NYCU Introduction to Machine Learning
Final Project Report

110550088 李杰穎

1 Environment

In this section, I will briefly introduce the environment I used in this project, including workstation settings and the Python environment.

1.1 Workstations Environment

- OS: Ubuntu 22.04.3 LTS
- CPU: AMD Ryzen Threadripper PRO 5955WX 16-Cores $\times$ 2
- GPU: NVIDIA GeForce RTX 24GB

1.2 Python Environment

- Package Manager: conda
- Python: 3.9.18
- Packages: `torch`, `pandas`, `torchvision`, `timm`... For the complete list, please refer to `requirements.txt`.

2 Implementation

In this project, I utilize FGVC-HERBS[1] (referred to as "HERBS" for brevity) as the primary model, adjusting multiple hyperparameters to enhance its performance.

2.1 Model Architecture

In this section, I will mainly discuss about the architecture of HERBS. And how it becomes the SOTA of fine-grained bird images classification.

HERBS consists of two main modules: the high-temperature refinement module and the background suppression module.

2.1.1 High-Temperature Refinement Module

This module allows the model to learn appropriate feature scales by refining the feature map at different scales and improving the learning of diverse features. It initially uses higher temperatures to learn feature maps so more global and contextual information can be captured. Then, the feature maps are refined using lower temperatures to capture finer details. This approach allows obtaining richer features to better classify similar objects and improve accuracy. It maintains an appropriate size of the discriminative region, crucial for this task.

2.1.2 Background Suppression Module

This module first splits the feature map into foreground and background using classification confidence scores. It suppresses feature values in low-confidence areas while enhancing discriminative features. This suppression helps improve the details of the target object and reduce noise, aiding in cases where it's difficult to distinguish between foreground and background areas.

2.1.3 Novelties

HERBS can be integrated with different backbones, including CNN like ResNet and visual transformer like Swin-Transformer. It also achieve significant accuracy improvements on the CUB-200-2011 and NABirds benchmarks, surpassing 93% accuracy on both datasets. By ablation study in paper, we can know that this improvement is because of two new modules introduced above.

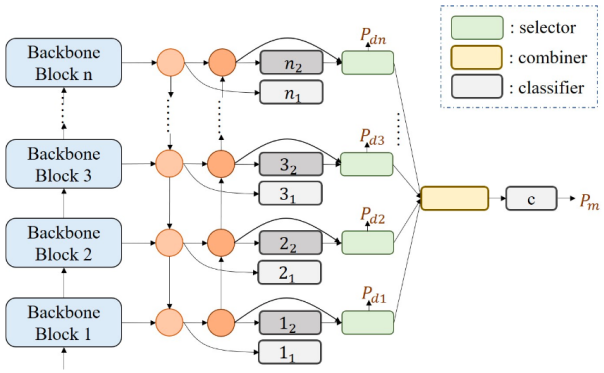


Figure 1: The architecture of HERBS. Screenshot from [1]. "The circles in the middle part denote the multi-scale feature fusion module, such as Feature Pyramid Network (FPN). The classifiers in the middle are used to generate classification map for background suppression." (quote from original paper)

2.2 Hyperparameters

2.2.1 Image size, batch size and epochs

All images will first be resized to 384×384 before being sent into models. This can reduce computing resources, which is also a common practice in image classification.

The batch size is set to 16, which can fit in our GPU.

As for epochs, all models used for experiments and evaluation are trained for 80 epochs. I further trained the model to 120 epochs for better performance; this model is also the submitted one.

2.2.2 Feature Pyramid Network and Number of selected features

Feature Pyramid Network (FPN) is the component for extracting the features generated by backbones. The number of features in each level of FPN is 1536.

And the numbers of selected features for layer 1 to layer 4 are 256, 128, 64 and 32, respectively.

2.2.3 Weights of Loss

In HERBS, they design multiple loss terms, each for different components. In this project, I won't modify these weights, because tuning them requires high computational resources. And the effect of tuning the weights is also unknown. Therefore, I will use the same value as original paper.

