is capable of predicting masked pixels, i.e. understands the distribution of dataset. Therefore, rather than immediately bootstrapping on target encoder's output from beginning of training, it is important for the target network to initially acquire the capability to recover masked pixels. The parameter analysis below demonstrates that, when the EMA coefficient τ remains fixed as 0.005, WSE values of 5 and 20 yield optimal performance, underscoring the necessity of the warm-up phase.



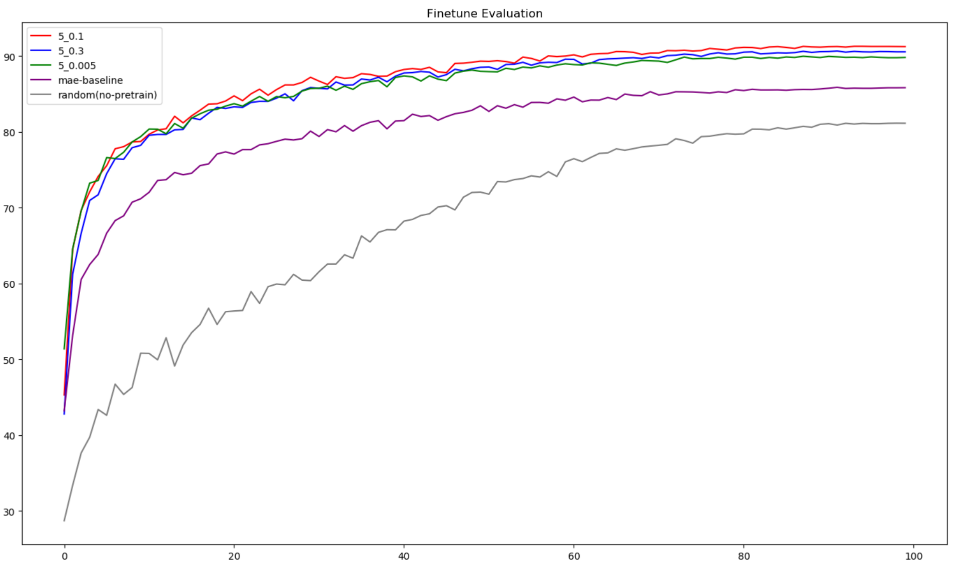


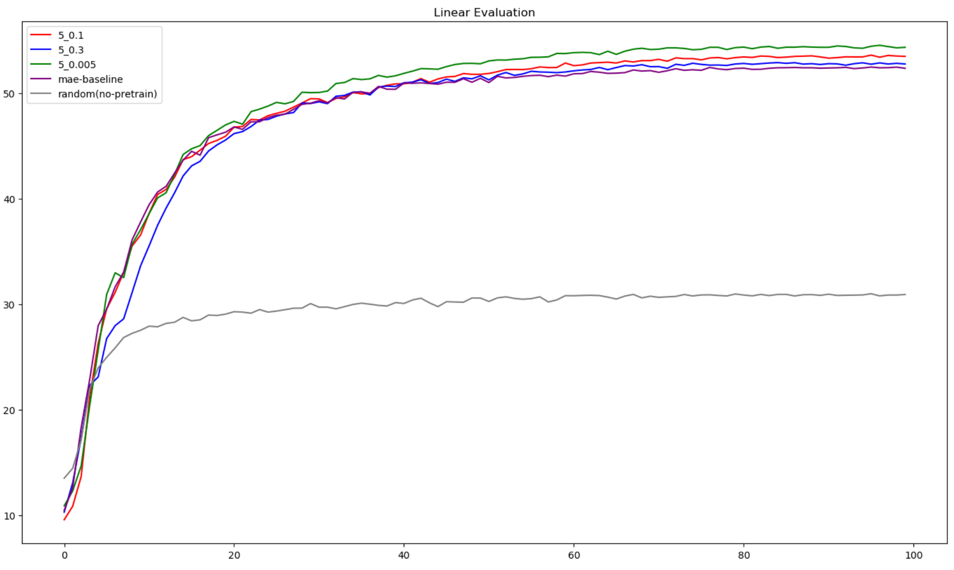
Exponential moving average coefficient τ: The target encoder is updated according to the following equation:

$$\mathrm{MAE}_{\mathrm{target}} = \mathrm{MAE}_{\mathrm{target}} \times (1-\tau) + \mathrm{MAE}_{\mathrm{train}} \times \tau$$

This equation is applied to the target encoder for each epoch after the warmup-stage. The approach of using EMA to construct a target network in this work is very similar to the utilization of target Q-network in reinforcement learning. Following that, we opt the default value of 0.005 as in Q-learning. This choice facilitates gradual updates to the target network, thereby mitigating abrupt alterations.

We conducted additional experiments to show how the value of τ effects the final performance. As illustrated below, during finetune evaluation, τ values of 0.1 and 0.3 consistently outperform the performance achieved with a τ value of 0.005. However, in linear evaluation, a τ value of 0.005 significantly surpasses the performance of the other values.


Finetune Evaluation
5_0.1
5_0.3
5_0.005
mae-baseline
random(no-pretrain)
30
40
50
60
70
80
90
0
20
40
60
80
100


Linear Evaluation
5_0.1
5_0.3
5_0.005
mae-baseline
random(no-pretrain)
10
20
30
40
50
0
20
40
60
80
100

Conclusion