**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**
**Đại Học Quốc Gia Hà Nội**


# Phân loại rác thải trong gia đình

Vũ Nguyễn Quỳnh Anh
Nguyễn Văn Duy
Đoàn Thị Minh Khuê

Mã học phần: MAT3508
Học kỳ 1, Năm học 2025-2026

# Thông tin Dự án

*[Thông tin này cũng cần được ghi trong README.md của kho GitHub.]*

**Học phần:** MAT3508 – Nhập môn Trí tuệ Nhân tạo
**Học kỳ:** Học kỳ 1, Năm học 2025-2026
**Trường:** VNU-HUS (Đại học Quốc gia Hà Nội – Trường Đại học Khoa học Tự nhiên)
**Tên dự án:** Phân loại rác thải trong gia đình
**Ngày nộp:** 30/11/2025
**Báo cáo PDF:** https://github.com/doanminhkhue/IntroAI-GC/tree/main/report
**Slide thuyết trình:** https://github.com/doanminhkhue/IntroAI-GC/tree/main/slide
**Kho GitHub:** `https://github.com/doanminhkhue/IntroAI_garbage_classification`

## Thành viên nhóm

| **Họ tên** | **Mã sinh viên** | **Tên GitHub** |
|---|---|---|
| Vũ Nguyễn Quỳnh Anh | 23001830 | quynhanh_2610 |
| Nguyễn Văn Duy | 23001854 | nvndy27 |
| Đoàn Thị Minh Khuê | 23001894 | doanminhkhue |

## Phân công công việc

| **Họ tên** | **Nhiệm vụ** |
|---|---|
| Vũ Nguyễn Quỳnh Anh | Giới thiệu, cơ sở lý thuyết, tiền xử lý |
| Nguyễn Văn Duy | Huấn luyện, triển khai mô hình, kết luận |
| Đoàn Thị Minh Khuê | Chia dữ liệu, thực nghiệm và đánh giá |

# Danh sách hình vẽ

# Danh sách bảng

# Mục lục

# Lời mở đầu

Trong thời đại công nghệ số phát triển mạnh mẽ, Trí tuệ Nhân tạo (Artificial Intelligence – AI) không còn là khái niệm xa vời trong sách vở hay phim ảnh, mà đã trở thành nền tảng quan trọng trong nhiều ứng dụng thực tiễn của đời sống hiện đại. Những tiến bộ vượt bậc trong lĩnh vực Học sâu (Deep Learning) và xử lý ảnh đã mở ra nhiều hướng tiếp cận mới, đặc biệt là trong việc giải quyết các vấn đề môi trường – một trong những thách thức cấp thiết của thế kỷ XXI.

Trong quá trình học môn Nhập môn Trí tuệ nhân tạo (Introduction to Artificial Intelligence), chúng tôi nhận thấy rằng việc ứng dụng AI vào những bài toán gần gũi đời sống không chỉ giúp củng cố kiến thức lý thuyết, mà còn đem lại cơ hội tiếp cận quy trình xây dựng một hệ thống AI hoàn chỉnh. Trong bối cảnh lượng rác thải sinh hoạt ngày càng gia tăng, nhu cầu phân loại rác tại hộ gia đình trở nên cấp bách, nhưng vẫn còn gặp nhiều hạn chế do thói quen và khả năng nhận diện của người dùng. Xuất phát từ thực tế đó, nhóm chúng tôi lựa chọn đề tài **"Phân loại rác thải sinh hoạt trong gia đình"** với mong muốn áp dụng các kỹ thuật xử lý ảnh và mô hình học sâu để hỗ trợ quá trình phân loại rác tự động.

Báo cáo này trình bày quá trình tìm hiểu lý thuyết về xử lý ảnh, mạng nơ-ron tích chập (Convolutional Neural Networks – CNN), quy trình huấn luyện mô hình, cách xây dựng ứng dụng demo và đánh giá hiệu quả phân loại. Mục tiêu chính của nhóm không phải tạo ra một hệ thống hoàn hảo, mà là nắm vững tư duy, quy trình và kỹ năng triển khai mô hình AI vào thực tế.

Chúng tôi xin gửi lời cảm ơn chân thành đến Giảng viên hướng dẫn – Thầy Nguyễn Hải Vinh, người đã đồng hành và cung cấp nền tảng kiến thức vững chắc giúp chúng tôi hoàn thành đề tài này.

Dù đã rất nỗ lực, nhưng do hạn chế về thời gian và kinh nghiệm, bài báo cáo khó tránh khỏi thiếu sót. Chúng tôi mong nhận được những ý kiến đóng góp của Thầy và các bạn để hoàn thiện hơn trong tương lai.

*Xin chân thành cảm ơn!*

# Chương 1

# Giới thiệu Đề tài

## 1.1 Đặt vấn đề

Trong những năm gần đây, vấn đề ô nhiễm môi trường, đặc biệt là ô nhiễm rác thải sinh hoạt, đã trở thành một thách thức nghiêm trọng đối với các đô thị lớn và khu dân cư. Việc phân loại rác tại nguồn được xem là giải pháp quan trọng giúp giảm tải khối lượng rác cần xử lý, tăng khả năng tái chế và nâng cao ý thức bảo vệ môi trường. Tuy nhiên, thực tế cho thấy phần lớn hộ gia đình vẫn gặp khó khăn trong việc nhận diện và phân loại đúng các nhóm rác do thiếu kiến thức, thói quen hoặc sự phức tạp trong phân loại.


Hình 1.1: Minh họa cách phân loại rác thải

Trong khuôn khổ môn học Nhập môn Trí tuệ Nhân tạo, việc ứng dụng kỹ thuật AI vào các bài toán thực tiễn là một bước quan trọng giúp sinh viên kết nối giữa lý thuyết và triển khai hệ thống thực tế. Tuy nhiên, tự xây dựng một mô hình xử lý ảnh phức tạp từ đầu là điều không khả thi đối với sinh viên năm 3 do hạn chế về dữ liệu, thời gian và tài nguyên phần cứng cần thiết để huấn luyện các mô hình Deep Learning lớn. Thách thức đặt ra cho nhóm là: Làm thế nào để ứng dụng các phương pháp học sâu vào một hệ thống phân loại rác hoàn chỉnh mà không cần xây dựng mọi thứ từ con số 0?

