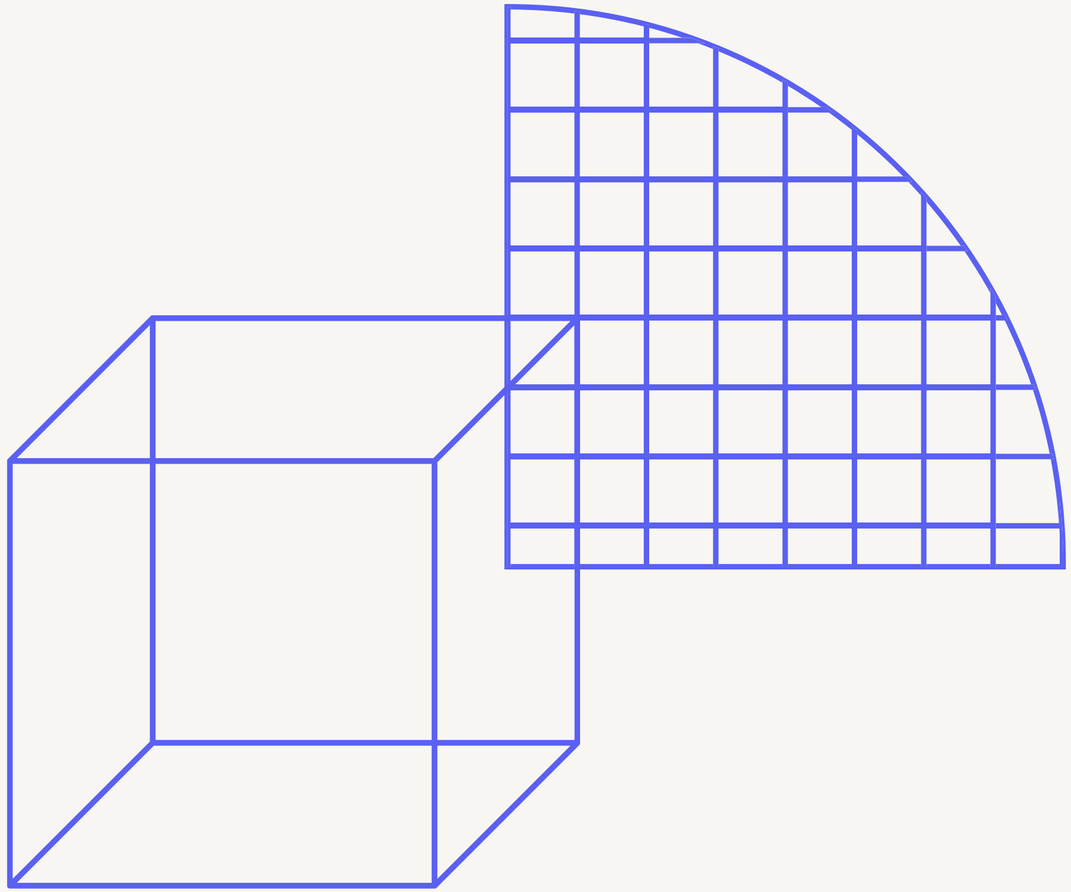
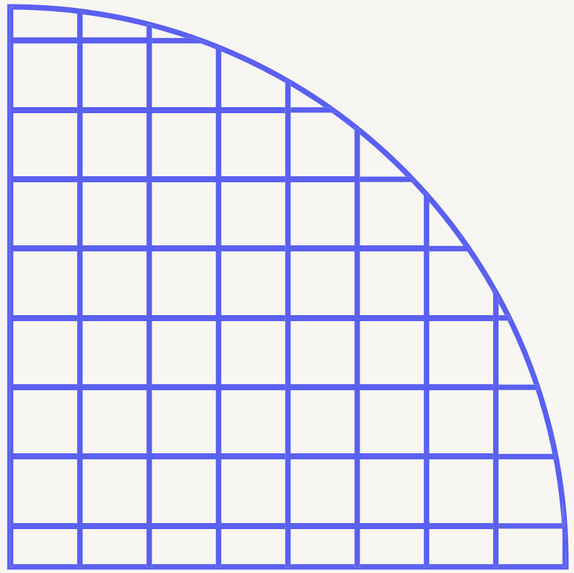
XỬ LÍ ẢNH

