Introduction to Machine Learning -Final Project

109350008, 張詠哲

The end of report has inference note. TAs can read it before execute inference.py

- **Environment detail**
  - Python version: 3.10.12
  - Framework: pytorch 2.1.2
  - Hardward: GTX 1660ti, cuda 12.1

- **Implementation details**

  - **Model architecture**

I use PIM (Plug-in module, https://github.com/chou141253/fgvc-pim) as my model architecture.

This model (Plug-in Module, PIM) operates by taking the input image through the network. At each block (each backbone), the resulting feature map undergoes a screening process with the Weakly Supervised Selector, which identifies regions exhibiting strong discrimination or those less relevant to classification. Subsequently, a Combiner integrates the features of the selected regions to generate the final prediction outcomes. Following is the detailed introduction about it.

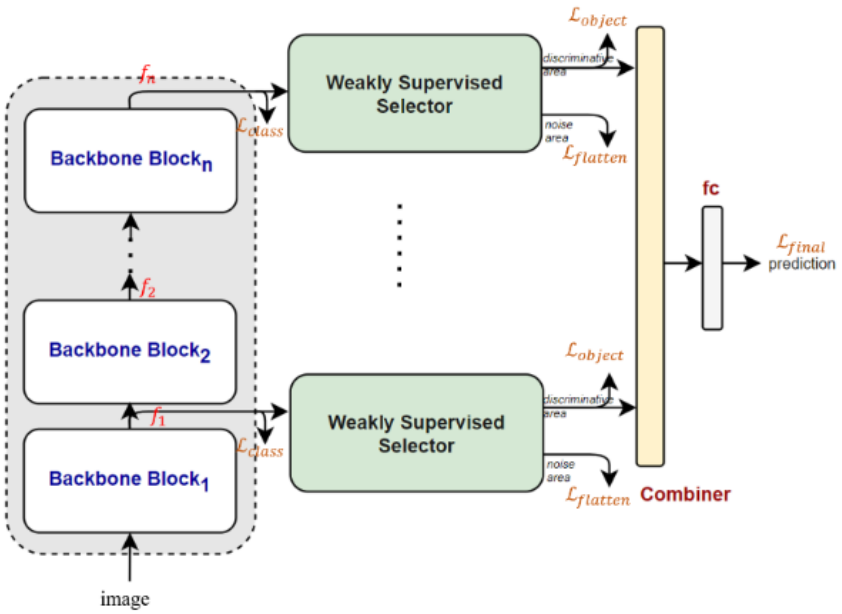
- **Module Design**:
  PIM architecture is designed to be integrated into backbone model, such as ResNet, EfficientNet, and Swin-T. Feature Pyramid Network (FPN) is also incorporated into the backbone model, which helps in capturing multi-scale information from the input images, enhancing the model's ability to analyze fine-grained details. And the fundamental concept of this model is to treat each pixel (or patch) on the feature map as an independent feature representing its region. These features are then classified, and the classification ability is used to distinguish regions.

- **Weakly Supervised Selector**:
  The feature map output by a block in the backbone network is fed into a weakly supervised selector. Each feature point is classified by a linear classifier, resulting in a new feature map Softmax is applied to obtain class prediction probabilities for each feature point. Then selects the first few feature points with high confidence scores.

- **Feature Fusion Mechanism (Combiner)**:
  The selected features are fused through a Combiner

