NYCU Introduction to Machine Learning, Final Project

110550089, 周冠辰

1. How to reproduce my kaggle submission

(1) First download the finetuned_model.zip from google drive. The link to google drive is in 110550089_weight.txt .

(2) Unzip the finetuned_model.zip with the folder structure:

```
- finetuned_model/
    - model_beit/…
    - model_lora/…
```

(3) Put the model folder and data folder to my submit folder. The necessary folder structure should be like:

```
- 110550089_final/
   - 110550089_inference.py
   - data/
      - test/…
   - finetuned_model/
      - model_beit/…
      - model_lora/…
   - other files and folders
```

(4) Create the virtual environment with python=3.9

(5) Install the packages with requirements.txt

(If error happen when installing PyTorch package, remove the following line in requirements.txt. Then directly download the PyTorch 2.1.1+cu118 in their official website)

```
--find-links https://download.pytorch.org/whl/torch_stable.html
torch==2.1.1+cu118
torchaudio==2.1.1+cu118
torchvision==0.16.1+cu118
```

(6) Run inference.py, the predictions would be in predictions.csv.

I have successfully reproduced my prediction on both Windows11 and Ubuntu22.04 with the latest cuda version, but I'm not sure how it works with old version of cuda.

2. Environment details

(1) Python version: 3.9

(2) Framework: PyTorch 2.1.1+cu118

(3) Hardware: This project is training on single NVIDIA GeForce RTX3060 with 12GB VRAM

3. Implementation details

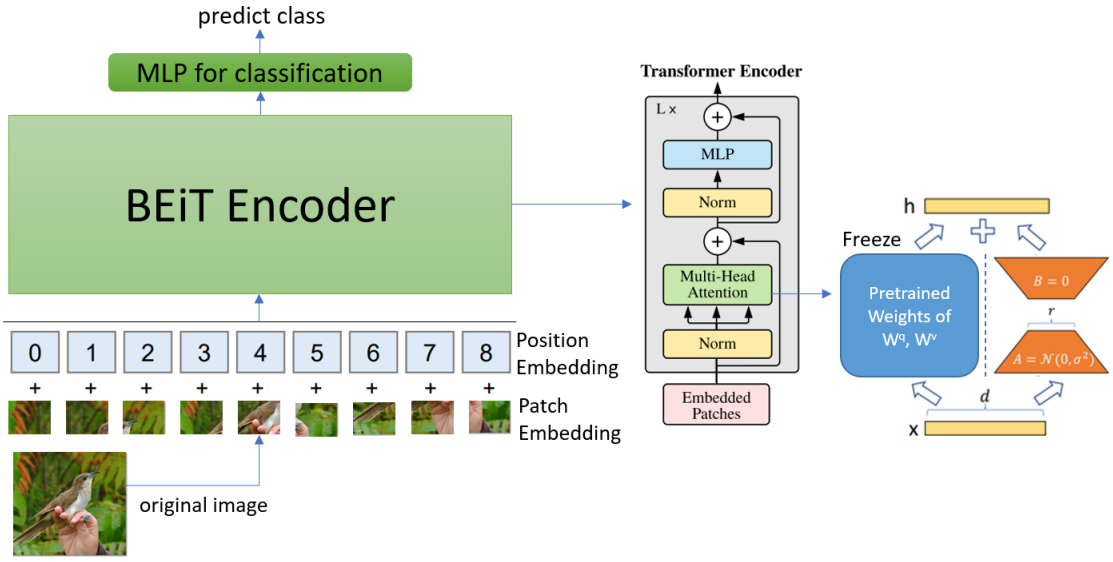


Figure 1: Overview of the architecture I used. The left part is BEiT transformer for image classification, the middle part is the encoder of BEiT and the right part is the LoRA module adding to original model.

(This image is referenced and modified from the original paper of BEiT, ViT and LoRA [1, 3, 2].)

(1) Model Architecture:

I choose BEiT (BERT Pre-Training of Image Transformers) [1] as my architecture backbone with parameter efficient fine-tuning method LoRA [2] by adding additional trainable module into original model. The overview of the architecture I used is showed in Figure 1.

The architecture of BEiT is quite similar to other transformers in computer vision domain such as ViT (Vision Transformer) [3]. The main difference between BEiT and ViT is that BEiT is pretraining with a self-supervise learning method which is conceptually inspired by language model BERT [4]. The pretraining method detail can be seen in the original paper. In image classification work, we first segment the image into patches, then use the encoder part of the transformer model to encode the image patches into feature-space embedding and adding addition classification layers at the end of the

model to classify the image with the embedding. The BEiT model I use is pre-trained in a self-supervised fashion on ImageNet-21k and further fine-tuned on ImageNet-1k which is provided by Microsoft on HuggingFace, which is a large-scale model with input image size at resolution 384x384.

