Digital Empowerment Network

Artificial Intelligence

Week 03

Image Classification using pre-trained CNNs and Deployment with Streamlet

Mentor: Hussain Shoaib

Image Classification using pre-trained CNNs and Deployment with Streamlit

Objective:

To implement an image classification model using pre-trained CNNs (transfer learning) and deploy it using Streamlit as a simple web application that accepts image uploads and shows predictions.

Step 1: Dataset Selection

- Choose one of the following image datasets:
- Cats' vs Dogs
- Flowers (5 Classes)
- Intel Image Classification
- Any dataset of your choice (2–10 classes)

Step 2: Model Implementation (Transfer Learning)

A. Data Preparation

- Organize dataset into folders (train/, val/, test/)
- Resize all images (e.g., 224×224)
- Normalize and apply data augmentation
- Split data appropriately

B. Pre-trained CNN Model

Use at least two of the following: VGG16, MobileNetV2, ResNet50, or EfficientNetB0

Steps:
- Load pre-trained model (without top layers)
- Add your own classification head
- Freeze base layers for initial training
- (Optional) fine-tune later layers

C. Evaluation

- Evaluate with Accuracy, Loss, Confusion Matrix
- Plot training/validation accuracy/loss
- Visualize a few predictions with probabilities

Step 3: Web App Deployment Using Streamlit

A. Streamlit App Features

- Upload image (.jpg/.png)
- Model processes the image and shows:
- Predicted class with confidence
- Optionally: top 3 probabilities
- Show visual feedback (image + prediction)

B. App Structure

- ➢ app.py (Streamlit script)
- ➢ model.h5 or saved .pt model
- ➢ requirements.txt for dependencies

Deliverables:

- Trained model file (e.g., .h5 or .pt)
- app.py Streamlit file
- Jupyter Notebook (for training phase)
- Sample predictions (screenshots or display)

Deadline:

Submit within 7 days of receiving the task.

Tools to Use:

- Python, TensorFlow/Keras or PyTorch
- Streamlit
- Google Colab or Jupyter
- Libraries: opencv-python, PIL, matplotlib, seaborn