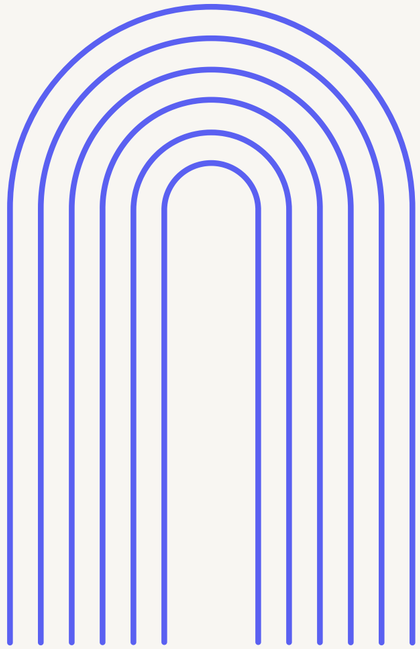
Đề tài số 6 : Nhận dạng hoa quả từ ảnh xám bằng CNN



I. Giới thiệu



Thành viên



- Vũ Ngọc Hùng - B22DCCN373
- Kim Duy Hưng - B22DCCN409
- Phùng Trung Kiên - B22DCCN433

1. GIỚI THIỆU ĐỀ TÀI

Nhận dạng ảnh là một nhánh quan trọng của Computer Vision.

- CNN là mô hình mạnh mẽ trong nhận dạng và phân loại ảnh.
- Ứng dụng của đề tài: phân loại hoa quả dựa trên hình dạng và kết cấu từ ảnh xám.

Mục tiêu :

- Xây dựng mô hình CNN để nhận dạng và phân loại hoa quả từ ảnh xám.
- Phân loại các loại quả như: táo, chuối, cam, xoài…

Định hướng ứng dụng:

- Nhận diện vị trí quả để robot có thể hái
- Phân loại hoa quả: nhận diện loại quả đang chạy trên băng chuyền để điều hướng tay gạt cơ khí đưa quả vào đúng thùng chứa

2. TẠI SAO DÙNG CNN ?

CNN là mô hình mạnh mẽ trong nhận dạng và phân loại ảnh , nó sở hữu các ưu điểm như :

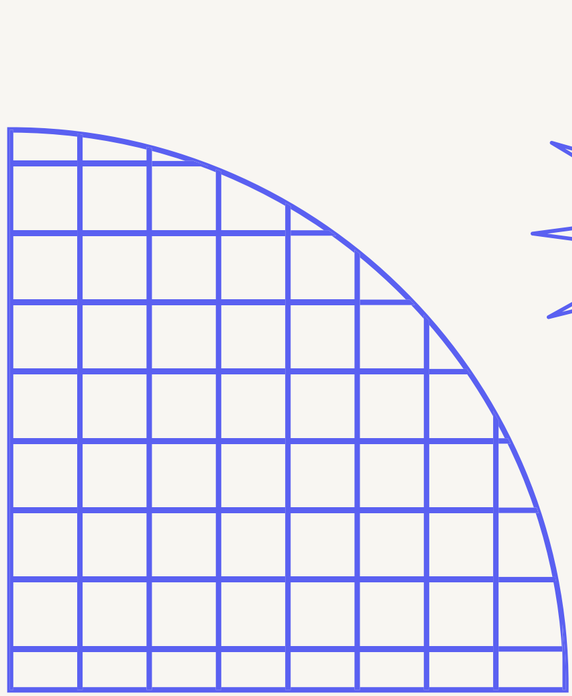
- CNN tự động học đặc trưng từ ảnh mà không cần trích xuất thủ công.
- Hoạt động tốt với ảnh 2D, có thể là ảnh xám hoặc màu.
- CNN đã được chứng minh hiệu quả trong các bài toán như :
  - Nhận dạng vật thể
  - Nhận diện chữ viết tay
  - Phân loại y tế, nông nghiệp...

