Machine Learning Final Project: Leaf Recognition

Zijie Chen
3230106106
Department of Computer Science and Technology
`jazzy.chen@zju.edu.cn`

Abstract

Leaf classification is a fundamental task in computer vision with applications in botany and ecology. This report presents my approach to the Kaggle "Classify Leaves" competition with 18,353 training images across 176 species. I propose an ensemble framework using EfficientNet-B4 and ResNet variants (ResNet50d, ResNet200d) trained with strong augmentation, mixup/cutmix (for EfficientNet), EMA, label smoothing, and loss-based data cleaning followed by low-learning-rate fine-tuning. Through rigorous experimentation with 5-fold cross-validation, model averaging, and optional test-time augmentation (TTA), I show that ensembling diverse backbones yields the best performance. My best public leaderboard score is 0.9879 and my best private leaderboard score is 0.9884.

1 Introduction

Plant species identification is a crucial problem in biodiversity conservation and agriculture. Traditional methods rely on expert botanists, which is time-consuming and not scalable. The "Classify Leaves" competition on Kaggle challenges participants to build automated systems for classifying 176 categories of leaves. The key challenges include high inter-class similarity, intra-class variance due to lighting/pose, and mild label noise. Fine-grained recognition depends on subtle shape and venation cues, so high-resolution inputs and strong regularization are essential for robust generalization.

In this project, I aim to develop a robust deep learning pipeline for leaf classification. My contributions are:

1. Implementation of a flexible training pipeline supporting multiple backbones (EfficientNet, ResNet) via the `timm` library and 5-fold stratified cross-validation.

2. Application of strong augmentations (RandomResizedCrop, flips, rotation, color jitter), regularization (label smoothing, EMA), and mixup/cutmix (EfficientNet) to mitigate overfitting.

3. Loss-based data cleaning with a two-stage clean-data fine-tuning protocol to improve CV stability.

4. A comprehensive evaluation of single models and ensemble strategies (with/without TTA), plus ablations on learning-rate sweeps and mixup removal.

2 Methodology

2.1 Data Preprocessing and Augmentation

The dataset consists of leaf images which are resized to fixed resolutions (e.g., 380×380 or 512×512) depending on the model architecture. I use the `albumentations` library for image transformations.

Training Augmentations: To handle the fine-grained nature of the task and improve model robustness, I apply a strong augmentation pipeline:

- `RandomResizedCrop`: Randomly cropping and resizing the image (scale 0.8–1.0, ratio 0.8–1.2) to encourage the model to focus on different parts of the leaf.
- `HorizontalFlip` and `Rotate`: Geometric transformations (rotation up to 30 degrees) to handle pose variations.
- `ColorJitter`: Adjusting brightness, contrast, saturation, and hue to improve lighting invariance.
- `Normalize`: Standardizing images using ImageNet mean and standard deviation.

Validation Preprocessing: I resize to a slightly larger size and apply center crop to reduce scale variance at evaluation time.

Mixup and Cutmix: I enable Mixup/Cutmix for EfficientNet-B4 (mixup alpha 0.2, cutmix alpha 0.1). For ResNet baselines and clean fine-tuning runs, I disable Mixup/Cutmix to keep optimization stable.

2.2 Model Architectures

I leverage transfer learning by fine-tuning models pre-trained on ImageNet. The models are selected to balance strong fine-grained recognition with architectural diversity for ensembling.

EfficientNet-B4 (Noisy Student). EfficientNet scales depth, width, and resolution with a compound coefficient ϕ:

$$d = \alpha^{\phi}, \quad w = \beta^{\phi}, \quad r = \gamma^{\phi}, \quad \text{s.t. } \alpha\beta^2\gamma^2 \approx 2.$$

This yields an efficient model that preserves accuracy under a fixed compute budget. I use input size 380×380 and replace the classifier with a linear layer to 176 classes.

