

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 Novelities

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.

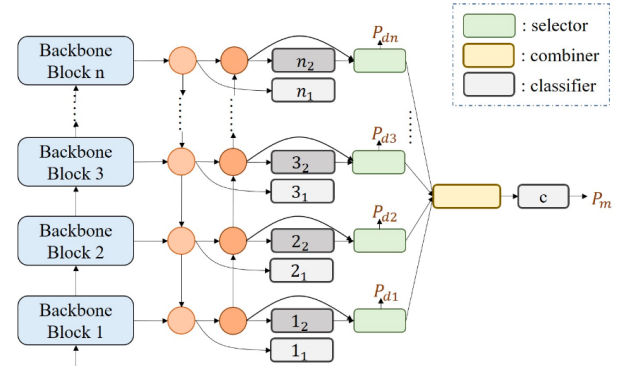


Figure 1: The architecture of HERBS. Screenshot from [1]. “The circles in the middle part denote the multi-scale feature fusion module, such as Feature Pyramid Network (FPN). The classifiers in the middle are used to generate classification map for background suppression.” (quote from original paper)

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.

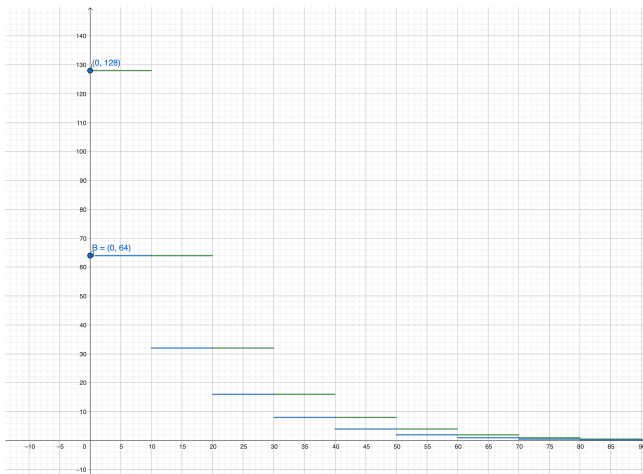


Figure 2: The temperature from 0 to 80 epochs, the blue line is temperature=64, the green line is temperature=128

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.

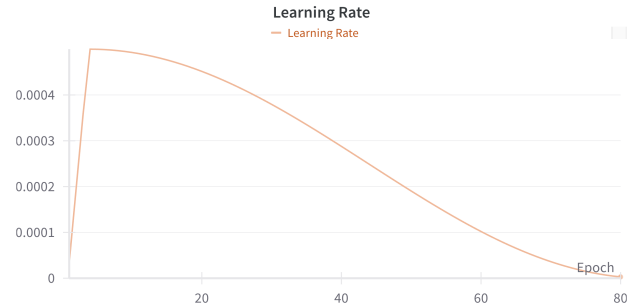


Figure 3: The learning rate overtime from 0 to 80 epochs.

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

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

```

3 transforms.RandomHorizontalFlip(),
4 transforms.RandomApply([transforms
  ↳ .GaussianBlur(kernel_size=(5, 5), sigma=(0.1,
  ↳ 5))], p=0.1),
5 transforms.RandomAdjustSharpness(sharpness_factor=1
  ↳ .5,
  ↳ p=0.1),
6 transforms.ToTensor(),
7 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

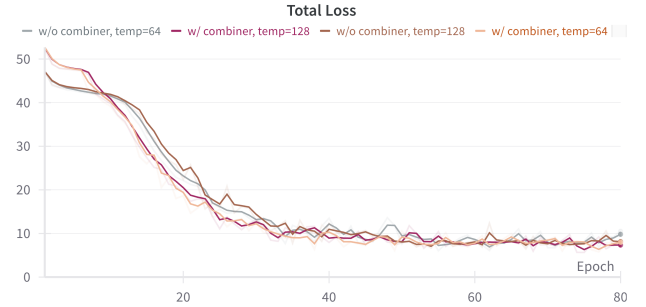


Figure 4: The training loss of four models over time

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

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

Code 3: No Crop Transformation

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

Code 4: Five Crop Transformation

```
1 transforms.Resize((510, 510), Image.BILINEAR),
2 transforms.FiveCrop((384, 384)),
3 transforms.Lambda(lambda crops:
  ↳ torch.stack([transforms.ToTensor()(crop) for
  ↳ crop in crops])),
4 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.

