NYCU Intro. to ML
Final Project Report

110550093, 蔡師睿

1 Environment Details

1.1 Python Version

Python 3.9.18

1.2 Virtual Environment

Conda

1.3 Framework

Pytorch

1.4 Hardware

On GPU RTX4090 (lab GPU) & RTX3050 (my loptop).

1.5 How to Inference

First, create a conda environment with Python version 3.9.18. After activating the created conda environment, rebuild the environment by using the command '**pip install -r requirements.txt**'. There is no need to manually download the weights from my Google Drive, as I have utilized the gdown package in my inference file to automatically download the model weights. All you need to do is place the dataset in the same directory. The example directory structure is as follows:

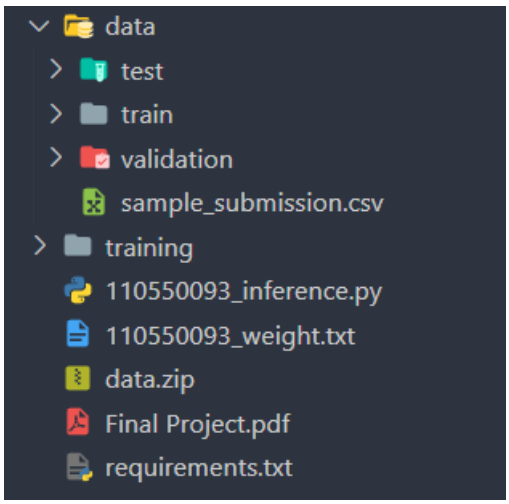

Figure 1: Example directory structure. Notice: There must be a folder named "test" under the "data" directory.

After executing my inference file, there will be two created folders named "weights" and "results." The downloaded model weights will be stored in the "weights" folder, and the CSV file will be placed in the "results" folder.

2 Implementation Details

2.1 Model Architecture

CAL

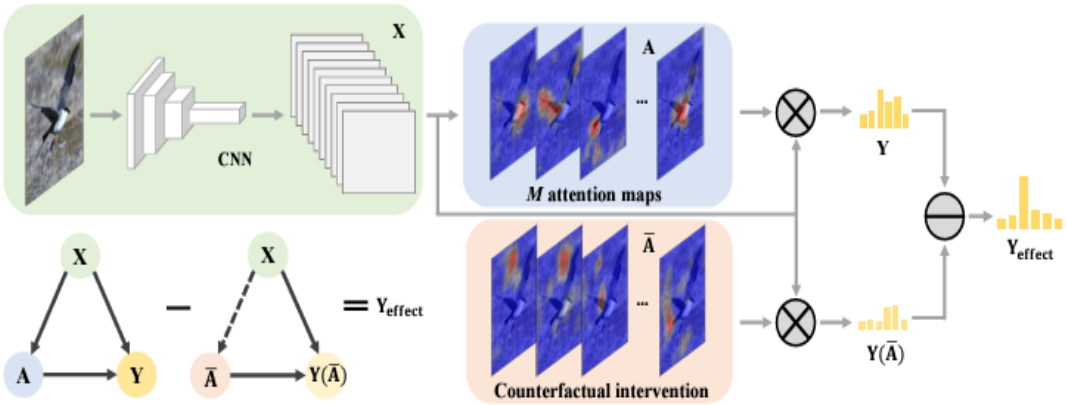

Figure 2: Model Architecture of CAL [1]. This framework involves initially implementing counterfactual intervention on the original attention mechanism, achieved by substituting it with random attentions. Subsequently, it deducts the counterfactual classification results from the original classification outcomes to analyze the impacts of learned visual attention and optimize them throughout the training process.