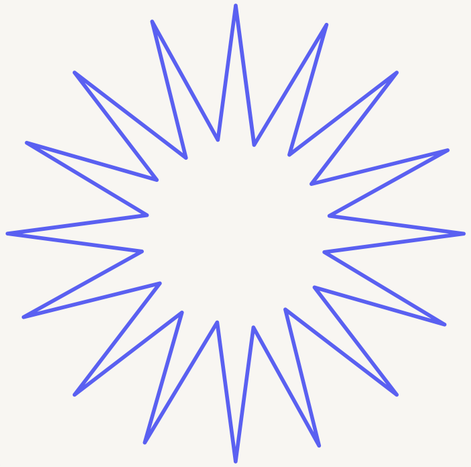
3. ĐẶC ĐIỂM CỦA ẢNH XÁM

Ảnh xám (gray image) hay còn gọi là **ảnh đơn sắc** (monochromatic), là loại ảnh mà mỗi giá trị điểm ảnh (pixel) trong ma trận điểm ảnh mang giá trị từ 0 đến 255.

Ưu điểm: giảm dữ liệu, dễ train, phù hợp camera đơn sắc.
Hạn chế: mất thông tin màu sắc → khó phân biệt quả cùng hình dạng khác màu.

2. Xây dựng





1. TẬP DỮ LIỆU

Tập dữ liệu Fruit Classification
từ kaggle

- Tổng số hình ảnh: 22495.
- Kích thước tập train : 16854 hình ảnh
- Kích thước tập test: 5641 hình ảnh
- Kích thước hình ảnh: 100x100 pixel.
- Chứa hình ảnh chụp các loại hoa quả từ các góc độ khác nhau
- Đa dạng nhiều loại quả và đều được lược bỏ background từ trước

1. TIỀN XỬ LÍ ẢNH

Tiền xử lý ảnh (Image Preprocessing) là một phần quan trọng nằm giữa
Thu nhận ảnh (Image Acquisition) và Xử lý chính (Computer/CNN)

Bước 1 : Làm sạch ảnh và Tăng cường chất lượng

Mục tiêu là giảm nhiễu mà vẫn giữ được biên (đường nét) của hình khối.

Quy trình :

- Chuyển ảnh xám: Nếu ảnh đầu vào là ảnh màu → chuyển sang ảnh xám để giảm chiều dữ liệu, vì hình dạng thường không phụ thuộc vào màu sắc
- Loại bỏ nhiễu:
  - Kỹ thuật: Sử dụng Bộ lọc trung vị (Median Filter).
  - Lý do: Bộ lọc trung vị loại bỏ nhiễu hiệu quả (đặc biệt là nhiễu muối tiêu - Impulse noise/Salt and pepper noise) và quan trọng hơn là bảo toàn đường biên (edge) tốt hơn so với Bộ lọc trung bình (Mean Filter)
- Cân bằng histogram

1. TIỀN XỬ LÍ ẢNH

1. TIỀN XỬ LÍ ẢNH

Bước 2 : Tách đối tượng (Segmentation)

Mục tiêu là tách hình khối ra khỏi nền (background) hoặc xác định chính xác đường biên của chúng

Phương pháp sử dụng :

- Phương pháp Canny. Canny được coi là tối ưu vì nó có tỉ lệ lỗi thấp và xác định tốt vị trí biên.
- Phân ngưỡng (Thresholding): Nếu ảnh là hình vẽ đơn giản (hai màu sáng/tối rõ rệt), phân ngưỡng là một lựa chọn nhanh chóng ví dụ : Otsu , ..

