NYCU Introduction to Machine Learning, Final Report

110550168, 賴御安

Environment details (5%)

1. Python version

   Python 3.8.18

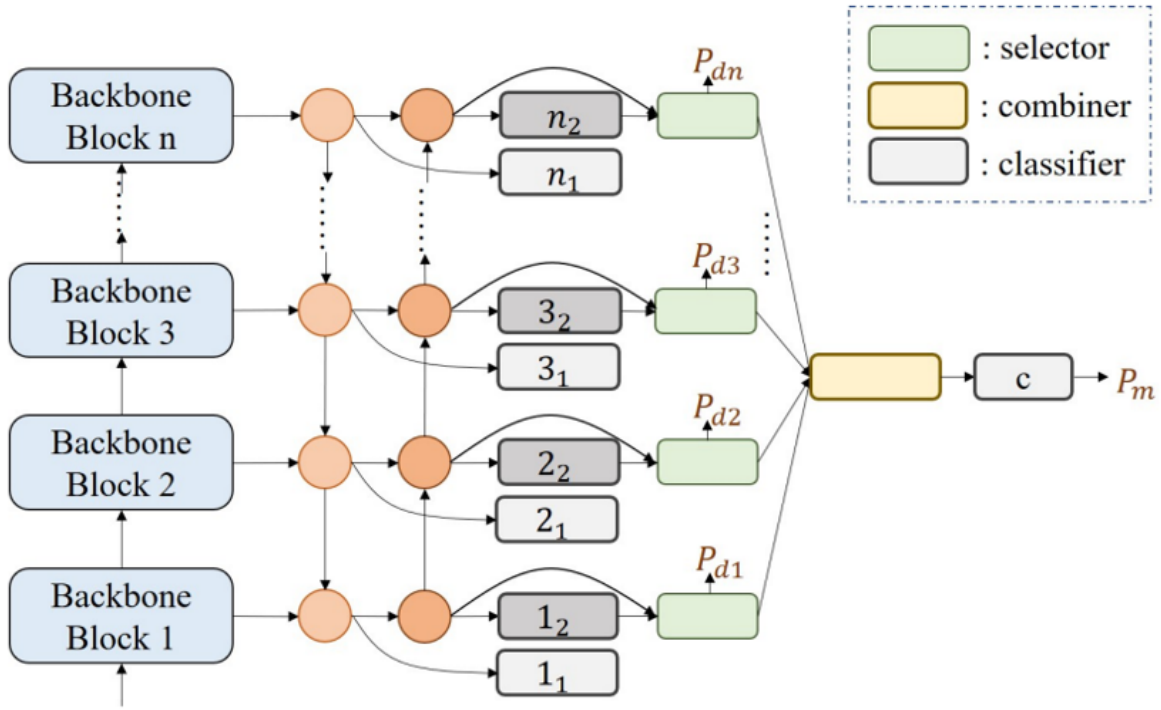
2. Framework
   a. torch==2.1.2
   b. Testing data needs to be put under ./training/final_training_data/test.
   c. The weight needs to be put in the same directory of inference.
3. Hardware

   CPU : AMD EPYC 7302 16-Core Processor

   GPU : NVIDIA RTX A5000

Implementation details (15%)

1. Model architecture



For my backbone blocks, I use a kind of Transformer-based model, Swin Transformer. By this, we can build hierarchical feature maps by merging image patches in deeper layers and has linear computation complexity to the input size.

Then I pass through the top-down and bottom-up features fusion module, which is just like FPN (feature pyramid network). This can allow us to take advantage of both high and low level feature maps.

After these, the Background Suppression module and the high-temperature refinement module will follow. In Background Suppression, we will generate classification maps and calculate the maximum score map by the maximum of each Softmax. And the

high-temperature refinement module is to let the top-down classifier cover more areas and the bottom-up classifier focus on specific areas. Then, selectors and combiners are used to merge the result and get the final prediction.

2. Hyperparameters

```yaml
project_name: FGVC-HERBS
notes: 'no inverse'
exp_name: basline
use_wandb: False
wandb_entity: poyung
train_root: ./final_training_data/train/
val_root: ./final_training_data/valid/
data_size: 384
num_workers: 2
batch_size: 16
model_name: swin-t
pretrained: ~
optimizer: SGD
max_lr: 0.0005
wdecay: 0.0003
max_epochs: 30 # 10 for ablation study
warmup_batchs: 1500
use_amp: True
use_fpn: True
fpn_size: 1536
use_selection: True
num_classes: 200
num_selects:
  layer1: 256
  layer2: 128
  layer3: 64
  layer4: 32
use_combiner: True # false for not using combine
lambda_b0: 1.375
lambda_b: 0.3
lambda_s: 0.0
lambda_n: 5.0
lambda_c: 1.0
update_freq: 4
log_freq: 100
eval_freq: 10
temperature: 64
```

3. Training strategy

I first train the model without using valid data and test for different epochs. Then I find out that not using valid data may lead to some error, which is that my model isn't tracing the best weight, so that I can only get the last weight I trained. This leads to unstable accuracy. So, I randomly pick 3 pictures from each class in train data and make a valid dataset.