Thay vì phát triển một mô hình mới, nhóm lựa chọn hướng tiếp cận thực tiễn hơn: Tìm hiểu lý thuyết nền tảng, tận dụng các mô hình học sâu đã được huấn luyện trước (Pre-trained Models) như MobileNetV2 hoặc EfficientNet, và tiến hành tinh chỉnh (Fine-tuning) để mô hình phù hợp với bài toán phân loại rác sinh hoạt trong gia đình. Từ đó, nhóm xây dựng một hệ

thống phân loại dựa trên xử lý ảnh, có khả năng nhận diện các loại rác phổ biến như *plastic*, *paper*, *glass*, *metal*, *organic* và *trash*.

## 1.2 Phương pháp tiếp cận

Để giải quyết bài toán đặt ra, đồ án triển khai theo mô hình tiếp cận **"Nghiên cứu – Xây dựng – Tích hợp"**:

### Nghiên cứu (Research)

Tìm hiểu các kiến thức nền tảng liên quan đến xử lý ảnh, mạng nơ-ron tích chập (CNN), cách hoạt động của mô hình phân loại hình ảnh và quy trình huấn luyện dữ liệu. Bên cạnh đó, nghiên cứu cấu trúc bộ dữ liệu rác thải, các kỹ thuật tiền xử lý (preprocessing) và tăng cường dữ liệu (data augmentation).

### Xây dựng và Huấn luyện (Model Training)

Sử dụng các mô hình CNN phổ biến đã được huấn luyện trước làm nền tảng (MobileNetV2, EfficientNetB0), sau đó tiến hành tinh chỉnh để mô hình đạt độ chính xác cao hơn trên bộ dữ liệu rác thải. Quá trình huấn luyện bao gồm chia tập dữ liệu, thiết lập tham số, tối ưu hóa và theo dõi hiệu suất.

### Tích hợp và Triển khai (Integration & Deployment)

Sau khi có mô hình tốt, nhóm tiến hành xây dựng ứng dụng demo như Camera App bằng OpenCV hoặc ứng dụng Web (Streamlit), giúp người dùng chụp ảnh và phân loại rác ngay lập tức. Các chức năng được tập trung vào độ ổn định, độ chính xác và tính dễ sử dụng.

## 1.3 Mục tiêu của báo cáo

Mục tiêu chính của đồ án không nhằm tạo ra một mô hình AI hoàn toàn mới, mà là làm chủ quy trình thiết kế và triển khai một ứng dụng AI thực tế bằng các thư viện sẵn có. Cụ thể:

1. Tìm hiểu về quy trình xử lý ảnh và kiến trúc CNN.
2. Biết cách sử dụng các mô hình pre-trained, tinh chỉnh và đánh giá mô hình phân loại rác.
3. Xây dựng hệ thống phân loại rác hoàn chỉnh, có thể dự đoán gần chính xác các loại rác và hỗ trợ người dùng trong quá trình phân loại.
4. Củng cố kiến thức môn học thông qua việc phát triển một sản phẩm AI có thể chạy được.

## 1.4 Bố cục báo cáo

Báo cáo gồm 5 chương chính:

- **Chương 1 - Giới thiệu Đề tài**: Trình bày bối cảnh, lý do chọn đề tài, phương pháp tiếp cận và mục tiêu.
- **Chương 2 - Cơ sở lý thuyết**: Tổng quan về xử lý ảnh, mạng nơ-ron tích chập và các mô hình pre-trained được sử dụng.

- **Chương 3 - Phân tích và Thiết kế Mô hình**: Mô tả bộ dữ liệu, quy trình tiền xử lý, kiến trúc mô hình và các tham số huấn luyện.
- **Chương 4 - Thực nghiệm và Đánh giá**: Trình bày kết quả huấn luyện, biểu đồ độ chính xác, ma trận nhầm lẫn và demo phân loại.
- **Chương 5 - Kết luận và Hướng phát triển**: Tổng kết quá trình thực hiện và đề xuất cải tiến mô hình trong tương lai.

# Chương 2

# Cơ sở lý thuyết

## 2.1 Tổng quan về xử lý ảnh

Xử lý ảnh (Image Processing) là bước nền tảng trước khi đưa dữ liệu ảnh vào mô hình học máy hoặc học sâu. Mục tiêu chính của tiền xử lý ảnh là chuẩn hóa dữ liệu, giảm nhiễu và tăng tính đại diện của ảnh để mô hình có thể học hiệu quả hơn.

### 2.1.1 Các bước tiền xử lý thường gặp

- **Đổi kích thước (Resize)**: Chuẩn hóa kích thước ảnh về một kích thước cố định (ví dụ $224 \times 224$) để phù hợp với kiến trúc mạng và batch processing.
- **Chuẩn hóa (Normalization)**: Thường đưa giá trị pixel về phạm vi $[0, 1]$ (chia cho 255) hoặc chuẩn hóa theo mean/std của tập ImageNet (nếu dùng mô hình pre-trained).
- **Cắt ảnh ngẫu nhiên (Random Crop) & Canh lề (Center Crop)**: Giúp mô hình học được các vị trí khác nhau của đối tượng.
- **Xoay, lật, thay đổi độ sáng/contrast (Data Augmentation)**: Tăng kích thước hiệu quả của bộ dữ liệu và giúp mô hình tổng quát hóa tốt hơn.
- **Loại nhiễu (Denoising) & Làm sắc nét (Sharpening)**: Khi dữ liệu có nhiều nhiễu, có thể áp dụng bộ lọc phù hợp.

### 2.1.2 Kỹ thuật tăng cường dữ liệu (Data Augmentation)

Một số thao tác phổ biến:

- Horizontal/Vertical flip, random rotation, random zoom.
- Color jitter: thay đổi brightness, contrast, saturation.
- Random erasing / Cutout: loại bỏ một vùng ảnh để mô hình học các đặc trưng còn lại.
- MixUp / CutMix: kết hợp hai ảnh/nhãn để tăng tính đa dạng.

Những kỹ thuật này đặc biệt hữu ích khi bộ dữ liệu nhỏ, trường hợp phổ biến trong bài toán phân loại rác thải gia đình.