ResNet50d and ResNet200d. ResNets use residual connections to ease optimization in deep networks. A residual block computes

$$y = x + F(x; \theta), \tag{1}$$

where F is a stack of convolution, normalization, and nonlinearity. The "d" variants modify the stem and downsampling to improve transfer performance. I use input size 512×512 and replace the final classification head with a 176-way linear layer.

2.3 Training Strategy

I use 5-fold stratified cross-validation. The training objective and regularization choices are designed to improve generalization under label noise and fine-grained visual variability.

Objective and label smoothing. Let the dataset be $\{(x_i, y_i)\}_{i=1}^{N}$ with $K = 176$ classes, $y_i \in \{0,1\}^K$ one-hot, logits $z = f_\theta(x)$, and $p = \mathrm{softmax}(z)$. The cross-entropy loss is

$$L_{\mathrm{CE}}(y, p) = -\sum_{k=1}^{K} y_k \log p_k. \tag{2}$$

I apply label smoothing with ε by using $y^{\mathrm{ls}} = (1-\varepsilon)y + \varepsilon/K$ and $L_{\mathrm{LS}} = L_{\mathrm{CE}}(y^{\mathrm{ls}}, p)$. This reduces overconfidence and improves calibration. I use $\varepsilon = 0.1$ for ResNet baselines and $\varepsilon = 0.05$ for fine-tuning.

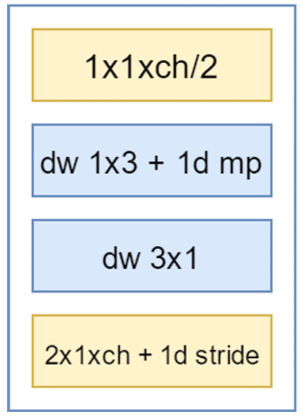

Figure 1: EfficientNet-B4 block diagram.

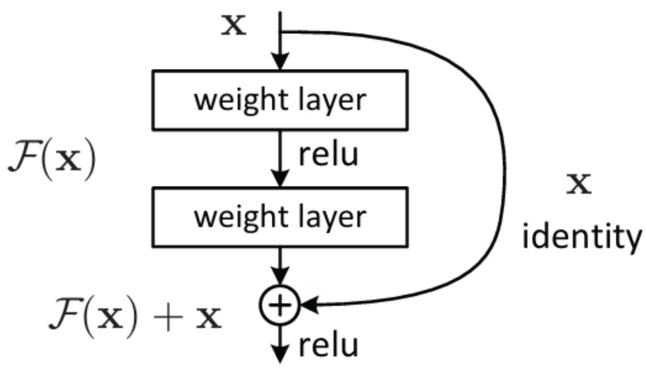

Figure 2: Residual block architecture diagram.

Mixup and CutMix. To further regularize EfficientNet-B4, I use mixup and cutmix. For mixup, sample $\lambda \sim \mathrm{Beta}(\alpha, \alpha)$ and form

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j, \quad \tilde{y} = \lambda y_i + (1-\lambda)y_j, \tag{3}$$

then optimize $L_{\mathrm{CE}}(\tilde{y}, p(\tilde{x}))$. For cutmix, sample a binary mask $M \in \{0,1\}^{H\times W}$ and set

$$\tilde{x} = M \odot x_i + (1-M) \odot x_j, \quad \tilde{y} = \lambda y_i + (1-\lambda)y_j, \quad \lambda = \frac{1}{HW}\sum M. \tag{4}$$

Mixup/cutmix are disabled for ResNet baselines and for clean fine-tuning to keep optimization stable.

Optimization and scheduling. I use AdamW with weight decay and gradient clipping, and a cosine annealing learning rate schedule. Given gradients g_t, AdamW maintains

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t, \quad v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2, \tag{5}$$

and updates parameters with decoupled weight decay,

$$\theta_{t+1} = \theta_t - \eta\frac{m_t}{\sqrt{v_t}+\epsilon} - \eta\lambda\theta_t. \tag{6}$$

I clip gradients by $g_t \leftarrow g_t \cdot \min(1, \tau/\|g_t\|_2)$. Cosine annealing sets

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})(1 + \cos(\pi t/T)). \tag{7}$$

Mixed precision accelerates training via FP16 compute with loss scaling: $L_s = sL$, $g_{\text{fp16}} = \nabla_{\theta_{\text{fp16}}} L_s$, $g_{\text{fp32}} = g_{\text{fp16}}/s$, and $\theta_{\text{fp32}} \leftarrow \theta_{\text{fp32}} - \eta g_{\text{fp32}}$.