1. TIỀN XỬ LÍ ẢNH

Bước 2 : Tách đối tượng (Segmentation)

Phương pháp Canny được coi là phương pháp phát hiện biên tối ưu vì nó giải quyết được các vấn đề mà phân ngưỡng toàn cục như Otsu không giải quyết được:

- Độ chính xác và Vị trí biên: Canny được thiết kế để có Tỉ lệ lỗi thấp và Xác định tốt vị trí biên
    - Phân ngưỡng Otsu dựa vào lược đồ xám để tìm ngưỡng tối ưu. Nếu ảnh có sự chiếu sáng không đều hoặc nhiều nhiễu, việc chọn một ngưỡng toàn cụcsẽ trở nên khó khăn, dẫn đến phân loại sai nhiều pixel.

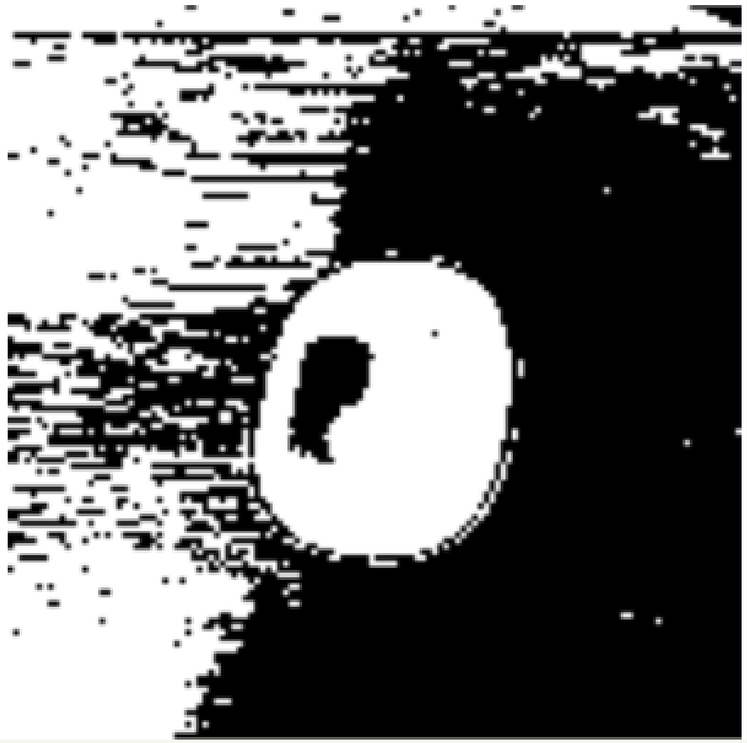
- Chống nhiễu: Canny tích hợp bước Làm mịn (Smoothing) bằng bộ lọc Gaussian ở đầu tiên. Điều này giúp giảm nhiễu trước khi tính toán gradient, khiến thuật toán ít nhạy cảm với nhiễu hơn.

