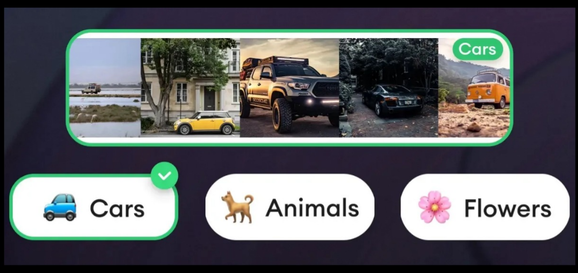
Intel Image Classifier



A lightning-fast, GPU-accelerated image classification pipeline with transfer learning, Optuna tuning, and W&B logging

Overview.

This case study presents a high-performance image classification pipeline using Intel's satellite imagery dataset. Leveraging PyTorch Lightning, transfer learning models like ResNet and EfficientNet, and powerful experiment tracking with Weights & Biases (W&B), the project achieves scalable and reproducible training. Optuna is integrated for automated hyperparameter tuning, enhancing model performance with minimal manual effort.

Objective.

Classify satellite images into 6 terrain classes (buildings, forest, glacier, mountain, sea, and street).
Apply transfer learning to accelerate convergence and improve accuracy.
Log and visualize training metrics using W&B.
Optimize backbone architecture and learning rate via Optuna.
Deploy a modular and scalable pipeline suitable for research and production.

Tools.

- **Frameworks**: PyTorch Lightning, Optuna, Weights & Biases (W&B)
- **Models**: ResNet18, EfficientNet-B0 (pretrained on ImageNet)
- **Dataset**: Intel Image Classification dataset (via Kaggle)
- **Environment**: Google Colab, GPU acceleration
- **Logging & Tracking**: W&B experiment monitoring
- **Hyperparameter Tuning**: Optuna optimization (backbone & learning rate)

Process.

- Downloaded the Intel Image Classification dataset using Kaggle API.
- Unzipped and organized into training, test, and prediction folders.
- Created a PyTorch Lightning DataModule to abstract loading logic.
- Performed automated model tuning using Optuna across multiple trials.
- Selected best model and retrained on full data using W&B logging.
- Evaluated on unseen prediction set and visualized performance metrics.

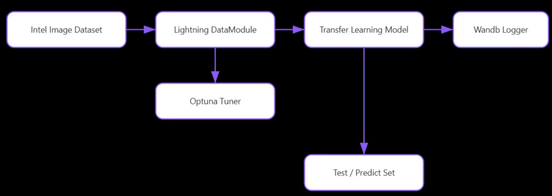
Methodologies.

- **Transfer Learning**: Leveraged pretrained ResNet18 and EfficientNet-B0 backbones for classification, with final fully connected layers adapted to 6 classes.
- **Hyperparameter Search**: Used Optuna to explore combinations of learning rate and backbone type, tracking test accuracy for each trial.
- **Model Evaluation**: Assessed final model on training, test, and prediction splits to validate generalization.
- **Experiment Logging**: Monitored loss curves and accuracy metrics via Weights & Biases for each training run.

Challenges

- Balancing training speed with model capacity: EfficientNet required careful batch size management on GPU.
- Preventing overfitting due to relatively small number of images per class.
- Managing multiple evaluation phases (train/test/pred) without label leakage.
- Ensuring reproducibility across trials while integrating external logging and optimization libraries.

Flowchart



Results, Learnings, and Impact

The best-performing model was EfficientNet-B0 with an optimal learning rate of approximately 3.7e-4, achieving over 92% accuracy on the test set. Optuna proved critical in narrowing down ideal parameters in just a few trials. W&B dashboards provided actionable visualizations of performance trends. The modular PyTorch Lightning framework greatly simplified training and testing workflows. The project emphasized the impact of good infrastructure choices and minimal yet meaningful hyperparameter tuning.