EMA for stable evaluation. I maintain an exponential moving average of parameters,

$$\theta_t^{\text{ema}} = \beta \theta_{t-1}^{\text{ema}} + (1-\beta)\theta_t, \tag{8}$$

and evaluate with θ_T^{ema} for EfficientNet-B4 and ResNet200d. EMA reduces variance and improves final generalization.

2.4 Inference, Ensembling, and TTA

At inference time, I mirror the validation preprocessing (resize + center crop + normalization) to reduce train-test shift. For each model configuration, I run all $F = 5$ fold checkpoints and average their predicted class probabilities, which smooths variance caused by different folds and yields a more stable prediction per model.

Probability averaging. For an input x, fold f, and model m, I denote probabilities as $p_{m,f}(x) = \mathrm{softmax}(f_{\theta_{m,f}}(x))$. The fold-averaged prediction is

$$\bar{p}_m(x) = \frac{1}{F} \sum_{f=1}^{F} p_{m,f}(x). \tag{9}$$

For an ensemble of M models, I use a uniform mean,

$$p_{\text{ens}}(x) = \frac{1}{M} \sum_{m=1}^{M} \bar{p}_m(x), \tag{10}$$

which is robust and emphasizes complementary representations across backbones.

Test-Time Augmentation (TTA). I evaluate a lightweight TTA using a horizontal flip $t(x)$ to probe invariance. Predictions are averaged across transforms,

$$p_{\text{tta}}(x) = \frac{1}{T} \sum_{t \in \mathcal{T}} p_{\text{ens}}(t(x)), \quad \mathcal{T} = \{\text{identity}, \text{hflip}\}, \tag{11}$$

and compared with the non-TTA baseline. In practice the gain is negligible, likely because the training augmentations already include flips and strong spatial transforms, and the ensemble itself reduces variance.

2.5 Data Cleaning

I rank training samples by their per-sample loss using out-of-fold (OOF) predictions from a strong baseline. The highest-loss samples are treated as potential label issues. I removed 97 samples out of 18,353 (creating `train_clean.csv` with 18,256 images), retrained models on the cleaned set, and then performed a low-learning-rate fine-tuning stage without mixup to refine decision boundaries.

Figure 3: Representative label issue samples identified by loss-based data cleaning. Each image shows the true label and the model's predicted label, demonstrating cases where the original labels may be incorrect or ambiguous.

3 Experiment Setup and Results

3.1 Setup

All experiments were conducted on a single GPU (RTX 4090, 24GB) using PyTorch. The dataset contains 18,353 images across 176 classes and was split into 5 stratified folds. I trained on both the full dataset and the cleaned subset, and evaluated models based on Top-1 Accuracy. Hyperparameters:

- Batch Size: 16 (EffNet-B4), 32 (ResNet50d), 8 (ResNet200d).
- Epochs: 20-25 for full training, 3-5 for fine-tuning sweeps.
- Learning Rate: 1e-3 (EffNet-B4), 2e-4 to 3e-4 (ResNet), and 5e-5 for clean fine-tuning.

3.2 Results

Tables 1–3 summarize the performance of single models, ensembles, and ablations on local Cross-Validation (CV) and the Kaggle Public/Private Leaderboard (LB). EffB4 denotes EfficientNet-B4.

Table 1: Single-model results on full vs. cleaned data.