SIM-Trans

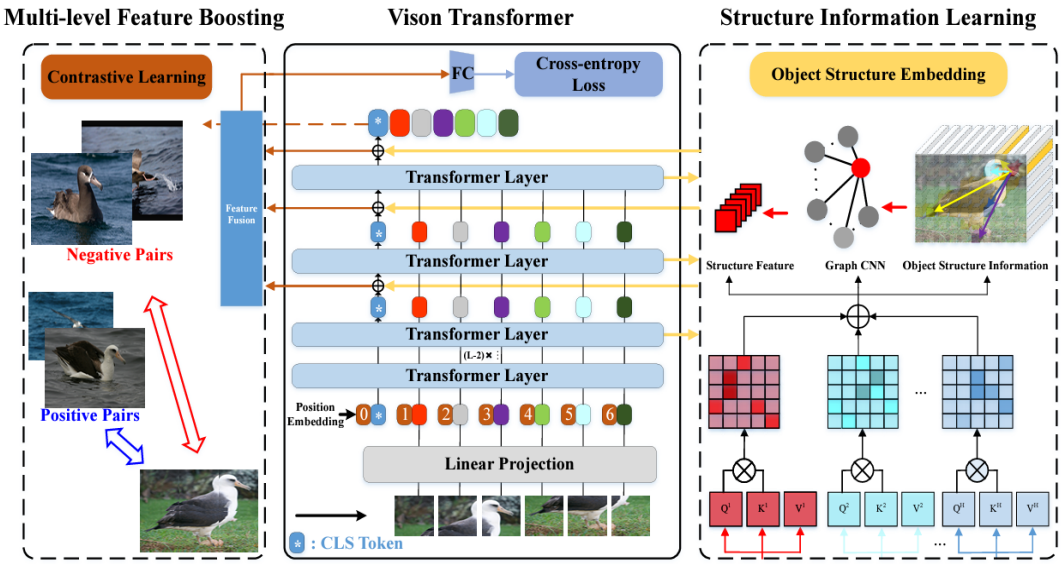

Figure 3: Model Architecture of SIM-Trans [2]. This framework includes a vision transformer backbone responsible for extracting features. A Structure Information Learning (SIL) module on the right side utilizes self-attention weights from the transformer layer to analyze the spatial context of discriminative patches within objects. Additionally, a Multi-Level Feature Boosting (MFB) module on the left fuses features from different levels. Simultaneously, the features are enhanced through contrastive learning.

