VRDL HW1: Image Classification Report

Yi-Hsiang Ho, 111550106

1. Introduction

The task is to classify creature images into 100 different categories using ResNet. The core idea of my work is to leverage advanced data augmentation techniques, MixUp and CutMix, and ensemble learning to improve the model's performance. ResNeXt is chosen to be the backbone of the model. GitHub repository is available at:

1.1. ResNet

ResNet (Residual Network) is a deep learning architecture introduced by Microsoft in 2015 to address the vanishing gradient problem in very deep neural networks. The main contribution is its residual block. It tries to learn the residual, or difference, between layers by skip connections, which bypass one or more layers. The method allows the network to learn identity mappings.

1.2. ResNeXt

ResNeXt is an improved variant of ResNet, introduced by Facebook AI in 2017, which enhances performance by separating residual blocks into multiple small branches. Instead of simply increasing depth or width, ResNeXt introduces the concept of "cardinality," which refers to the number of parallel convolutional paths in each residual block. This design improves feature learning while maintaining computational efficiency. [3]

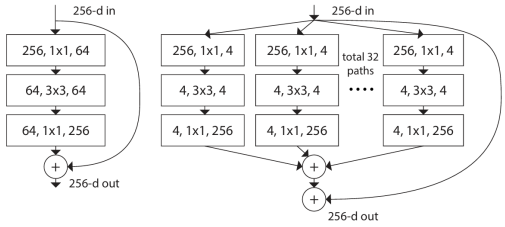

Figure 1. **Comparison of ResNet and ResNeXt architecture.** Left: ResNet. Right: ResNeXt. Image source: [3].

1.3. Mixup-Cutmix

Mixup [5] and CutMix [4] are data augmentation techniques that improve model generalization. Mixup blends two images and their corresponding labels through linear interpolation, while CutMix replaces a region of one image with a patch from another and mix the labels. Both methods enhance robustness, reduce overfitting, and improve classification accuracy by promoting diverse feature learning.

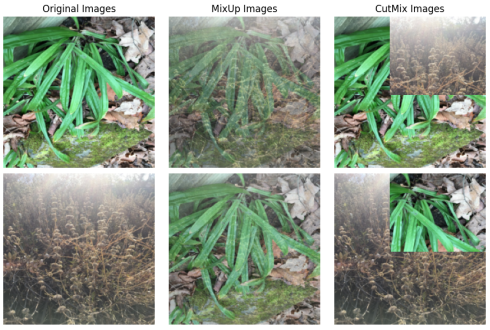

Figure 2. **Visualization of Mixup and CutMix.**

1.4. K-fold Cross Validation & Ensemble learning

Cross-validation is a technique used to evaluate model performance by training on different subsets of the data. It helps to reduce overfitting and improve generalization. K-fold cross-validation divides the data into K subsets, or folds. The model is trained on $K-1$ folds and validated on the remaining fold. This process is repeated K times, with each fold serving as the validation set once.

Ensemble learning combines multiple models to improve performance. It averages the predictions of multiple models to make the final prediction. This technique helps to reduce variance, increase accuracy, and improve model robustness.

2. Method

2.1. Data Pre-processing

Data preprocessing plays a crucial role in the training pipeline. Basic augmentations such as random horizontal flip and rotation are applied before training. Random resized cropping is also added to both increase robustness and meet the model's input requirements. Then, Mixup or CutMix is randomly chosen as an additional data augmentation technique to improve the diversity of the training dataset and smooth the labels. Note that no data augmentations are performed during validation or testing. The only preprocessing in these period is resizing to 224x224.

2.2. K-fold and Test-Time Ensemble

To further enhance model performance, I leverage the concept of K-fold. In this work, I use K-fold to generate K datasets, each with a different training and validation set, and train K sub-models on these datasets. In test-time ensemble, I average the predictions of these K sub-models to generate the final prediction. This technique helps to improve generalization and enhance model performance.

2.3. Model Architecture

The model is built upon ResNeXt-101 64x4d, which has been pretrained on ImageNet and the weights can be retrieved from PyTorch as `ResNeXt101_64X4D_Weights.IMAGENET1K_V1`. Also, to adapt the model to this task, the final fully connected layer is modified to output predictions for 100 classes.

2.4. Optimization

For optimization, AdamW is used with an initial learning rate of 1e-4. A cosine annealing warm restart scheduler dynamically adjusts the learning rate throughout training. Cross-entropy loss is employed as the loss function, and training is performed with a batch size of 64 over 80 epochs. Checkpoints are periodically saved, ensuring that the best-performing model based on validation accuracy is retained.

2.5. Hyperparameters

The hyperparameter details are listed below:

- Learning rate: 1e-4
- Optimizer: AdamW
- Scheduler: CosineAnnealingWarmRestarts
- Loss Function: CrossEntropyLoss
- Batch Size: 64
- Epochs: 80
- Number of Folds (K): 5

3. Results

In this section, I compare the performance of each component in the model. The details of each method are listed:

- ResNeXt: ResNeXt-101 64x4d pretrained on ImageNet.
- 5-Fold: Train ResNeXt on 5 different dataset seperated by 5-fold.
- Mixup & Cutmix: ResNeXt with Mixup or CutMix data augmentation.
- All: Combines all the techniques mentioned in Sec. 2.