Model	Data	Res	CV Acc	Public LB	Private LB
ResNet50d (baseline)	full	512	0.9744	0.9798	0.9818
ResNet200d (baseline)	full	512	0.9769	0.9802	0.9834
EfficientNet-B4 (baseline)	full	380	0.9777	0.9834	**0.9868**
EfficientNet-B4 (clean)	clean	380	0.9832	**0.9839**	0.9857
ResNet50d (clean)	clean	512	0.9812	0.9805	0.9845
ResNet200d (clean)	clean	512	0.9819	0.9802	0.9834
EfficientNet-B4 (clean ft)	clean	380	**0.9841**	0.9839	0.9866
ResNet50d (clean ft)	clean	512	0.9814	0.9809	0.9845
ResNet200d (clean ft)	clean	512	0.9821	0.9807	0.9841

Table 2: Ensemble results.

Ensemble	CV Acc	Public LB	Private LB	Notes
EffB4 + Res50d (baseline)	0.9767	0.9839	0.9882	no TTA
EffB4 + Res50d + Res200d (baseline)	0.9767	0.9834	0.9879	with/without TTA
EffB4 + Res50d (clean)	0.9832	**0.9879**	0.9843	best public
EffB4 + Res50d (clean ft)	**0.9841**	0.9839	**0.9884**	best private

Table 3: EfficientNet-B4 fine-tuning sweeps (CV only when LB is not available).

Setting	CV Acc	Public LB	Private LB
No-mixup fine-tune (5 ep)	0.9779	0.9827	0.9877
LR=1e-4, 3 ep	0.9778	-	-
LR=1e-4, 5 ep	0.9777	-	-
LR=5e-5, 3 ep	0.9775	-	-
LR=5e-5, 5 ep	0.9777	-	-

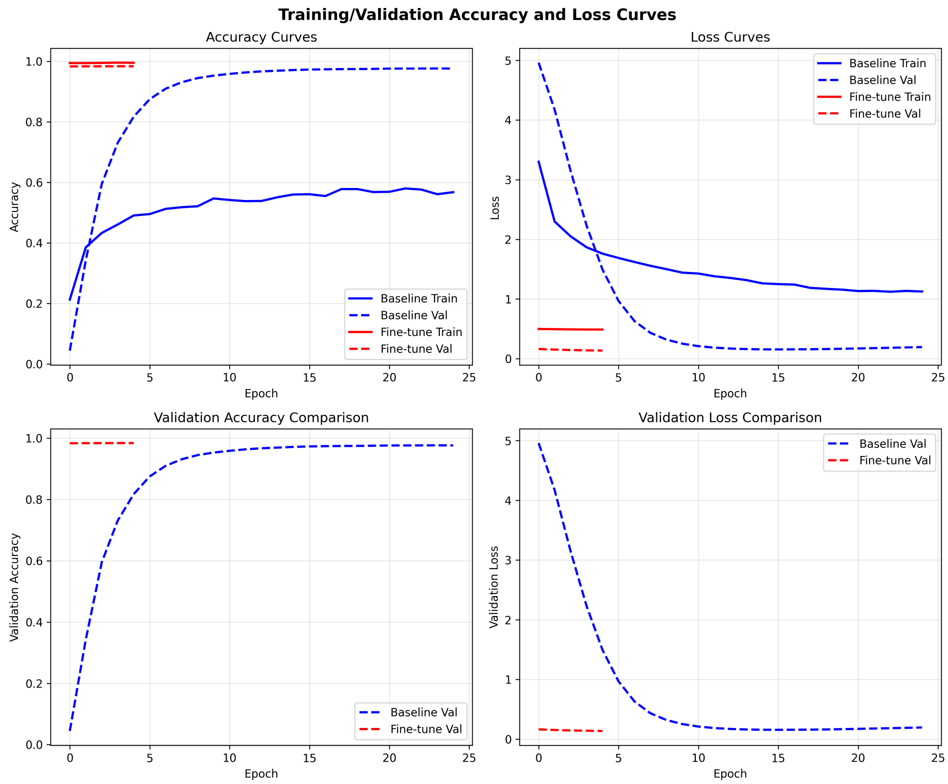

Figure 4: Training/validation accuracy and loss curves for baseline and clean fine-tuning runs.