1. TIỀN XỬ LÍ ẢNH

2. Ảnh xám

5. Phát hiện biên (Canny)

6. Phương pháp otsu



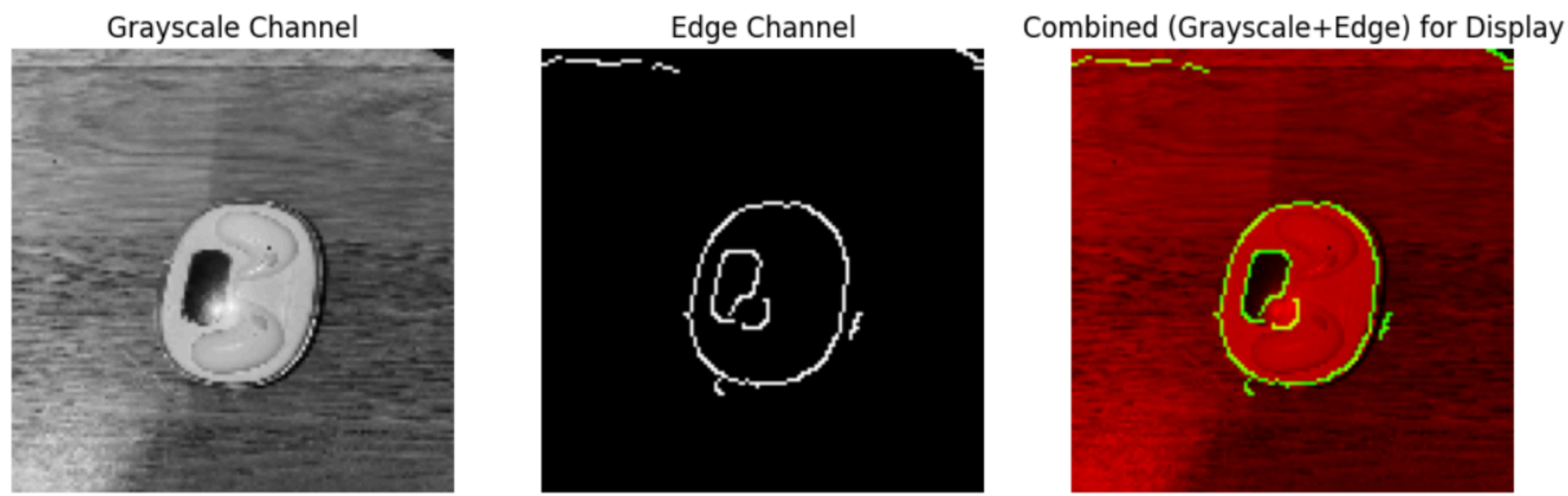
1. TIỀN XỬ LÍ ẢNH

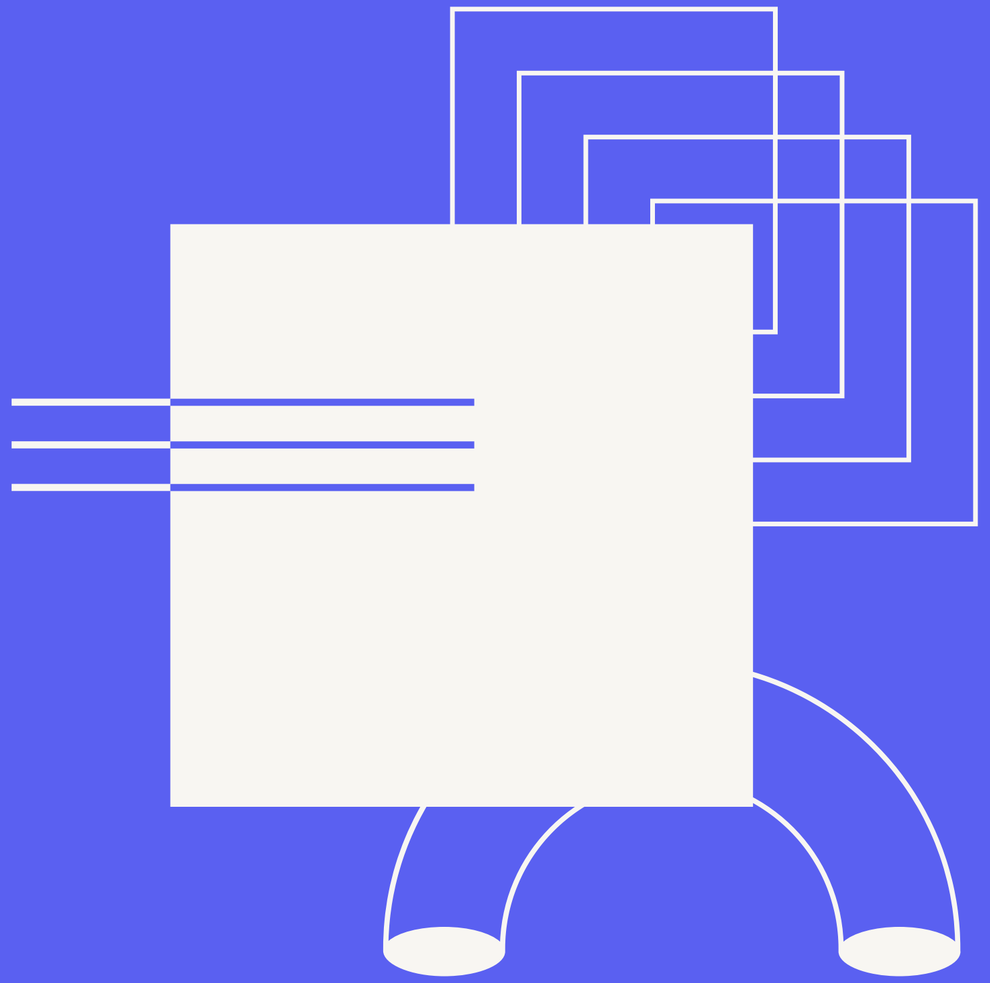
Ảnh kết quả sau bước tiền xử lí :