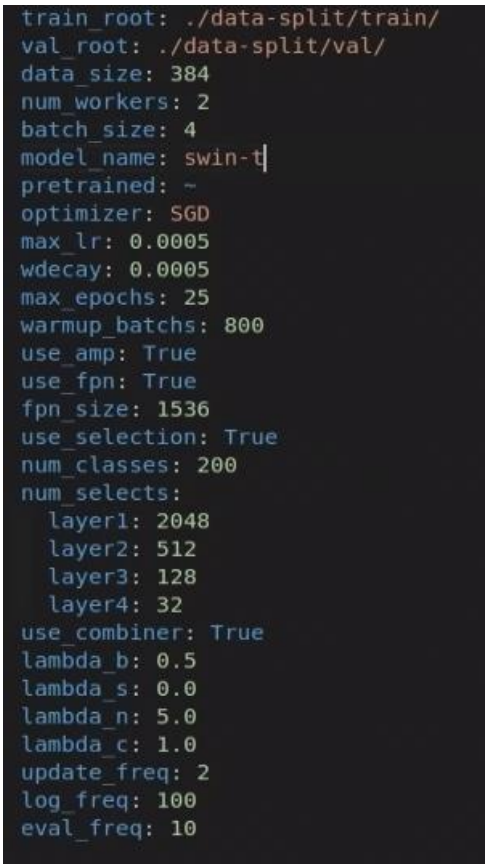
  - **Graph Convolution Architecture:**
    All selected feature points are treated as a graph structure, and a graph convolutional network (GCN) is applied. The relationship between different nodes is learned, and features are aggregated into super nodes through pooling layers. Then the features of these super nodes are averaged, and a linear classifier is used for final predictions.

The schematic flow of PIM



- **Hyperparameters**

Following picture is the hyperparameters I select.



There are the main hyperparameter I choose:

- Base model (backbone model): swin-T
  (swin-T use pretrain weight:
  'swin_large_patch4_window12_384_in22k')
- Optimizer: SGD
- Initial learning rate: 0.0005
- Weight decay: 0.0005
- Batch size: 4
- Epoch number: 50
- FPN size: 1536
- number of regions selected by the weak selector (num_selects):
  (Because the swin-t has 4 block, therefore there has 4 layer)
  - layer1: 2048
  - layer2: 512
  - layer3: 128
  - layer4: 32

■ Training strategy

1. I split 30% of training data as validation set. Then use the remain training set (70%) to train the model.
2. About the transformation on training set. Randomly crop horizontal flip, sharpness, normalize is used. To prevent model from overfitting.
3. After training, it will record the weights of best performance on validation set and the last training.

Result of train on 50 epochs

```
Start Training 47 Epoch..0%..10%..20%..30%..40%..50%..60%..70%..80%..90%....[success][1277.0276 sceonds]
Start Evaluating 47 Epoch..0%..10%..20%..30%..40%..50%..60%..70%..80%..90%....BEST_ACC: 92.185% (91.847%)....[success][1
90.1299 sceonds]
Start Training 48 Epoch..0%..10%..20%..30%..40%..50%..60%..70%..80%..90%....[success][1277.259 sceonds]
Start Evaluating 48 Epoch..0%..10%..20%..30%..40%..50%..60%..70%..80%..90%....BEST_ACC: 92.185% (91.915%)....[success][1
90.8071 sceonds]
Start Training 49 Epoch..0%..10%..20%..30%..40%..50%..60%..70%..80%..90%....[success][1277.0096 sceonds]
Start Evaluating 49 Epoch..0%..10%..20%..30%..40%..50%..60%..70%..80%..90%....BEST_ACC: 92.185% (91.915%)....[success][1
90.3418 sceonds]
Start Training 50 Epoch..0%..10%..20%..30%..40%..50%..60%..70%..80%..90%....[success][1276.7862 sceonds]
Start Evaluating 50 Epoch..0%..10%..20%..30%..40%..50%..60%..70%..80%..90%....BEST_ACC: 92.185% (91.982%)....[success][1
90.4966 sceonds]
```