Ensemble

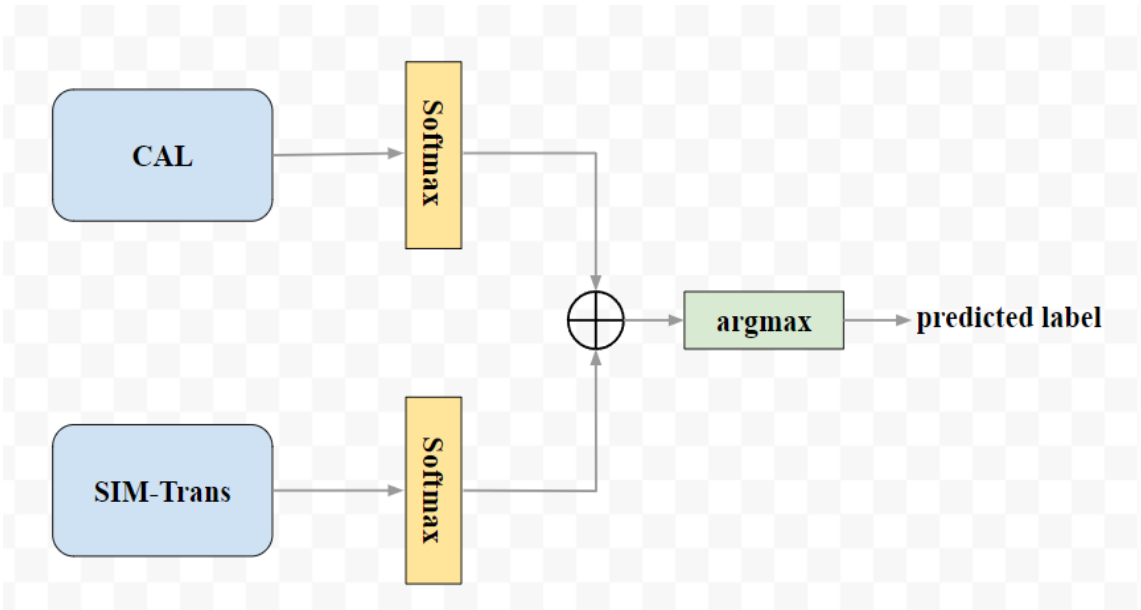

Figure 4: Model Architecture of ensemble model. The output logits of the two models will undergo the softmax function first, and will be followed by addition. The predicted label will then be determined by taking the argmax of the combined logits.

2.2 Hyperparameters

CAL

- random seed: 666
- epochs: 30
- batch size: 12
- SGD:
  - learning rate: 0.001
  - momentum: 0.9
  - weight_decay: 1e-5

SIM-Trans

- random seed: 666
- epochs: 25
- batch size: 8
- SGD:
  - learning rate: 0.03
  - momentum: 0.9
  - weight_decay: 0
- WarmupCosineSchedule:
  - warmup_steps: 10
  - t_total: 25

2.3 Training Strategy

At first, I believe that ResNet would effectively handle fgvc tasks, and thus I explored and exprimented with several models which utilized ResNet as their backbone, including PMG [3] and CAL. However, the public score of these models consistently remained below 0.9. Consequently, I started searching for a transformer to replace ResNet because transformer has demonstrated superior performance in various scenarios, such as fine-tuning, as evidenced by recent research.

Finally, I utilized SIM-Trans to fine-tune on our CUB dataset. During training, I discovered that the training time was too long, and due to some environment

problem, my conda couldn't import apex package provided by the source code to accelerate. To address this issue, I imported *autocast* from *torch.cuda.amp*, resulting in a successful reduction of training time from 8 minutes per epoch to 5 minutes per epoch.

Afterwards, I conducted experiments with ensembling to assess whether combining transformer and ResNet could yield improved results. I selected SIM-Trans and CAL to ensemble, because CAL is nearly achieved the highest accuracy among ResNet backbone models based on my survey of the papers and SIM-Trans has the highest accuracy among my models. Surprisingly, the accuracy showed an improvement, rising from 90.7% to 91.4%.

2.4 Training Details

Dataset

To establish validation dataset for assessing the performance during training, I split the dataset in "train" folder by selecting only one image in each class as my validation set. There will be 9588 training images, 200 validation images, and 2000 testing images.

Pre-trained Weights

In my fine-tuning process, the pre-trained weights of ResNet50 and ResNet101 were trained on ImageNet, and the pre-trained weights of ViT-B_16 were trained on ImageNet21k.

3 Experiment Results

3.1 Evaluation Metrics

Utilize accuracy as my evaluation metric.

$$accuracy = \frac{1}{n}\sum_{i=1}^{n} f(x),\ where\ f(x) = \begin{cases} 1, & if\ x = label \\ 0, & otherwise \end{cases}$$

3.2 Learning Curve

CAL

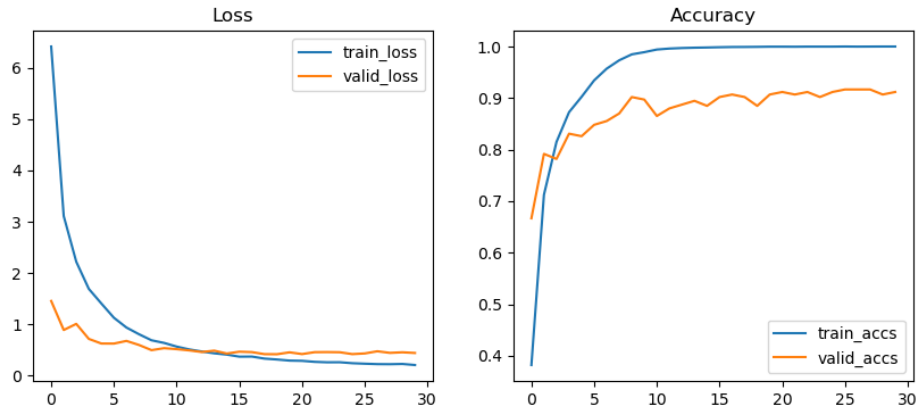

Figure 5: The accuracy and loss for each epoch of the CAL model.