- Ảnh có dạng (H, W, C) = (100 , 100 , 2) với :
  - H : chiều cao của ảnh
  - W : chiều rộng của ảnh
  - C : số channel của ảnh
- Ghép 2 kênh :
  - Kênh 1 : Ảnh xám chứa thông tin vùng (độ sáng, nội dung bên trong).
    - biết “bên trong hình khối có gì” ?
  - Kênh 2 : Ảnh Canny chứa biên cạnh (shape rõ ràng hơn).
    - biết “đường biên của hình như thế nào”

=> CNN học hình dạng mạnh hơn và nhanh hơn.

1. TIỀN XỬ LÍ ẢNH





2. HUẤN LUYỆN MÔ HÌNH

2.1 TẢI DỮ LIỆU HUẤN LUYỆN

processed_data_for_cnn: List lưu lại cặp giá trị (đường dẫn ảnh gốc, đường dẫn file .npy tương ứng của ảnh)

Nạp ma trận ảnh (100, 100, 2) vào danh sách dữ liệu đầu vào X.

Trích xuất tên quả từ đường dẫn thư mục làm nhãn (Label) Y.

Mạng nơ-ron hoạt động tốt nhất với các số nhỏ trong khoảng [0, 1].

```
X = []
y = []

print("Đang tải dữ liệu từ các file .npy...")
for original_path, npy_path in processed_data_for_cnn:
    try:
        # Load ảnh 2 kênh
        data = np.load(npy_path)
        X.append(data)

        # Lấy nhãn
        label = os.path.basename(os.path.dirname(original_path))
        y.append(label)
    except Exception as e:
        print(f"Lỗi khi đọc file {npy_path}: {e}")

X = np.array(X)
y = np.array(y)

X = X.astype('float32') / 255.0
```

Chuẩn hóa dữ liệu về khoảng [0, 1] giúp thuật toán tối ưu hóa hội tụ nhanh hơn.

2.1 TẢI DỮ LIỆU HUẤN LUYỆN

Mã hóa nhãn

- Mô hình học máy không hiểu văn bản (ví dụ: "Apple", "Banana"). Cần
- Mã hóa thành số nguyên(Apple -> 0, Banana -> 1).
- Mã hóa One-hot: Chuyển số thành vector xác suất.
- Từ đó “Apple” => [1, 0, ...]. Phù hợp với hàm mất mát categorical_crossentropy.

```
# 2. Mã hóa nhãn (Label Encoding)
le = LabelEncoder()
y_encoded = le.fit_transform(y)
y_categorical = to_categorical(y_encoded)
num_classes = len(le.classes_)
```

Chia tập dữ liệu

```
X_train, X_test, y_train, y_test =
train_test_split(X, y_categorical, test_size=0.2, random_state=42)
```

2.2 KIẾN TRÚC MẠNG CNN

Mô hình được xây dựng theo dạng Sequential (Tuần tự)
Lớp Input khai báo kích thước đầu vào: (100, 100, 2)
Khối trích xuất đặc trưng (Feature Extraction): Khối 1 trích xuất đặc trưng cấp thấp. Khối 2 trích xuất đặc trưng cấp cao
Khối phân loại (Classification): đưa ra kết quả đầu ra của mô hình

```python
def build_cnn_model(input_shape, num_classes):
    model = Sequential()

    # Input layer
    model.add(Input(shape=input_shape))

    model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D((2, 2)))

    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D((2, 2)))
    model.add(Dropout(0.25)) # Tránh overfitting

    # Flatten: Duỗi thẳng để đưa vào mạng nơ-ron đầy đủ (Dense)
    model.add(Flatten())

    # Phân lớp
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation='softmax')) # Output layer

    return model

# Khởi tạo model
model = build_cnn_model((100, 100, 2), num_classes)
model.summary()
```

2.2 KIẾN TRÚC MẠNG CNN

Khối trích xuất đặc trưng (Feature Extraction): học các đặc điểm hình ảnh từ thấp đến cao.