## 2.2 Mạng nơ-ron tích chập (Convolutional Neural Networks - CNN)

Mạng nơ-ron tích chập là kiến trúc tiêu chuẩn cho các bài toán xử lý ảnh. Dưới đây là phần mô tả chi tiết các thành phần và nguyên lý hoạt động. Tuy nhiên, nhóm sẽ chỉ tìm hiểu sơ qua, và chủ yếu sử dụng thư viện trong báo cáo này.

### 2.2.1 Phép tích chập (Convolution)

Phép tích chập giữa ảnh đầu vào $I$ và kernel (filter) $K$ được định nghĩa (cho trường hợp liên tục rời rạc và đơn giản) bởi:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)\, K(m,n)$$

Trong thực tế, mỗi lớp tích chập học được nhiều kernel ($C_{out}$) để tạo ra nhiều bản đồ đặc trưng (feature maps).

**Các tham số chính**

- **Kernel size** (ví dụ $3 \times 3$, $5 \times 5$).
- **Stride**: bước dịch chuyển kernel trên ảnh. Stride lớn làm giảm kích thước đầu ra.
- **Padding**: zero-padding để điều chỉnh kích thước đầu ra (same vs valid).
- **Number of filters**: số lượng feature maps đầu ra (tăng số filter tăng khả năng mô hình hoá nhưng cũng tăng tham số).

### 2.2.2 Lớp Pooling

Pooling (MaxPooling, AveragePooling) giảm kích thước không gian của feature maps, giúp giảm tính toán và khuyếch đại tính bất biến vị trí:

$$\text{MaxPool}_{2\times2}(x) = \max\{x_{2i,2j}, x_{2i+1,2j}, x_{2i,2j+1}, x_{2i+1,2j+1}\}$$

### 2.2.3 Hàm kích hoạt (Activation functions)

- **ReLU** ($f(x) = \max(0, x)$): phổ biến vì gradient không bão hòa cho các giá trị dương.
- **Leaky ReLU**, **ELU**: biến thể để tránh dead ReLU.
- **Softmax**: được dùng ở lớp cuối cho bài toán phân loại đa lớp để chuyển logits thành xác suất:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

### 2.2.4 Batch Normalization & Dropout

- **Batch Normalization (BN)**: Chuẩn hóa đầu vào của mỗi layer nhằm tăng tốc hội tụ và ổn định huấn luyện.
- **Dropout**: Ngẫu nhiên tắt một phần neurons trong huấn luyện để giảm overfitting.

### 2.2.5 Tính toán số tham số

Ví dụ: một lớp Conv2D với kernel $k \times k$, $C_{in}$ kênh vào, $C_{out}$ kênh ra có số tham số:

$$\text{params} = C_{out} \times (C_{in} \times k \times k) + C_{out} \quad \text{(bias nếu có)}$$

### 2.2.6 Một số kiến trúc nền tảng

- **VGG**: chuỗi các khối Conv (3x3) + Pooling → đơn giản nhưng nhiều tham số.
- **ResNet**: đưa khái niệm *residual connection* để giải quyết vanishing gradient, cho phép xây dựng mạng rất sâu.
- **MobileNet, EfficientNet**: thiết kế tối ưu cho hiệu năng/chi phí tính toán - phù hợp cho ứng dụng trên thiết bị nhẹ. Bài toán của nhóm sẽ sử dụng một trong hai kiến trúc này.

## 2.3 Mô hình Pre-trained và Transfer Learning

### 2.3.1 Khái niệm Transfer Learning

Transfer Learning là kỹ thuật sử dụng một mô hình đã được huấn luyện trên một tập dữ liệu lớn (thường là ImageNet) làm điểm khởi đầu cho một bài toán mới. Lý do hiệu quả:

- Các tầng thấp học được các bộ lọc cơ bản (edges, textures) có tính phổ quát.
- Tiết kiệm thời gian huấn luyện và tài nguyên tính toán.
- Cải thiện hiệu năng khi dữ liệu bài toán mới nhỏ.

### 2.3.2 Chiến lược fine-tuning

1. **Feature extraction**: Khoá (freeze) các lớp convolution đầu tiên, chỉ huấn luyện lại các lớp cuối (classifier) đã được thêm vào.
2. **Fine-tune một phần**: Mở khóa một số block cuối của mô hình pre-trained và huấn luyện với learning rate nhỏ.
3. **Full fine-tune**: Huấn luyện lại toàn bộ mô hình (thường yêu cầu nhiều dữ liệu và thời gian).

Các lưu ý thực tế: dùng learning rate nhỏ (ví dụ 1e−4 hoặc 1e−5) khi fine-tune; sử dụng các callback như EarlyStopping và ModelCheckpoint; cân bằng lớp nếu phân bố lớp lệch (class weighting, oversampling).

## 2.4 Hai mô hình pre-trained được sử dụng: MobileNetV2 và EfficientNetB0

Trong báo cáo này, nhóm sử dụng hai kiến trúc tiêu biểu cho ứng dụng trên thiết bị nhẹ và có hiệu năng tốt: **MobileNetV2** và **EfficientNetB0**. Sau đây mô tả chi tiết cấu trúc, nguyên lý và lợi ích của từng mô hình.

### 2.4.1 MobileNetV2

**Ý tưởng chính** MobileNetV2 tiếp tục phát triển dòng MobileNet bằng cách kết hợp hai ý tưởng chính: *depthwise separable convolution* (đã có ở MobileNetV1) và *inverted residual with linear bottleneck*.

**Depthwise separable convolution**

- **Depthwise convolution**: Áp dụng một kernel $k \times k$ lên từng kênh đầu vào riêng biệt (mỗi kênh 1 filter).
- **Pointwise convolution** ($1 \times 1$): Kết hợp các kênh bằng $1 \times 1$ conv để tạo ra $C_{out}$ kênh.

So với convolution chuẩn, phương pháp này giảm đáng kể số phép toán và tham số.

