BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC ĐIỆN LỰC

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO CHUYÊN ĐỀ HỌC PHẦN
PHÂN TÍCH VÀ TRỰC QUAN HÓA DỮ LIỆU

ĐỀ TÀI:

PHÂN TÍCH VÀ TRỰC QUAN HÓA DỮ LIỆU PHÂN LOẠI CHÓ VÀ MÈO

Sinh viên thực hiện	**: TRỊNH THANH TÙNG**
Giảng viên hướng dẫn	**: TS. VŨ VĂN ĐỊNH**
Ngành	**: CÔNG NGHỆ THÔNG TIN**
Chuyên ngành	**: CÔNG NGHỆ PHẦN MỀM**
Lớp	**: D17CNPM3**
Khóa	**: 2022-2027**

Hà Nội, tháng … năm 2025

PHIẾU CHẤM ĐIỂM

Sinh viên thực hiện:

Họ và tên	Chữ ký	Điểm
Trịnh Thanh Tùng 22810310321		

Giảng viên chấm:

Họ và tên	Chữ ký	Ghi chú
Giảng viên chấm 1:		
Giảng viên chấm 2:		

MỤC LỤC

DANH MỤC BẢNG BIỂU

DANH MỤC HÌNH ẢNH

LỜI CẢM ƠN

Trong bối cảnh trí tuệ nhân tạo và học sâu (Deep Learning) đang ngày càng phát triển mạnh mẽ, các kỹ thuật xử lý và phân tích dữ liệu ảnh đã trở thành nền tảng quan trọng trong nhiều ứng dụng thực tiễn, từ y tế, giao thông cho đến nhận dạng hình ảnh trong đời sống hằng ngày. Việc nghiên cứu các mô hình phân loại ảnh, đặc biệt trong bài toán nhận diện chó và mèo, không chỉ giúp hiểu rõ hơn về quy trình xử lý dữ liệu hình ảnh mà còn mở ra cơ hội tiếp cận với các phương pháp học máy tiên tiến, tối ưu hoá mô hình và trực quan hóa dữ liệu nhằm nâng cao hiệu quả phân loại.

Với tinh thần học hỏi và mong muốn vận dụng kiến thức vào thực tiễn, chúng em đã thực hiện đề tài **"Phân tích và trực quan hóa dữ liệu phân loại chó và mèo"**. Thông qua đề tài này, chúng em có cơ hội tìm hiểu quy trình thu thập và tiền xử lý dữ liệu ảnh, áp dụng các kỹ thuật trực quan hóa dữ liệu, xây dựng và huấn luyện mô hình phân loại dựa trên mạng nơ-ron tích chập (CNN), từ đó rèn luyện kỹ năng lập trình, phân tích dữ liệu và tư duy thuật toán.

Chúng em nhận thức rằng kinh nghiệm nghiên cứu thực tế của bản thân còn hạn chế, do đó đề tài khó tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự góp ý và chia sẻ kinh nghiệm từ thầy cô và các bạn để có thể hoàn thiện hơn trong tương lai. Những đóng góp đó sẽ là nguồn động lực quý báu giúp chúng em tiếp tục học tập và phát triển trong lĩnh vực trí tuệ nhân tạo và xử lý ảnh.

Cuối cùng, chúng em xin gửi lời cảm ơn chân thành đến **thầy Vũ Văn Định** – người đã tận tình hướng dẫn, định hướng và hỗ trợ chúng em trong suốt quá trình thực hiện đề tài. Chúng em cũng xin trân trọng cảm ơn tất cả những người đã tạo điều kiện, hỗ trợ và đồng hành cùng chúng em trong quá trình học tập và nghiên cứu. Những kiến thức và kinh nghiệm thu được từ đề tài sẽ là hành trang quý giá giúp chúng em tự tin hơn trên con đường nghiên cứu và phát triển nghề nghiệp sau này.

LỜI MỞ ĐẦU

Trong kỷ nguyên của dữ liệu lớn và Trí tuệ Nhân tạo, việc xây dựng các hệ thống **Phân loại hình ảnh** tự động đã trở thành một thách thức cốt lõi trong lĩnh vực Thị giác Máy tính (Computer Vision). Bài toán **Phân loại Chó và Mèo** là một bài toán phân loại nhị phân điển hình, đòi hỏi mô hình không chỉ nhận diện được sự khác biệt cơ bản về hình dạng mà còn phải xử lý được sự đa dạng lớn về góc chụp, kích thước, ánh sáng và màu sắc của dữ liệu ảnh. Để giải quyết hiệu quả sự phức tạp này, trọng tâm của nghiên cứu không chỉ nằm ở kiến trúc mô hình Mạng nơ-ron Tích chập (CNN) mà còn nằm ở quá trình **phân tích và xử lý dữ liệu đầu vào**.

Quá trình **Phân tích dữ liệu ảnh (Image Data Analysis)** đóng vai trò tiên quyết, giúp tôi hiểu rõ về bản chất của tập dữ liệu: từ sự khác biệt về kích thước ảnh, dải giá trị pixel, cho đến sự cân bằng hay mất cân bằng giữa hai lớp. Những phân tích này là cơ sở để thiết kế các bước **tiền xử lý dữ liệu** bắt buộc, bao gồm việc **chuẩn hóa** (normalization) dải giá trị pixel và **tái định cỡ** (resizing) các ảnh về cùng một kích thước đồng nhất, nhằm đảm bảo mô hình nhận được đầu vào ổn định và tối ưu hóa tốc độ hội tụ trong quá trình huấn luyện.

Đặc biệt, nhằm giải quyết vấn đề overfitting – một rào cản lớn khi làm việc với tập dữ liệu ảnh giới hạn tôi tập trung sâu vào kỹ thuật **Tăng cường Dữ liệu (Data Augmentation)**. Kỹ thuật này là một hình thức **xử lý dữ liệu** tại chỗ, tự động tạo ra các biến thể mới của ảnh gốc (như xoay, lật, dịch chuyển, zoom) để mở rộng tập huấn luyện "ảo", giúp mô hình học được các đặc trưng mạnh mẽ hơn và tổng quát hóa tốt hơn. Toàn bộ quá trình từ phân tích dữ liệu thô, áp dụng các giải thuật xử lý, cho đến đánh giá hiệu suất của mô hình sẽ được **trực quan hóa** triệt để (qua biểu đồ phân bố và đường cong huấn luyện), nhằm minh chứng rõ ràng tác động của việc xử lý dữ liệu đến sự gia tăng hiệu suất phân loại ảnh cuối cùng.

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1.1. Tổng quan về phân tích và trực quan hóa dữ liệu

1.1.1. Phân tích dữ liệu (Data Analysis)

Phân tích dữ liệu là quá trình kiểm tra, làm sạch, biến đổi và mô hình hóa dữ liệu nhằm mục đích khám phá thông tin hữu ích, đưa ra kết luận và hỗ trợ việc ra quyết định. Trong lĩnh vực học máy, phân tích dữ liệu là bước đầu tiên và quan trọng nhất.

Đối với dữ liệu ảnh, phân tích dữ liệu không chỉ kiểm tra số lượng mẫu ảnh trong mỗi lớp (Chó và Mèo) mà còn là xem xét các yếu tố như: dải giá trị của pixel (từ 0 đến 255), sự khác biệt về kích thước và định dạng tệp. Việc phân tích này giúp xác định các bước tiền xử lý cần thiết.

a. Các giai đoạn của phân tích dữ liệu

- **Phân tích Thăm dò (Exploratory Data Analysis - EDA):** Đây là bước đầu tiên, sử dụng các phương pháp thống kê và trực quan để khám phá tập dữ liệu, tóm tắt các đặc điểm chính, và xác định các bất thường. EDA giúp đặt nền móng cho việc lựa chọn mô hình và chiến lược tiền xử lý.
- **Làm sạch Dữ liệu (Data Cleaning/Wrangling):** Xử lý các vấn đề như dữ liệu bị thiếu (Missing Values), dữ liệu ngoại lai (Outliers), hoặc định dạng không nhất quán. Trong tập dữ liệu ảnh, bước này bao gồm việc chuẩn hóa tên tệp, kiểm tra tệp hỏng (corrupted files), và đảm bảo tính đồng nhất của nhãn (labels).
- **Chuyển đổi Dữ liệu (Data Transformation):** Áp dụng các phép toán để đưa dữ liệu về dạng phù hợp với mô hình. Ví dụ, trong bài toán ảnh, việc **chuẩn hóa pixel** (đưa giá trị 0 - 255 về 0 - 1 hoặc phân phối Gauss) là một kỹ thuật chuyển đổi thiết yếu giúp mô hình học hiệu quả hơn.

b. Phân tích dữ liệu trong lĩnh vực Thị giác máy tính:

Đối với dữ liệu ảnh, phân tích không chỉ dừng lại ở thống kê số lượng mẫu ảnh trong mỗi lớp (**Chó** và **Mèo** - nhằm kiểm tra sự mất cân bằng lớp hay *Class Imbalance*), mà còn đi sâu vào các yếu tố kỹ thuật của hình ảnh:

- **Phân phối Giá trị Pixel:** Kiểm tra dải giá trị của pixel (thường là 0 đến 255 cho ảnh 8-bit). Việc hiểu rõ phân phối này rất quan trọng để quyết định phương pháp chuẩn hóa (Normalization) hay tiêu chuẩn hóa ($Standardization$) phù hợp.
- **Tính đa dạng về Kích thước và Tỷ lệ Khung hình:** Phân tích sự khác biệt về chiều cao (H), chiều rộng (W), và số kênh (C) của ảnh. Các mô hình CNN thường yêu cầu đầu vào có kích thước cố định, do đó, phân tích này giúp xác định kích thước ảnh đầu vào lý tưởng ($Input$ $Shape$) và chiến lược thay đổi kích thước (Resizing Strategy).
- **Kiểm tra Siêu dữ liệu (Metadata):** Xem xét định dạng tệp (JPEG, PNG), thông tin Exif (thông số camera, ngày chụp), có thể ảnh hưởng đến kết quả cuối cùng.

1.1.2. Trực quan hóa dữ liệu và kết quả (Data Visualization)

Trực quan hóa dữ liệu là việc sử dụng các yếu tố thị giác (biểu đồ, đồ thị, sơ đồ) để truyền đạt thông tin một cách rõ ràng và hiệu quả:

a. Trực quan hóa Dữ liệu Nguồn (Source Data Visualization):

Đây là việc trực quan hóa các mẫu dữ liệu thô, giúp con người có cái nhìn tổng quan về sự đa dạng và chất lượng của tập dữ liệu:

- **Hiển thị Ảnh Mẫu:** Trình bày ngẫu nhiên một lưới các ảnh mẫu từ mỗi lớp (ví dụ: 4 ảnh Chó và 4 ảnh Mèo). Điều này giúp **nhanh chóng nhận ra tính đa dạng** của tập dữ liệu về:

- **Ánh sáng và Độ tương phản:** Ảnh bị phơi sáng quá mức hay thiếu sáng.
- **Màu sắc và Kênh màu:** Ảnh màu (RGB) hay ảnh xám (Grayscale).
- **Góc chụp và Bối cảnh:** Mức độ phức tạp của hậu cảnh, góc nhìn của chủ thể.

- **Biểu đồ Tần suất (Histogram):** Vẽ biểu đồ tần suất của các giá trị pixel trung bình để phát hiện các vấn đề như ảnh quá tối (tập trung ở giá trị thấp) hay quá sáng (tập trung ở giá trị cao).
- **Biểu đồ Kích thước Ảnh (Scatter Plot):** Trực quan hóa mối quan hệ giữa chiều rộng và chiều cao của các ảnh trong tập dữ liệu để xác định kích thước phổ biến nhất.

b. Trực quan hóa Kết quả Mô hình (Model Result Visualization):

Đây là công cụ **không thể thiếu** để giám sát quá trình học và đánh giá hiệu suất của mô hình. Các biểu đồ này cung cấp bằng chứng trực quan về cách mô hình đang học:

- **Đường cong Độ chính xác (Accuracy Curve):** Biểu đồ thể hiện **Độ chính xác (Accuracy)** trên tập huấn luyện (Training Set) và tập kiểm tra (Validation Set) theo số lượng epoch.
  - *Mô hình Học Tốt:* Hai đường cong tăng dần và song hành gần nhau.
  - *Mô hình **Overfitting** (Học thuộc lòng):* Độ chính xác huấn luyện tiếp tục tăng cao, trong khi độ chính xác kiểm tra đi ngang hoặc bắt đầu giảm. Điều này cho thấy mô hình đang học các nhiễu cụ thể của dữ liệu huấn luyện thay vì các đặc trưng tổng quát.
- **Hàm Mất mát (Loss Curve):** Biểu đồ thể hiện **Giá trị Hàm mất mát (Loss Value)** theo số lượng epoch. Đây là hình ảnh trực quan nhất thể hiện tốc độ và hiệu quả của việc tối ưu hóa.

- Sự khác biệt lớn giữa Loss trên tập huấn luyện và Loss trên tập kiểm tra là dấu hiệu mạnh mẽ của **Overfitting**.

- **Ma trận Nhầm lẫn (Confusion Matrix):** Một bảng cho thấy hiệu suất của thuật toán phân loại. Nó cho biết có bao nhiêu trường hợp được phân loại đúng (True Positives, True Negatives) và bao nhiêu trường hợp bị phân loại sai (False Positives, False Negatives).
- **Bản đồ Kích hoạt Grad-CAM (Grad-CAM Activation Maps):** Một kỹ thuật trực quan hóa cho biết khu vực nào của ảnh được mô hình chú ý nhất khi đưa ra quyết định phân loại. Ví dụ, nếu mô hình phân loại một bức ảnh là "Mèo" và Grad-CAM highlight vào phần tai và râu, điều này xác nhận mô hình đang học các đặc trưng đúng đắn.

1.2. Mô hình mạng nơ-ron tích chập (CNN) và ứng dụng trong phân loại ảnh

1.2.1. Sự cần thiết của mạng nơ-ron tích chập (CNN)

Trong bài toán xử lý ảnh, mỗi bức ảnh là một ma trận dữ liệu ba chiều (chiều cao H, chiều rộng W và số kênh màu C).

Trong bài toán phân loại ảnh, nếu sử dụng mạng nơ-ron truyền thống (Fully Connected Network), chúng ta phải "làm phẳng" (flatten) toàn bộ pixel của ảnh thành một vector duy nhất. Điều này dẫn đến hai vấn đề lớn:

- Mất cấu trúc không gian: Mô hình không thể nhận biết được mối quan hệ gần gũi giữa các pixel (ví dụ: góc tai mèo và đường viền trên đầu mèo).
- Quá nhiều tham số: Với ảnh có độ phân giải lớn, số lượng tham số trở nên khổng lồ, dẫn đến chi phí tính toán cao và dễ gây overfitting.

Từ đây, Convolutional Neural Network (CNN) ra đời để giải quyết các vấn đề trên. CNN hoạt động tương tự cách mắt người xử lý hình ảnh, bằng cách tập trung vào các đặc trưng cục bộ (như đương nét, góc cạnh, màu sắc) trước khi kết hợp chúng lại thành một hình ảnh hoàn chỉnh.

1.2.2. Cơ chế hoạt động của các lớp cơ bản trong CNN

Bảng 1 Ba lớp chính hoạt động liên tiếp của kiến trúc CNN

Lớp	Vai trò	Phép so sánh
Lớp Tích chập (Convolutional Layer)	**Trích xuất đặc trưng**: Sử dụng các bộ lọc (filters) để quét qua ảnh, tìm kiếm các mẫu nhỏ như đường nét, màu sắc	Giống như một **đội thám tử** tìm kiếm các manh mối nhỏ (như tai, mũi, lông) trên toàn bộ bức ảnh.
Lớp Gộp (Pooling Layer)	**Giảm chiều dữ liệu**: Giảm kích thước bản đồ đặc trưng, loại bỏ các chi tiết không quan trọng	Giống như việc **tóm tắt** các manh mối của thám tử để giữ lại những đặc trưng quan trọng nhất.
Lớp Kết nối đầy đủ (Fully Connected Layer)	**Đưa ra quyết định phân loại**: Sử dụng các đặc trưng cuối cùng đã được tổng hợp để đưa ra xác suất ảnh là Cho hay Mèo.	Giống như một **thẩm phán** đưa ra phán quyết cuối cùng dựa trên tất cả các bằng chứng đã được tóm tắt.

1.3. Các kỹ thuật Tiền xử lý và Tăng cường dữ liệu ảnh

1.3.1. Tiền xử lý Dữ liệu (Preprocessing)

Đây là bước bắt buộc nhằm đảm bảo dữ liệu đầu vào có chất lượng cao và đồng nhất, phù hợp với yêu cầu của mô hình.

a. Thay đổi Kích thước (Resizing):

Tất cả các ảnh phải được chuyển về cùng một kích thước đầu vào (ví dụ: 128 x 128 hoặc 224 x 224). Kỹ thuật padding (thêm viền) hoặc cropping (cắt xén) có thể được sử dụng để giữ tỷ lệ khung hình.

b. Chuẩn hóa Pixel (Normalization):

- Đưa dải giá trị pixel về một phạm vi nhỏ và đồng nhất.
- *Chuẩn hóa Min-Max:* Thường đưa giá trị từ [0, 255] về [0, 1] bằng cách chia cho 255. Kỹ thuật này giúp tốc độ hội tụ của mô hình nhanh hơn và ổn định hơn.

c. Mã hóa Nhãn (Label Encoding):

Chuyển các nhãn phân loại dạng chuỗi (ví dụ: "Chó", "Mèo") sang dạng số. Trong phân loại đa lớp, thường dùng One-Hot Encoding (ví dụ: Chó → [1, 0], Mèo → [0, 1]).

1.3.2. Tăng cường Dữ liệu (Data Augmentation)

Tăng cường dữ liệu là một kỹ thuật mạnh mẽ được sử dụng để làm giàu tập dữ liệu huấn luyện bằng cách tạo ra các phiên bản biến thể của các mẫu ảnh gốc.

Mục đích: Tăng cường dữ liệu giúp tăng tính tổng quát hóa (Generalization) của mô hình và là một biện pháp hữu hiệu để chống Overfitting. Bằng cách giới thiệu các góc chụp, ánh sáng, và vị trí vật thể khác nhau, mô hình học được các đặc trưng thực sự của vật thể thay vì ghi nhớ bối cảnh cụ thể.

Bảng 2 Các kỹ thuật tăng cường ảnh

Kỹ thuật Tăng cường	**Mô tả**	**Ứng dụng**
Xoay (Rotation)	Xoay ảnh một góc ngẫu nhiên ($\pm 15^o$)	Giúp mô hình nhận diện vật thể ở các góc nghiêng khác nhau.
Lật (Flipping)	Lật ảnh theo chiều ngang (Horizontal Flip)	Rất hữu ích vì vật thể thường đối xứng và việc lật không làm thay đổi nhãn.
Biến đổi Màu sắc	Thay đổi ngẫu nhiên độ sáng (Brightness), độ tương phản (Contrast), hoặc bão hòa (Saturation).	Giúp mô hình bền vững với các điều kiện ánh sáng khác nhau trong môi trường thực tế.
Cắt xén ngẫu nhiên (Random Crop)	Cắt lấy một phần ngẫu nhiên của ảnh.	Đảm bảo mô hình không chỉ tập trung vào trung tâm ảnh.

CHƯƠNG 2: PHÂN TÍCH VÀ TIỀN XỬ LÝ DỮ LIỆU ẢNH

2.1. Khám phá bộ dữ liệu ban đầu (Dataset Exploration)

Bước đầu tiên của quy trình là khảo sát xem dữ liệu chứa những gì, số lượng ảnh mỗi lớp, và kiểm tra một vài tệp mẫu.

Code dưới đây duyệt toàn bộ file trong thư mục gốc, tách thành hai danh sách: **dog_files** và **cat_files**.

```
def explore_dataset():
    print("\nExploring dataset...")

    if not train_files:
        print("No training files found!")
        return [], [], []

    # Count dogs and cats
    dog_files = []
    cat_files = []

    for filename in train_files:
        if 'dog' in filename.lower():
            dog_files.append(filename)
        elif 'cat' in filename.lower():
            cat_files.append(filename)

    print("Dog images:", len(dog_files))
    print("Cat images:", len(cat_files))
    print("Total training images:", len(train_files))

    if dog_files:
        print("Sample dog files:", dog_files[:3])
    if cat_files:
        print("Sample cat files:", cat_files[:3])

    return train_files, dog_files, cat_files
```

Hình 2.1 Mã nguồn khám phá bộ dữ liệu ban đầu

Kết quả thu được:

- Dog images: 12,500
- Cat images: 12,500
- Tổng ảnh: 25,000

Bộ dữ liệu được cân bằng hoàn toàn → thuận lợi cho huấn luyện mô hình.

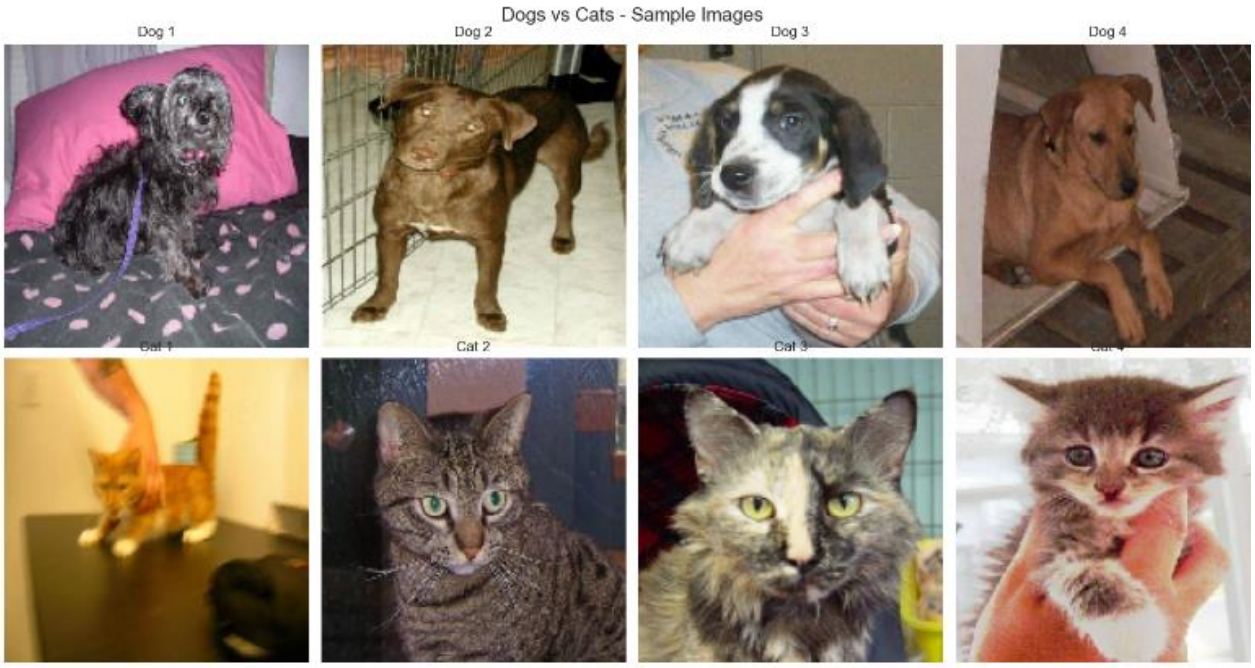
2.2. Hiển thị một số ảnh mẫu

Để quan sát trực tiếp cấu trúc dữ liệu, mã nguồn hiển thị 8 ảnh đầu tiên của chó và mèo.

```
# Display sample images
def show_sample_images():
    print("\nShowing sample images...")

    if not train_files:
        print("No images to display!")
        return

    fig, axes = plt.subplots(2, 4, figsize=(15, 8))
    fig.suptitle('Dogs vs Cats - Sample Images', fontsize=16)

    if dog_files and cat_files:
        # Show 4 dog images
        for i in range(min(4, len(dog_files))):
            img_path = os.path.join(TRAIN_DIR, dog_files[i])
            img = load_img(img_path, target_size=(150, 150))
            axes[0, i].imshow(img)
            axes[0, i].set_title('Dog ' + str(i+1))
            axes[0, i].axis('off')

        # Show 4 cat images
        for i in range(min(4, len(cat_files))):
            img_path = os.path.join(TRAIN_DIR, cat_files[i])
            img = load_img(img_path, target_size=(150, 150))
            axes[1, i].imshow(img)
            axes[1, i].set_title('Cat ' + str(i+1))
            axes[1, i].axis('off')
```

Hình 2.2 Mã nguồn hiển thị một số ảnh mẫu

```
Exploring dataset...
Dog images: 12500
Cat images: 12500
Total training images: 25000
Sample dog files: ['dog.0.jpg', 'dog.1.jpg', 'dog.10.jpg']
Sample cat files: ['cat.0.jpg', 'cat.1.jpg', 'cat.10.jpg']

Showing sample images...
```



