110612117 張仲瑜 HW1 report

I. Introduction

1. Goal

In this task, we aim to achieve as higher accuracy as possible in a 100-class image classification competition, with Resnet based CNN models.

2. Environment

- ✧ Laptop: Windows / RTX4060 * 1
- ✧ Kaggle account * 3: Linux / Tesla T4 * 2

3. Core Idea

I conducted a series of experiments to test whether designs such as Cardinality, Squeeze-and-Excitation and various data augmentation can boost the performance compared to original Resnet.

II. Method

1. Data augmentation set [1]

- ✧ Gaussian Blur + RandomResizeCrop
  Since the size of the training images is not identical, we have to perform the down sampling on images to fit most of the pretrained models. To avoid aliasing and mitigating noises, Gaussian Blur is applied before RandomResizeCrop.

- ✧ (Gaussian Blur) + Sharpening
  After observing the dataset, I found that it's close to a multi-class (flower, grass, bird) fine-grained classification task.
  In my experience, increasing the sharpness of the data will help the model to identify tiny differences between them.

✧ Color Jittering + Greyscale

Since lots of grass and flowers are similar in colors, Color Jittering and Greyscale are applied to help the model learning features except for the color.

✧ Mixup training [2]

Severe overfitting happens when finetuning a deep network, the label-mixing strategy increases the difficulty of training data, preventing the training loss from reducing to tiny numbers within few epochs

2. Model Architecture

✧ Model name: SEresneXt101d_32x8d

Output size	SE-ResNeXt-101 (32×8d)		
144 × 144	conv, 7 × 7, 64, stride 2		
72 * 72	max pool, 3 × 3, stride 2		
72 × 72	conv, 1 × 1, 256 conv, 3 × 3, 256 conv, 1 × 1, 256 SE module	C=32	x3
36 x 36	conv, 1 × 1, 512 conv, 3 × 3, 512 conv, 1 × 1, 512 SE module	C=32	x4
18x18	conv, 1 × 1, 1024 conv, 3 × 3, 1024 conv, 1 × 1, 1024 SE module	C=32	x23
9x9	conv, 1 × 1, 2048 conv, 3 × 3, 2048 conv, 1 × 1, 2048 SE module	C=32	x3
1x1	dropout (p = 0.5)		
1x1	global average pooling, 100-d fc, softmax		

