HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA THỰC TẬP CƠ SỞ

THỰC TẬP CƠ SỞ

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

Báo Cáo Hàng Tuần Lần 8 ngày (27/4-4/5/2025)

Hà Nội – 2025

I.NHIỆM VỤ TUẦN NÀY

1. Nhận diện từ webcam

Nhận ảnh từ webcam(base 64)

```python
def inference_from_base64(base64_data, size=224, checkpoint_path="precheckpoint/animals/last.pt"):
    image_bytes = base64.b64decode(base64_data)
    np_arr = np.frombuffer(image_bytes, np.uint8)

    image = cv2.imdecode(np_arr, cv2.IMREAD_COLOR)
    if image is None:
        raise ValueError("Không thể giải mã ảnh từ base64")

    img_tensor = preprocess_image(image, size)
    model = load_model(checkpoint_path)
    softmax = nn.Softmax(dim=1)

    with torch.no_grad():
        predict = model(img_tensor)
        prob = softmax(predict)
        max_value, max_index = torch.max(prob, 1)

    label = categories[max_index.item()]
    score = max_value.item()
    return label, score
```

- Ảnh được mã hóa dưới dạng chuỗi base64 (thường dùng khi ảnh được gửi từ webcam qua HTTP POST)
- Giải mã chuỗi về ảnh gốc, resize ảnh, gọi mô hình và dự đoán

* Ưu điểm:
- Rất đơn giản (API nhận ảnh + trả JSON)
- Nhẹ hơn, chỉ xử lý ảnh mỗi 1–2 giây
* Nhược điểm:
- Không mượt vì phải gửi từng khung ảnh về base64, xử lí và trả lại kết quả dự đoán

2. Tạo Web với Flask

- Tạo backend app.py

B1: Tạo Flask và thiết lập lưu thư mục ảnh

```python
from flask import Flask, render_template, request, url_for, jsonify
from Inference import inference_from_path, inference_from_base64
import os
from werkzeug.utils import secure_filename

#Initialize Flask and saving photo folder
app = Flask(__name__)
app.config["UPLOAD_FOLDER"] = "static/"
```

B2: Giao diện upload ảnh

```python
@app.route( rule: "/", methods=["GET", "POST"])
def index():
    if request.method == "POST":
        if "file" not in request.files:
            return "No file", 400
        file = request.files["file"]
        if file.filename == "":
            return "No selected file", 400
        filename = secure_filename(file.filename)
        filepath = os.path.join(app.config["UPLOAD_FOLDER"], filename)
        file.save(filepath)
        label, score, resized_image_path = inference_from_path(filepath)
        resized_image_url = url_for( endpoint: 'static', filename=os.path.basename(resized_image_path))
        return render_template( template_name_or_list: "index.html", prediction=f"{label} ({score:.2f})", image_path=resized_image_
    return render_template("index.html")
```

• Nếu người dùng gửi form (POST):

• Lấy ảnh từ form (request.files["file"])

• Lưu vào thư mục static/

• Gọi hàm inference_from_path() để dự đoán label

• Hiển thị ảnh và kết quả dự đoán trên index.html

*Lưu ý: Khi hiển thị ảnh thì phải resize lại ảnh để hiển thị đúng khu vực hiển thị ảnh

```python
# Save resized image for UI
resized_image_path = os.path.join("static", "resized_" + os.path.basename(image_path))
cv2.imwrite(resized_image_path, cv2.resize(image, dsize: (224, 224)))
```

- Lưu ảnh đã resize về thư mục static để hiển thị ảnh

B3: Trả về giao diện webcam

```python
@app.route("/webcam")
def webcam():
    return render_template("webcam.html")

@app.route( rule: "/api/predict_webcam", methods=["POST"])
def predict_webcam():
    data = request.get_json()
    if not data or 'image' not in data:
        return jsonify({"error": "No image data"}), 400

    #Loại bỏ prefix ""
    base64_data = data["image"].split(",")[1]
    label, score = inference_from_base64(base64_data)
    return jsonify({"label": label, "score": f"{score:.2f}"})
```

- Dùng JavaScript để bật webcam, chụp ảnh định kỳ, gửi về backend để phân loại
- Nhận ảnh webcam dạng base64 gửi từ frontend (JavaScript).
- Gọi *inference_from_base64()* để phân loại ảnh
- Trả kết quả dạng JSON (label và score)

B4: Tạo 2 file index.html và webcam.html

```html
<!-- Form upload ảnh -->
<div class="image">
    <form method="POST" enctype="multipart/form-data">
        <input type="file" name="file" required>
        <br><br>
        <button type="submit">Phân loại từ ảnh</button>
    </form>


    <div class="result">
        <!-- Kết quả -->
        {% if prediction %}
        <div class="result">
            <p><strong>Kết quả:</strong> {{ prediction }}</p>
            <img src="{{ image_path }}" alt="Uploaded Image">
        </div>


        {% endif %}
    </div>
</div>
```