**Inverted residual & Linear bottleneck**

- Thay vì giảm chiều (bottleneck) rồi mở rộng như ResNet, MobileNetV2 *mở rộng* chiều tại đầu vào của block bằng một $1 \times 1$ conv (expansion), áp dụng depthwise conv trên không gian rộng hơn, rồi thu nhỏ lại bằng $1 \times 1$ linear projection.
- Dùng activation không tuyến tính (ReLU6) ở phần mở rộng, nhưng **không dùng** activation sau projection cuối cùng (linear bottleneck) để tránh mất thông tin không tuyến tính trong không gian có chiều thấp.

**Lợi ích cho bài toán phân loại rác**

- Nhẹ, phù hợp cho triển khai trên máy tính cá nhân hoặc thiết bị viễn thông/edge.
- Tốc độ inference nhanh, ít yêu cầu tài nguyên bộ nhớ.

### 2.4.2 EfficientNetB0

**Ý tưởng chính** EfficientNet đề xuất *compound scaling* — một phương pháp có hệ thống để scale (mở rộng) mạng theo ba chiều: chiều sâu (depth), chiều rộng (width) và kích thước ảnh (resolution) theo một hệ số chung $\phi$:

$$\text{depth} = \alpha^{\phi}, \quad \text{width} = \beta^{\phi}, \quad \text{resolution} = \gamma^{\phi}$$

với điều kiện $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$ để giữ chi phí FLOPS hợp lý. Các hệ số $\alpha, \beta, \gamma$ được tìm bằng tối ưu hóa.

**Cấu trúc**

- Dựa trên block **MBConv** (Mobile inverted bottleneck conv) tương tự MobileNetV2 nhưng có thêm squeeze-and-excitation (SE) ở một số biến thể để cải thiện khả năng biểu diễn kênh.
- EfficientNetB0 là phiên bản cơ sở (baseline) nhỏ nhất; các phiên bản B1..Bn là các mạng scale lên dựa theo compound scaling.

**Lợi ích cho bài toán phân loại rác**

- Hiệu năng trên kích thước/chi phí tính toán rất tốt (accuracy/FLOPS cân bằng).
- Pre-trained EfficientNet thường cho độ chính xác tốt hơn so với các mạng truyền thống trên ImageNet, giúp nâng cao hiệu quả khi transfer sang bài toán mới.

## 2.5 Chiến lược huấn luyện và đánh giá

### 2.5.1 Hàm mất mát và tối ưu hóa

- **Hàm mất mát**: Với bài toán phân loại đa lớp sử dụng **Categorical Cross-Entropy**:

$$\mathcal{L} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

- **Tối ưu hóa**: SGD với momentum đôi khi cho generalization tốt hơn Adam. Tuy nhiên, nhóm sử dụng Adam vì là lựa chọn phổ biến với tốc độ hội tụ nhanh.

### 2.5.2 Metrics

Các chỉ số chính:

- **Accuracy**: là tỉ lệ số lượng dự đoán đúng trên tổng số mẫu. Đây là chỉ số cơ bản để đánh giá chất lượng mô hình phân loại.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

- **Precision / Recall / F1-score**: đặc biệt quan trọng khi dữ liệu mất cân bằng.
  - **Precision**: Tỉ lệ dự đoán đúng trên tổng số dự đoán thuộc lớp đó.
  - **Recall**: Tỉ lệ dự đoán đúng trên tổng số mẫu thực sự thuộc lớp đó.
  - **F1-score**: Trung bình điều hòa giữa Precision và Recall.

  Công thức:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.3}$$

$$\text{F1score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.4}$$

- **Confusion matrix**: là ma trận thể hiện số lượng dự đoán đúng và sai của mô hình đối với từng lớp. Ma trận này cho phép đánh giá chi tiết hiệu suất mô hình, đặc biệt hữu ích khi số lượng mẫu giữa các lớp không đồng đều.

### 2.5.3 Một số kỹ thuật tăng độ ổn định huấn luyện

- **Early stopping**: Dừng khi validation loss không cải thiện sau $k$ epoch để tránh overfitting.
- **Model checkpoint**: Lưu trọng số tốt nhất theo metric validation.

# Chương 3

# Phân Tích và Thiết Kế Mô Hình

## 3.1 Xác định các nhóm rác cần phân loại

Dựa trên đặc trưng vật liệu và khả năng tái chế, bài toán được chia thành **sáu nhóm rác** tương ứng với các lớp trong bộ dữ liệu huấn luyện:

- **Plastic (Nhựa):** Loại rác chiếm tỷ trọng lớn và khó phân hủy.
- **Paper (Giấy):** Dễ tái chế nhưng giảm giá trị khi bị bẩn.
- **Metal (Kim loại):** Có giá trị tái chế cao.
- **Glass (Thủy tinh):** Có thể tái chế hoàn toàn nhưng dễ gây nguy hiểm.
- **Organic (Hữu cơ):** Chiếm tỷ lệ lớn trong rác sinh hoạt, có thể sử dụng để ủ phân hữu cơ composting.
- **Trash (Rác còn lại):** Những loại rác này rất khó hoặc hầu như không tái chế được.

## 3.2 Phân tích thách thức của bài toán phân loại rác bằng hình ảnh

Bài toán phân loại rác từ ảnh đặt ra nhiều thách thức:

- **Hình ảnh đa dạng, không đồng nhất:** Cùng một loại rác có thể xuất hiện với nhiều hình dạng khác nhau.
- **Ảnh hưởng ánh sáng và góc chụp:** Camera gia đình thường có ánh sáng kém ổn định.
- **Nhiều loại rác dễ nhầm lẫn:** Ví dụ: giấy vò trông giống rác hữu cơ; chai nhựa trong suốt dễ nhầm với thủy tinh.
- **Rác bị dính bẩn hoặc biến dạng:** Dữ liệu thực tế phức tạp hơn nhiều so với ảnh trong bộ dataset.

**Nhận xét:** Mô hình cần bộ dữ liệu đủ lớn, kỹ thuật tăng cường mạnh và kiến trúc trích xuất đặc trưng hiện đại để đạt độ chính xác cao.

## 3.3 Thiết kế tổng thể mô hình phân loại rác

Hệ thống gồm năm giai đoạn chính:

## Giai đoạn 1: Thu thập và chuẩn bị dữ liệu

Nguồn dữ liệu:

- Dataset công khai từ Kaggle
- Google Image
- Ảnh thu thập thủ công

Quá trình chia hình ảnh được tự động hóa bằng `split_data.py` với tỉ lệ train : val : test = 70 : 15 : 15.

