Mục Lục

Danh mục bảng

Danh mục hình

Bài toán nhận dạng dáng người

Chương 1: Giới Thiệu

1.1 Đặt vấn đề

Phân loại tư thế con người như **đứng**, **ngồi**, **nằm** và **cúi người** là một bài toán quan trọng trong nhiều lĩnh vực như giám sát an ninh, chăm sóc người già, y tế, và tương tác người – máy. Trong các tình huống thực tế, việc hệ thống có thể tự động nhận diện tư thế con người giúp phát hiện sớm các hành vi bất thường (té ngã, bất động) hoặc hỗ trợ đưa ra phản hồi phù hợp trong các hệ thống thông minh.

Một trong những cách tiếp cận phổ biến hiện nay là sử dụng **ảnh hình bóng (silhouette)** – dạng ảnh chỉ giữ lại đường viền của cơ thể người, bỏ qua chi tiết về màu sắc hay khuôn mặt. Cách tiếp cận này giúp giảm thiểu sự phụ thuộc vào ánh sáng, trang phục hay đặc điểm cá nhân, đồng thời phù hợp trong các môi trường cần bảo vệ quyền riêng tư.

Tuy nhiên, việc phân loại tư thế từ hình bóng gặp nhiều thách thức do sự đa dạng về hình thể, góc nhìn và độ giống nhau giữa các tư thế (ví dụ: cúi người và ngồi có thể trông rất giống nhau khi nhìn từ phía trước). Do đó, cần xây dựng một mô hình học sâu đủ mạnh để rút trích đặc trưng hình dáng và phân loại chính xác các tư thế này.

Trong báo cáo này sẽ đề xuất bài toán phân loại tư thế con người từ ảnh hình bóng với 4 lớp: **đứng**, **ngồi**, **nằm**, và **cúi người**. Mục tiêu là xây dựng một mô hình có khả năng phân biệt chính xác các tư thế trên, hướng đến các ứng dụng thực tiễn trong giám sát và hỗ trợ thông minh.

1.2 Mục tiêu đề tài

Mục tiêu chính của đề tài này là xây dựng và đánh giá các mô hình CNN để tự động phân loại bốn tư thế con người: Đứng, ngồi, nằm và cúi người. Cụ thể, đề tài sẽ tập trung vào:

- Thu thập và chuẩn bị tập dữ liệu hình ảnh (**silhouette**) đa dạng cho bốn tư thế là đứng, ngồi, nằm và cúi người.
- Xây dựng và huấn luyện nhiều kiến trúc mạng CNN khác nhau.
- So sánh hiệu suất (độ chính xác) của các kiến trúc CNN đã thực nghiệm trên tập dữ liệu ảnh test.

1.3 Đối tượng và phạm vi nghiên cứu

- Đối tượng nghiên cứu: Các kiến trúc và nguyên lý hoạt động của Mạng Nơ-ron Tích chập (CNN) trong bài toán phân loại hình ảnh.

- Phạm vi nghiên cứu: Đề tài chỉ giới hạn trong việc phân loại bốn tư thế của con người: Đứng, ngồi, nằm và cúi người dựa trên dữ liệu hình ảnh **silhouette**. Việc nghiên cứu không đi sâu vào các khía cạnh khác như phân tích chi tiết hình dáng hoặc đặc điểm nhận dạng cá nhân.

1.4 Bố cục đề tài

Báo cáo này được cấu trúc thành 4 chương chính như sau:

Chương 1: Giới thiệu - Trình bày tổng quan về đề tài, mục tiêu và phạm vi nghiên cứu.

Chương 2: Phương pháp và Dữ liệu - Mô tả quy trình thu thập, tiền xử lý dữ liệu và cách thiết kế, xây dựng mô hình CNN.

Chương 3: Kết quả và Thực nghiệm - Trình bày kết quả thực nghiệm, so sánh hiệu suất các kiến trúc và thảo luận những phát hiện quan trọng.

Chương 4: Kết luận và Hướng phát triển - Tóm tắt các kết quả đạt được và đề xuất các hướng phát triển tiếp theo của đề tài.