As fine-tuning such a large transformer model is quite computational expensive, I adapt the parameter-efficient fine-tuning (PEFT) methods LoRA (Low-Rank Adaptation of Large Language Models) to the BEiT model. LoRA is first developed for fine-tuning large language model but currently used in computer vision works such as tuning a personal stable diffusion model. The implementation of LoRA is to freeze the original pretrained weights of transformer model and inject additional trainable module with less trainable parameters, which is usually used in the weights of query and key in the attention layer. The implementation detail and concept can be seen in the original paper. Though I didn't find any image classification work combining BEiT and LoRA, I think it is a great attempt and it also performed well. With LoRA, I can significantly reduce the usage of ram and speed up the training time without dropping the accuracy.

(2) Hyperparameter:

Learning rate: 5e-5
Batch size: 8
Accumulation steps: 8
Epochs: 25
Optimizer: AdamW
Scheduler: linear scheduler
Warmup ratio: 0.05
(With batch size 8 and accumulation steps 8, the model will perform one backpropagation per 8*8=64 images.)

(3) Training Strategy:

I choose AdamW as the optimizer and cross-entropy loss as loss function. AdamW is widely used in training a transformer and cross-entropy loss is the most used loss function in image classification.

I use some data augmentation tricks such as random resize cropping and random horizontal flip to prevent overfitting. Also, I use a linear scheduler with warm up steps which the former can avoid oscillations and prevents overfitting, the latter can enhance the stability and prevents the early divergence. The dynamic learning rate is showed in the Figure 2.

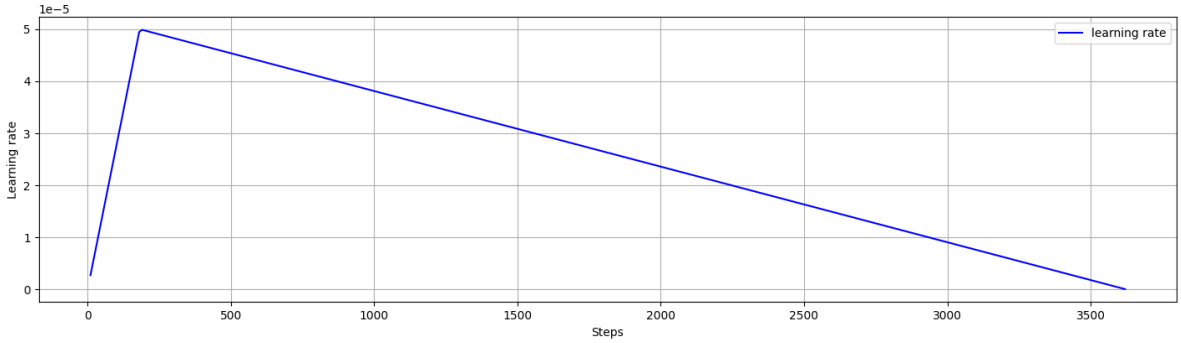


Figure 2. The curve of dynamic learning rate. With warm up steps, the learning rate will firstly increase to the threshold and slowing decreased by linear scheduler.

4. Experimental results

(1) Evaluation Metrices

I use accuracy as the evaluation metric and I split the provided dataset with ground truth into 95% for training and 5% for evaluation. In each epoch, I perform evaluation once. The accuracy curve is showed in Figure 3. It is showed that the model can finally achieve higher than 0.94 accuracy on evaluation dataset. And can achieve 0.915 on public score with test dataset. The reason of a higher score on evaluation dataset I think is that I use less data for evaluating so the class distribution might not be evenly spilt.