Experimental results (15%)

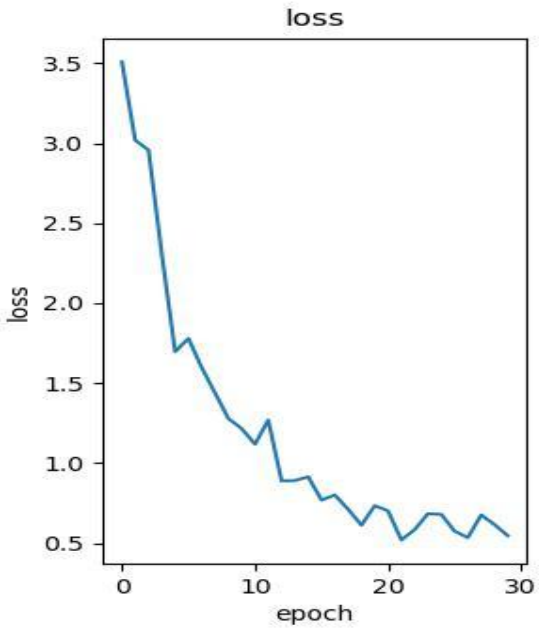
1. Evaluation metrics

```
""" calculate accuracy """
# total_samples = len(test_loader.dataset)

best_top1 = 0.0
best_top1_name = ""
eval_acces = {}
for name in corrects:
    acc = corrects[name] / total_samples[name]
    acc = round(100 * acc, 3)
    eval_acces[name] = acc
    ### only compare top-1 accuracy
    if "top-1" in name or "highest" in name:
        if acc >= best_top1:
            best_top1 = acc
            best_top1_name = name
```

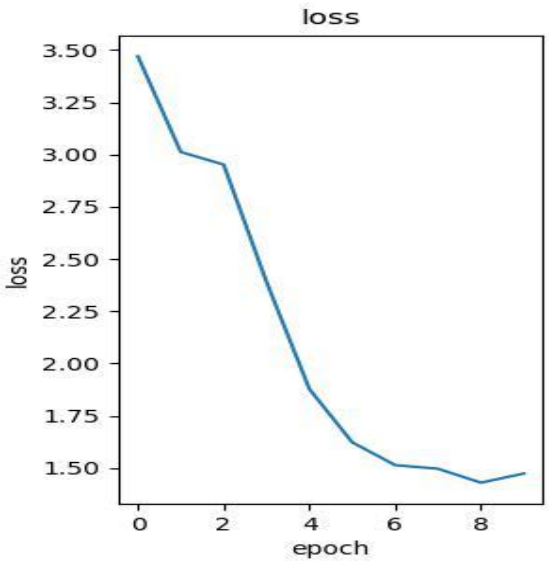
By evaluation metrics, I use the basic one, which is calculating the total correct prediction and dividing by the total samples, and I will show the accuracy when getting better accuracy, like the picture below.

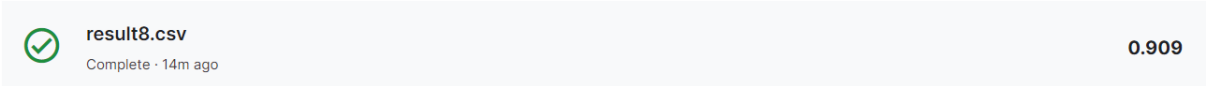
2. Learning curve



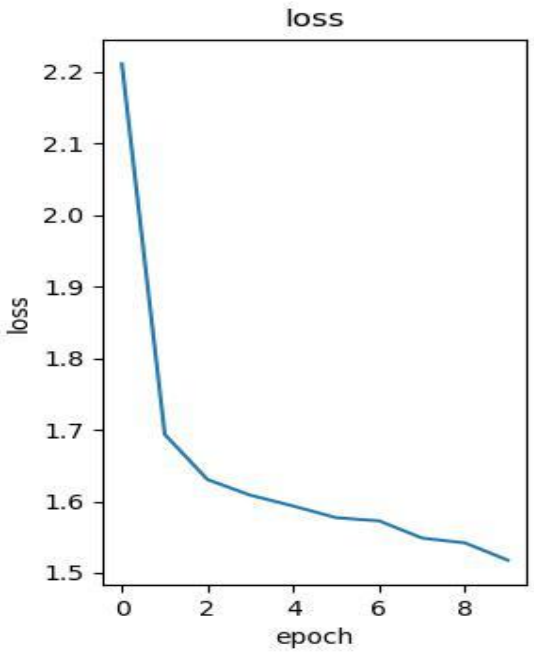
3. Ablation Study

For this part, I use a smaller epoch to test whether having a combiner matters the training accuracy. In this test, I use 10 epochs. And the following picture is the original loss curve and the accuracy on the kaggle.





And the following graph is without using a combiner.



result9.csv
Complete · 4h ago
0.334

We can see that the loss curve doesn’t change a lot, but the prediction of not using a combiner matters when we train less time.

Bonus (5%)

I think that my ablation study isn’t fully done, since I didn't try to raise my epochs because of the large time cost to test for different epochs. By the paper, I think that if we raise the epoch, the result will be similar to their PA, thus the accuracy of which is slightly less than the full model.