- $\lambda_m = 1$
- $\lambda_d = 5$
- $\lambda_l = 0.3$
- $\lambda_r = 1$

2.2.4 Temperature

In the official code, the temperature in a given epoch is calculated by,

$$\text{Temperature} = \text{Init. Temperature} \cdot (0.5^{\lfloor \frac{\text{epoch}}{10} \rfloor}). \quad (1)$$

The above equation is actually equivalent with the equation in the paper.

Figure 2 further demonstrate the temperature over epoch.

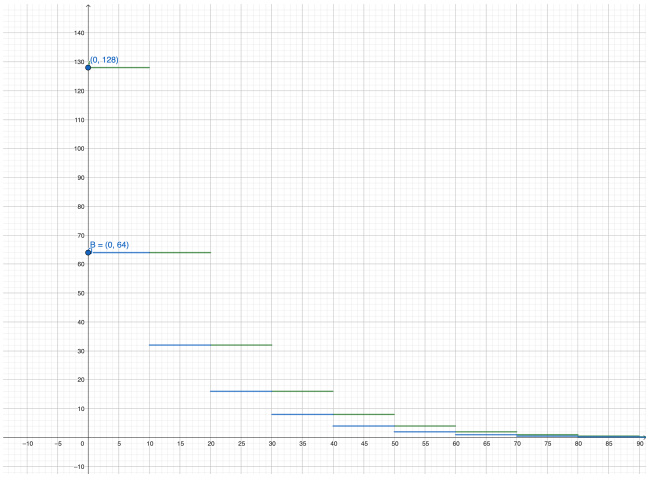


Figure 2: The temperature from 0 to 80 epochs, the blue line is temperature=64, the green line is temperature=128

2.3 Training Strategy

2.3.1 Optimizer

The official code used SGD as the optimizer. In section 3, I will conduct a experiment and discuss the difference of using SGD and AdamW.

The parameters of SGD are:

- `momentum = 0.9`
- `max_lr = 0.0005`
- `weight_decay = 0.0003`
- `nesterov = True`

2.3.2 Learning Rate and Warm-up Batches

As we can see in Figure 3, the learning rate is initially set low, this is because of the introducing of warm-up batches. This concept first be introduced in [2]. This idea is that the weights of the model are initialized randomly, thus, the gradient of model's weights may be unstable. Therefore, if we first make the learning rate lower in the early stage of training, the model can see all the data without updating the weight too much. After the model sees all the data, we can fallback to normal learning rate scheduling. In this way, we can reduce the probability of falling into local minima.

The learning rate scheduler used in HERBS's official code is cosine decay.

The number of warm-up batches is 1500.

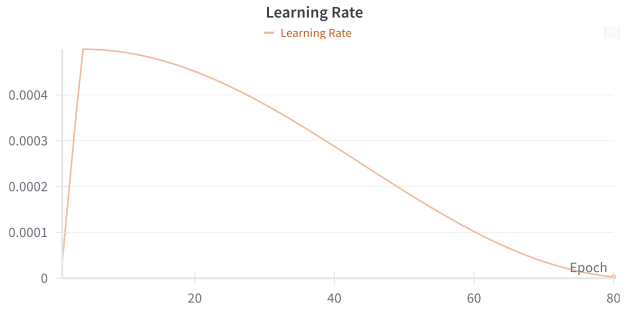


Figure 3: The learning rate overtime from 0 to 80 epoches.

2.3.3 Data Argumentation

The training time augmentation is aligned with HERBS's official code, as shown in Code 1.

Code 1: Training Time Data Argumentation

```
transforms.Resize((510, 510), Image.BILINEAR),
transforms.RandomCrop((384, 384)),
```

```
transforms.RandomHorizontalFlip(),
transforms.RandomApply([transforms.GaussianBlur(kernel_size=(5, 5), sigma=(0.1, 5))], p=0.1),
transforms.RandomAdjustSharpness(sharpness_factor=1.5, p=0.1),
transforms.ToTensor(),
transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
```

In my opinion, I believe that this data argumentation is strong enough, I've tried adding other data argumentation, like color jitter, but didn't achieve better performance. In addition, I think applying data argumentation that would change the color in the image would make the model unable to correctly classify images. This is because color is a critical part when we classify different birds.