Chương 2 Phương pháp và Dữ liệu

2.1 Quy trình thu thập và tiền xử lý dữ liệu

Để huấn luyện các mô hình CNN, một tập dữ liệu ảnh chất lượng cao là yếu tố then chốt. Quy trình chuẩn bị dữ liệu của đề tài được thực hiện qua các bước sau:

2.1.1 Thu thập dữ liệu

- Nguồn dữ liệu: Dữ liệu hình ảnh dùng để phân loại bốn tư thế của con người được thu thập chủ yếu từ nền tảng Kaggle, một kho dữ liệu mở phổ biến trong cộng đồng Machine Learning.
- Số lượng ảnh: Sau quá trình thu thập, mỗi lớp (đứng, nằm, ngồi và cúi người) được đảm bảo có từ 1200 ảnh, đạt yêu cầu về số lượng tối thiểu cho huấn luyện mô hình học sâu.

- Định dạng: Tất cả các ảnh được thu thập đều là ảnh màu (RGB).

2.1.2 Phương pháp chia dữ liệu

2.1.2.1. 80% Training, 10% Validation, 10% Testing

Phương pháp này là một cách chia dữ liệu phổ biến và được khuyến nghị hơn trong hầu hết các trường hợp phát triển mô hình học máy.

- Ưu điểm:
    - Giúp tinh chỉnh siêu tham số hiệu quả: Tập validation cho phép thử nghiệm các cấu hình khác nhau của mô hình (ví dụ: tốc độ học, số lượng lớp, số lượng neuron) và chọn ra cấu hình tốt nhất mà không làm "ô nhiễm" tập kiểm thử cuối cùng.
    - Phát hiện và giảm thiểu quá khớp (Overfitting): Bằng cách theo dõi hiệu suất của mô hình trên tập validation trong quá trình huấn luyện, có thể nhận biết sớm khi mô hình bắt đầu quá khớp (tức là học quá kỹ dữ liệu huấn luyện và mất khả năng tổng quát hóa). Khi đó, có thể dừng quá trình huấn luyện hoặc áp dụng các kỹ thuật điều hòa (regularization).
    - Đánh giá khách quan hơn: Tập testing hoàn toàn độc lập và chỉ được sử dụng một lần để đánh giá cuối cùng, đảm bảo kết quả đánh giá là khách quan nhất.
- Nhược điểm:
    - Cần một lượng dữ liệu đủ lớn để chia thành ba tập mà vẫn đảm bảo mỗi tập có đủ mẫu đại diện.

2.1.2.2. 80% Training, 20% Testing

Phương pháp này đơn giản hơn nhưng có một số hạn chế.

- Ưu điểm:
    - Đơn giản để thực hiện.

    - Thích hợp khi có một lượng dữ liệu hạn chế và việc tạo thêm tập validation sẽ làm giảm đáng kể kích thước của tập huấn luyện hoặc tập kiểm thử.

- Nhược điểm:

    - Khó khăn trong việc tinh chỉnh siêu tham số: Nếu không có tập validation, sẽ phải sử dụng tập testing để tinh chỉnh siêu tham số. Điều này sẽ làm cho tập testing không còn "mới lạ" với mô hình nữa, dẫn đến việc đánh giá hiệu suất cuối cùng không còn hoàn toàn khách quan.

    - Nguy cơ quá khớp cao hơn: Việc không có tập validation để theo dõi hiệu suất trong quá trình huấn luyện có thể khiến khó phát hiện và ngăn chặn hiện tượng quá khớp.

    - Đánh giá không khách quan: Việc lặp đi lặp lại việc kiểm thử trên cùng một tập kiểm thử để tinh chỉnh mô hình có thể dẫn đến việc mô hình của chỉ hoạt động tốt trên tập kiểm thử đó, nhưng lại kém trên dữ liệu mới thực tế.

2.2 Cấu hình tải dữ liệu với TensorFlow

Dữ liệu được tải và chuẩn bị cho mô hình bằng thư viện TensorFlow và Keras image_dataset_from_directory. Các tham số cấu hình như sau:

- · Kích thước ảnh: img_height = 256, img_width = 256.
- · Kích thước lô (Batch Size): batch_size theo độ tối ưu của kiến trúc.
- · Chế độ màu: color_mode='rgb' để đọc ảnh màu.
- · Chế độ nhãn: label_mode='int' để nhãn được mã hóa dưới dạng số nguyên.
- · Kỹ thuật nội suy: interpolation='bilinear' được sử dụng khi thay đổi kích thước ảnh.
- · Số lượng lớp: num_classes = 4 (tương ứng với đứng, ngồi, nằm và cúi người).
- · Tên các lớp: class_names được tự động nhận diện từ cấu trúc thư mục dữ liệu ([‘standing;,’sitting’, ‘lying’, ‘bending’] hoặc tương tự tùy theo thứ tự alphabet).