Hình 2.3 Hình ảnh ví dụ từ tập dữ liệu nguồn

Nhận xét từ hình ảnh: Kích thước ảnh không đồng nhất; Tư thế, màu sắc, phông nền đa dạng; Đây là bài toán phân loại "tự nhiên" khó, phù hợp dùng CNN

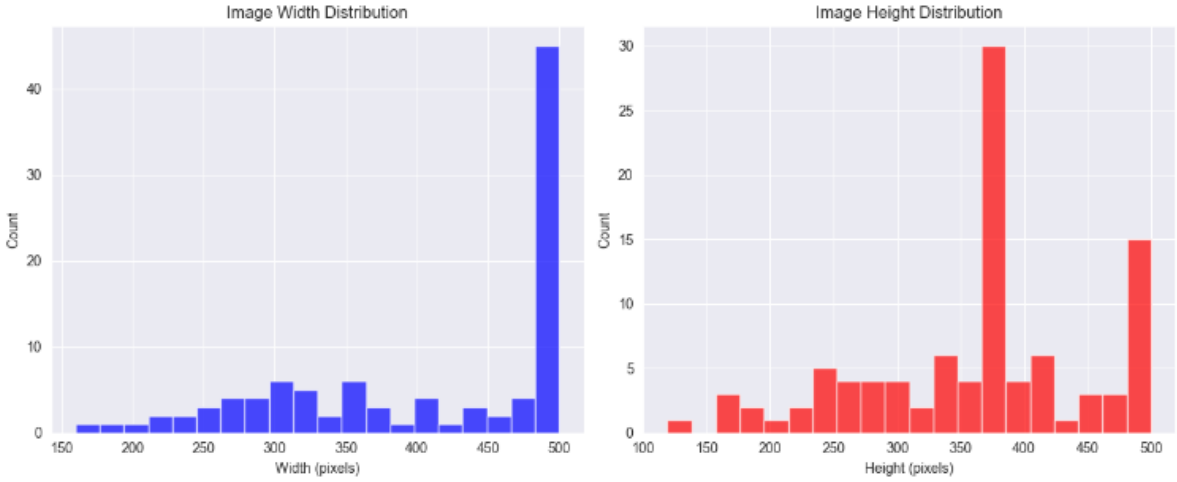
2.3. Phân tích kích thước ảnh

Việc phân tích kích thước ảnh giúp ta lựa chọn kích thước chuẩn (input size) cho mô hình.

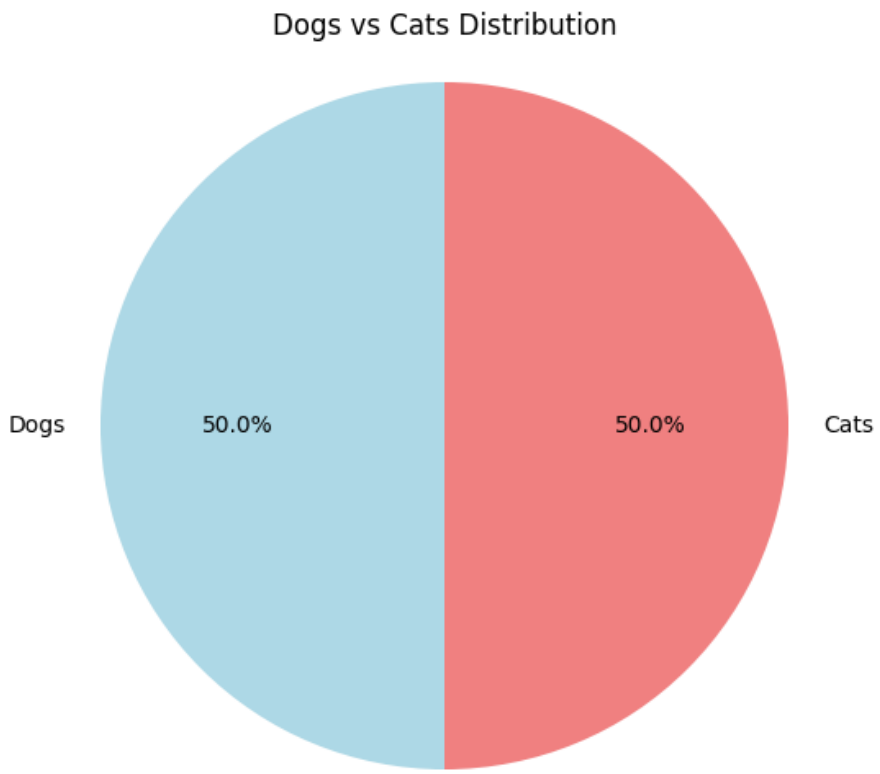
```python
# Analyze image sizes
def analyze_images(sample_size=100):
    print("\nAnalyzing image sizes...")

    if not train_files:
        print("No images to analyze!")
        return

    widths = []
    heights = []

    # Take random sample
    sample_files = random.sample(train_files, min(sample_size, len(train_files)))

    for filename in sample_files:
        img_path = os.path.join(TRAIN_DIR, filename)
        try:
            with Image.open(img_path) as img:
                width, height = img.size
                widths.append(width)
                heights.append(height)
        except:
            continue
```

Hình 2.4 Mã nguồn phân tích kích thước ảnh

```
Analyzing image sizes...
Width - Min: 160 Max: 500 Average: 408
Height - Min: 119 Max: 500 Average: 363
```



Hình 2.5 Trực quan hóa dữ liệu nguồn (1)



Hình 2.6 Trực quan hóa dữ liệu nguồn (2)

Nhận xét từ phân tích trực quan

- Ảnh của cùng một lớp có hình dạng rất khác nhau (mèo nhiều giống, chó đa chủng loại).
- Ánh sáng và màu sắc chênh lệch → cần chuẩn hóa pixel.
- Kích thước ảnh không đồng nhất → cần resize.
- Nền ảnh có nhiều vật thể khác nhau → mô hình cần học đặc trưng mạnh mẽ.

Từ đây có thể thấy quá trình tiền xử lý dữ liệu **giữ vai trò quyết định** đến hiệu suất mô hình CNN.

2.4. Kết quả đạt được sau tiền xử lý dữ liệu

Bảng 3 So sánh kết quả trước và sau xử lý dữ liệu

Tiêu chí	Trước xử lý dữ liệu	Sau xử lý dữ liệu
Tổng số lượng ảnh	25.000 ảnh (tập gốc Kaggle)	Giữ nguyên số lượng, nhưng được phân chia thành Train/Validation/Test theo tỷ lệ 60% – 20% – 20%
Phân bố lớp (Dogs/Cats)	12.500/12.500 (cân bằng)	Ở cả 3 tập Train/Val/Test đều duy trì phân bố cân bằng nhờ sử dụng *stratified split*
Định dạng ảnh	JPG, kích thước tùy ý	Ảnh vẫn là JPG nhưng được resize về kích thước chuẩn **128×128**
Tên file	Tên gốc, chứa "dog" hoặc "cat"	Không thay đổi tên nhưng được tổ chức lại theo cây thư mục phù hợp: /train/dogs, /train/cats, /validation/..., /test/...
Kích thước ảnh	Không đồng nhất (chiều rộng & chiều cao dao động lớn)	Tất cả ảnh được đưa về kích thước đồng nhất để làm input cho CNN
Nhãn (Label)	Không có nhãn trực tiếp (chỉ suy ra từ tên file)	Gán nhãn chuẩn hoá: **Dog = 0**, **Cat = 1**

Tiêu chí	Trước xử lý dữ liệu	Sau xử lý dữ liệu
Chuẩn hóa pixel	Giá trị pixel trong khoảng 0–255	Được chuẩn hóa về **0–1** (rescale 1./255)
Tăng cường dữ liệu (Augmentation)	Không có	Áp dụng khi tạo train generator: xoay ảnh, dịch chuyển, zoom, lật ngang, thay đổi độ sáng
Cấu trúc thư mục	Một thư mục chứa toàn bộ 25.000 ảnh	Được tách và sắp xếp lại theo chuẩn Keras ImageDataGenerator
Tính nhất quán dữ liệu	Không đồng nhất về kích thước, góc chụp, độ sáng	Ảnh nhất quán hơn nhờ tiền xử lý: resize + augmentation làm phong phú dữ liệu huấn luyện