It's noteworthy that the mean and standard deviation used in normalization are calculated based on ImageNet[3], which is a common practice in image classification. Adding the normalization term can stable the training process.

3 Experiments

3.1 Evaluation Method

Because the provided datasets only contains a labeled training set and a unlabeled test set, it's hard to evaluate the model's performance. Thus, I randomly split the original training set to a training set and validation set using 8/2 split. For the experiments provides below, I trained the model using the smaller training set, and use the validation set for evaluation. This is possible because every image in validation set is labeled.

For evaluation metric, I mainly use accuracy and F1-score to evaluate the performance of models.

Moreover, following the evaluation in HERBS[1], I also performed evaluation on fined-class. In particular, I will use 9 generic classes, including flycatcher, gull, kingfisher, sparrow, tern, vireo, warbler, woodpecker, wren. The mapping between each generic classes and species will be listed in section 4.

3.2 Ablation Study

In this section, I will mainly discuss about one component and one hyperparameter in HERBS, which are combiner and temperature.

By adjusting these two configurations, I trained 4 models, which are,

1. With combiner, temperature=64 (default setting)
2. With combiner, temperature=128
3. Without combiner, temperature=64
4. Without combiner, temperature=128

Table 1: Ablation study on combiner and temperature. The F1-score is calculated with a weighted manner.

Model		Accuracy	F1-score
Combiner	Temperature		
✓	64	0.9235	0.9227
✓	128	**0.9245**	**0.9238**
	64	0.9216	0.908
	128	0.9144	0.912

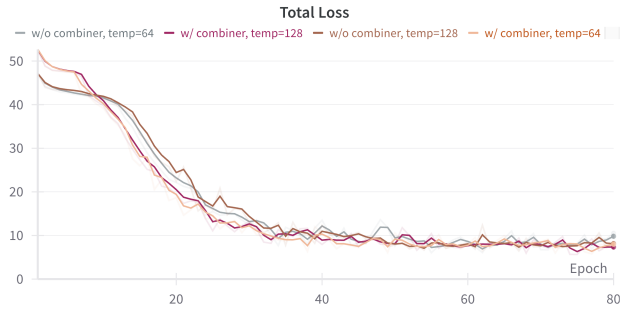


Figure 4: The training loss of four models over time

As we can see in Table 1, the performance of the model with combiner, temperature=128 is the best, achieving the accuracy of 0.9245. This might due to the high exploration rate in features in the early phase of training. However, I believe that tuning the temperature is not very useful, the increasing randomness of selecting features may not directly improve the performance. As for the combiner, it's clear that the accuracy decrease after removing the combiner, for both low and high temperature. Therefore, this ablation study proves that the combiner introduced in HERBS has positive effect in performance.

For training loss, we can observe in Figure 4, the loss of four models are all decreasing and finally converge to nearly the same value. We can also notice that the loss of two high temperature model are higher in first 10 epochs, but quickly decreased afterward. I think this is mainly because of the diversity of features in the early stage of training.

Finally, I will use model with combiner and temperature=128 to conduct following experiments, because it has the highest accuracy and F1-score.

3.3 Inference Time Transformation

It's a common practice to perform some transformations on image when inference. In evaluation, I will perform several type of

transformation combinations, to evaluate how transformations effect final result.

I tested three inference time transformation on the model with default setting. **Original transformation** is the implementation used in HERBS's official code, which first resize the image to 510×510 and center crop the image to 384×384; This transformation is also a common practice in image classification. **No crop transformation** directly resize the image to 384×384. **Five crop transformation** first resize the image to 510×510, and then crop the image into four corners and the central crop.

