NYCU Visual Recognition using Deep Learning
Homework 1

[312551185], [郭晏涵]

GitHub Link:

https://github.com/slovengel/312551185_HW1

Introduction:

This project aims to tackle an image classification problem using ResNet as the model backbone. The dataset consists of RGB images, with 21024 samples for training and validation and 2344 samples for testing. The goal is to predict the corresponding object category ID from 100 different categories.

To enhance model performance, I employed several key techniques, including data augmentation, transfer learning, adaptive learning rate scheduling, and model ensembling. The core idea of my method is to ensemble three models based on ResNeXt as the backbone, forming a more robust and accurate classifier.

The proposed model achieved 93% accuracy in the CodaBench competition (on public leaderboard), demonstrating its effectiveness in handling the classification task.

Method:

1. **Data pre-processing**

   For the entire dataset, I resize the images to (224, 224) to match the ResNeXt input size. The original train/validation split was used. To enhance model performance, different transformation pipelines are applied to the training and validation datasets.

   For the training data, augmentations such as random cropping, horizontal flipping, affine transformations, and perspective distortion are applied to improve robustness. Images are then converted to tensors and normalized using ImageNet mean and standard deviation. Meanwhile, the training data is shuffled to ensure random sampling.

   For the validation data, only resizing, tensor conversion, and normalization are applied for consistency.

2. **Model architecture**

   Three model variants were used, all based on the ResNeXt backbone with modifications to adapt to the dataset's 100-class classification task. The following are the detailed architectures:

Model Name	Modifications	Model Size
ResNeXt	Replaced the final fully connected layer (2048 → 100)	86M
ResNeXt_freeze	Same as ResNeXt, with frozen lower layers	86M
ResNeXt_ViT	Integrated WinKawaks/vit-tiny-patch16-224[1] in layer 4, modified fully connected layer (2048 → 100)	92M

3. Hyperparameters

The three models are trained using **SGD** with momentum of 0.9 and a **batch size** of **32** for at leaset 50 epochs.

CosineAnnealingLR scheduler is applied to adjust the learning rate dynamically throughout training. It creates a smooth and continuous decay of the learning rate without interruptions.

For ResNeXt and ResNeXt_ViT, the learning rate starts at **1e-3** in the first epoch and gradually decreases to **1e-6** by the final epoch.

For ResNeXt_freeze, the learning rate follows the same **1e-3** to **1e-6** schedule within each training section. At the beginning of each section, specific layers are unfrozen to allow progressive fine-tuning.

Additional experiments:

1. Gradual unfreezing

This experiment is conducted on the ResNeXt_freeze model. Gradual unfreezing is a transfer learning strategy where a pretrained model is fine-tuned by progressively unfreezing its layers. This allows the model to retain low-level features from pretrained layers while adapting higher-level features to the new dataset.

The following are the experiment results:

Epoch	Action	Best Training Acc.	Best Validation Acc.
1-10	Unfreeze the final fully connected layer	57.49	49.00
10-20	Unfreeze layer 4	85.02	79.66
20-30	Unfreeze layer 3	92.88	86.00
30-50	Unfreeze layer 2 and 1	95.85	85.33

Analysis:

Training only the feature extractor (the final fully connected layer) does not achieve satisfactory classification performance. However, after unfreezing layer 4, the model's performance improves significantly, indicating that the higher-level features of this dataset differ from those of the pretrained model’s dataset.

After unfreezing layers 3, 2, and 1, the accuracy improves only slightly. This suggests that the low-level features of the two datasets are likely highly similar. Notably, the best validation accuracy decreases after unfreezing layers 2 and 1, which may be due to the small validation dataset size, causing overfitting.

Based on this analysis, it may be beneficial to keep layers 2 and 1 frozen while increasing the number of training epochs to further improve accuracy.

2. Ensemble of models

This experiment is conducted on the ensemble model. According to [2], an ensemble of multiple CNN models can significantly enhance prediction accuracy, surpassing the performance of any individual model within the ensemble. The design ideas and example codes are based on [3]. The detailed implementation is as follows:

1. Train the three models separately—ResNeXt, ResNeXt_freeze, and ResNeXt_ViT.
2. Feed the same input into each model to obtain three separate outputs.
3. Use torch.cat() to concatenate the three output tensors and use them as input for training a linear classifier (100×3 → 100).