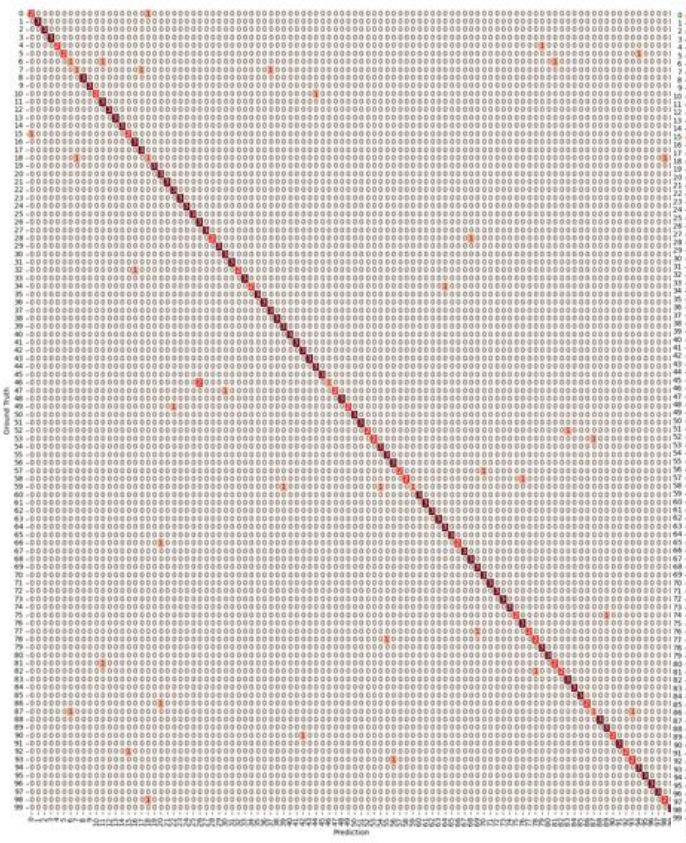
Total parameters: 93.6 M

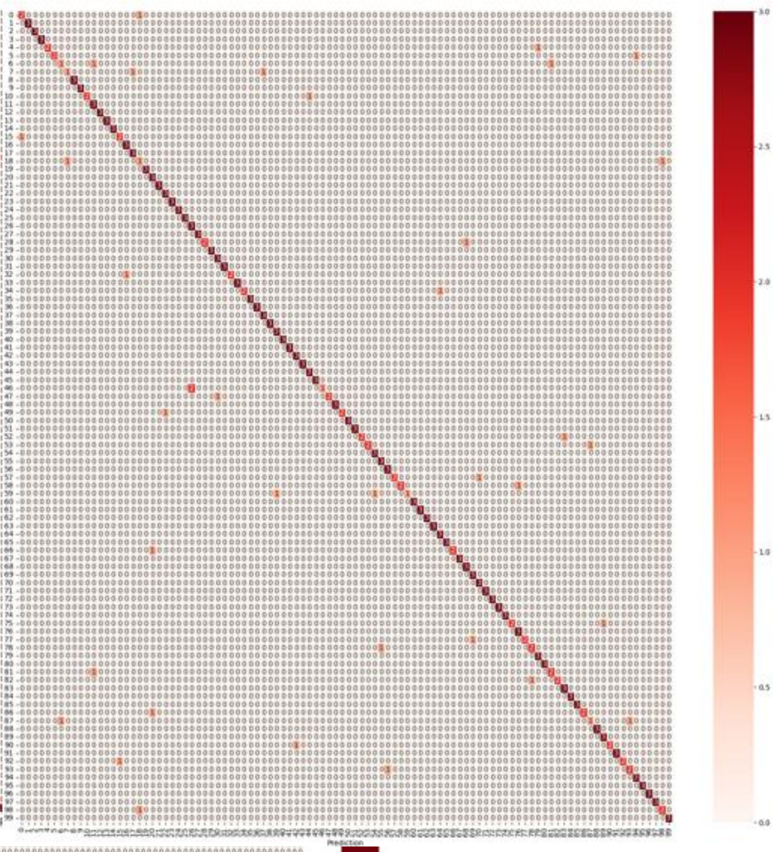
3. Hyperparameters (best one)
   - ✧ Pretrained on imagenet 1k
   - ✧ Training size: 224 * 224
   - ✧ Inferencing size: 320 * 320
   - ✧ Learning rate: Linear 10 warm-up [3] epochs to 5e-5 and * 0.5 decay factor whenever both validation accuracy and loss didn't decrease in 2 consecutive epochs.
     Start from epoch 27, I reset the learning rate = 1.248e-6 with the same decay factor since I observe there exists a extreme point.
   - ✧ Loss: cross entropy with counter-data unbalance weight
   - ✧ Optimizer: AdamW
   - ✧ Batch size: 32, [4]
   - ✧ Epoch: Max is 100 but with early-stopping patience = 7 epochs, best epoch = 29
   - ✧ Weight decay = 1e-3
   - ✧ Mix-up probability = 0.5, alpha = 1.3