- Lớp Conv2D (Tích chập)
  - Block 1: Conv2D(32, (3, 3), activation='relu', padding='same')
  - Sử dụng 32 bộ lọc (filters) kích thước 3x3 để quét qua ảnh.
  - Mục đích: Phát hiện các đặc trưng cơ bản như cạnh ngang, dọc, đường chéo từ ảnh biên Canny và ảnh xám.

  - Block 2: Conv2D(64, (3, 3), activation='relu', padding='same')
  - Tăng số lượng bộ lọc lên 64.
  - Mục đích: Kết hợp các đặc trưng cơ bản để nhận diện các hình dạng phức tạp hơn (ví dụ: độ cong của quả táo, vỏ sần sùi của quả cam).

2.2 KIẾN TRÚC MẠNG CNN

- Lớp Gộp (MaxPooling2D): MaxPooling2D((2, 2))
  - Cơ chế: Chọn giá trị lớn nhất trong vùng 2x2 pixel.
  - Mục đích:
  a. Giảm kích thước dữ liệu đi một nửa => Giảm khối lượng tính toán.
  b. Giữ lại các đặc trưng nổi bật nhất, giảm nhiễu.
- BatchNormalization:
  - Mục đích: Giúp chuẩn hóa dữ liệu ở giữa các lớp, làm cho mạng học ổn định hơn và nhanh hơn.
- Drop out:
  - Kỹ thuật chống học vẹt (Overfitting). Nó ngẫu nhiên "tắt" một số nơ-ron trong quá trình học, buộc mạng phải học các đặc trưng mạnh mẽ hơn thay vì phụ thuộc vào một vài nơ-ron cụ thể.

2.2 KIẾN TRÚC MẠNG CNN

Khối phân loại:

- Flatten: Duỗi mảng 3 chiều (100x100x2) từ các lớp Conv thành một vector dài 1 chiều để đưa vào mạng Dense.
- Lớp ẩn (Dense Layer):
  - Dense(128, activation='relu')
  - Mục đích: Mạng nơ-ron đầy đủ kết nối với 128 nút ẩn để học mối quan hệ phi tuyến tính giữa các đặc trưng.

- Dense(num_classes, activation='softmax'): Lớp cuối cùng có số nơ-ron bằng số loại quả. Hàm softmax sẽ tính phần trăm xác suất của nhãn (ví dụ: 90% là Táo, 10% là Lê) rồi đưa ra kết quả đầu ra

2.3 CẤU HÌNH BIÊN DỊCH MÔ HÌNH

Để mô hình có thể học được, chúng ta cần xác định cách nó sửa lỗi và tối ưu hóa.

