Visual Recognition using Deep Learning HW1 Report

110550093 蔡師睿

1 Introduction

The task involves image classification with 100 classes, covering both animals and plants. The dataset consists of 21,024 images for training and validation, and 2,344 images for testing. This challenge imposes several constraints: only the ResNet[1] backbone is allowed, the model size must not exceed 100M, and the use of external data is prohibited. Rather than directly applying a state-of-the-art ResNet-based model for transfer learning, I incorporate various advanced techniques and models to achieve higher accuracy and surpass the strong baseline. The implementation can be found at *https://github.com/Ray-1026/Visual-Recognition/tree/main/HW1*.

2 Method

2.1 Models

ResNeSt[2] is built upon ResNet[1] by introducing split-attention blocks. This design enhances feature representation and generalization by integrating channel-wise attention within grouped convolutions. ResNeSt[2] surpasses ResNet[1] and other architectures like ResNeXt[3] on benchmark datasets such as ImageNet[4] and COCO[5], demonstrating strong generalization across various vision tasks. Furthermore, it significantly boosts the performance of downstream models, including Mask R-CNN[6], Cascade R-CNN[7], and DeepLabV3[8].

SE-ResNeXt[9][3] is an enhanced deep learning architecture that builds upon ResNeXt[3] by incorporating Squeeze-and-Excitation (SE) blocks[9]. ResNeXt[3] improves upon ResNet[1] by utilizing grouped convolutions for greater computational efficiency, while SE blocks[9] adaptively recalibrate feature maps through channel-wise attention to enhance representational power. By integrating these two techniques, the goal is to achieve superior performance in the image classification task.

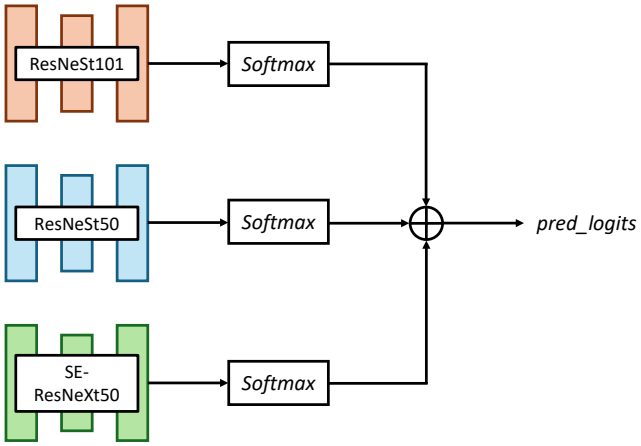

Figure 1: Inference with ensemble models

Ensemble models combine multiple individual models to improve predictive accuracy, robustness, and generalization. By leveraging the strengths of different models, ensembles help reduce overfitting, and often outperform single models in most deep learning tasks. As shown in Fig. 1, I employ three models for ensemble: *ResNeSt101*, *ResNeSt50*, and *SE-ResNeXt50*, with a total parameter count of 97.79M. During inference, the predicted logits from each model are passed through the softmax activation function,

$$softmax(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)},$$

and then summed to obtain the final predictions.

2.2 Data Augmentation

Data augmentation is a technique that artificially expands the training dataset by applying transformations to existing data. It helps improve model generalization, reduce overfitting, and enhance robustness without collecting additional data.

TrivialAugment[10] is a lightweight and efficient data augmentation technique that requires minimal hyperparameter tuning. Inspired by AutoAugment[11] and RandAugment[12], it eliminates the need for manually selecting augmentation policies. I adopt this technique for its simplicity, effectiveness, and low computational overhead.

Data Augmentation During Training. As shown in Algo. 1, I first apply random resized cropping to 224, followed by random horizontal and vertical flips, as well as TrivialAugment[10]. The image is then converted from PIL to a tensor and normalized. Additionally, I apply random erasing with a probability of 0.2.

Algorithm 1 Data Augmentation During Training

$train\_tfm = transforms.Compose([$
$\quad transforms.RandomResizedCrop(224),$
$\quad transforms.RandomHorizontalFlip(),$
$\quad transforms.RandomVerticalFlip(),$
$\quad transforms.TrivialAugmentWide(),$
$\quad transforms.ToTensor(),$
$\quad transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]),$
$\quad transforms.RandomErasing(p = 0.2, scale = (0.02, 0.33), ratio = (0.3, 3.3)),$
$])$

Algorithm 2 Data Augmentation During Testing

$test\_tfm = transforms.Compose([$
$\quad transforms.Resize((256, 256)),$
$\quad transforms.CenterCrop(224),$
$\quad transforms.ToTensor(),$
$\quad transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]),$
$])$

Data Augmentation During Testing. As shown in Algorithm 2, I first resize the images to 256 and then apply center cropping to 224. The image is subsequently converted from PIL to a tensor and normalized.