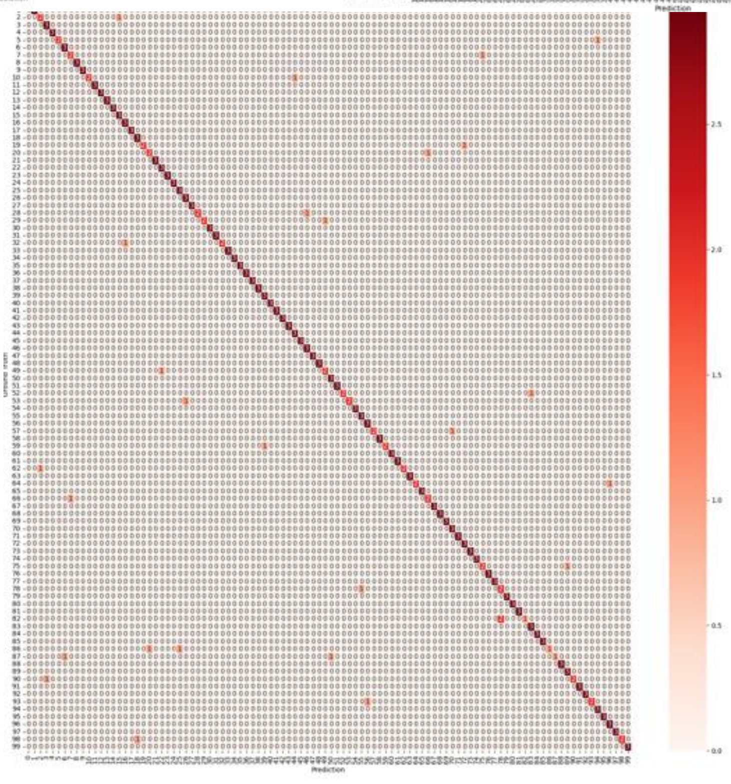
III. Result:

1. Best performance
   - ✧ Accuracy: 0.9633 on private testing
     Validation Loss: 0.6744, Validation Accuracy: 0.8933, Validation Precision: 0.9155, Validation Recall: 0.8933, Validation F1: 0.8891

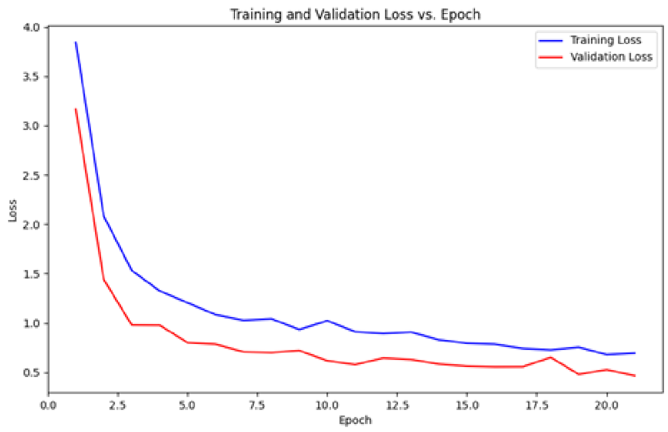
✧ Confusion Matrix (It breaks into three parts, I use strategy of continuing training with the same scalar, weight and optimizer state… in the first two one, and manually adjust the learning rate in the final state.):

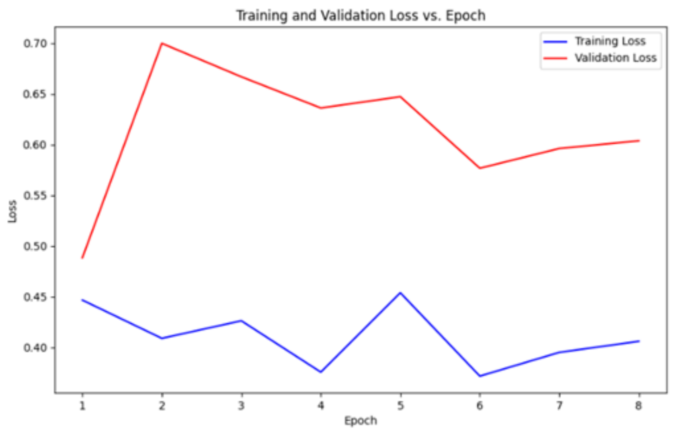


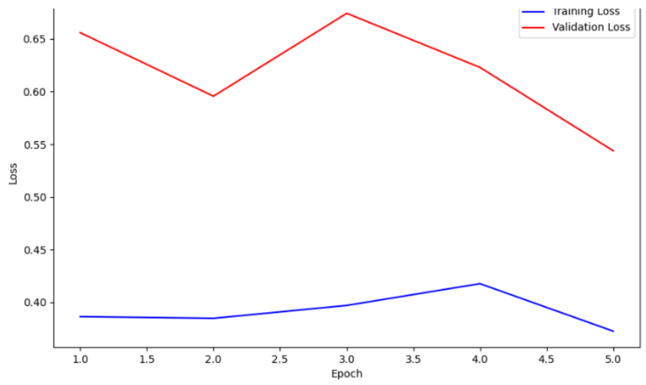




✧ Learning Curve (loss):







IV. Additional experiment:

1. Hypothesis: Group Convolution is effective. The original paper claims that it can boost the performance of the model, so I want to verify it.