Hình ảnh file index.html

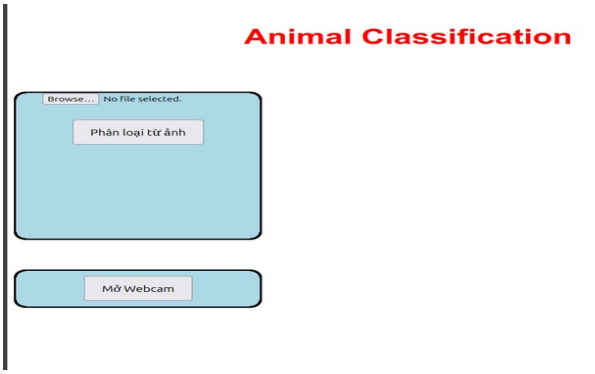
```html
<video id="video" width="900" height="500" autoplay></video>
<canvas id="canvas" width="400" height="300" style="display: none;"></canvas>
<p id="result" style="text-align: center; font-size: 20px;"></p>
<p style="text-align: center;"><a href="/">← Quay lại trang chính</a></p>
<script>
    const video = document.getElementById("video");
    const canvas = document.getElementById("canvas");
    const result = document.getElementById("result");
    const context = canvas.getContext("2d");

    navigator.mediaDevices.getUserMedia({ video: true })
        .then(stream => {
            video.srcObject = stream;
            setInterval(() => {
                context.drawImage(video, 0, 0, canvas.width, canvas.height);
                const dataURL = canvas.toDataURL("image/jpeg");
                fetch("/api/predict_webcam", {
                    method: "POST",
                    headers: { "Content-Type": "application/json" },
                    body: JSON.stringify({ image: dataURL })
                })
                .then(res => res.json())
                .then(data => {
                    if (data.label) {
                        result.textContent = `${data.label} (${data.score})`;
                    }
                });
            }, 1000); // gửi ảnh mỗi 1 giây
        })
        .catch(err => {
            alert("Không thể mở webcam: " + err);
        });
```

Hình ảnh file webcam.html

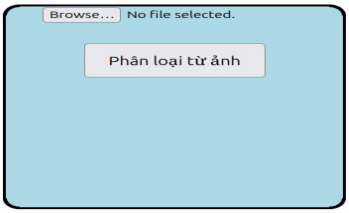
- Sử dụng JavaScript để:

- Mở webcam qua getUserMedia.
- Vẽ ảnh lên canvas.
- Mã hóa ảnh thành base64.
- Gửi ảnh lên Flask backend (/api/predict_webcam) bằng fetch.
- Hiển thị kết quả dự đoán lên trang.

B5: Chạy web bằng cổng post: http://127.0.0.1:5000/



- Test phân loại ảnh:



- Test webcam:

3. Tạo môi trường trên Docker

- Đóng gói phần triển khai web

B1: Tạo Requirements.txt

```
# Core deep learning
torch>=2.0.0
torchvision>=0.15.0
torchsummary
opencv-python-headless
# Data handling
numpy
scikit-learn
opencv-python
flask
# Progress bar
tqdm
# TensorBoard
tensorboard
# Others
matplotlib
numpy
```

B2: Tạo Dockerfile

```
# Base image với PyTorch + CUDA (nếu dùng GPU)
FROM pytorch/pytorch:2.1.0-cuda11.8-cudnn8-runtime

# Set môi trường để không hỏi múi giờ
ENV DEBIAN_FRONTEND=noninteractive
ENV TZ=Asia/Ho_Chi_Minh

# Cài đặt thư viện hệ thống
RUN apt-get update && apt-get install -y \
    tzdata \
    gcc \
   apt-utils \
    git \
    wget \
    curl \
    nano \
    ffmpeg \
    libsm6 \


   libxext6 \
   libgl1-mesa-glx \
   python3-opencv \
    && rm -rf /var/lib/apt/lists/*
# Tạo thư mục app
WORKDIR /app
# Copy toàn bộ project vào container
RUN mkdir -p /app/checkpoints
COPY efficientnet_b0_rwightman-3dd342df.pth /app/checkpoints/
COPY . .
# Cài đặt thư viện Python
RUN pip install --upgrade pip
```

```
RUN pip install -r Requirements.txt
# Mở cổng Flask
EXPOSE 5050
# Lệnh mặc định
CMD ["python", "app.py"]
```

B3: Tạo .dockerignore

```
__pycache__/
*.pyc
*.pyo
*.pyd
*.swp
.env
.vscode/
.idea/
```

- dockerignore giúp container **nhẹ hơn, sạch hơn, an toàn hơn.**
- Khi bạn build Docker image, Docker sẽ tự động đọc tệp .dockerignore và bỏ qua những thư mục hoặc file được chỉ định trong đó

B4: Chạy và kiểm thử container

 # Build image
docker build -t animal-webcam-app .

 # Chạy thử trên local
docker run -p 5000:5000 animal-webcam-app

Sau khi chạy post: http://127.0.0.1:5000/ sẽ ra kết quả như sau: