HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BỘ MÔN THỰC TẬP CƠ SỞ

THỰC TẬP CƠ SỞ

Đề Tài: Animal Classifier with AI

Giảng viên hướng dẫn : **KIM NGỌC BÁCH**
Họ và tên sinh viên : **NGUYỄN XUÂN HẢI**
Mã sinh viên : **B22DCCN271**
Lớp : **D22CQCN07-B**
Nhóm : **13**

Hà Nội 2025

Lộ trình/KẾ HOẠCH DỰ ÁN: PHÂN LOẠI ĐỘNG VẬT

I. Mục tiêu dự án

- Xây dựng mô hình **phân loại động vật** từ hình ảnh, video.
- Nhận diện **loài động vật** (chó, mèo, gà, bò, bướm, voi, chim, ...).
- Cung cấp giao diện đơn giản để tải ảnh và hiển thị kết quả.
- Hỗ trợ **nhận diện từ webcam**.
- Tạo **API** để tích hợp vào ứng dụng khác.
- **Triển khai trên Git và Docker**: Đảm bảo dự án dễ dàng quản lý và triển khai trên môi trường sản xuất.

II. Công nghệ sử dụng

- **Ngôn ngữ**: Python
- **Thư viện**: PyTorch , OpenCV, Matplotlib, Flask, torchvision, scikit_learn
- **Mô hình**: EfficientNet, RESNET (nhẹ, chạy tốt trên CPU)
- **Dữ liệu**: ImageNet, Kaggle, hoặc dataset tự thu thập
- **Triển khai**: Git, Docker

III. Kế hoạch thực hiện (2 tháng)

Tuần 1: Thu thập dữ liệu và tiền xử lý

- **Thu thập dữ liệu**:
    - Tải các bộ dữ liệu như **Animal-10n** (cho 10 loài động vật) từ Kaggle hoặc tự thu thập hình ảnh động vật từ các nguồn mở.
    - **Tạo Git repository**:
        - Tạo một repository trên GitHub để quản lý mã nguồn.
        - Đẩy dự án lên Git ngay từ đầu để dễ dàng theo dõi tiến độ và phiên bản.

- **Tiền xử lý dữ liệu**:
    - Resize ảnh về kích thước chuẩn .
    - Chuẩn hóa màu sắc và chuyển đổi ảnh sang tensor để huấn luyện.
    - Chia dữ liệu thành tập **Train** và **Test** .

Tuần 2: Xây dựng mô hình phân loại

- **Xây dựng mô hình**:
    - Sử dụng **ResNet** hoặc **EfficientNet** để tạo mô hình phân loại động vật.
- **Huấn luyện mô hình**:
    - Huấn luyện mô hình trên tập **Train**, sử dụng kỹ thuật Data Augmentation.
    - Kiểm thử mô hình trên tập **Test**, đo lường độ chính xác , độ mất mát và tối uu hóa.
- **Triển khai lên Git**:
    - Commit mã nguồn, các file cấu hình và dữ liệu huấn luyện lên GitHub.

Tuần 3: Tối ưu mô hình và chuyển đổi

- **Điều chỉnh tham số**:
    - Tinh chỉnh các tham số như **learning rate**, **batch size**, và **data augmentation** để cải thiện mô hình.
    - Áp dụng các kỹ thuật **Early Stopping** hoặc **Model Checkpoints** để tránh overfitting.
- **Dockerization**:
    - Tạo **Dockerfile** để đóng gói ứng dụng và mô hình vào container.
    - Đảm bảo rằng môi trường phát triển và triển khai giống nhau, giúp dễ dàng chuyển giao mã nguồn.

Tuần 4: Sử dụng `SummaryWriter` và trực quan hóa huấn luyện

- **Sử dụng `SummaryWriter` của TensorBoard**:
    - **Lưu trữ thông số mô hình** như loss và accuracy sau mỗi epoch.
    - **Theo dõi tiến trình huấn luyện** và lưu lại các tham số cần thiết trong suốt quá trình huấn luyện để dễ dàng phân tích.
    - Sử dụng **TensorBoard** để hiển thị đồ thị của độ chính xác và loss qua từng epoch, giúp phân tích và điều chỉnh tham số.