The following are the experiment results:

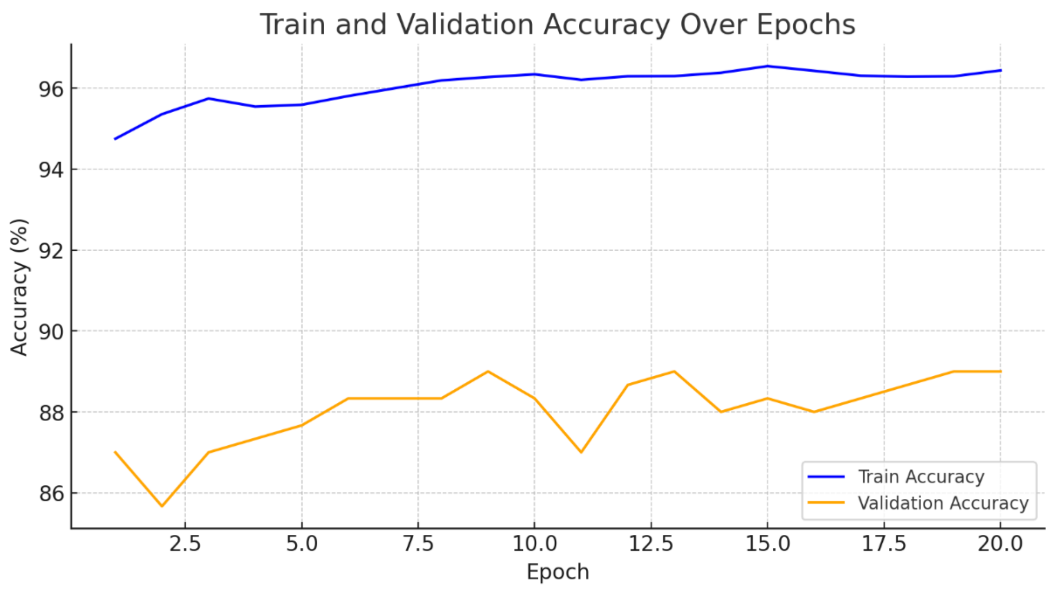
Model Name	**Best Validation Acc.**
ResNeXt	87.33
ResNeXt_freeze	86.00
ResNeXt_ViT	87.66
ensemble (train for 20 epochs)	89.00

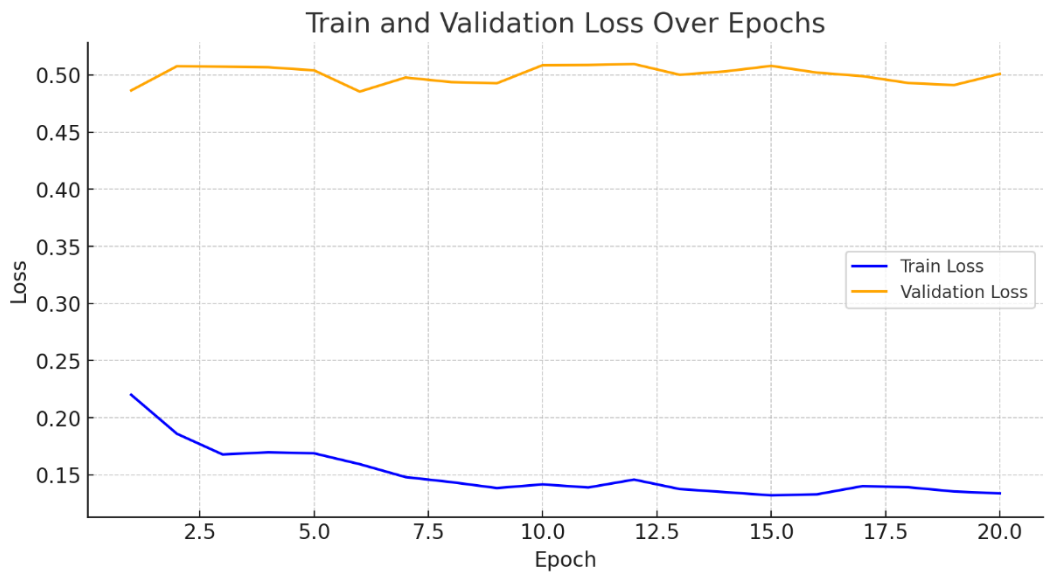
Analysis:

The ensemble model outperforms each individual model, achieving the highest validation accuracy of 89 %. This improvement demonstrates that combining multiple models captures diverse features and reduces prediction errors. The slight accuracy gains suggest that the three models complement each other, but their learned features are not entirely independent. Further improvements could be explored by incorporating additional models with different architectures or optimizing the weighting strategy for combining predictions.

Results:

Based on the experimental results, neither adding a ViT block nor freezing certain layers led to significant improvements. Since the ensemble model achieved the highest training accuracy (96.43%), it was selected as the final model for inference. The following are the learning curves for training the ensemble model.





Code Reliability:

I lint the code using Flake8 to follow the PEP8 instructions.

References:

[1] https://huggingface.co/WinKawaks/vit-tiny-patch16-224
[2] https://www.sciencedirect.com/science/article/pii/S1319157823000228
[3] https://github.com/alexppppp/tinyimagenet-classification-ensemble