2.3 Loss Function

Instead of using cross-entropy loss, I apply label smoothing loss[13]. Label smoothing modifies the standard cross-entropy loss by smoothing the target labels, reducing overconfidence in the model's predictions and improving generalization. It is defined as follows:

$$\mathcal{L}_{\text{LS}} = -\sum_{i=1}^{K} q_i \log p_i,$$

where K is the number of classes, p_i is the predicted probability for class i, and q_i is the smoothed label distribution, given by:

$$q_i = \begin{cases} 1 - \epsilon + \frac{\epsilon}{K}, & \text{if } i \text{ is the correct class} \\ \frac{\epsilon}{K}, & \text{otherwise} \end{cases}$$

, where ϵ represents smoothing factor and is set to 0.1.

2.4 Hyperparameters

All of these models utilize pretrained weights trained on ImageNet-1K[4], with their codebases adapted from torchvision[14] and timm[15].

- random seed: 666

- epochs:
  - *ResNeSt101* and *ResNeSt50*: 50
  - *SE-ResNeXt50*: 80
- batch size: 64
- AdamW:
  - learning rate:
    - fc layer: $1e-3$
    - others: $1e-4$
  - weight_decay: $1e{-}3$
- Cosine Annealing Schedule:
  - T_max:
    - *ResNeSt101* and *ResNeSt50*: 50
    - *SE-ResNeXt50*: 80
  - eta_min: $1e-6$

3 Results

3.1 Training Curve

ResNeSt101

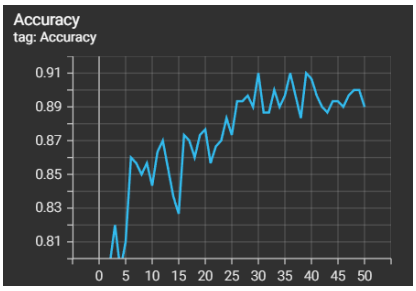

Figure 2: Validation Accuracy

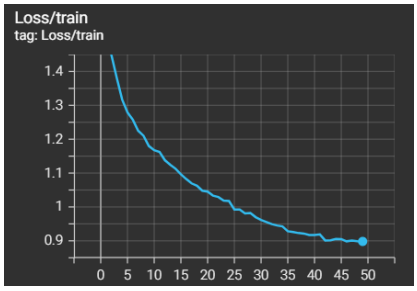

Figure 3: Training loss

ResNeSt50

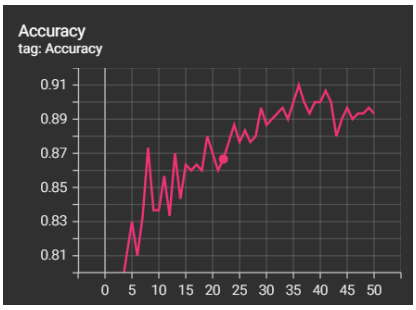

Figure 4: Validation Accuracy

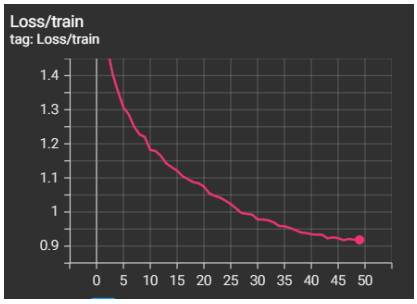

Figure 5: Training loss

SE-ResNeXt50

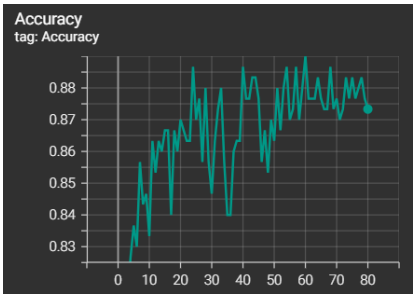

Figure 6: Validation Accuracy

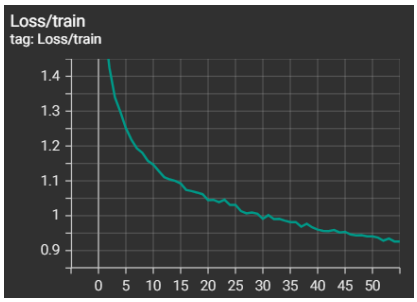

Figure 7: Training loss

3.2 Public score

ensemble.zip	2025-03-17 19:03	Finished	0.95

Figure 8: Public score on Codabench

4 Additional Experiments

4.1 Comparison

Model	#Params (M)	Public Score
SE-ResNeXt50	25.72	0.93
ResNeSt50	25.64	0.94
ResNeSt101	46.43	0.94
Ensemble	97.79	**0.95**

4.2 Ablation Study: TrivialAugment

As shown in Fig. 9, Fig. 10, and Fig. 11, incorporating TrivialAugment leads to a lower loss compared to the baseline. This indicates that TrivialAugment effectively

enhances the model's ability to generalize by introducing diverse transformations, reducing overfitting, and improving robustness during training.