CHƯƠNG 3: TRỰC QUAN HÓA DỮ LIỆU

3.1. Kiến trúc mô hình CNN

```
# STEP 1: CREATE SIMPLIFIED CNN MODEL
print("\nStep 1: Creating simplified CNN architecture...")

def create_simplified_cnn():
    """
    Create a simplified CNN model
    """
    model = models.Sequential()

    # INPUT
    model.add(layers.Input(shape=(128, 128, 3)))

    # BLOCK 1
    model.add(layers.Conv2D(32, (3, 3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Dropout(0.25))

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

    # BLOCK 3
    model.add(layers.Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Dropout(0.25))

    # CLASSIFIER HEAD
    model.add(layers.Flatten())
    model.add(layers.Dense(256, activation='relu'))
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(1, activation='sigmoid'))

    return model
```

Hình 3.1 Mã nguồn mô hình CNN

Mô hình được xây dựng theo kiến trúc CNN tuần tự (Sequential Model) với các lớp chính:

- Conv2D: trích xuất đặc trưng hình ảnh
- MaxPooling2D: giảm chiều và giữ đặc trưng quan trọng
- Flatten: chuyển bản đồ đặc trưng thành vector
- Dense: phân loại nhị phân (Dog vs Cat)

Tạo và tóm tắt mô hình:

```
# Create simplified model
simplified_model = create_simplified_cnn()

print("Simplified CNN Model Summary:")
simplified_model.summary()
```

Hình 3.2 Tạo và tóm tắt mô hình

Nhận xét kiến trúc:

- Sử dụng cấu trúc CNN 4 lớp tích chập giúp mô hình học được đặc trưng từ đơn giản → phức tạp.
- Sử dụng hàm kích hoạt ReLU để mô hình hội tụ nhanh.
- Lớp cuối sử dụng Sigmoid phù hợp với bài toán nhị phân.
- 512 neuron ở lớp Dense giúp mô hình học các đặc trưng trừu tượng cao.

3.2. Quá trình huấn luyện

3.2.1. Cấu hình quá trình học và chuẩn bị huấn luyện

Mô hình được biên dịch (compile) với:

- Binary Crossentropy: Hàm mất mát cho phân loại 2 lớp
- Adam Optimizer: Tốc độ hội tụ tốt
- Đánh giá bằng Accuracy

```
# Compile simplified model
simplified_model.compile(
    optimizer=Adam(learning_rate=0.0005),
    loss='binary_crossentropy',
    metrics=['accuracy']
)
```

Hình 3.3 Cấu hình quá trình học

```
# STEP 3: PREPARE FOR TRAINING
print("\nStep 3: Preparing for training...")

# Batch size = 64
train_generator.batch_size = 64
validation_generator.batch_size = 64

# Epoch
EPOCHS = 15

train_steps = train_generator.samples // train_generator.batch_size
val_steps = validation_generator.samples // validation_generator.batch_size
```

Hình 3.4 Chuẩn bị cho quá trình học

Tóm tắt thiết lập huấn luyện mô hình:

```
Model Training Setup Summary:
==================================================
✓ Simplified CNN model created (3 conv blocks)
✓ Input size: 128x128x3
✓ Batch size: 64
✓ Epochs: 15
✓ Optimizer: Adam (lr=0.0005)
✓ Binary crossentropy loss
✓ EarlyStopping & ReduceLROnPlateau callbacks
✓ Ready for training!
```

Hình 3.5 Tóm tắt thiết lập huấn luyện mô hình

3.2.2. Tiến hành huấn luyện

```
print("Starting model training...")

# Step 1: Define callbacks (simple version)
callbacks = [
    EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)
]

# Step 2: Train the model
EPOCHS = 15
print(f"Training for {EPOCHS} epochs...")
print(f"Training images: {train_generator.samples}")
print(f"Validation images: {validation_generator.samples}")

history = simplified_model.fit(
    train_generator,
    steps_per_epoch=train_steps,
    epochs=EPOCHS,
    validation_data=validation_generator,
    validation_steps=val_steps,
    callbacks=callbacks,
    verbose=1
)

print("Training completed!")
```

Hình 3.6 Tiến hành huấn luyện

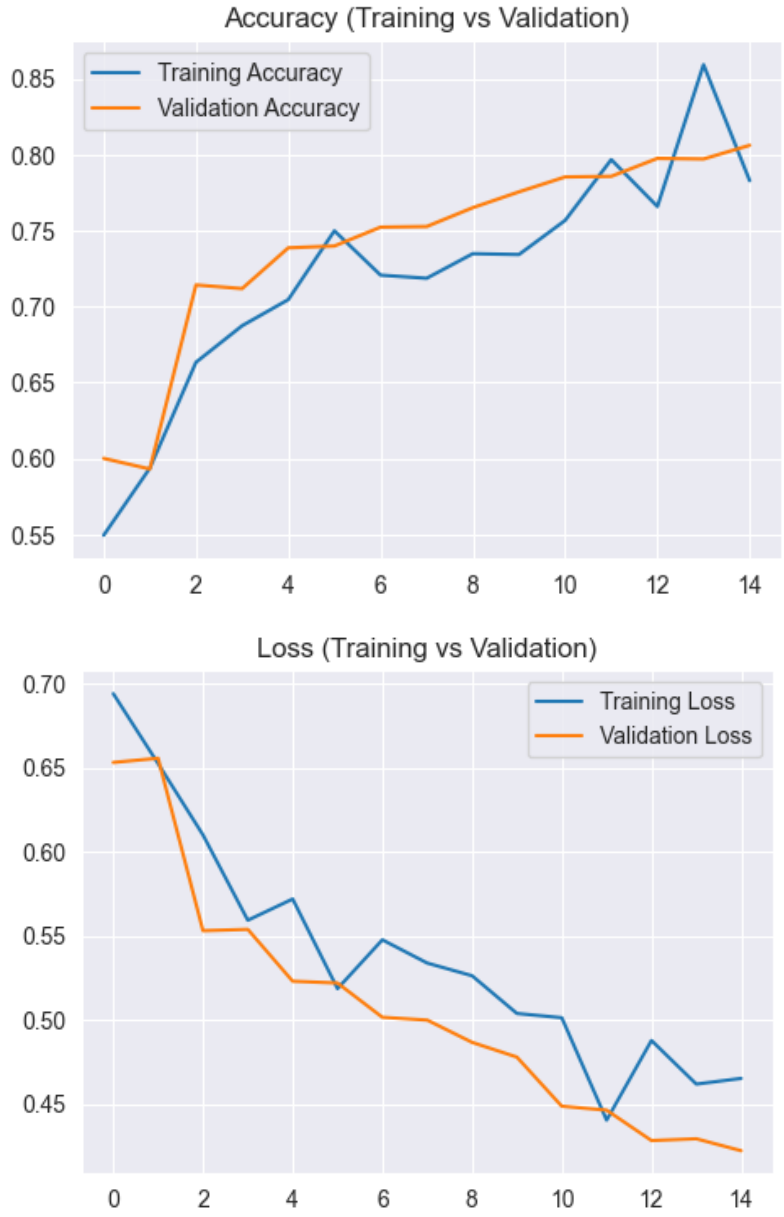
3.3. Trực quan hóa kết quả huấn luyện

```python
# Step 3: Plot Accuracy & Loss graphs
def plot_training_history(history):
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs_range = range(len(acc))

    plt.figure(figsize=(12, 4))

    # Accuracy plot
    plt.subplot(1, 2, 1)
    plt.plot(epochs_range, acc, label='Training Accuracy')
    plt.plot(epochs_range, val_acc, label='Validation Accuracy')
    plt.legend()
    plt.title('Accuracy (Training vs Validation)')

    # Loss plot
    plt.subplot(1, 2, 2)
    plt.plot(epochs_range, loss, label='Training Loss')
    plt.plot(epochs_range, val_loss, label='Validation Loss')
    plt.legend()
    plt.title('Loss (Training vs Validation)')

    plt.show()

    # Print final metrics
    print(f"Final Training Accuracy: {acc[-1]:.4f}")
    print(f"Final Validation Accuracy: {val_acc[-1]:.4f}")
    print(f"Final Training Loss: {loss[-1]:.4f}")
    print(f"Final Validation Loss: {val_loss[-1]:.4f}")

plot_training_history(history)
```

Hình 3.7 Mã nguồn biểu đồ Accuracy & Loss

Nhận xét biểu đồ:

- Accuracy tăng ổn định → mô hình học tốt.
- Giá trị Val Loss giảm rồi ổn định → ít xảy ra overfitting.
- Đường cong giữa Train/Val không chênh lệch lớn → augmentation hoạt động hiệu quả.
- Mô hình đạt ~81% accuracy