SIM-Trans

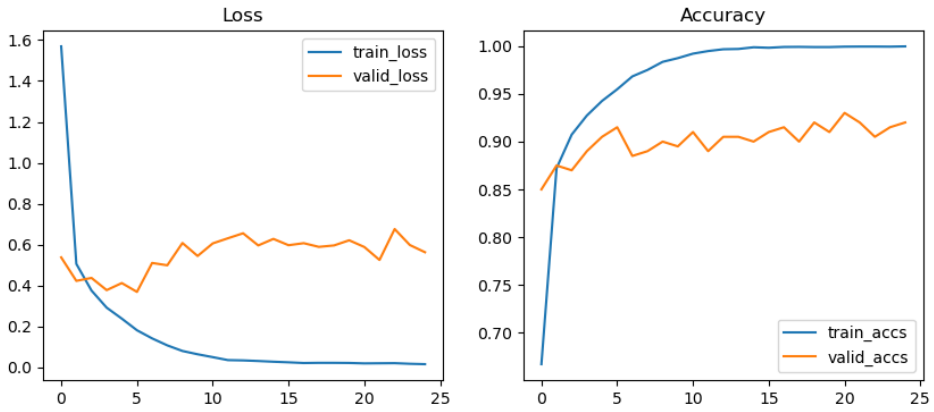

Figure 6: The accuracy and loss for each epoch of the SIM-Trans model.

3.3 Ablation Study

CAL

Method	Accuracy (%)
ResNet101	81.7
ResNet101 + CAL	89.2

Table 1: This ablation study demonstrates a significant improvement in our task when using our CAL model compared to only using ResNet101.

Multi-level feature boosting	Contrastive learning	Accuracy (%)
		88.6
✓		90.2
✓	✓	90.7

Table 2: In this table, it is evident that multi-level feature boosting contributes to robust feature learning. Furthermore, in multi-level feature boosting, contrastive learning can further enhance the model's performance in terms of classification accuracy.

3.4 Comparisons

Method	Backbone	Accuracy (%)
ResNet50	ResNet50	81.6
PMG	ResNet50	86.3
ResNet101	ResNet101	81.7
PMG	ResNet101	87.3
CAL	ResNet101	89.2
SIM-Trans	ViT-B_16	90.7
Ensemble (CAL + SIM-Trans)	Resnet101 + ViT-B_16	**91.4**

Table 3: Compare the training performance of various models, such as ResNet50, ResNet101, PMG, CAL, and SIM-Trans, on our custom dataset. Evaluate their accuracy based on the public submission scores obtained from Kaggle.

4 Discussion

Attention is All You Need!!! However, the accuracy and loss of the validation set in SIM-Trans didn't exhibit improvements in each iteration. I consider that my validation set was divided too small, making it challenging to observe significant improvements.

References

[1] Y. Rao, G. Chen, J. Lu, and J. Zhou, "Counterfactual attention learning for fine-grained visual categorization and re-identification," in *ICCV*, 2021.

[2] H. Sun, X. He, and Y. Peng, "Sim-trans: Structure information modeling transformer for fine-grained visual categorization," in *Proceedings of the 30th ACM International Conference on Multimedia*, 2022, pp. 5853–5861.

[3] R. Du, D. Chang, A. K. Bhunia, *et al.*, "Fine-grained visual classification via progressive multi-granularity training of jigsaw patches," in *European Conference on Computer Vision*, 2020.