Image Classification using CNN on Imagenette

Course: CSE 6363 — Machine Learning
Name: Elsheikh Mohamed

UTA ID: 1001486517
Date: *7/20/2025*

1. Objective

This assignment explores deep learning for image classification using the **Imagenette** dataset. The project involves:

- Building a basic CNN from scratch.
- Training and validating the model using PyTorch Lightning.
- Implementing early stopping and model checkpointing.
- Evaluating the trained model on a separate test set.

2. Dataset

The **Imagenette** dataset is a subset of ImageNet with 10 easily classifiable classes. The version used contains images resized to **160x160**, and the final input to the CNN is converted to grayscale and resized to **64x64** for computational efficiency.

- **Train/Validation Split**: 90% / 10%
- **Test Split**: Provided separately
- **Input Channels**: 1 (Grayscale)
- **Image Size**: 64x64

3. Model Architecture

We implemented a simple convolutional neural network with the following structure:

```
text
CopyEdit
```

```
Conv2D(1, 32, kernel_size=3, padding=1) → ReLU → MaxPool(2x2)
Conv2D(32, 64, kernel_size=3, padding=1) → ReLU → MaxPool(2x2)
Flatten
Linear(64×16×16, 256) → ReLU
Linear(256, 10)
```

- **Trainable Parameters**: ~4.2 million
- **Loss Function**: Cross-entropy
- **Optimizer**: Adam
- **Metric**: Multiclass Accuracy

4. Training Setup

- **Framework**: PyTorch Lightning
- **Hardware**: CPU-only (No GPU detected)
- **Batch Size**: 128
- **Epochs**: Up to 50 (early stopping applied)
- **Early Stopping**: Monitored `val_loss` with patience = 5
- **Checkpointing**: Saved best model based on `val_acc`

5. Results

Metric	Value
Validation Accuracy	**53.0%**
Validation Loss	**2.060**
Test Accuracy	**53.9%**
Test Loss	**2.213**

- Final model was evaluated on unseen test data.
- Results suggest that while the model is learning, performance can be improved with deeper architecture, better regularization, or data augmentation.

6. Conclusion

We successfully built and trained a basic CNN on the Imagenette dataset using PyTorch Lightning. Although the model achieved ~54% test accuracy, further improvement is possible via:

- Using ResNet-18 or other pretrained models.
- Adding data augmentation techniques.
- Training longer with GPU acceleration.