✧ Concept of Group Convolution [5][6]:

The idea is similar to the inception module, using splitting, transforming and aggregating to reduce the error with the same number of parameters.

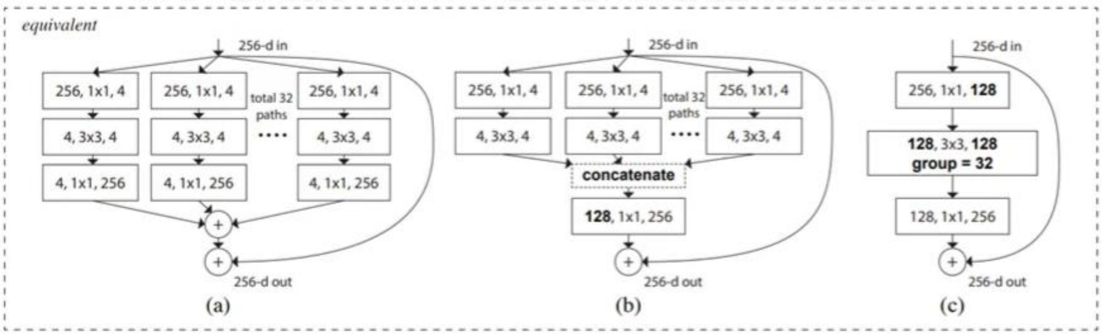


Fig from original paper

- ✧ Control variables:
  Training size = Inferencing size = 224 * 224
  Batch size = 20, dropout rate = 0.5, data augmentation set is used, learning rate = 1e-3, lr decay factor = 1e-3, no warm-up, no mix-up, weight decay = 1e-3

- ✧ Independent variable:
  Cardinality in every bottleneck block

- ✧ Result

Model Architecture	Accuracy
resnet50 with 1x64d	0.86
resneXt50 with 32x4d	0.88

- ✧ Implication
  As the author claims, the group convolution can boost model performance

2. Hypothesis: SE module is effective. The original paper claims that it can boost the performance of the model, so I want to verify it.

   - ✧ Concept of SE [7][8]:
     Adding a squeeze and excitation block (pooling -> fc -> non-linear activation function) at the end of every bottleneck block on top of the original resnet model, aiming to achieve channel-weight recalibration, aiming to let the model learn the relationship between channels and adjust their weight.

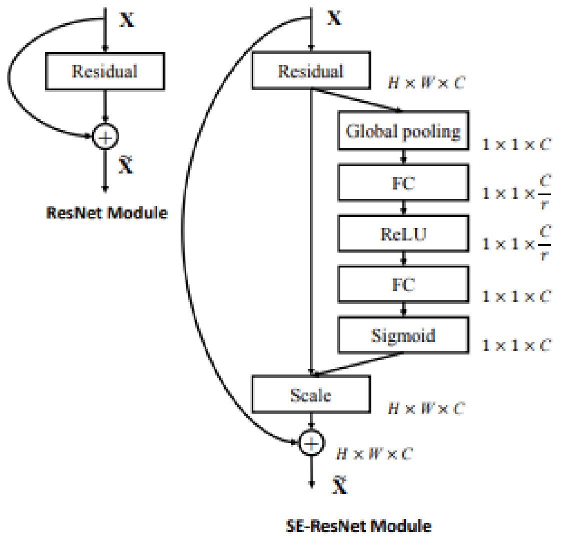


Fig from original paper

- ✧ Control variables:
  Training size = Inferencing size = 224 * 224
  Batch size = 20, dropout rate = 0.5, data augmentation set is used, learning rate = 1e-3, lr decay factor = 1e-3, no warm-up, no mix-up, weight decay = 1e-3

- ✧ Independent variable:
  SE block in every bottleneck block
- ✧ Result

Model Architecture	Accuracy
resneXt50_324D w/o SE	0.88
resneXt50_324D with SE	0.91

- ✧ Implication
  As the author of the claims, adding SE blocks is significantly effective.

