

Track 1 – Image Classification (Category 1: Pretrained Allowed)

Team: sangram-vaishnavi

1) Problem

We solve fine-grained single-label image classification across 102 categories. The model predicts one integer label (1–102) for each test image.

2) Data

We use only the provided competition data:

- train/images + train.csv (noisy labels)
- val/images + val.csv (clean labels for local evaluation)
- label_list.txt for valid labels (1–102)

No external datasets or internet resources are used.

3) Preprocessing & Augmentation

Input resolution: 384x384.

Training augmentations:

- RandomResizedCrop(384, scale=(0.6, 1.0))
- RandomHorizontalFlip(p=0.5)
- RandomApply(ColorJitter(0.25,0.25,0.25,0.1), p=0.5)
- RandomRotation(12 degrees)
- Normalize with ImageNet mean/std: mean=(0.485,0.456,0.406), std=(0.229,0.224,0.225)

Validation / inference preprocessing:

- Resize(int(384*1.14)) then CenterCrop(384)
- Same ImageNet normalization

4) Model

Backbone: ConvNeXt-Tiny (timm) pretrained on ImageNet.

We replace the classifier head to output 102 logits (one per class).

5) Training Procedure

- Loss: CrossEntropyLoss with label smoothing = 0.1
- Optimizer: AdamW (lr=2e-4, weight_decay=1e-4)
- Scheduler: CosineAnnealingLR with T_max=20 epochs
- Batch size: 32 (image size 384)
- Seed: 42

Checkpointing:

- We evaluate each epoch on val set and save the best model weights to model/weights.pt.
- We save model/meta.json containing model_name and img_size for predict.py to reconstruct the architecture.

6) Inference & Submission

predict.py loads model/weights.pt and meta.json, applies Resize+CenterCrop preprocessing, runs batched inference, and outputs predictions.csv with (image_id,label). Output labels are integers in 1–102.

7) Category Compliance

This submission is Category 1. We fine-tune pretrained vision model weights (ConvNeXt-Tiny ImageNet weights). No LLMs are used. No external data is used. All steps are reproducible via `train.py` + `requirements.txt`.