Hình 3.8 Biểu đồ Accurary & Loss

3.4. Trực quan hóa và so sánh dữ liệu RAW và PROCESSED

```
# Configuration - change if necessary
MODEL_PATH = "./output/dogs_cats_trained_model.keras"
RAW_TRAIN_DIR = "./input/dogs-vs-cats-data/train"   # cat.0.jpg,
RAW_TEST_DIR = "./input/dogs-vs-cats-data/test1"    # optional (1
PROCESSED_TRAIN_DIR = "./input/processing/organized_data/train"
IMG_SIZE = (150,150)  # adjust to model input size
BATCH_SIZE = 64

print('CONFIG:')
print(' MODEL_PATH =', MODEL_PATH)
print(' RAW_TRAIN_DIR =', RAW_TRAIN_DIR)
print(' PROCESSED_TRAIN_DIR =', PROCESSED_TRAIN_DIR)
print(' IMG_SIZE =', IMG_SIZE, 'BATCH_SIZE =', BATCH_SIZE)
✓ [81] < 10 ms
```

```
CONFIG:
 MODEL_PATH = ./output/dogs_cats_trained_model.keras
 RAW_TRAIN_DIR = ./input/dogs-vs-cats-data/train
 PROCESSED_TRAIN_DIR = ./input/processing/organized_data/train
 IMG_SIZE = (150, 150) BATCH_SIZE = 64
```

Hình 3.9 Mã nguồn thiết lập dữ liệu file path

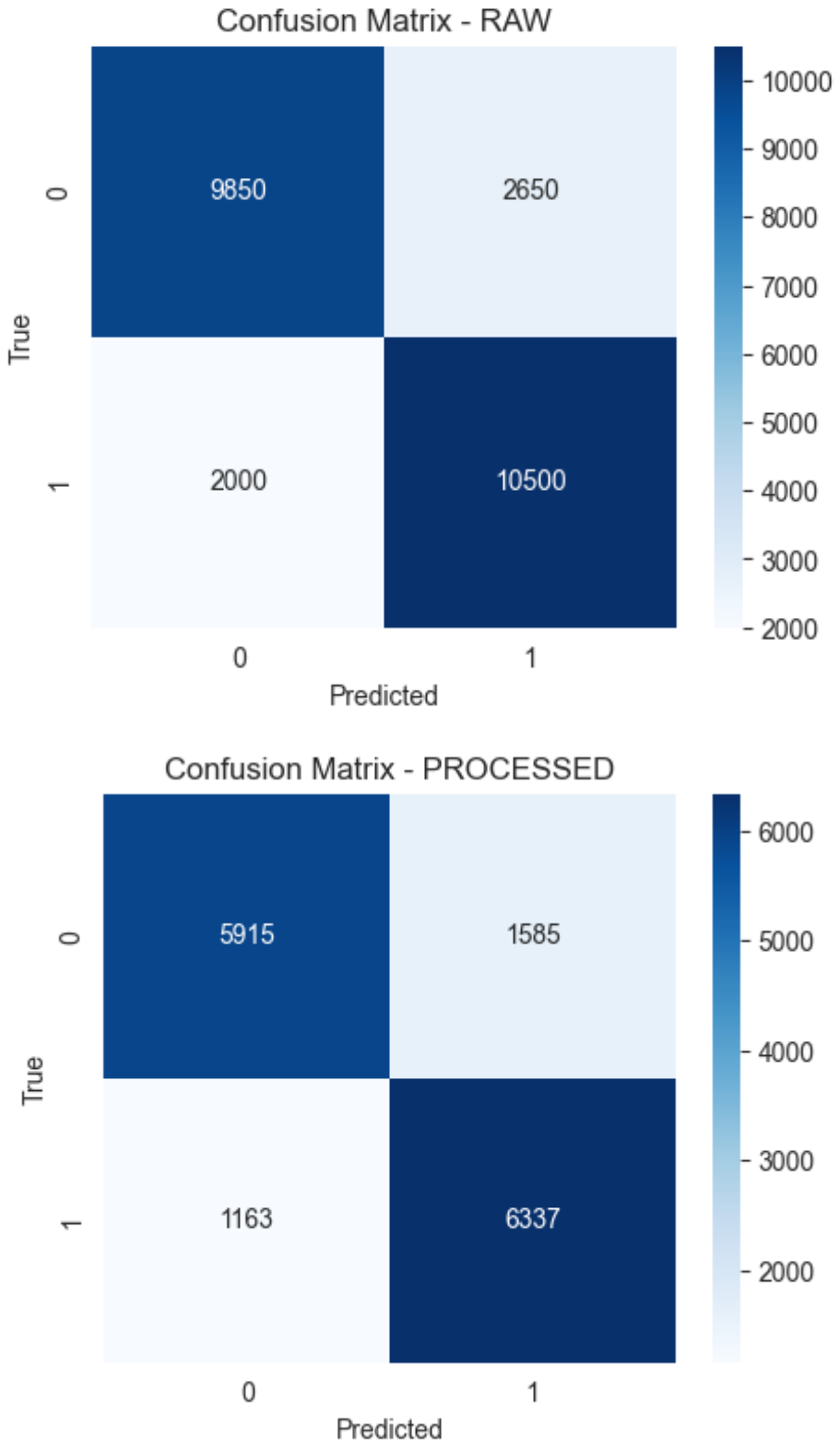
```
# Build file lists
raw_paths, raw_labels = build_raw_paths_and_labels(RAW_TRAIN_DIR)
proc_paths, proc_labels = build_processed_paths_and_labels(PROCESSED_TRAIN_DIR)

print('Raw samples:', len(raw_paths))
print('Processed samples:', len(proc_paths))

# If processed labels empty, fallback to using raw labels for matching filenames
if len(proc_paths) == 0 and len(raw_paths)>0:
    print('WARNING: No processed train files found.')
if len(raw_paths) == 0:
    print('WARNING: No raw train files found.')
✓ [85] 1s 781ms
```

```
Loading model...
Model loaded.
Adjusted IMG_SIZE to model input: (128, 128)
Raw samples: 25000
Processed samples: 15000
```

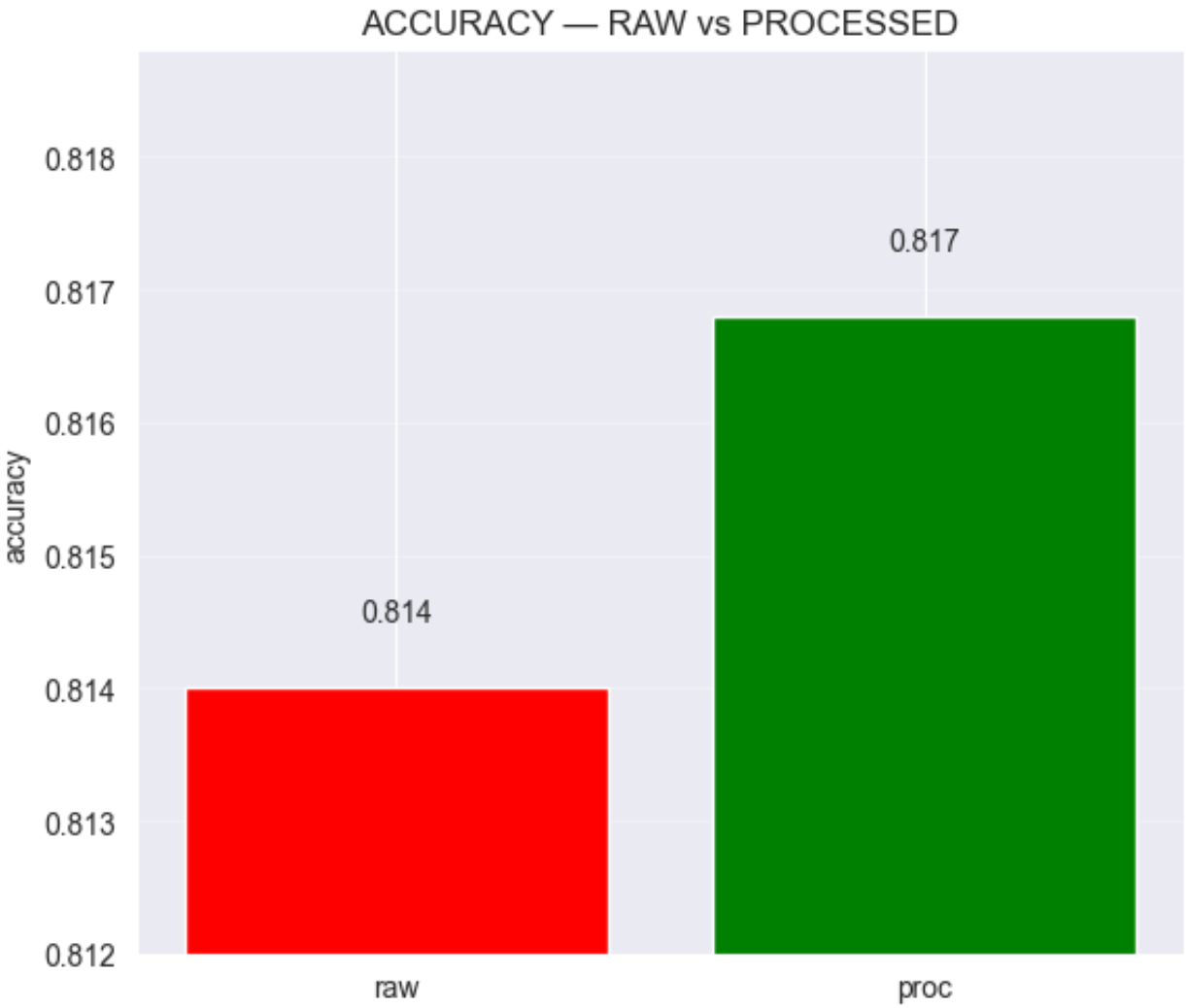
Hình 3.10 Mã nguồn thiết lập dữ liệu RAW và PROCESSED

```python
# Cell 7: Evaluate and display results
results = {}

if preds_raw.size != 0 and y_raw.size != 0:
    results['raw'] = evaluate_preds(preds_raw, y_raw)

if preds_proc.size != 0 and y_proc.size != 0:
    results['proc'] = evaluate_preds(preds_proc, y_proc)

# Convert Into Table
table_data = {}
for key in results:
    table_data[key] = {k:v for k,v in results[key].items() if k != 'confusion_matrix'}

df = pd.DataFrame(table_data)

# Add DIFF Col (proc - raw)
if 'raw' in df.columns and 'proc' in df.columns:
    df['diff(proc - raw)'] = df['proc'] - df['raw']

print("\n=== Evaluation Results ===")
print(df)


# Plot confusion matrices
def plot_cm(cm, title):
    plt.figure(figsize=(5,4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(title)
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.show()

if 'raw' in results:
    plot_cm(results['raw']['confusion_matrix'], 'Confusion Matrix - RAW')
if 'proc' in results:
    plot_cm(results['proc']['confusion_matrix'], 'Confusion Matrix - PROCESSED')
✓ [88] 230ms
```