The results are shown in Tab. 1. All the results here are evaluated by choosing the best checkpoint based on the validation accuracy. Mixup & Cutmix significantly improve the performance compared to the ResNeXt backbone, indicating the importance of these data augmentation techniques. The 5-fold approach also enhances accuracy by training on different subsets of the dataset. The best model achieves 95.22% accuracy on the public test set and 95.14% on the private test set. The validation accuracy for 5-Fold and All is 100% since these two method have already seen the validation set during training.

Method	Val	Test pub.	Test priv.
ResNeXt	0.9000	0.9317	0.9266
5-Fold	1.0000	0.9386	0.9292
Mixup & Cutmix	0.8933	0.9497	0.9437
all	1.0000	**0.9522**	**0.9514**

Table 1. **Accuracy results of different models.** "Val" refers to validation accuracy. "Test pub." and "Test priv." refer to public and private test set accuracy, respectively. Gray words indicate that the model has seen the validation set during training. Thus the validatio accuracy cannot be used to evaluate the model.

The training loss curve and validation accuracy curve are shown in Fig. 3 and Fig. 4, respectively. For those methods without mixup and cutmix. The training loss decreases steadily over time, while the validation accuracy increases. The model converges after approximately 50 epochs, with no signs of overfitting. The model with mixup and cutmix shows an unstable training loss curve, which may be due to the augmentation. However, the validation accuracy curve is still increasing, indicating that the model is learning effectively. It may increase the model's robustness and generalization.

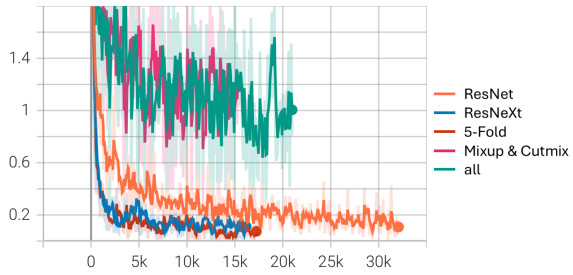

Figure 3. **Training loss curve.** Only a single sub-model is selected for 5-Fold and All to better visualize.

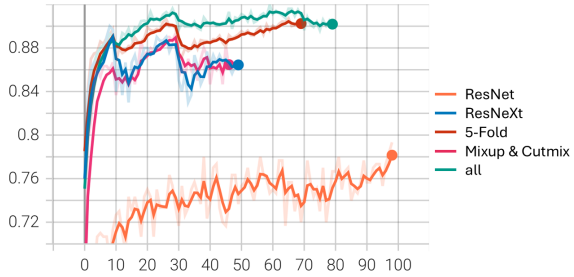

Figure 4. **Validation accuracy curve.** The selection of those two curve is the same as Fig. 3.

Other Experiments

The Selection of Backbone

The selection of the backbone is crucial for the model's performance. I've conducted experiments with different backbones, including a variant of ResNet, ResNeXt and ResNeSt [6]. The results are shown in Fig. 5. While ResNeSt achieves the highest validation accuracy, it performs poorly on the test set. ResNeXt is the most stable backbone, with consistent performance across all datasets. Therefore, I choose ResNeXt as the backbone for the following experiments.

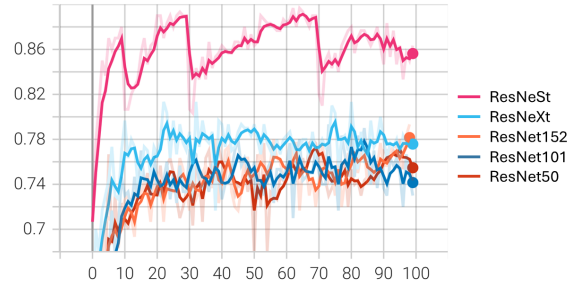

Figure 5. **Validation accuracy of different backbones.** The results are evaluated by choosing the best checkpoint based on the validation accuracy.

I've also experimented with different tricks, such as focal loss [2] , image normalization (normalize the image before feeding into the model), trivial data augmentation, and random data augmentation [1]. All these experiments are based on the backbone, ResNeXt. The results are shown in Tab. 2. However, none of these tricks improve the model's performance. Therefore, they are not included in the final model.

Also, different strategies for combining test-time ensemble are tested. It includes output average, output probability average (softmax), and majority vote. The results are shown in Tab. 3. The output average achieves the best performance, so it is used in the final model.

Trick	Val	Test pub.	Test priv.
Backbone	0.9000	0.9317	0.9266
Focal loss	0.8933	0.9326	0.9104
Normalization	0.8833	0.9138	0.9078
Random Aug.	0.8933	0.9215	0.9078
Trivial Aug.	0.9033	0.9326	0.9317

Table 2. **Accuracy results of different tricks.** The red background indicates a decrease in accuracy, while the blue one indicates an increase.

Trick	Test pub.	Test priv.
Average	**0.9522**	**0.9514**
Softmax avg.	0.9514	0.9480
Vote	0.9480	0.9445
Single sub-model	0.9420	0.9300

Table 3. **Accuracy results of different ensemble strategies.** The bold words indicate the best performance.

References

[1] Transforming and augmenting images - torchvision main documentation. 3

[2] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 3

[3] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 1

[4] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019. 1

[5] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 1

[6] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Muller, R. Manmatha, Mu Li, and Alexander Smola. Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955*, 2020. 3