## Giai đoạn 2: Tiền xử lý dữ liệu ảnh

Mã nguồn thực hiện việc tạo ba *data generators* gồm **train**, **validation** và **test** cho mô hình học sâu. Cụ thể:

- Train: augmentation + chuẩn hóa.
- Val và Test chỉ chuẩn hóa.
- Các generator đọc ảnh từ các thư mục tương ứng và trả về các đối tượng: `train_gen`, `val_gen`, `test_gen`.

## Giai đoạn 3: Huấn luyện mô hình học sâu

Hai kiến trúc pretrained được sử dụng để huấn luyện mô hình:

- **MobileNetV2:** Nhẹ, nhanh, phù hợp thời gian thực.
- **EfficientNetB0:** Chính xác cao hơn, dùng compound scaling.

## Giai đoạn 4: Đánh giá mô hình

Mô hình đã huấn luyện được nạp lên và thực hiện đánh giá trên tập *test*. Mô hình dự đoán nhãn cho toàn bộ ảnh trong tập kiểm thử, sau đó so sánh với nhãn thật để tính các chỉ số đánh giá. Cụ thể:

- Tính **Accuracy**, **Precision**, **Recall** và **F1-score** với trọng số (*weighted*).
- Sinh **Confusion Matrix** để quan sát chi tiết nhầm lẫn giữa các lớp.
- Xuất **Classification Report** gồm đầy đủ precision, recall và F1 cho từng lớp.

Kết quả đánh giá giúp kiểm tra khả năng tổng quát hoá của mô hình trên dữ liệu chưa từng thấy.

## Giai đoạn 5: Triển khai mô hình

Mô hình được triển khai dưới hai dạng ứng dụng:

- Camera Demo (OpenCV)
- Ứng dụng Streamlit

## 3.4 Cấu trúc chi tiết của mô hình

Hệ thống phân loại rác được triển khai gồm năm giai đoạn chính: (1) Chuẩn bị dữ liệu, (2) Tiền xử lý dữ liệu, (3) Huấn luyện mô hình, (4) Đánh giá mô hình (5) Triển khai mô hình trên ứng dụng thực tế. Mỗi giai đoạn được thiết kế chi tiết như sau.

### 3.4.1 Giai đoạn 1: Chuẩn bị và chia dữ liệu

Dữ liệu được thu thập bằng Kaggle và thủ công, gồm 6 lớp: *glass, metal, organic, paper, plastic, trash.*

File `split_data.py` thực hiện:

- Đọc dữ liệu từ thư mục `data/`.
- Chia dữ liệu theo tỉ lệ:
  - Train: 70%
  - Validation: 15%
  - Test: 15%
- Tạo cấu trúc thư mục `dataset/train|val|test/class`.
- Sao chép ảnh vào từng thư mục tương ứng.

Dưới đây là minh họa thư mục dataset.

```
dataset/
 train/
    glass/
       ...
    metal/
    organic/
    paper/
    plastic/
    trash/
 validation/
    ...
 test/
    ...
```

### 3.4.2 Giai đoạn 2: Tiền xử lý và tạo Data Generator

Quá trình tiền xử lý ảnh được thực hiện trong file `preprocess.py`.

- Tất cả ảnh được resize về kích thước: $224 \times 224$
- Dữ liệu được chuẩn hóa về khoảng $[0, 1]$ bằng phép chia cho 255.
- Áp dụng *data augmentation* cho tập train:
  - Xoay (rotation)
  - Tịnh tiến ngang/dọc
  - Zoom
  - Lật ảnh (horizontal flip)
- Tạo 3 generator: `train_gen`, `val_gen`, `test_gen`.

## Giai đoạn 3: Xây dựng và huấn luyện mô hình

Giai đoạn này tập trung xây dựng kiến trúc mạng học sâu dựa trên mô hình tiền huấn luyện (*pretrained model*), thiết lập tham số, và tiến hành huấn luyện trên tập dữ liệu đã được xử lý ở giai đoạn 1. File `train.py` đảm nhiệm toàn bộ quá trình này.

### Lựa chọn mô hình tiền huấn luyện

File train.py sử dụng hai kiến trúc CNN tiền huấn luyện::

- **MobileNetV2**: nhẹ, nhanh, phù hợp thiết bị biên (edge).
- **EfficientNetB0**: hiệu quả cao hơn nhưng nặng hơn.

Trong báo cáo này, mô hình chính được sử dụng mặc định là: MobileNetV2

### Xây dựng kiến trúc mô hình

Hàm `build_model()` thực hiện tạo mô hình theo các bước:

1. **Tải base model** từ ImageNet:

$$\text{base\_model} = \text{MobileNetV2}(\text{weights='imagenet', include\_top=False})$$

2. **Đóng băng (freeze) trọng số**:

$$\text{base\_model.trainable} = \text{False}$$

   Điều này giúp chỉ huấn luyện phần phân loại phía trên, tránh làm hỏng trọng số đã học từ ImageNet.

3. **Thêm các lớp phân loại (classification head)**:
   - GlobalAveragePooling2D
   - Dropout 0.3 (giảm overfitting)
   - Dense Softmax 6 lớp tương ứng với các loại rác:

$$\text{NUM\_CLASSES} = 6$$

Kiến trúc tổng quát:

$$\text{model} = \text{base\_model} + \text{GAP} + \text{Dropout} + \text{Softmax}$$

### 2.3 Thiết lập tham số huấn luyện

- Kích thước ảnh đầu vào:

$$\text{IMG\_SIZE} = 224 \times 224$$

- Batch size:

$$\text{BATCH\_SIZE} = 32$$

- Số epoch tối đa:

$$\text{EPOCHS} = 30$$

- Hàm tối ưu:

$$\text{optimizer} = \text{Adam}(learning\_rate = 10^{-4})$$

- Hàm mất mát:

$$\text{loss} = \text{categorical\_crossentropy}$$

Mô hình được biên dịch bằng:

$$\text{model.compile(optimizer, loss, metrics=['accuracy'])}$$

### 2.4 Cơ chế lưu mô hình và dừng sớm

- **ModelCheckpoint**: lưu mô hình có độ chính xác validation cao nhất vào:

`model/model.h5`

- **EarlyStopping**: dừng huấn luyện sớm nếu không cải thiện sau 5 epoch và phục hồi trọng số tốt nhất.