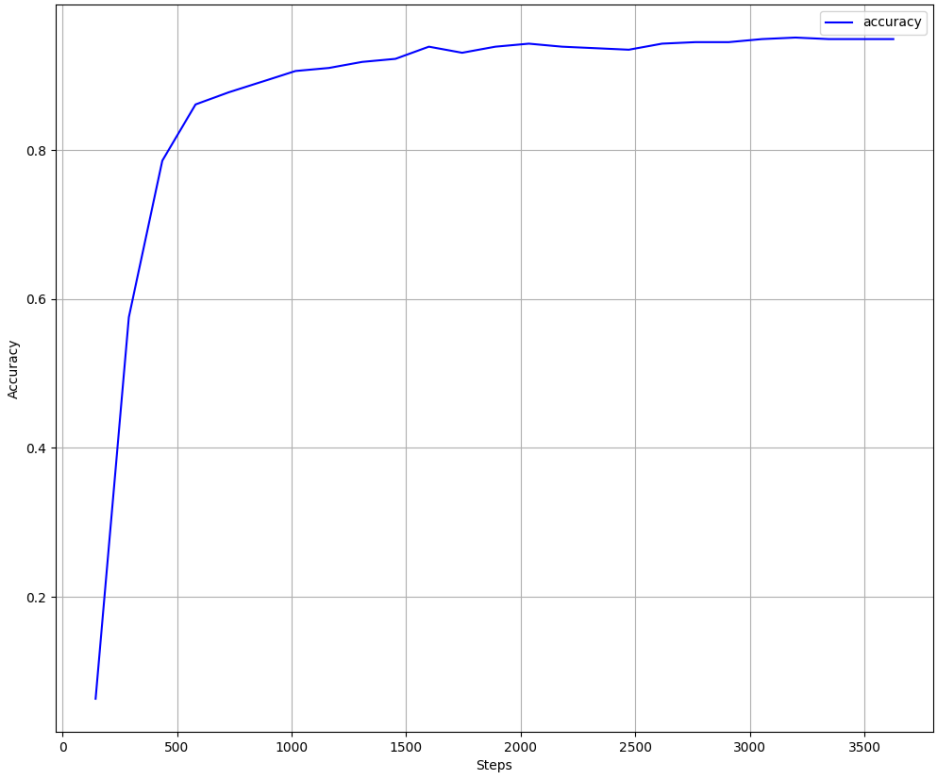
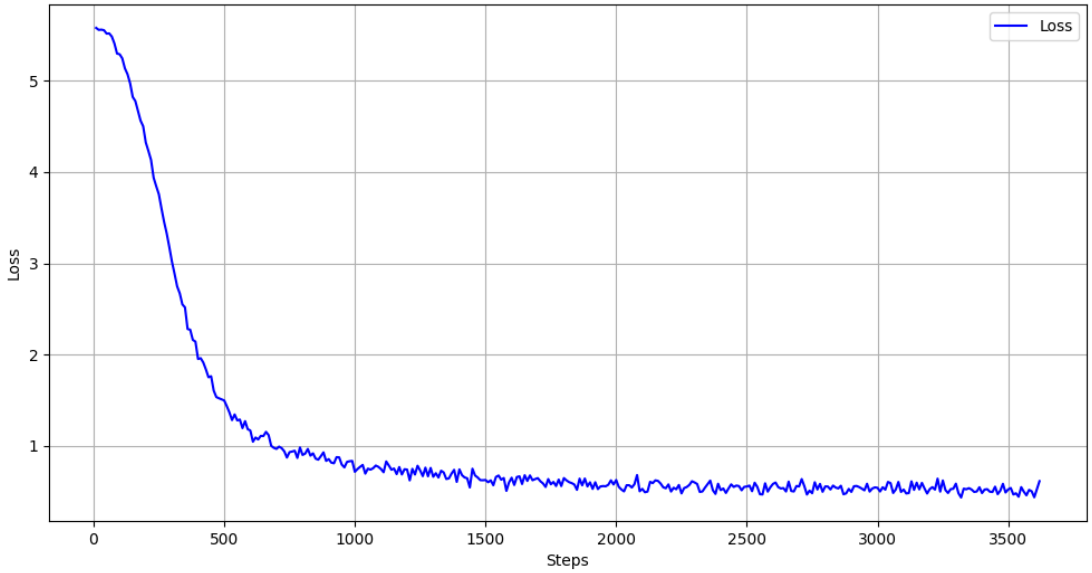


Figure 3. The accuracy curve of evaluation dataset while training.

(2) Learning Curve

The curve showing below implied that the model converged well.



(3) Ablation Study

(a) Model Decision:

I have try tuning different transformer for image classification. I use public score of leader board as the evaluation metric, which is the accuracy of 1000 images in test dataset. It is showed that the larger models always have a better performance. Also, the models with input images in higher resolution can enhance the model performance. With four different transformers for image classification, BEiT get the highest score, so I use it at my model backbone.

Model	Public Score
ViT-base-224	0.865
ViT-base-384	0.889
Swin-base-224	0.882
Swin-large-384	0.909
Swinv2-base-192	0.880
Swinv2-large-384	0.904
BEiT-large-384	0.911

(b) BEiT with LoRA:

I make a comparison between BEiT with and without LoRA module. I use a BEiT-large-384 with the same batch size 2. (Using a larger batch size can exceed the ram that RTX3060 12GB has. But I think using a larger batch size can has a more significant improvement on both ram usage and training time, small batch

size might be limited by model overhead.) It is showed that though LoRA can not really enhance the performance, it can significantly reduce the ram usage and speed up the training process. The reason is quite simple, with LoRA module, the optimizer is no need to calculate and store the gradient information of the freeze weights, especially with AdamW which need more ram to store momentum and other information. Also, LoRA module is easy to train due to less trainable parameter.

Model	Public Score	Ram Usage	Training Time	Trainable parameter
BEiT w/o LoRA	0.911	8.2GB	14h44m	304178888 (100%)
BEiT w/ LoRA	0.915	3.8GB	12h03m	1777864 (0.58%)

(c) Data Augmentation:

I make a comparison between training with and without data augmentation. It is obvious that training with data augmentation can enhance the overall accuracy.

Model	Public Score
BEiT-LoRA w/o DA	0.909
BEiT-LoRA w/ DA	0.915

Reference

[1]. Hangbo Bao, Li Dong, Songhao Piao, Furu Wei. BEIT: BERT Pre-Training of Image Transformers.

[2] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby. AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

[5] Jun Yu, Hao Chang, Keda Lu, Guochen Xie, Liwen Zhang, Zhongpeng Cai, Shenshen Du, Zhihong Wei, Zepeng Liu. Bag of Tricks and a Strong Baseline for FGVC.