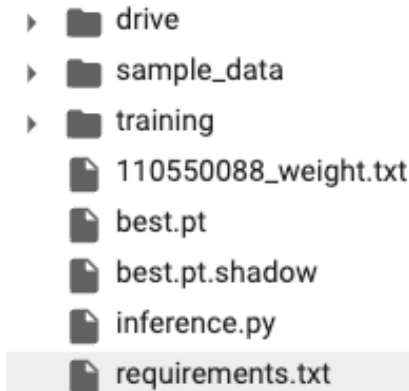


Figure 6: The screenshot after completing step 3.

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.

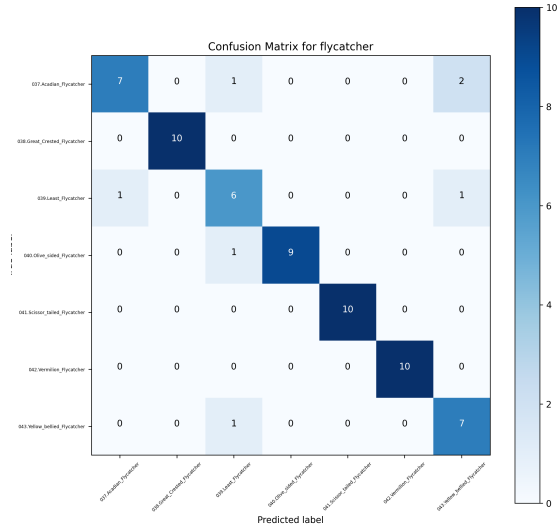


Figure 7: The confusion matrix of flycatchers

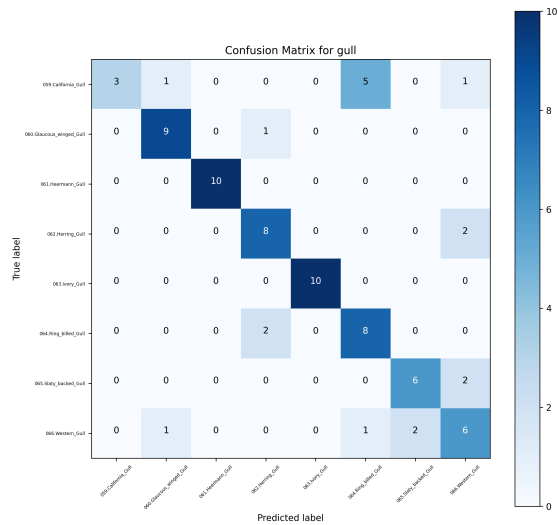


Figure 8: The confusion matrix of gulls

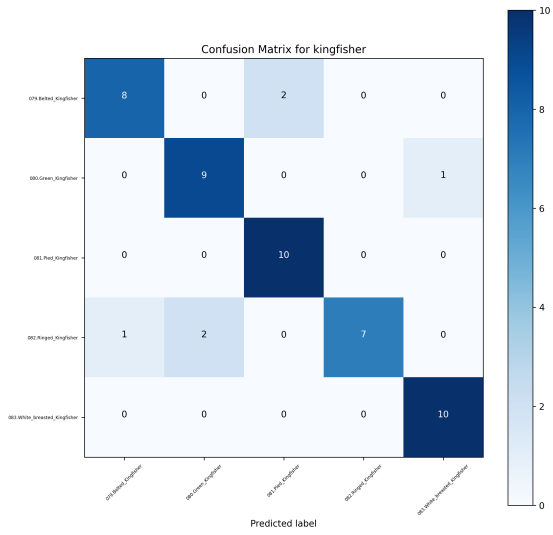


Figure 9: The confusion matrix of kingfishers

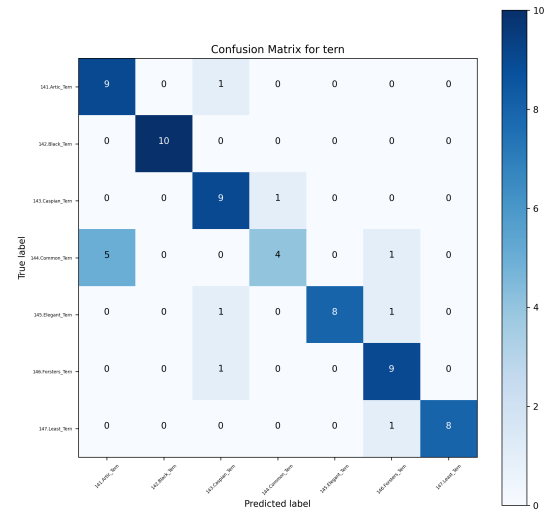


Figure 11: The confusion matrix of terns

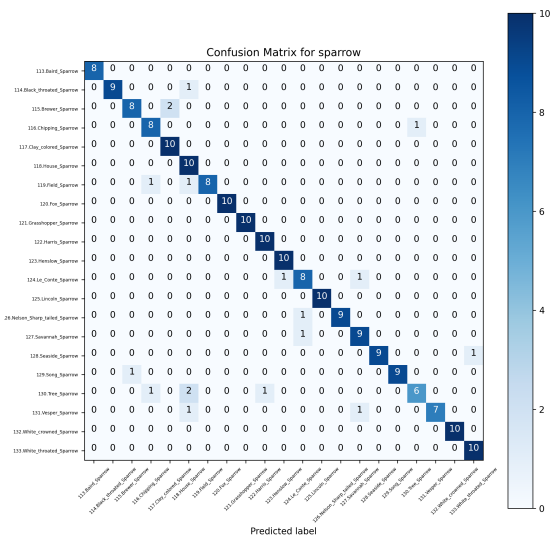


Figure 10: The confusion matrix of sparrows

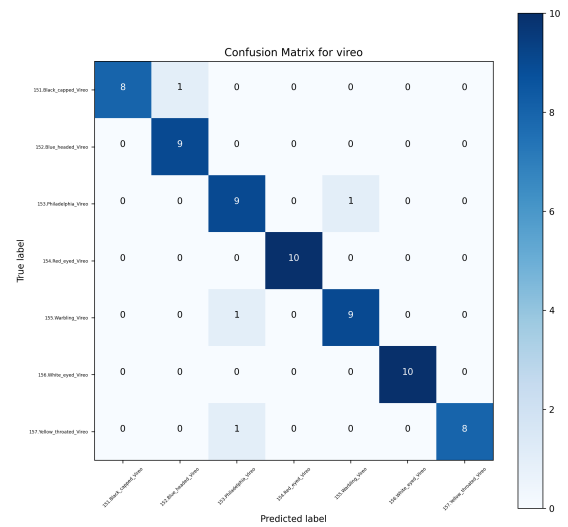


Figure 12: The confusion matrix of vireos

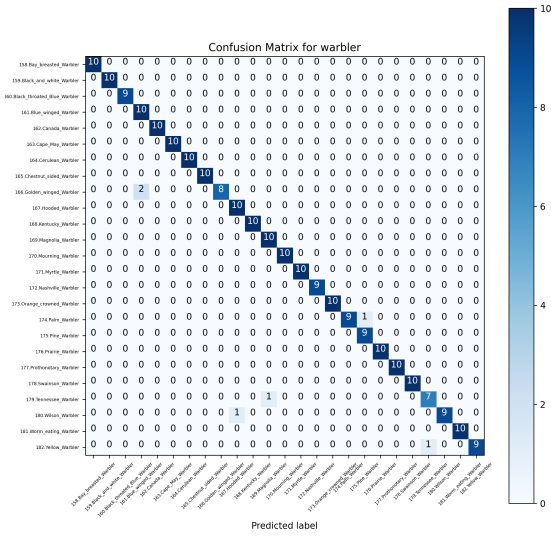


Figure 13: The confusion matrix of warblers

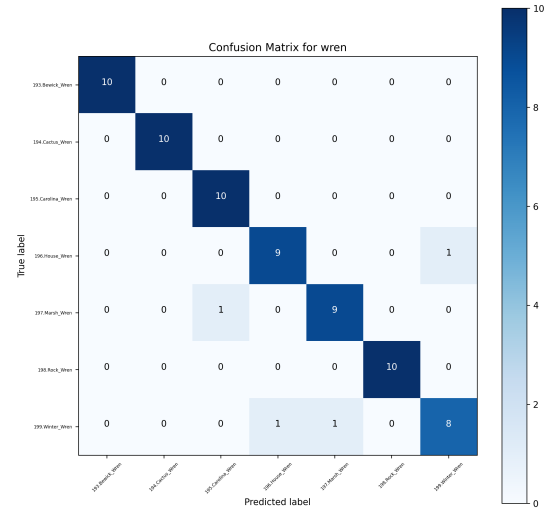


Figure 15: The confusion matrix of wrens

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.

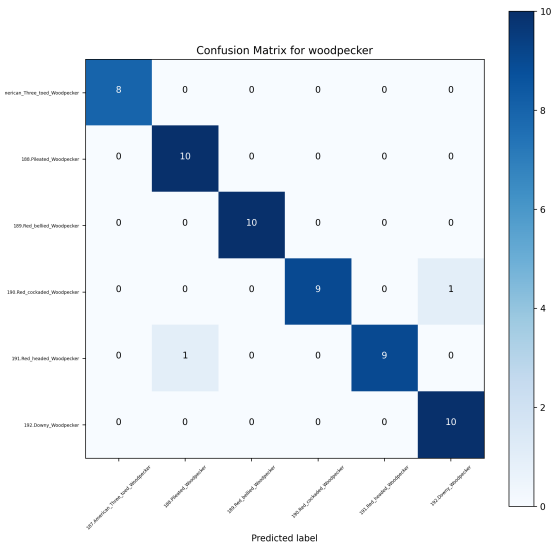