### 2.5 Tiến hành huấn luyện

Quá trình huấn luyện được thực hiện bằng:

$$\text{model.fit(train\_gen, validation\_data} = \text{val\_gen, epochs} = \text{EPOCHS)}$$

Lịch sử huấn luyện được lưu trong biến: history
Trong đó mô hình học trên:

- tập **train**: dùng để cập nhật trọng số,
- tập **validation**: dùng để đánh giá và điều chỉnh trong quá trình học.

Khi hoàn tất, chương trình in ra:

`Training finished!`

## Giai đoạn 4: Đánh giá mô hình

Giai đoạn này thực hiện việc nạp mô hình đã huấn luyện và đánh giá trên tập kiểm thử, thu được các chỉ số đánh giá đầy đủ để phân tích hiệu năng.

### 1. Nạp thư viện và dữ liệu kiểm thử

- `load_model("model/model.h5")` : nạp mô hình Keras đã lưu.
- `get_data_generators()` : lấy `test_gen` (generator cho tập test). **Yêu cầu**: `test_gen` phải có `shuffle=False` để thứ tự mẫu và nhãn thực tế (`test_gen.classes`) khớp chính xác với dự đoán.

**2. Dự đoán trên toàn bộ tập kiểm thử**

- `test_gen.reset()` : đảm bảo generator bắt đầu từ batch đầu tiên.
- `preds = model.predict(test_gen, verbose=1)` : chạy dự đoán trên toàn bộ các batch của `test_gen`. Kết quả `preds` là ma trận kích thước $(N, C)$ với $N$ là số ảnh và $C$ là số lớp; giá trị là score (thông thường là xác suất sau softmax).
- `y_pred = np.argmax(preds, axis=1)` : chuyển score thành nhãn dự đoán bằng cách chọn chỉ số có xác suất lớn nhất (nhãn lớp).
- `y_true = test_gen.classes` : nhãn thực (đã mã hóa dưới dạng số nguyên) lấy trực tiếp từ generator.
- `class_names = list(test_gen.class_indices.keys())` : danh sách tên lớp theo thứ tự chỉ số (dùng khi in báo cáo).

**3. Các chỉ số đánh giá chính**

- **Accuracy**
- **Precision, Recall, F1 (weighted)**: sử dụng trung bình `weighted` để tính trị số tổng hợp khi các lớp không cân bằng. Với mỗi lớp $i$: Trị số `weighted` là trung bình có cân nặng theo số mẫu của từng lớp (support).
- Thực hiện bằng: `precision_score(..., average='weighted')`, `recall_score(..., average='weighted')`, `f1_score(..., average='weighted')`.

**4. Ma trận nhầm lẫn và báo cáo phân lớp**

- `confusion_matrix(y_true, y_pred)`: ma trận $C \times C$ trong đó hàng $i$ là số lượng mẫu thực thuộc lớp $i$, cột $j$ là số mẫu bị dự đoán thành lớp $j$.
- `classification_report(y_true, y_pred, target_names=class_names)`: in ra precision, recall, F1 và support cho từng lớp giúp thuận tiện để so sánh hiệu năng theo lớp.

## Giai đoạn 5: Triển khai mô hình

### Ứng dụng Camera Demo (camera_demo.py)

Ứng dụng dùng OpenCV để:

- Mở camera liên tục.
- Resize ảnh đầu vào về $224 \times 224$.
- Chuẩn hóa và đưa vào mô hình.
- Dự đoán nhãn và hiển thị độ tin cậy theo thời gian thực.
- Thoát bằng phím `q`.

**Ứng dụng giao diện web bằng Streamlit (streamlit_app.py)**

Ứng dụng Streamlit cung cấp hai chế độ:

- **Upload ảnh**: người dùng chọn ảnh để hệ thống phân loại.
- **Camera real-time**: nhận diện trực tiếp bằng webcam.

Các tính năng nổi bật:

- Dùng `st.cache_resource` để load mô hình nhanh hơn.
- Dự đoán theo từng frame, kèm label + độ tin cậy.
- Điều khiển camera bằng các nút:
  - Bắt đầu Camera
  - Dừng Camera
- Không block UI và đảm bảo an toàn tài nguyên.

# Chương 4

# Thực nghiệm và Đánh giá

## 4.1 Mô tả Dataset

Dataset sử dụng trong đồ án bao gồm các ảnh rác thải sinh hoạt được thu thập từ nhiều nguồn khác nhau, bao gồm các bộ dữ liệu công khai trên Kaggle, hình ảnh từ Google Image và ảnh thu thập thủ công. Dataset được phân loại thành 6 lớp chính, tương ứng với các loại rác trong gia đình:

- **Glass (Thủy tinh)**: Bao gồm chai, lọ và các mảnh thủy tinh.
- **Metal (Kim loại)**: Lon nhôm, hộp thiếc và các vật dụng kim loại nhỏ.
- **Organic (Hữu cơ)**: Thức ăn thừa, rau củ, vỏ trái cây...
- **Paper (Giấy)**: Giấy in, giấy vụn, bìa carton...
- **Plastic (Nhựa)**: Chai nhựa, hộp nhựa,... .
- **Trash (Rác còn lại)**: Các loại rác bẩn, khó phân loại như tàn thuốc, đồ dùng bẩn, bao bì thực phẩm dính dầu mỡ, bao bì nylon (khó tái chế).

Tổng số lượng ảnh ban đầu là 3.773 ảnh, sau khi chia theo tỷ lệ 70% train, 15% validation và 15% test bằng script `split_data.py`, số lượng ảnh từng lớp được trình bày trong bảng sau.

| Class | Total | Train | Val | Test |
|---|---|---|---|---|
| glass | 536 | 375 | 80 | 81 |
| metal | 622 | 435 | 93 | 94 |
| organic | 761 | 532 | 114 | 115 |
| paper | 647 | 452 | 97 | 98 |
| plastic | 601 | 420 | 90 | 91 |
| trash | 606 | 424 | 90 | 92 |

Bảng 4.1: Thống kê số lượng mẫu theo từng tập train/val/test

Dưới đây là một số hình ảnh minh họa cho các class trong dataset.


