This project applies ensemble learning techniques—**Bagging**, **Boosting**, and **Stacking**—on the CIFAR-10 image dataset using custom deep learning and machine learning models. The goal was to evaluate the performance improvements and integration challenges of ensemble methods using real-world computer vision data.

Dataset

- **Source**: CIFAR-10
- **Classes**: 10 image classes (airplane, automobile, bird, cat, etc.)
- **Preprocessing**: Images resized to 64x64, converted to tensors, normalized
- Link ; https://www.kaggle.com/c/cifar-10/
- 60k images of airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

Models Used

Bagging

- **Model**: LightCNN
- **Ensemble**: Trained 5 individual CNNs using bootstrapped data and aggregated via majority voting.
- **Output**: Models saved as `bagging_model_0.pth` to `bagging_model_4.pth`

Boosting

- **Model**: XGBoost
- **Features**: Flattened image tensors as input (64x64x3 = 12288 features)
- **Output**: Model saved as `boosting_xgboost_model.pkl`

Stacking

- **Base Learners**: LightCNN, SmallCNN, XGBoost
- **Meta Learner**: Logistic Regression (trained separately)
- **Feature Extraction**: Combined outputs from all base models
- **Output**: Stacked features saved in `stacking_features.npy` and labels in `stacking_labels.npy`

Workflow

1. Train individual bagging models (LightCNNs) and save `.pth` files
2. Train boosting model (XGBoost) on flattened images
3. Train SmallCNN and use it as a stacking base model
4. Extract softmax outputs from all models
5. Combine outputs as meta-features for the stacking classifier
6. Save features and labels to `.npy` files

Challenges Faced

1. **Repeated Model Retraining**
    - Issue: Models were being retrained on every run of the feature extraction script
    - Fix: Ensured model `.pth`/`.pkl` files are loaded in `eval()` mode without retraining logic in importable model scripts
2. **Shape Mismatch in XGBoost**
    - Issue: XGBoost expected flattened images of a specific shape (12288), but received mismatched input
    - Fix: Resized all images to 64x64 and flattened properly using `.view(images.size(0), -1)`
3. **Model File Overwrites**
    - Issue: Saved files like `stacking_smallcnn.pth` and `bagging_model_X.pth` were being overwritten
    - Fix: Removed auto-training in importable scripts and manually verified model loading paths
4. **Torch Warnings for Pickle**
    - Issue: Security warnings when using `torch.load`
    - Fix: Plan to switch to `weights_only=True` in future for safer model loading
5. **Efficiency in Feature Extraction**
    - Issue: Processing large batches for stacking was slow
    - Fix: Optimized dataloader and used `.no_grad()` context for inference-only usage

Outcomes

- Efficient and modular implementation of Bagging, Boosting, and Stacking
- Extracted robust meta-features for stacking using trained models
- Built a solid base for ensemble-based model evaluation with future integration into Flask-based APIs

File Outputs

- `bagging_model_*.pth` — Trained LightCNNs
- `boosting_xgboost_model.pkl` — Trained XGBoost
- `stacking_smallcnn.pth` — Stacking base CNN
- `stacking_features.npy`, `stacking_labels.npy` — Final feature and label files