- Thuật toán tối ưu (Optimizer): Adam. Tự động điều chỉnh learning rate cho từng tham số, giúp mô hình hội tụ nhanh hơn và tránh bị kẹt ở các điểm cực tiểu cục bộ
- Hàm mất mát (Loss Function): Categorical Crossentropy. Đây là hàm mất mát tiêu chuẩn cho bài toán phân loại đa lớp. Hàm này đo lường sự khác biệt giữa phân phối xác suất dự đoán (Output Softmax) và nhãn thực tế.
- Độ đo đánh giá (Metrics): Accuracy. Sử dụng độ chính xác để theo dõi hiệu năng trong quá trình huấn luyện

```
# Compile model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

2.4 CẤU HÌNH CÁC SIÊU THAM SỐ HUẤN LUYỆN

- Số vòng lặp (Epochs): 20
  - Toàn bộ tập dữ liệu sẽ được đưa qua mạng 20 lần
- Kích thước lô (Batch Size): 32
  - Thay vì cập nhật trọng số sau mỗi ảnh (quá chậm) hoặc sau toàn bộ dữ liệu (quá tốn RAM), mô hình gom 32 ảnh thành 1 nhóm để xử lý cùng lúc
- Chiến lược kiểm thử : Sử dụng tập kiểm thử để đánh giá chéo ngay sau mỗi epoch, giúp theo dõi xem mô hình có đang học tốt trên dữ liệu lạ hay không

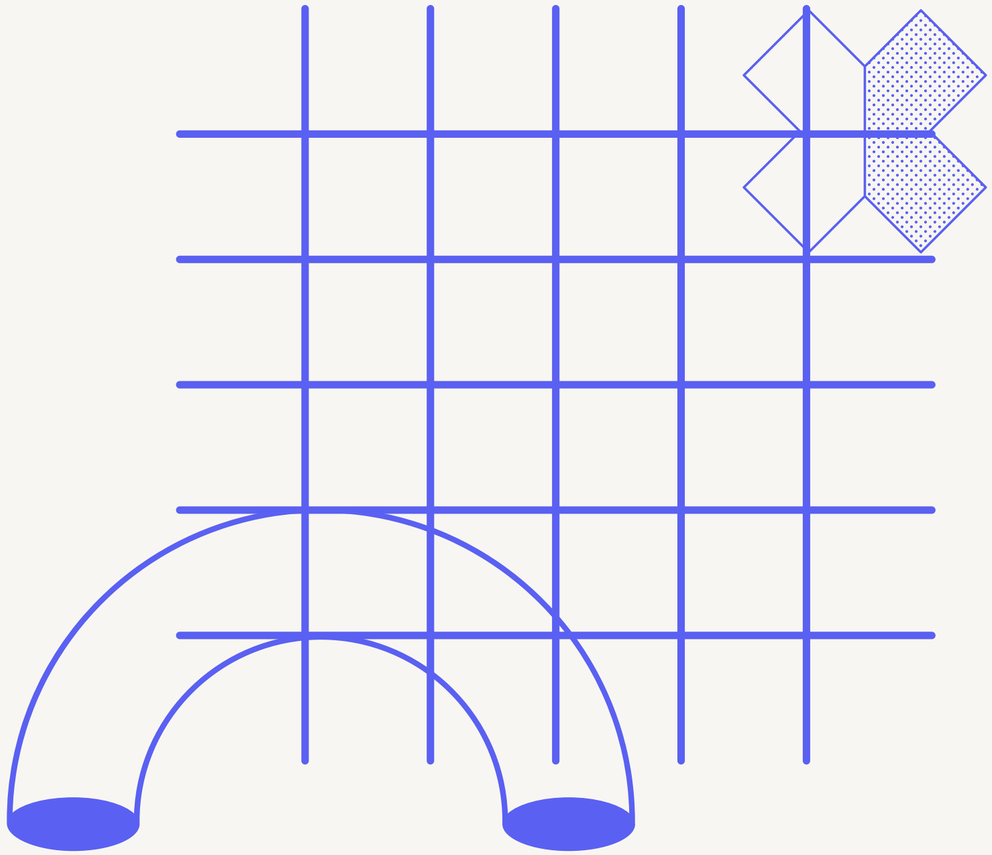
```
# Huấn luyện
EPOCHS = 20
BATCH_SIZE = 32

history = model.fit(
    X_train, y_train,
    epochs=EPOCHS,
    batch_size=BATCH_SIZE,
    validation_data=(X_test, y_test),
    verbose=1
)
```

2.5 KẾT QUẢ HUẤN LUYỆN

Độ chính xác tăng dần theo số vòng lặp (epochs). Điều này chứng tỏ mô hình đang "học" hiệu quả các đặc trưng từ ảnh 2 kênh (xám + biên). Sau 20 epochs, độ chính xác trên tập kiểm thử đạt mức rất cao (xấp xỉ 98-99%)

DEMO



Cảm ơn thầy và các bạn đã lắng nghe!