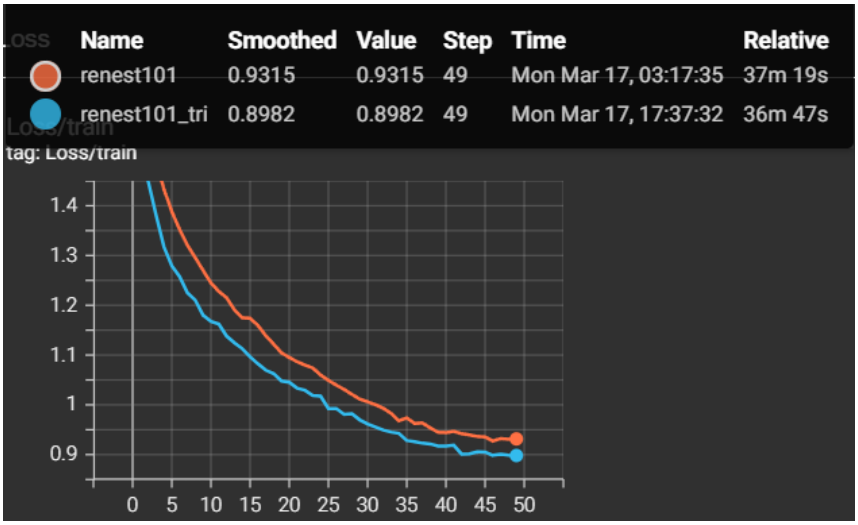

Figure 9: ResNeSt101

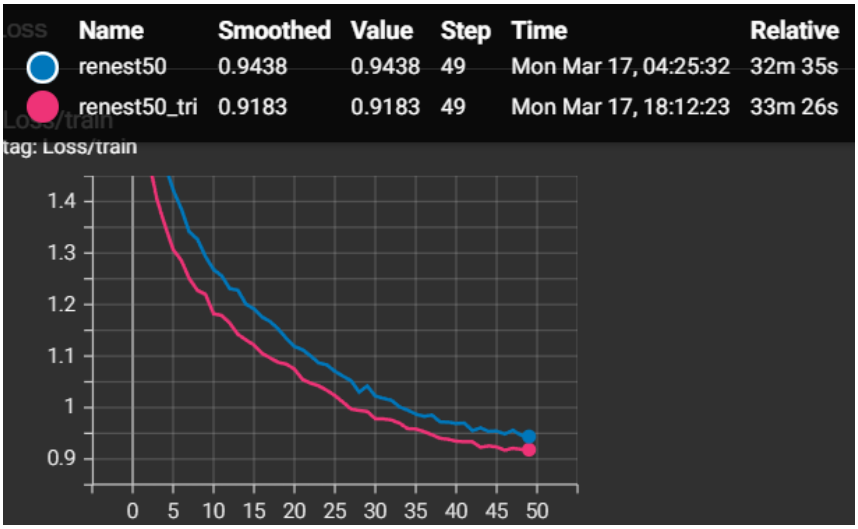

Figure 10: ResNeSt50

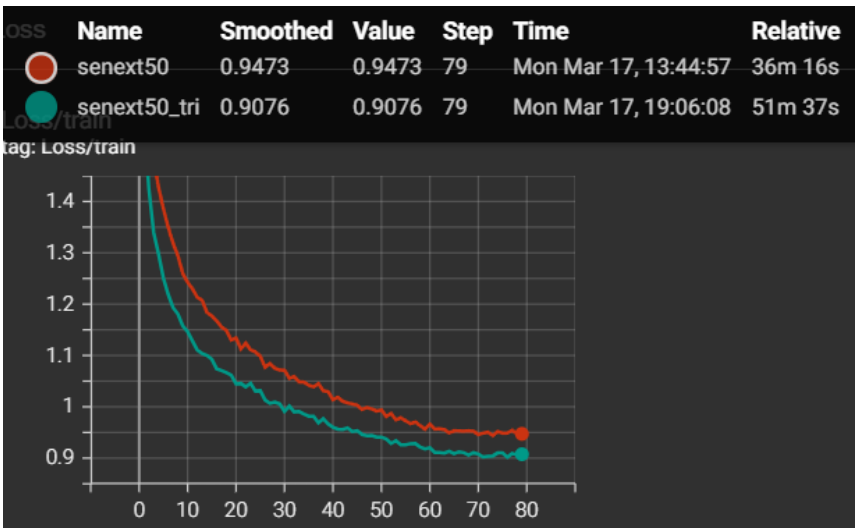

Figure 11: SE-ResNeXt50

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[2] H. Zhang, C. Wu, Z. Zhang, *et al.*, "Resnest: Split-attention networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 2736–2746.

[3] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[4] 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.

[5] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft coco: Common objects in context," in *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*, Springer, 2014, pp. 740–755.

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[7] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6154–6162.

[8] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[9] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[10] S. G. Müller and F. Hutter, "Trivialaugment: Tuning-free yet state-of-the-art data augmentation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 774–782.

[11] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," *arXiv preprint arXiv:1805.09501*, 2018.

[12] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 702–703.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[14] T. maintainers and contributors, *Torchvision: Pytorch's computer vision library*, `https://github.com/pytorch/vision`, 2016.

[15] R. Wightman, *Pytorch image models*, `https://github.com/rwightman/pytorch-image-models`, 2019. DOI: `10.5281/zenodo.4414861`.