- **Sử dụng `scikit-learn` để trực quan hóa kết quả**:
    - **Matrix Confusion**: Sử dụng thư viện **scikit-learn** để vẽ confusion matrix, giúp trực quan hóa hiệu quả phân loại của mô hình.

Tuần 5: Xây dựng giao diện và triển khai API

- **Nhận diện từ webcam**:
    - Sử dụng **OpenCV** để truy xuất video từ webcam, chạy mô hình phân loại theo thời gian thực.
    - Hiển thị kết quả dự đoán lên webcam: ví dụ, hiển thị tên loài động vật trên ảnh động.
- **Tạo API với Flask**:
    - Xây dựng API RESTful cho phép nhận ảnh từ người dùng và trả về kết quả phân loại.
    - Tạo endpoint `/predict` để người dùng có thể gửi ảnh và nhận kết quả phân loại.
- **Triển khai trên Docker**:
    - Đóng gói ứng dụng Flask và mô hình phân loại vào Docker container.
    - Tạo môi trường Docker có thể chạy trên bất kỳ máy chủ nào với các phụ thuộc đã được cài đặt.

Tuần 6: Kiểm thử và triển khai

- **Kiểm thử mô hình**:
    - Kiểm tra mô hình phân loại với bộ dữ liệu mới để đảm bảo tính chính xác và độ ổn định của hệ thống.
    - Đánh giá hiệu suất khi chạy nhận diện từ webcam và API trên các môi trường khác nhau.
- **Tạo môi trường Docker**:
    - Kiểm tra môi trường Docker trên máy local và trên cloud (ví dụ: AWS, Azure) để đảm bảo tính tương thích.
    - Đảm bảo ứng dụng hoạt động ổn định, dễ dàng scale lên nếu cần.

Tuần 7: Hoàn thiện và viết báo cáo

- **Cải tiến giao diện**:

    - Tinh chỉnh giao diện người dùng và bổ sung tính năng mới nếu cần (ví dụ: nhận diện nhiều loài động vật cùng lúc).
- **Viết tài liệu hướng dẫn**:
    - Cung cấp tài liệu hướng dẫn sử dụng API và cách chạy mô hình từ webcam.
    - Cung cấp hướng dẫn triển khai dự án trên Docker và môi trường đám mây.
- **Hoàn thiện báo cáo**:
    - Tổng kết các kết quả đạt được, bao gồm độ chính xác mô hình, tốc độ dự đoán trên webcam và API, cũng như cách thức triển khai dự án.

IV. Kết luận

- **Mục tiêu đạt được**:
    - Xây dựng thành công một mô hình phân loại động vật với độ chính xác cao.
    - Tạo chức năng nhận diện động vật qua webcam, cho phép phân loại theo thời gian thực.
    - Cung cấp API cho phép tích hợp mô hình phân loại vào các ứng dụng khác.
    - Sử dụng **TensorBoard** để theo dõi quá trình huấn luyện và cải thiện mô hình.
    - Trực quan hóa kết quả với **scikit-learn** và **Matplotlib** để đánh giá và phân tích mô hìn

- **Kết quả mong đợi**:
    - Mô hình phân loại có thể nhận diện đúng động vật trong ảnh và hoạt động hiệu quả trên CPU.
    - API hoạt động ổn định, có thể mở rộng trong tương lai để nhận diện thêm các đối tượng khác ngoài động vật.
    - Các tính năng nhận diện qua webcam và API sẽ tạo cơ hội ứng dụng trong các dự án thực tế, như giám sát động vật hoang dã, nhận diện động vật trong nông nghiệp, hoặc trong các hệ thống bảo mật.