Code 2: Original Transformation

```
transforms.Resize((510, 510), Image.BILINEAR),
transforms.CenterCrop((384, 384)),
transforms.ToTensor(),
transforms.Normalize(mean=[0.485, 0.456, 0.406],
    std=[0.229, 0.224, 0.225])
```

Code 3: No Crop Transformation

```
transforms.Resize((384, 384), Image.BILINEAR),
transforms.Lambda(lambda crops:
    transforms.ToTensor(),
transforms.Normalize(mean=[0.485, 0.456, 0.406],
    std=[0.229, 0.224, 0.225])
```

Code 4: Five Crop Transformation

```
transforms.Resize((510, 510), Image.BILINEAR),
transforms.FiveCrop((384, 384)),
transforms.Lambda(lambda crops:
    torch.stack([transforms.ToTensor()(crop) for
    crop in crops])),
transforms.Lambda(lambda crops:
    torch.stack([transforms.Normalize(mean=[0.485,
    0.456, 0.406],std=[0.229, 0.224, 0.225])(crop)
    for crop in crops]))
```

I tested the above three kinds of transformations on the model with default hyperparameters. Table 2 shows the results, five crop can improve the accuracy by 0.001, compared with original transformation.

Table 2: The accuracy and F1-score using different transformation methods. The F1-score is calculated in weighed manner.

Transformation	Accuracy	F1-score
Original	0.9245	**0.9238**
No Crop	0.9189	0.9176
Five Crop	**0.9255**	**0.9238**

3.4 Generic Classes

In this section, I will mainly discuss about the accuracy within generic class. This experiment is useful because we can observe species in which generic class are more similar, thus the model can't distinguish than very well.

As we can observe in Table 3, the accuracy within "Gull" class is significant lower than other classes. In Figure 5, each kinds of gull is looks very similar, I think this is why the accuracy is the lowest in this class.

I further plot the confusion matrices for each generic class in subsection 4.4.

Table 3: The precision and number of false positive samples for each generic class.

Generic Class	Precision	FP
Flycatcher	0.81429	8
Gull	0.79487	1
Kingfisher	0.9	0
Sparrow	0.89423	2
Tern	0.8	1
Vireo	0.89855	5
Warbler	0.932	6
Woodpecker	0.96552	0
Wren	0.97143	0

Figure 5: Each kind of gulls in the datasets.

4 Bonus

4.1 Training the model

You should able to train the model by running,
`python main.py --c ./configs/config.yaml`.

I've also provided the configuration files I used for every experiment.

4.2 Running the the inference code

I used Jupyter Notebook for inference, because I can test it on Google Colab before submitting. If TAs want to run the notebook on Google Colab, the process is as follow:

1. Open `110550088_inference.ipynb` in Google Colab
2. Upload all files and directories in the zip file to Colab, placed them in `/content`.
3. Download model's weights from Google Drive link, and upload it to `/content`.
4. Now, the `/content` should look like Figure 6.
5. I put the test datasets in `training/` `datasets/` `test`. Therefore, it's no need to download the test datasets.
6. Finally, after running all cells in the notebook, you should able to see `submission.csv` appears in the folder.

Also, after testing, I found that there is no need to install additional packages when running in colab. Though I still provide the `requirements.txt`, I don't use pip to install them in the jupyter notebook.

Noted that the use T4 GPU runtime in Google Colab, my inference notebook needs to run under environment with GPU.

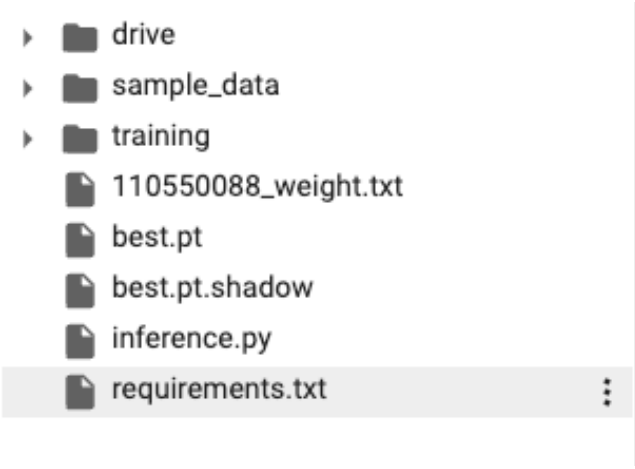


Figure 6: The screenshot after completing step 3.

4.3 Future Work

I think that the performance of HERBS is actually bounded by the size of training data. The accuracy in my own experiments is always lower than the number in [1]. Also, as mentioned in class, transformer-based model requires more data for training in order to achieve strong performance. Thus, for future work, I would like to figure out if there has any way to improve the data argumentation, let the model can learn even from small amount of data.