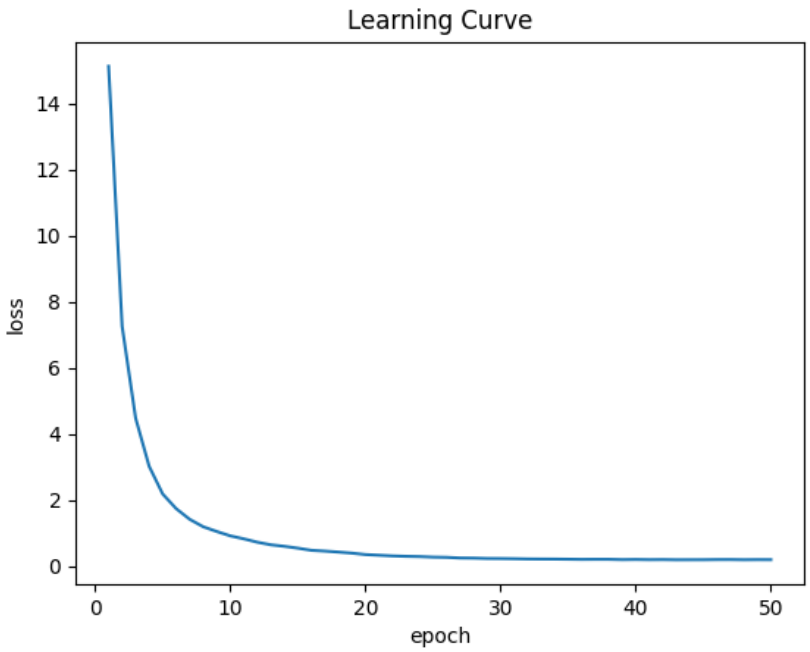
● Experimental results

■ Evaluation metrics

In this final project I only use **accuracy** as my evaluation metric

■ Learning curve

Following picture is the learning curve of my final submission training.
(The loss function is Cross Entropy)



■ Ablation study

I experiment with 4 types of ablation study: different num_selects, backbone, FPN size, additional structure. Because the time issue, in this section I only train on 20 epochs. And beside experiment factor, the remain hyperparameters are same as hyperparameters in **hyperparameters** section. There are the results on validation and some discussion.

1. **Backbone model:**

model	Accuracy
Resnet50	86.468
Efficient	84.202
Swin-t	92.185

By the above table. Swin Transformer (Swin-T) outperforms both ResNet-50 and EfficientNet with an accuracy of 92.185%. Compare to the both models (ResNet50 and EfficientNet). I think swin-t's attention mechanism and hierarchical structure might be more suitable for capturing intricate features crucial for fine-grained classification and contributing to better feature representation and discrimination.

2. Num_selects (left to right respond to layer1 to layer 4):

Backbone model: swin-t

Num_selects	Accuracy
[32, 32, 32, 32]	92.135
[512, 256, 128, 64]	92.118
[1024, 256, 128, 32]	92.126
[2048, 512, 128, 32]	92.185

The results in this part suggest that carefully tuning the number of selected regions in each block can impact accuracy, with a moderate to larger number of selections providing better results for fine-grained visual classification with Swin-T backbone, such as [2048, 512, 128, 32]. But the computation time is greatly increase by the layer number increase and the accuracy may not significant increase, this is a factor need to trade-off.

3. FPN size:

Backbone model: swin-t

FPN size	Accuracy
512	91.949
1024	91.949
1536	92.185
2048	92.287

The result of size 1024 is similar to the size 512, a size of 1024 doesn't provide a significant improvement. It suggests that increasing FPN size beyond a certain point might not bring additional benefits for the given architecture or dataset. Increase in FPN size to 1536 results in a noticeable improvement in accuracy. The highest accuracy is achieved with an FPN size of 2048. This suggests that, for this particular experiment, a larger FPN size contributes to capturing a broader range of contextual information, enabling the model to perform better in fine-grained visual classification tasks.. But I submit the 2048 FPN size to Kaggle get the lower accuacy than size of 1536 (2048 get 0.904, 1536 get 0.907). This may shows that more higher FPN size also tend to overfitting.

4. Additional structure (backbone only use swin-t):

structure	Accuracy
Just backbone	78.124
Backbone + FPN + selector + combiner	92.185

This result is quite intuitive. The inclusion of additional structures, including FPN for feature fusion, a selector for identifying discriminative regions, and a combiner for integrating selected features, significantly boosts the accuracy to 92.185%. Demonstrates the importance of these components in enhancing the model's ability to perform fine-grained visual classification.

✧ Inference Note

For TAs: You can put the test file and the model weights in ./training/For-inference file. Then execute the 109350008_inference.py
(python ./109350008_inference.py)