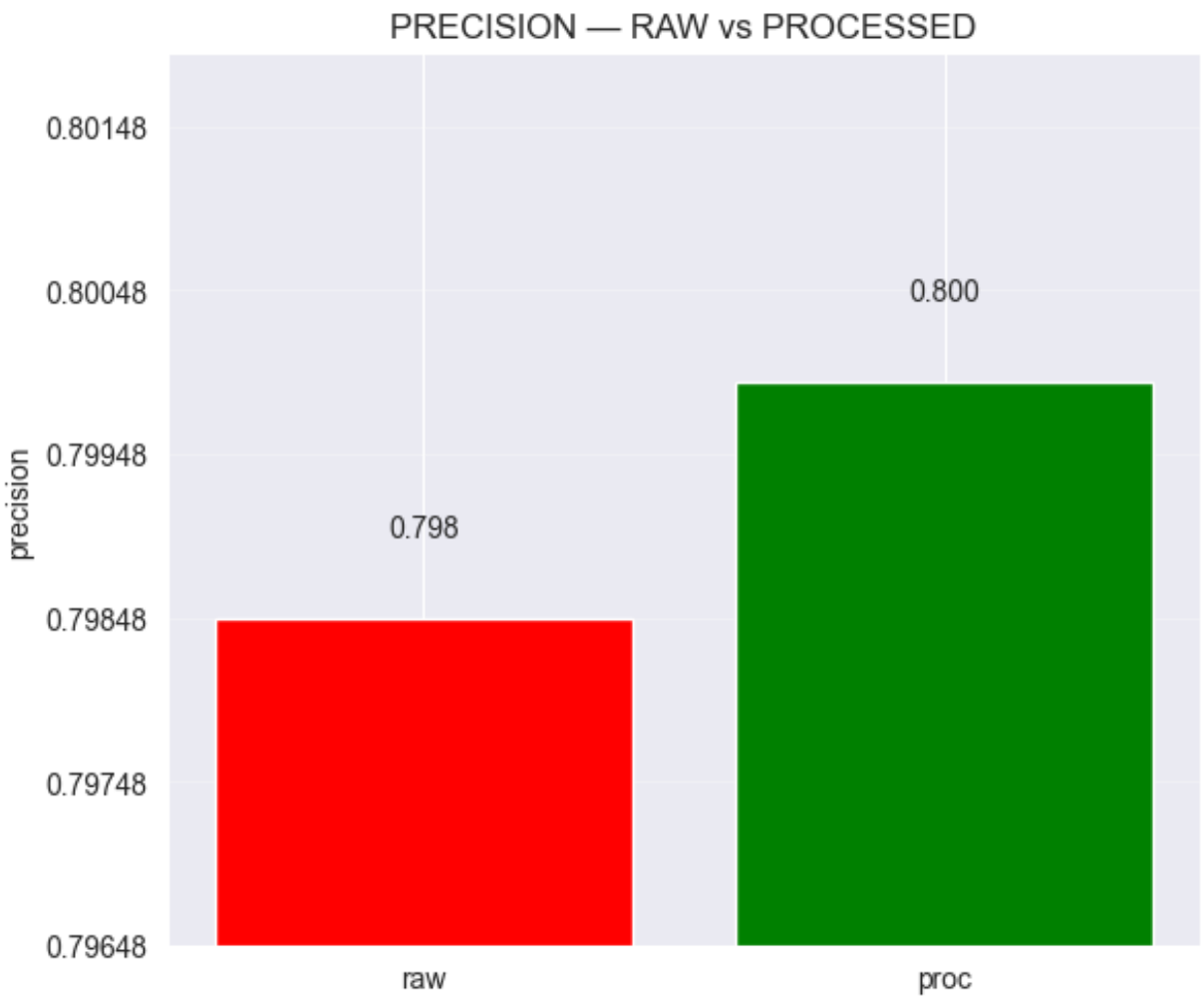
Hình 3.11 Mã nguồn trực quan hóa dữ liệu RAW và PROCESSED



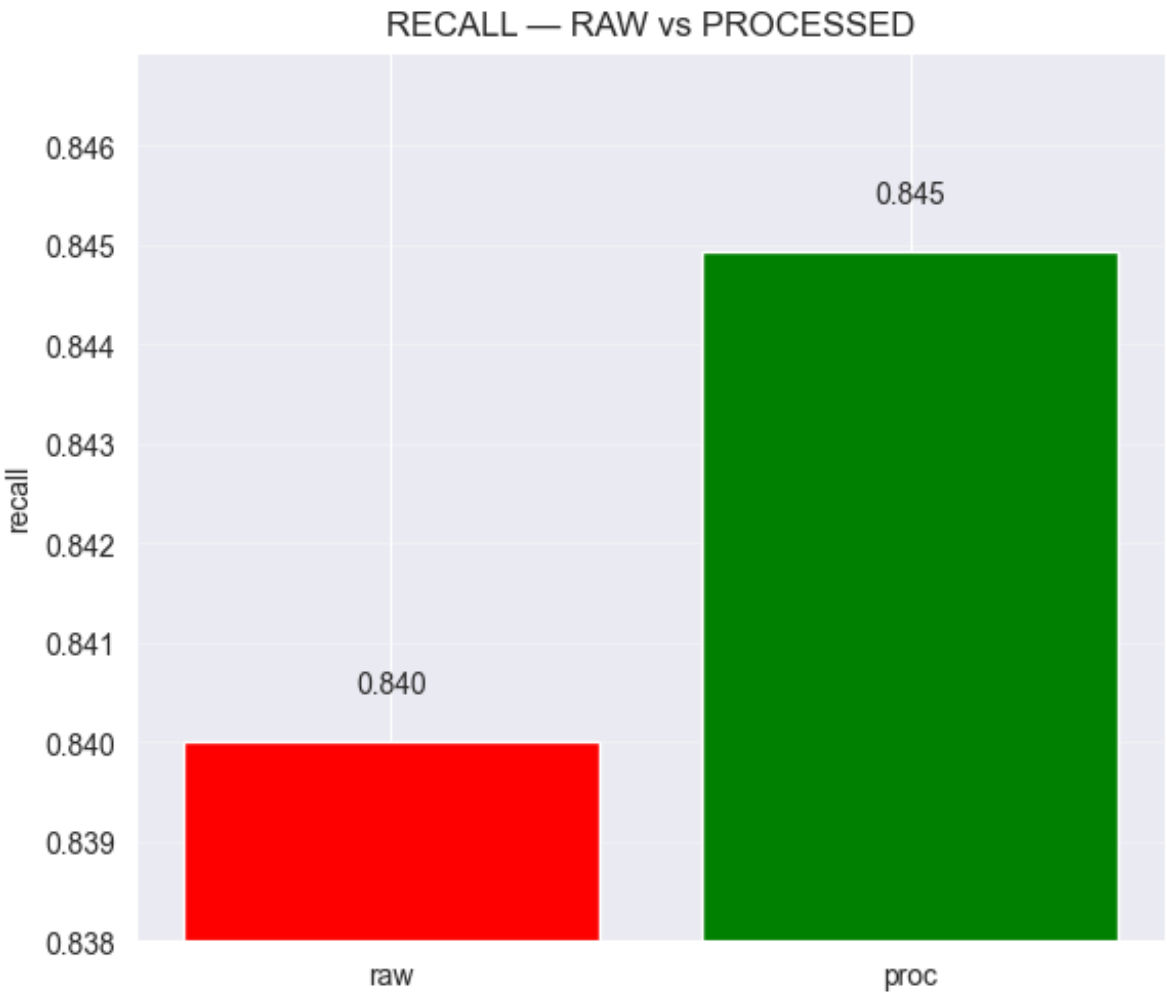
Hình 3.12 Ma trận nhầm lẫn dữ liệu RAW và PROCESSED