Các tập dữ liệu train_ds, val_ds, test_ds sau khi được tải đã được tối ưu hóa bằng cách sử dụng .cache(), .shuffle(), và .prefetch(buffer_size=tf.data.AUTOTUNE) để tăng tốc độ đọc và xử lý dữ liệu trong quá trình huấn luyện.

2.3 Thiết kế và Xây dựng mô hình CNN

Trong đề tài này, một số kiến trúc CNN đã được thiết kế và thử nghiệm, bao gồm cả các mô hình tự xây dựng và các mô hình tiền huấn luyện (Pre-trained Models) để so sánh hiệu suất. Tất cả các mô hình đều có lớp đầu vào Rescaling(1./255) để chuẩn hóa giá trị pixel về khoảng [0, 1].

2.3.1 Các mô hình CNN tự xây dựng

Ba mô hình CNN đơn giản (cnn_1, cnn_2, cnn_3) đã được thiết kế với các biến thể về số lượng lớp tích chập, số lượng bộ lọc, và tỷ lệ Dropout để tìm ra cấu hình tối ưu. Các mô hình này đều sử dụng các lớp cơ bản của CNN:

- Conv2D (Lớp tích chập): Trích xuất các đặc trưng cục bộ từ ảnh.
- BatchNormalization: Chuẩn hóa đầu ra của lớp trước, giúp tăng tốc độ và ổn định quá trình huấn luyện.
- MaxPooling2D (Lớp gộp cực đại): Giảm kích thước không gian của đặc trưng, giảm số lượng tham số.
- Dropout: Ngẫu nhiên vô hiệu hóa một số nơ-ron trong quá trình huấn luyện để chống overfitting.
- Flatten: Chuyển đổi ma trận đặc trưng thành vector một chiều.
- Dense (Lớp kết nối đầy đủ): Lớp đầu ra để phân loại. Lớp cuối cùng sử dụng hàm kích hoạt softmax để dự đoán xác suất cho mỗi lớp.
- Hàm mất mát: SparseCategoricalCrossentropy được sử dụng do nhãn được mã hóa dưới dạng số nguyên.
- Tối ưu hóa: Sử dụng bộ tối ưu Adam với learning rate cố định là 0.001.

Các mô hình được huấn luyện với epochs phù hợp với thiết kế của từng kiến trúc cnn và sử dụng các Callbacks EarlyStopping (ngừng huấn luyện sớm nếu val_loss không cải thiện sau 10 epoch) và ReduceLROnPlateau (giảm learning rate nếu val_loss không cải thiện sau 5 epoch) để tối ưu quá trình huấn luyện.

2.4 Các lớp ẩn trong mô hình

2.4.1 CNN - 1 (LPQ)

2.4.1.1 Kiến trúc mô hình CNN - 1

Việc xây dựng một mô hình CNN hiệu quả là một quá trình đòi hỏi sự thử nghiệm và tinh chỉnh liên tục. Để tìm ra kiến trúc tối ưu cho bài toán phân loại tư thế, quá trình này đã tiêu tốn **gần 9 giờ đồng hồ** để nghiên cứu và triển khai **hơn 10 biến thể kiến trúc CNN**. Mỗi kiến trúc đều được đánh giá kỹ lưỡng, từ việc điều chỉnh số lượng lớp tích chập, bộ lọc, kích thước pooling, cho đến việc áp dụng các kỹ thuật như Batch Normalization và Dropout ở các vị trí khác nhau. Chính nhờ quá trình tìm tòi và tối ưu hóa chuyên sâu này, cuối cùng đã có thể phát triển được mô hình như bến dưới, đạt được độ chính xác ấn tượng và khả năng tổng quát hóa mạnh mẽ.