3. Hypothesis: Using reciprocal of class-number as its weight in the Cross Entropy loss can improve the performance
   - ✧ Concept: Through my observation, I found that the dataset is unbalanced. Comparing two models with similar loss, the one with a higher recall rate performs better on the coda bench, so I set classes weight in CE as the reciprocal of their number of data, aiming to solve this problem.

   - ✧ Result

Loss formula	Accuracy
average weighted	0.9570
reciprocal version	0.9633

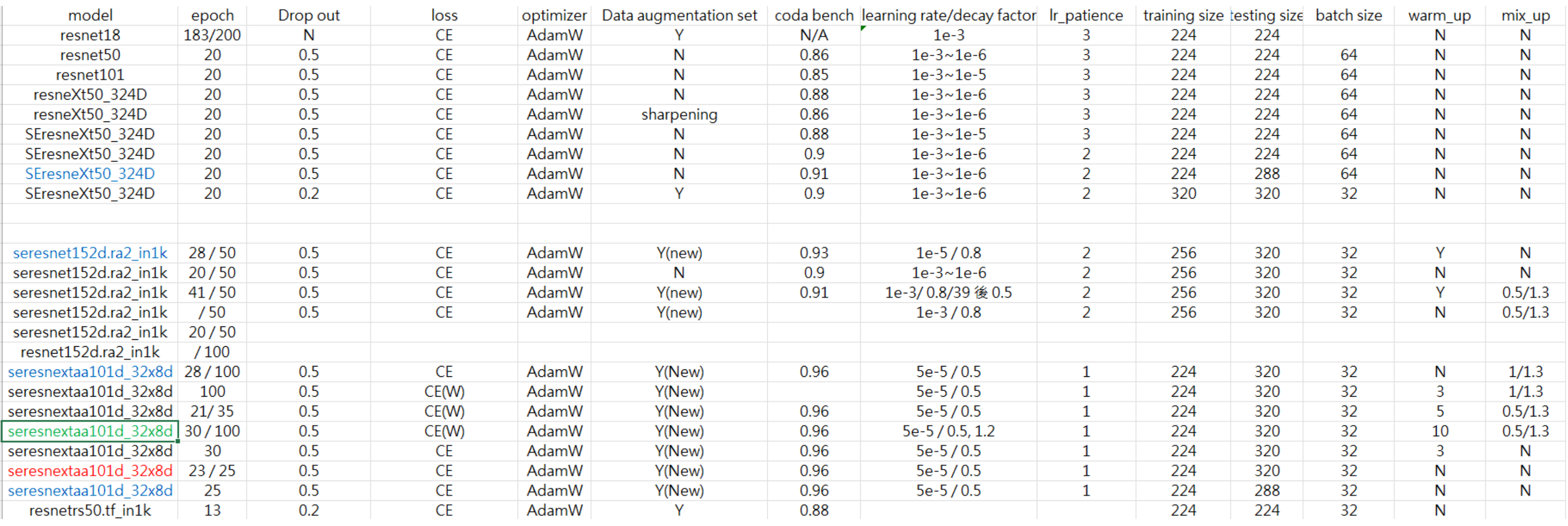
   - ✧ Implication:
     It works, dealing with those classes is important!