4.4 Confusion Matrix of Each Generic Class

In this section, I will provide the confusion matrices within each generic class. By them, we can observe what kinds of species are more likely to be wrongly classified. As I mentioned previously, gulls are very hard to distinguish, this can be also proved in Figure 8.

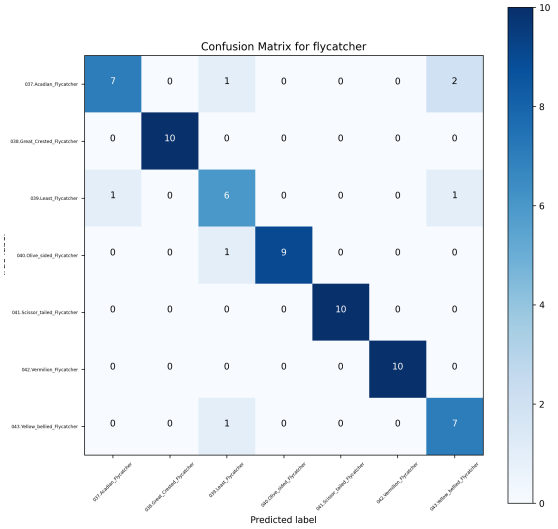


Figure 7: The confusion matrix of flycatchers

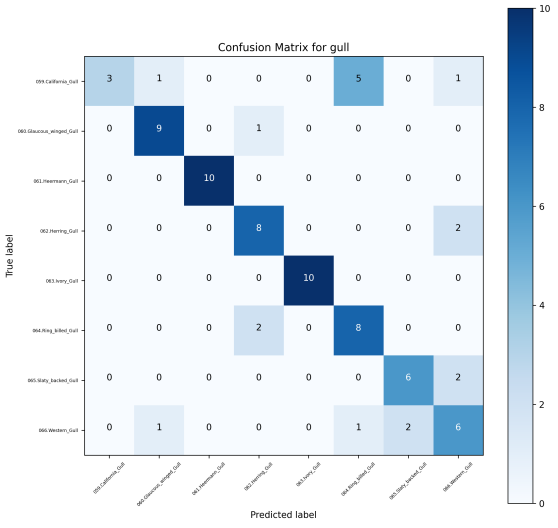


Figure 8: The confusion matrix of gulls

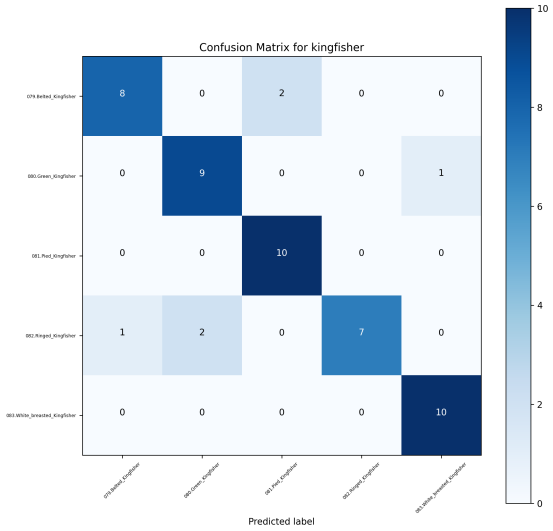


Figure 9: The confusion matrix of kingfishers

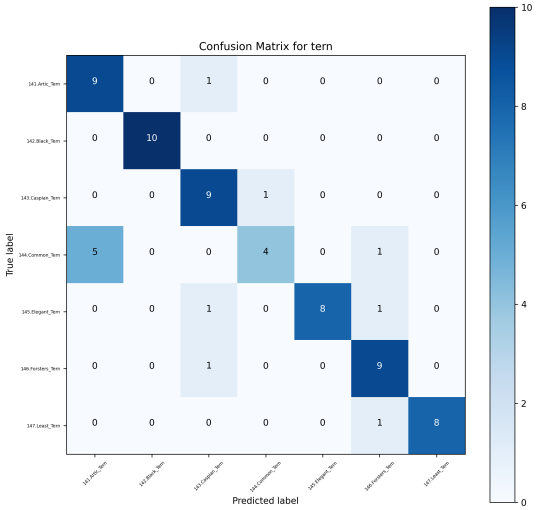


Figure 11: The confusion matrix of terns

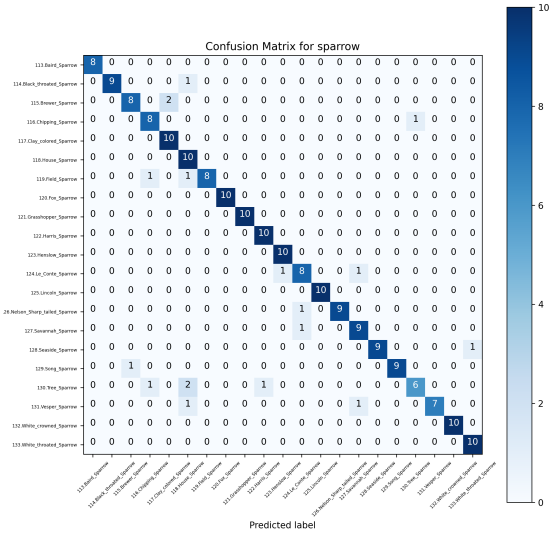


Figure 10: The confusion matrix of sparrows

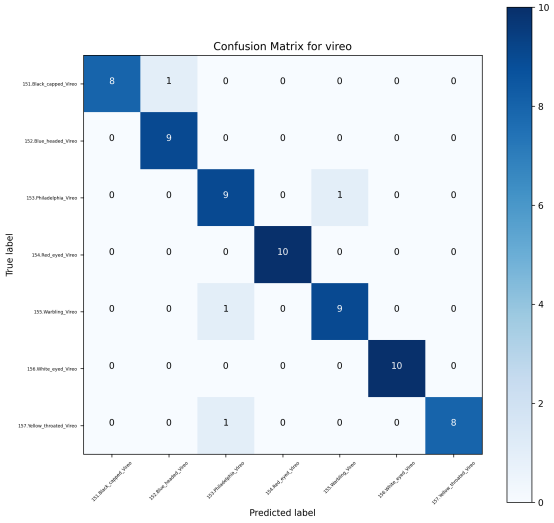


Figure 12: The confusion matrix of vireos

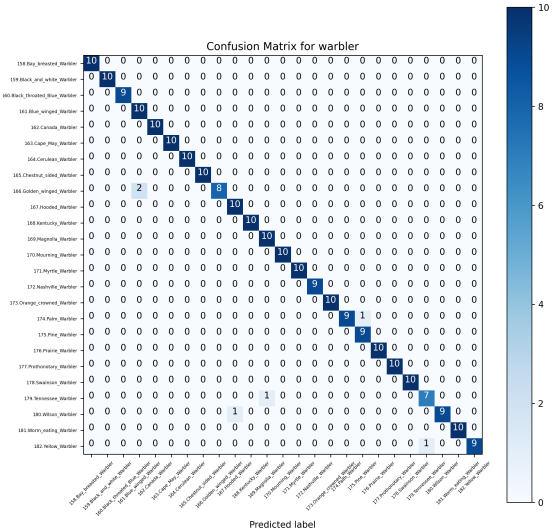


Figure 13: The confusion matrix of warblers

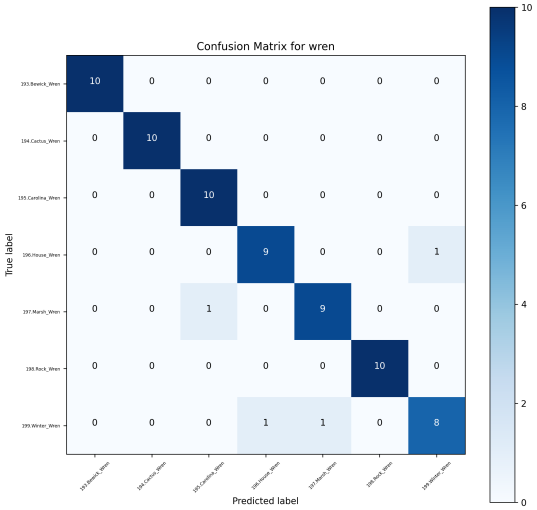


Figure 15: The confusion matrix of wrens

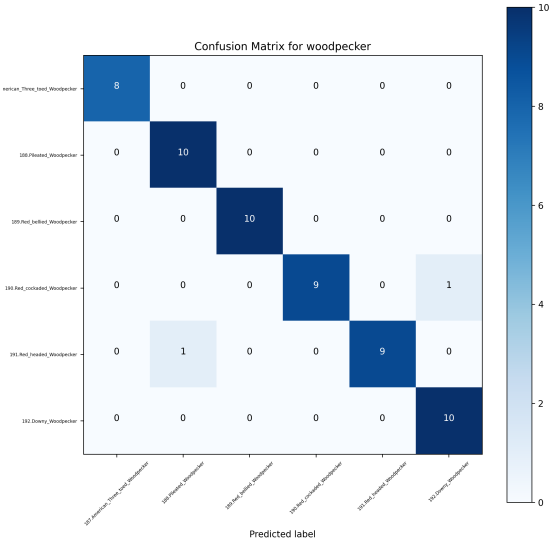
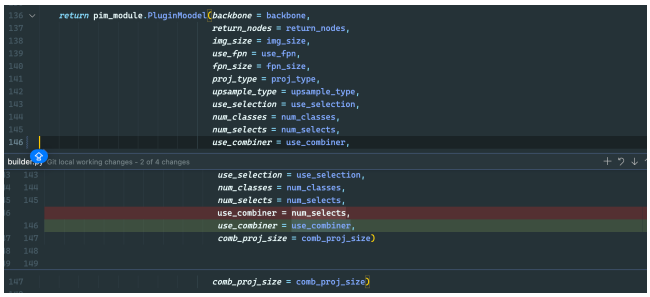