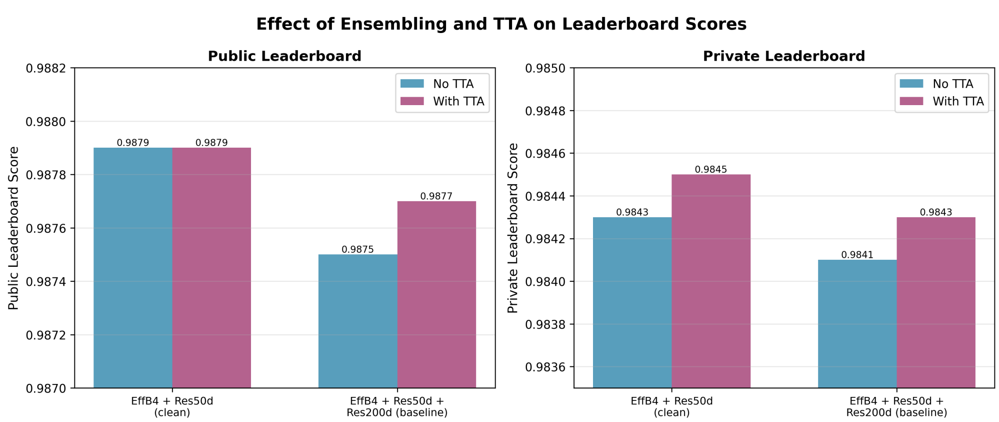

Figure 5: Bar chart showing the effect of ensembling and TTA on leaderboard scores. The chart compares ensemble results with and without test-time augmentation (TTA) for two ensemble configurations.

3.3 Analysis

Single Models: EfficientNet-B4 is the strongest single backbone, reaching a private LB of 0.9868. Training on cleaned data improves CV from 0.9777 to 0.9832, indicating that removing high-loss samples reduces noise and stabilizes learning.

Ensembling: Averaging predictions of EfficientNet-B4 and ResNet50d provides consistent gains, achieving 0.9882 private LB on the baseline ensemble. Adding ResNet200d does not improve the leaderboard. A plausible explanation is diminishing returns from similar feature representations, combined with a less stable optimization regime due to the small batch size forced by GPU memory limits. The ResNet200d runs use batch size 8, which can make batch-norm statistics noisier and reduce optimization efficiency compared with larger-batch baselines (e.g., 32 or 64). This likely blunts the extra capacity of ResNet200d.

Fine-Tuning and Sweeps: Clean-data fine-tuning yields the best private LB (0.9884) with a modest CV gain. Learning-rate sweeps produce only marginal changes, suggesting that the main gains come from data cleaning and ensembling rather than hyperparameter tuning.

Test Time Augmentation (TTA): Horizontal-flip TTA shows negligible improvement (differences within 0.0002). This may be because the training augmentations already include flips and strong spatial transforms, and the ensemble itself already reduces variance. As a result, TTA provides little additional diversity in predictions and is treated as optional to save inference time.

4 Conclusion

In this project, I successfully implemented a high-performance leaf classification system. By leveraging transfer learning with EfficientNet and ResNet architectures, combined with robust data augmentation, data cleaning, and ensemble strategies, I achieved best leaderboard scores of 98.79% (public) and 98.84% (private). Future work could explore Vision Transformers (ViT) or Swin Transformers to better capture global context in leaf structures.

Acknowledgments

I verify that all code and experiments are original. I acknowledge the open-source community for `timm` and `albumentations`. All code is available at `https://github.com/JazZyzJ/ML_project`.

References

[1] Tan, M. & Le, Q. (2019) EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *ICML*.

[2] He, K., Zhang, X., Ren, S. & Sun, J. (2016) Deep Residual Learning for Image Recognition. *CVPR*.

[3] Wightman, R. (2019) PyTorch Image Models. `https://github.com/rwightman/pytorch-image-models`.

[4] Yun, S., et al. (2019) CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features. *ICCV*.