Layer (type)	Output Shape	Param #
rescaling_3 (Rescaling)	(None, 256, 256, 1)	0
conv2d_36 (Conv2D)	(None, 256, 256, 32)	320
batch_normalization_9 (BatchNormalization)	(None, 256, 256, 32)	128
conv2d_37 (Conv2D)	(None, 256, 256, 32)	9248
batch_normalization_10 (BatchNormalization)	(None, 256, 256, 32)	128

max_pooling2d_18 (MaxPooling2D)	(None, 128, 128, 32)	0
dropout_21 (Dropout)	(None, 128, 128, 32)	0
conv2d_38 (Conv2D)	(None, 128, 128, 64)	18496
batch_normalization_11 (BatchNormalization)	(None, 128, 128, 64)	256
conv2d_39 (Conv2D)	(None, 128, 128, 64)	36928
batch_normalization_12 (BatchNormalization)	(None, 128, 128, 64)	256
max_pooling2d_19 (MaxPooling2D)	(None, 64, 64, 64)	0
dropout_22 (Dropout)	(None, 64, 64, 64)	0
conv2d_40 (Conv2D)	(None, 64, 64, 128)	73856
batch_normalization_13 (BatchNormalization)	(None, 64, 64, 128)	512

conv2d_41 (Conv2D)	(None, 64, 64, 128)	147584
batch_normalization_14 (BatchNormalization)	(None, 64, 64, 128)	512
max_pooling2d_20 (MaxPooling2D)	(None, 32, 32, 128)	0
dropout_23 (Dropout)	(None, 32, 32, 128)	0
conv2d_42 (Conv2D)	(None, 32, 32, 256)	295168
batch_normalization_15 (BatchNormalization)	(None, 32, 32, 256)	1024
conv2d_43 (Conv2D)	(None, 32, 32, 256)	590080
batch_normalization_16 (BatchNormalization)	(None, 32, 32, 256)	1024
max_pooling2d_21 (MaxPooling2D)	(None, 16, 16, 256)	0
dropout_24 (Dropout)	(None, 16, 16, 256)	0

global_average_pooling2d_4 (GlobalAveragePooling2D)	(None, 256)	0
dense_6 (Dense)	(None, 256)	65792
batch_normalization_17 (BatchNormalization)	(None, 256)	1024
dropout_25 (Dropout)	(None, 256)	0
dense_7 (Dense)	(None, 4)	1028

Bảng 1. Sơ đồ kiến trúc CNN - 1

```python
model = models.Sequential([
    layers.Rescaling(1./255, input_shape=(img_height, img_width, 1)),

    # --- BLOCK 1 ---
    layers.Conv2D(32, (3, 3), activation='relu', padding='same',
                  kernel_regularizer=regularizers.l2(0.0001)),
    layers.BatchNormalization(),
    layers.Conv2D(32, (3, 3), activation='relu', padding='same',
                  kernel_regularizer=regularizers.l2(0.0001)),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2, 2)),
    layers.Dropout(0.25),

    # --- BLOCK 2 ---
    layers.Conv2D(64, (3, 3), activation='relu', padding='same',
                  kernel_regularizer=regularizers.l2(0.0001)),
    layers.BatchNormalization(),
    layers.Conv2D(64, (3, 3), activation='relu', padding='same',
                  kernel_regularizer=regularizers.l2(0.0001)),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2, 2)),
    layers.Dropout(0.25),

    # --- BLOCK 3 ---
    layers.Conv2D(128, (3, 3), activation='relu', padding='same',
                  kernel_regularizer=regularizers.l2(0.0001)),
    layers.BatchNormalization(),
    layers.Conv2D(128, (3, 3), activation='relu', padding='same',
                  kernel_regularizer=regularizers.l2(0.0001)),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2, 2)),
    layers.Dropout(0.3),

    # --- BLOCK 4 ---
    layers.Conv2D(256, (3, 3), activation='relu', padding='same',
                  kernel_regularizer=regularizers.l2(0.0001)),
    layers.BatchNormalization(),
    layers.Conv2D(256, (3, 3), activation='relu', padding='same',
                  kernel_regularizer=regularizers.l2(0.0001)),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2, 2)),
    layers.Dropout(0.3),
    layers.GlobalAveragePooling2D(),
    layers.Dense(256, activation='relu',
                 kernel_regularizer=regularizers.l2(0.0001)),
    layers.BatchNormalization(),
    layers.Dropout(0.5),
    layers.Dense(num_classes, activation='softmax')
])
```

Hình 1. Code python kiến trúc CNN - 1

```
Total params: 1,243,364 (4.74 MB)
Trainable params: 1,240,932 (4.73 MB)
Non-trainable params: 2,432 (9.50 KB)
```