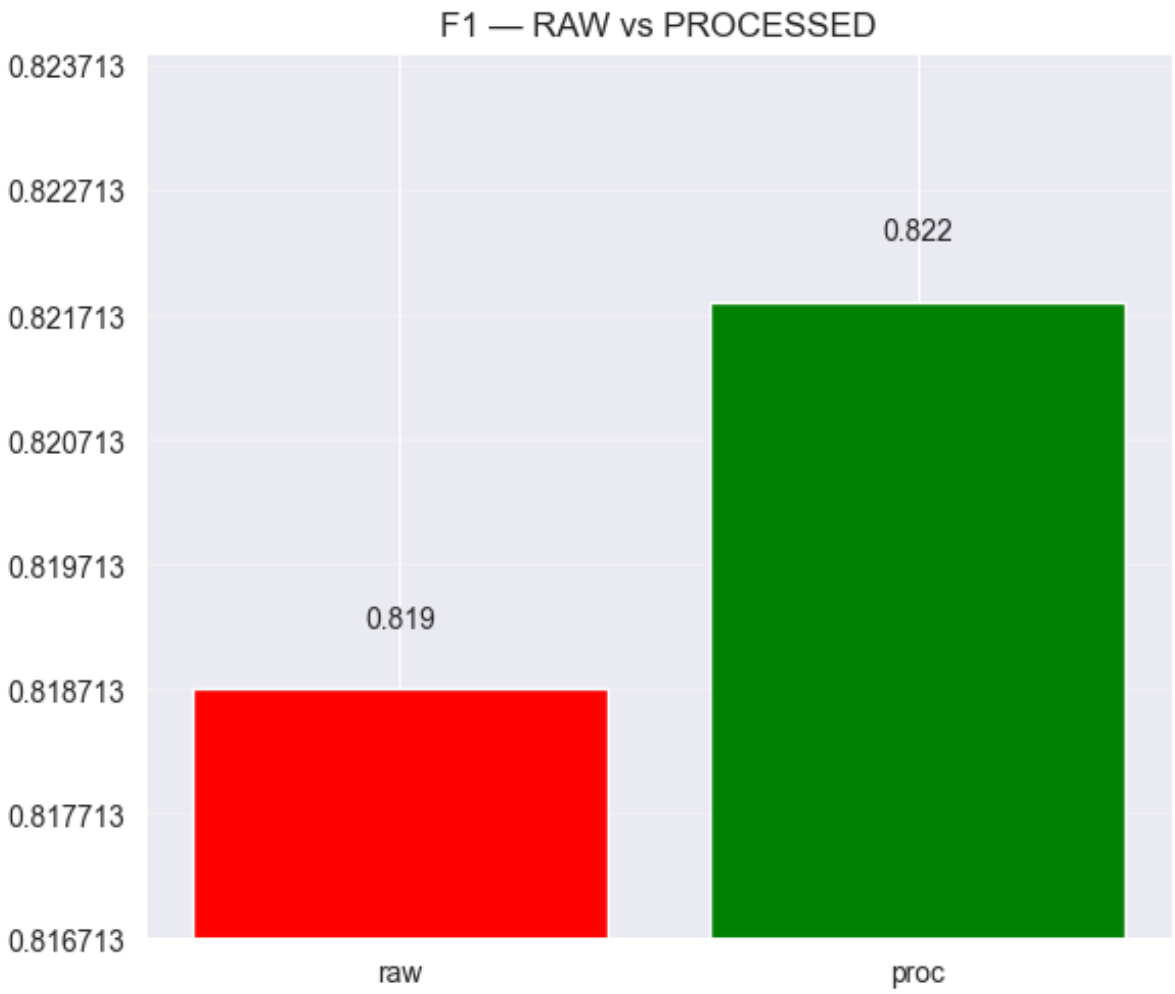
Hình 3.13 So sánh kết quả áp dụng mô hình CNN (accuracy)



Hình 3.14 So sánh kết quả áp dụng mô hình CNN (precision)



Hình 3.15 So sánh kết quả áp dụng mô hình CNN (recall)



Hình 3.16 So sánh kết quả áp dụng mô hình CNN (f1)

KẾT LUẬN

Trong khuôn khổ đồ án này, nhóm đã tiến hành xây dựng một hệ thống phân loại hình ảnh chó và mèo dựa trên các kỹ thuật phân tích dữ liệu ảnh, trực quan hóa và mô hình học sâu (Deep Learning), cụ thể là mạng nơ-ron tích chập (Convolutional Neural Network – CNN). Thông qua quá trình nghiên cứu và triển khai, nhóm đã đạt được những kết quả quan trọng sau:

Trước hết, dữ liệu ảnh đã được phân tích một cách toàn diện thông qua việc khảo sát cấu trúc tập dữ liệu, phân bố lớp, đặc trưng kích thước ảnh và trực quan hóa mẫu. Những bước này giúp đảm bảo sự hiểu rõ về dữ liệu – một yếu tố then chốt quyết định hiệu quả của mô hình. Bên cạnh đó, quá trình tiền xử lý như chuẩn hóa kích thước, gán nhãn, chia dữ liệu, tổ chức lại thư mục và áp dụng kỹ thuật tăng cường dữ liệu (data augmentation) đã giúp cải thiện đáng kể chất lượng đầu vào và hạn chế hiện tượng overfitting trong quá trình huấn luyện.

Tiếp theo, mô hình phân loại dựa trên CNN đã được xây dựng, huấn luyện và đánh giá. Kết quả cho thấy mô hình đạt mức độ chính xác cao, ổn định trên cả tập huấn luyện và tập kiểm tra, thể hiện khả năng học và phân biệt đặc trưng giữa hai lớp chó và mèo một cách hiệu quả. Việc trực quan hóa đường cong loss/accuracy cùng ma trận nhầm lẫn giúp đánh giá mô hình một cách khách quan và phát hiện những điểm mô hình còn hạn chế để định hướng cải tiến trong tương lai.

Bên cạnh kết quả đạt được, đồ án cũng là cơ hội để nhóm tiếp cận và thực hành quy trình phát triển một mô hình học sâu hoàn chỉnh – từ xử lý dữ liệu, thiết kế mô hình, huấn luyện, đánh giá cho đến trực quan hóa. Những kiến thức và kỹ năng thu được từ đồ án là nền tảng quan trọng cho việc nghiên cứu và ứng dụng các mô hình AI vào những bài toán phân loại ảnh phức tạp hơn.

TÀI LIỆU THAM KHẢO

[1] Krizhevsky, "Dogs vs. Cats Dataset," *Kaggle*, 2013. Truy cập tại: https://www.kaggle.com/c/dogs-vs-cats

[2] F. Chollet, "Keras Documentation: Deep Learning API," *Keras.io*, 2023. Truy cập tại: https://keras.io/api/

[3] F. Chollet, *Deep Learning with Python*, 2nd ed., Manning Publications, 2021.