V. Github

https://github.com/sharkccy/NYCU_CV_2025_Spring

VI. Thoughts

I learned more about how to run a huge model within limited resources. At the very beginning, my poor code can only run a resnet50 model and taking 10 minute per epoch. I thought running a SEresneXt101 would be impossible, it turned out that properly garbage collection, CUDA management, properly num_workers and closing plots are key during training. It's an unforgettable experience to witness "threshing" happened in 2025. After those adjustments, I can run SEresneXt101 in 15 minutes per epoch.



model	epoch	Drop out	loss	optimizer	Data augmentation set	coda bench	learning rate/decay factor	lr_patience	training size	testing size	batch size	warm_up	mix_up
resnet18	183/200	N	CE	AdamW	Y	N/A	1e-3	3	224	224		N	N
resnet50	20	0.5	CE	AdamW	N	0.86	1e-3~1e-6	3	224	224	64	N	N
resnet101	20	0.5	CE	AdamW	N	0.85	1e-3~1e-5	3	224	224	64	N	N
resneXt50_324D	20	0.5	CE	AdamW	N	0.88	1e-3~1e-6	3	224	224	64	N	N
resneXt50_324D	20	0.5	CE	AdamW	sharpening	0.86	1e-3~1e-6	3	224	224	64	N	N
SEresneXt50_324D	20	0.5	CE	AdamW	N	0.88	1e-3~1e-5	3	224	224	64	N	N
SEresneXt50_324D	20	0.5	CE	AdamW	N	0.9	1e-3~1e-6	2	224	224	64	N	N
SEresneXt50_324D	20	0.5	CE	AdamW	N	0.91	1e-3~1e-6	2	224	288	64	N	N
SEresneXt50_324D	20	0.2	CE	AdamW	Y	0.9	1e-3~1e-6	2	320	320	32	N	N
seresnet152d.ra2_in1k	28 / 50	0.5	CE	AdamW	Y(new)	0.93	1e-5 / 0.8	2	256	320	32	Y	N
seresnet152d.ra2_in1k	20 / 50	0.5	CE	AdamW	N	0.9	1e-3~1e-6	2	256	320	32	N	N
seresnet152d.ra2_in1k	41 / 50	0.5	CE	AdamW	Y(new)	0.91	1e-3/ 0.8/39 後 0.5	2	256	320	32	Y	0.5/1.3
seresnet152d.ra2_in1k	/ 50	0.5	CE	AdamW	Y(new)		1e-3 / 0.8	2	256	320	32	N	0.5/1.3
seresnet152d.ra2_in1k	20 / 50												
resnet152d.ra2_in1k	/ 100												
seresnextaa101d_32x8d	28 / 100	0.5	CE	AdamW	Y(New)	0.96	5e-5 / 0.5	1	224	320	32	N	1/1.3
seresnextaa101d_32x8d	100	0.5	CE(W)	AdamW	Y(New)		5e-5 / 0.5	1	224	320	32	3	1/1.3
seresnextaa101d_32x8d	21/ 35	0.5	CE(W)	AdamW	Y(New)	0.96	5e-5 / 0.5	1	224	320	32	5	0.5/1.3
seresnextaa101d_32x8d	30 / 100	0.5	CE(W)	AdamW	Y(New)	0.96	5e-5 / 0.5, 1.2	1	224	320	32	10	0.5/1.3
seresnextaa101d_32x8d	30	0.5	CE	AdamW	Y(New)	0.96	5e-5 / 0.5	1	224	320	32	3	N
seresnextaa101d_32x8d	23 / 25	0.5	CE	AdamW	Y(New)	0.96	5e-5 / 0.5	1	224	320	32	N	N
seresnextaa101d_32x8d	25	0.5	CE	AdamW	Y(New)	0.96	5e-5 / 0.5	1	224	288	32	N	N
resnetrs50.tf_in1k	13	0.2	CE	AdamW	Y	0.88			224	224	32	N	

Fig. Some of my experiments

VII. Reference:

1. https://pytorch.org/vision/main/transforms.html
2. https://blog.csdn.net/xiaoxifei/article/details/90416997
3. https://reurl.cc/Z4jjkM
4. https://reurl.cc/QY66AM
5. https://liaowc.github.io/blog/resnext-structure/
6. https://reurl.cc/W0XX9L
7. https://meetonfriday.com/posts/79fdff34/
8. https://liaowc.github.io/blog/SENet-structure/
9. https://github.com/huggingface/pytorch-image-models/blob/main/results/results-imagenet.csv