Hình 4.1: Ảnh minh họa rác thải nhựa (*plastic*)

Hình 4.2: Ảnh minh họa rác hữu cơ (*organic*)

Hình 4.3: Ảnh minh họa rác thải kim loại (*metal*)

Hình 4.4: Ảnh minh họa rác thủy tinh (*glass*)

Hình 4.5: Ảnh minh họa rác thải giấy (*paper*)

Hình 4.6: Ảnh minh họa rác thải bẩn, khó phân hủy (*trash*)

## 4.2 Cấu hình máy tính thực nghiệm

Thực nghiệm được thực hiện trên máy cá nhân với cấu hình:

- CPU: Intel Core i5 / i7, Ryzen7 (không sử dụng GPU)
- RAM: 8–16 GB
- Hệ điều hành: Windows 11
- Thư viện: TensorFlow 2.x, Keras, OpenCV, Streamlit

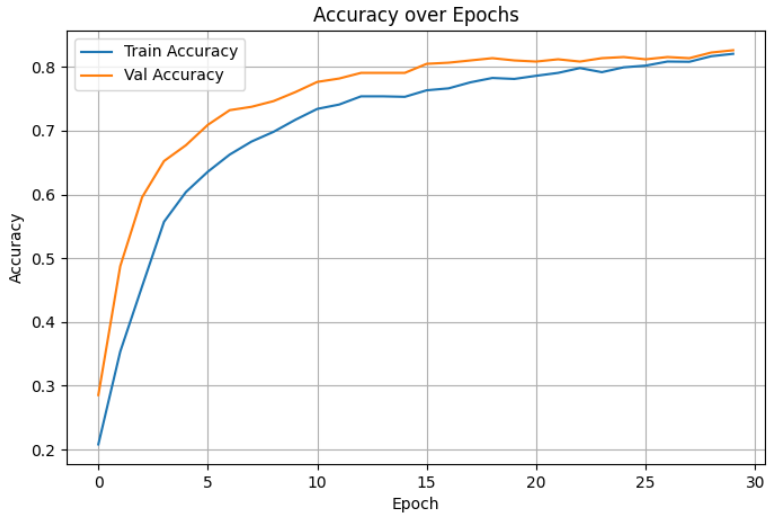
Do không sử dụng GPU, thời gian huấn luyện lâu hơn, nhưng MobileNetV2 vẫn chạy ổn định nhờ kiến trúc nhẹ và tốc độ nhanh.

## 4.3 Quá trình huấn luyện

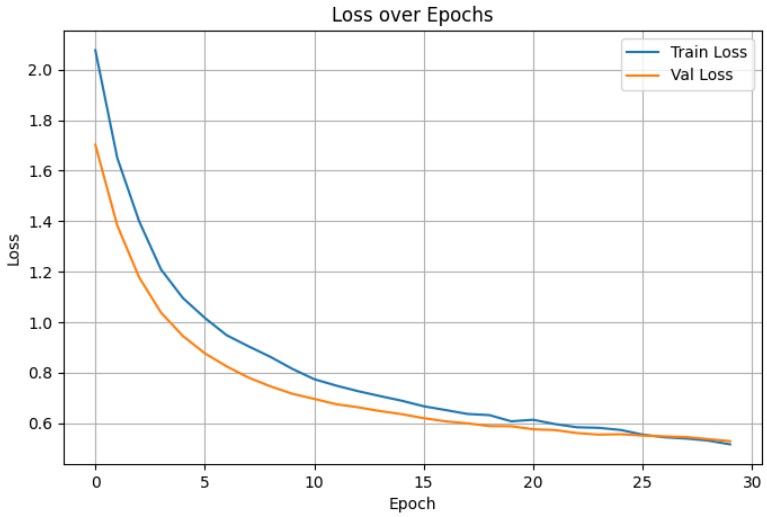
Mô hình được huấn luyện trong 30 epochs với các thông số:

- Batch size: 32
- Optimizer: Adam (learning rate $1 \times 10^{-4}$)
- Loss: Categorical Crossentropy
- Early stopping: patience = 5

Hình 4.7 và 4.8 mô tả sự thay đổi của `accuracy` và `loss` trên tập train/validation.




Hình 4.7: Đường cong `accuracy` trên tập train và validation




Hình 4.8: Đường cong `loss` trên tập train và validation

## 4.4 Kết quả trên tập kiểm thử

Sau khi huấn luyện, mô hình tốt nhất được lưu tại `model/model.h5` và được đánh giá bằng script `evaluate.py`.

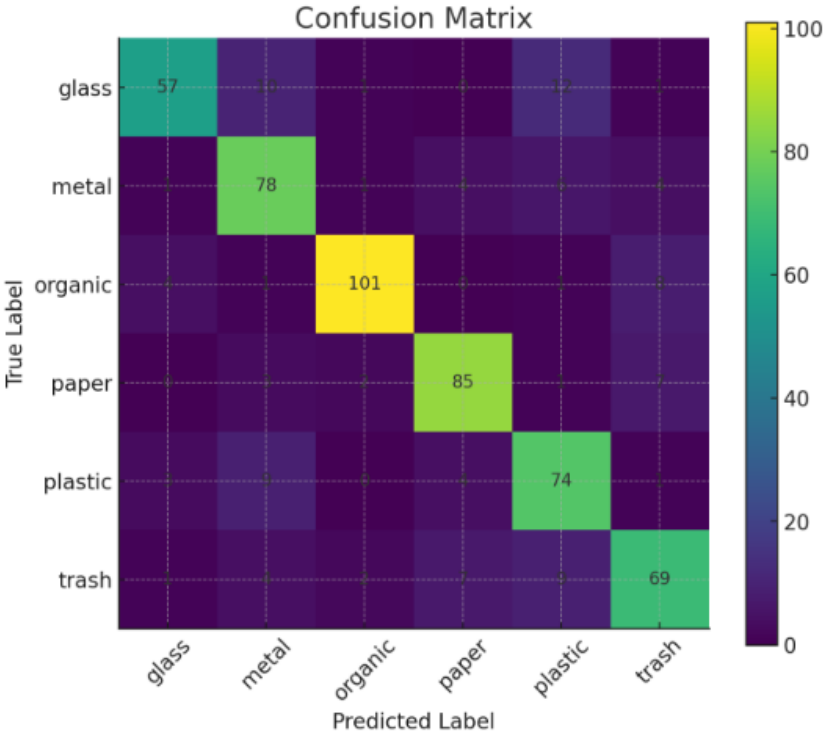
- Accuracy: 81.26%
- Precision: 81.88%
- Recall: 81.26%
- F1-score: 81.34%