Figure 14: The confusion matrix of woodpeckers

4.5 Bugs in HERBS's Official Code

When running the official code, I found several bugs. For example, in Figure 16, the author didn't properly pass the `use_combiner` parameter into model when building the model. Therefore, if I don't fix this bug, when conducting ablation study, there won't have any difference in model. Another example is Figure 17, when importing `PluginModel`, the original code append "`.py`" to module name.



```
return pim_module.PluginModel(backbone = backbone,
                              return_nodes = return_nodes,
                              img_size = img_size,
                              use_fpn = use_fpn,
                              fpn_size = fpn_size,
                              proj_type = proj_type,
                              upsample_type = upsample_type,
                              use_selection = use_selection,
                              num_classes = num_classes,
                              num_selects = num_selects,
                              use_combiner = use_combiner,
```

```
builder.py  Last local working changes - 2 of 4 changes
                              use_selection = use_selection,
                              num_classes = num_classes,
                              num_selects = num_selects,
                              use_combiner = num_selects,
                              use_combiner = use_combiner,
                              comb_proj_size = comb_proj_size)


                              comb_proj_size = comb_proj_size)
```

Figure 16: `FGVC-HERBS/models/builder.py`. Parameter `use_combiner` are not properly passed into model.

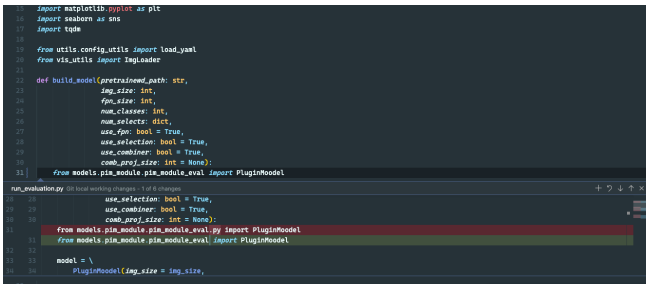


Figure 17: `FGVC-HERBS/run_evaluation.py`. Error when importing the package.

References

[1] P.-Y. Chou, Y.-Y. Kao, and C.-H. Lin, "Fine-grained visual classification with high-temperature refinement and background suppression," *arXiv preprint arXiv:2303.06442*, 2023.

[2] P. Goyal, P. Dollár, R. Girshick, *et al.*, "Accurate, large minibatch sgd: Training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.