Figure 14: The confusion matrix of woodpeckers

```

136 return pss_module.PluginModel(backbone = backbone,
137                               return_nodes = return_nodes,
138                               img_size = img_size,
139                               use_fpn = use_fpn,
140                               fpn_size = fpn_size,
141                               proj_type = proj_type,
142                               upsample_type = upsample_type,
143                               use_selection = use_selection,
144                               num_classes = num_classes,
145                               num_selects = num_selects,
146                               use_combiner = use_combiner,
147                               comb_proj_size = comb_proj_size)
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

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


```
1 import matplotlib.pyplot as plt
2 import sklearn as sns
3 import sys
4
5 from utils.config_utils import load_yaml
6 from vis_utils import ImageLoader
7
8 def build_model(pretrained_path, str,
9               img_size: int,
10              fpu_size: int,
11              num_classes: int,
12              num_selects: dict,
13              use_fpu: bool = True,
14              use_selection: bool = True,
15              use_combiner: bool = True,
16              comb_proj_size: int = None):
17     from models.pia_module pia_module_eval import PluginModel
18
19 run_evaluation.py (31 local working changes · 1 of 6 changes)
20
21 use_selection: bool = True,
22 use_combiner: bool = True,
23 comb_proj_size: int = None):
24     from models.pia_module pia_module_eval.py import PluginModel
25     from models.pia_module pia_module_eval import PluginModel
26
27 model = \
28     PluginModel(img_size = img_size,
```

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.