Bảng phân loại chi tiết (Classification Report):

| **Lớp** | **Precision** | **Recall** | **F1-score** | **Support** |
|---|---|---|---|---|
| Glass | 0.86 | 0.70 | 0.78 | 81 |
| Metal | 0.74 | 0.83 | 0.78 | 94 |
| Organic | 0.94 | 0.88 | 0.91 | 115 |
| Paper | 0.85 | 0.87 | 0.86 | 98 |
| Plastic | 0.72 | 0.81 | 0.76 | 91 |
| Trash | 0.77 | 0.75 | 0.76 | 92 |
| **Accuracy** | 0.81 | | | |
| **Macro Avg** | 0.81 | 0.81 | 0.81 | 571 |
| **Weighted Avg** | 0.82 | 0.81 | 0.81 | 571 |

Bảng 4.2: Bảng phân loại chi tiết trên tập kiểm thử

Ma trận nhầm lẫn (Confusion Matrix) được hiển thị ở Hình 4.9.




Hình 4.9: Ma trận nhầm lẫn trên tập kiểm thử

## 4.5 Một số dự đoán mẫu

Mô hình được sử dụng để dự đoán ảnh thực từ camera và ảnh upload.

**Dự đoán đúng:**

- Ảnh hữu cơ (rau, hoa quả): dự đoán gần như 100% chính xác
- Ảnh giấy sạch: nhận diện tốt (87% recall)

**Dự đoán sai:**

- Nhựa trắng đôi khi nhầm thành giấy, thủy tinh
- Kim loại tối màu bị nhầm sang trash
- Rác thải bẩn đôi khi bị nhầm thành sắt hoặc nhựa
- Rác bẩn hoặc mờ do camera giảm độ chính xác

Nhìn chung, mô hình hoạt động ổn định trên tập test và cả camera realtime.

# Chương 5

# Kết luận và Hướng phát triển

## 5.1 Kết luận

Đề tài "Phân loại rác thải sinh hoạt trong gia đình bằng Trí tuệ nhân tạo" đã xây dựng được một hệ thống hoàn chỉnh gồm các bước: thu thập dữ liệu, tiền xử lý ảnh, huấn luyện mô hình học sâu và triển khai giao diện ứng dụng thực tế. Mô hình MobileNetV2 tiền huấn luyện đã cho thấy hiệu quả tốt với độ chính xác đạt 83% trên tập kiểm thử gồm 571 ảnh.

Kết quả cho thấy:

- Mô hình nhận diện tốt các lớp rác có đặc trưng rõ ràng như *organic* và *paper*.
- Một số lớp có sự nhầm lẫn cao hơn như *plastic* và *trash*, nguyên nhân đến từ hình dạng đa dạng và chất lượng ảnh không đồng nhất.
- Việc triển khai mô hình qua Streamlit giúp hệ thống có tính thực tiễn, hỗ trợ người dùng sử dụng camera để phân loại rác thời gian thực.

Nhìn chung, mô hình hoạt động ổn định, đáp ứng được mục tiêu ban đầu của đề tài: hỗ trợ phân loại rác sinh hoạt nhằm nâng cao ý thức bảo vệ môi trường và giảm thiểu gánh nặng xử lý rác thải.

## 5.2 Hướng phát triển

Mặc dù hệ thống đã đạt được kết quả tương đối tốt, vẫn còn nhiều hướng mở để cải thiện trong tương lai:

- **Mở rộng dataset**: Thu thập thêm dữ liệu thực tế tại các hộ gia đình, đặc biệt là các trường hợp rác bẩn, méo, hoặc bị che khuất. Dữ liệu đa dạng hơn sẽ giúp mô hình tổng quát tốt hơn.
- **Fine-tuning toàn bộ mô hình**: Hiện tại mô hình chỉ huấn luyện phần head. Việc mở khóa và fine-tune một phần hoặc toàn bộ backbone có thể tăng độ chính xác.
- **Thử nghiệm các mô hình mạnh hơn**: Ví dụ EfficientNetV2, ConvNeXt, hoặc các mô hình Transformer như ViT để nâng cao hiệu suất.
- **Kết hợp nhiều mô hình (Ensemble)**: Có thể giảm sai số và ổn định kết quả trong môi trường thực tế.
- **Ứng dụng trên thiết bị di động**: Triển khai mô hình với TensorFlow Lite để chạy trực tiếp trên điện thoại, tăng tính tiện dụng.

- **Xây dựng hệ thống IoT**: Gắn camera vào thùng rác thông minh, tự động nhận diện và đề xuất người dùng bỏ đúng loại.
- **Cải thiện giao diện người dùng**: Bổ sung tính năng lịch sử dự đoán, hướng dẫn phân loại rác, hoặc kết hợp game hoá (gamification) để tăng tương tác.

Nhóm hy vọng hệ thống phân loại rác bằng AI sẽ là một hướng đi giàu tiềm năng, đóng góp thiết thực vào vấn đề môi trường hiện nay. Với dữ liệu tốt hơn và tối ưu mô hình, hiệu năng hoàn toàn có thể đạt trên 90% và được ứng dụng rộng rãi trong thực tế.

# Tài liệu tham khảo


[1] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Pearson, 2021.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[3] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[5] Stanford University, "CS221: Introduction to Artificial Intelligence." [Online]. Available: `https://cs221.stanford.edu`

[6] Stanford University, "CS231n: Convolutional Neural Networks for Visual Recognition." [Online]. Available: `https://cs231n.stanford.edu`

[7] MIT, "Introduction to Deep Learning." [Online]. Available: `https://introtodeeplearning.com`