Hình 2. Tổng tham số có trong CNN 1

2.4.1.2 Giải thích mô hình CNN - 1

- **rescaling_2 (Rescaling)**:
    - **Chức năng**: Lớp này được sử dụng để chuẩn hóa các giá trị pixel của hình ảnh đầu vào. Thông thường, các giá trị pixel nằm trong khoảng từ 0 đến 255. Lớp Rescaling sẽ chia các giá trị này cho một hệ số (ví dụ: 255.0) để đưa chúng về khoảng từ 0 đến 1.
    - **Ý nghĩa**: Chuẩn hóa dữ liệu đầu vào là một bước quan trọng trong học máy, đặc biệt là trong mạng nơ-ron. Nó giúp các thuật toán học hiệu quả hơn bằng cách giữ cho các giá trị trong một phạm vi nhỏ hơn và tránh các vấn đề liên quan đến sự chênh lệch lớn về độ lớn của dữ liệu.
    - **Output Shape**: (None, 256, 256, 1). Điều này có nghĩa là đầu ra của lớp này là hình ảnh có kích thước 256x256 pixel và 1 kênh (có thể là ảnh xám). None cho biết kích thước lô (batch size) có thể linh hoạt.
    - **Param #**: 0. Không có tham số nào được học trong lớp này vì nó chỉ thực hiện một phép biến đổi toán học đơn giản.
- **conv2d_28 (Conv2D)**:
    - **Chức năng**: Đây là một lớp tích chập (Convolutional Layer). Nó áp dụng một tập hợp các bộ lọc (kernels) lên hình ảnh đầu vào để trích xuất các đặc trưng. Mỗi bộ lọc sẽ quét qua hình ảnh và tính toán tích chập, tạo ra một bản đồ đặc trưng (feature map).
    - **Ý nghĩa**: Các lớp tích chập là cốt lõi của CNN. Chúng giúp mô hình học các mẫu và đặc trưng cục bộ trong hình ảnh, chẳng hạn như các cạnh, góc, hoặc kết cấu.
    - **Output Shape**: (None, 256, 256, 32). Điều này có nghĩa là có 32 bản đồ đặc trưng được tạo ra, mỗi bản đồ có kích thước 256x256. Số 32 là số lượng bộ lọc được sử dụng.
    - **Param #**: 320. Số tham số này bao gồm các trọng số của 32 bộ lọc (mỗi bộ lọc có kích thước nhất định, ví dụ 3x3 hoặc 5x5) và các bias tương ứng.
- **batch_normalization (BatchNormalization)**:
    - **Chức năng**: Lớp này chuẩn hóa các đầu ra của lớp trước đó bằng cách điều chỉnh và mở rộng kích thước dữ liệu để có giá trị trung bình bằng 0 và độ lệch chuẩn bằng 1.

    - **Ý nghĩa**: Batch Normalization giúp ổn định quá trình huấn luyện mạng, cho phép sử dụng tốc độ học (learning rate) cao hơn và làm cho quá trình hội tụ nhanh hơn. Nó cũng giúp giảm thiểu vấn đề covariate shift nội bộ.
    - **Output Shape**: (None, 256, 256, 32). Kích thước không thay đổi.
    - **Param #**: 128. Các tham số này bao gồm các giá trị trung bình, phương sai, tham số scale và shift được học trong quá trình huấn luyện.
- **conv2d_29 (Conv2D)**:
    - **Chức năng**: Một lớp tích chập khác, tiếp tục trích xuất các đặc trưng phức tạp hơn từ các bản đồ đặc trưng của lớp trước đó.
    - **Ý nghĩa**: Việc xếp chồng các lớp tích chập cho phép mô hình học các đặc trưng ngày càng phức tạp và trừu tượng hơn.
    - **Output Shape**: (None, 256, 256, 32). Kích thước hình ảnh vẫn giữ nguyên, số lượng bộ lọc là 32.
    - **Param #**: 9,248. Nhiều hơn lớp conv2d_28 vì nó có thể đang xử lý nhiều kênh đầu vào hơn (từ 32 kênh của lớp trước).
- **batch_normalization_1 (BatchNormalization)**: Tương tự như batch_normalization.
- **max_pooling2d_14 (MaxPooling2D)**:
    - **Chức năng**: Lớp này thực hiện thao tác gộp tối đa (max pooling). Nó chia đầu vào thành các "cửa sổ" nhỏ và lấy giá trị lớn nhất từ mỗi cửa sổ.
    - **Ý nghĩa**: Max Pooling giúp giảm chiều không gian của các bản đồ đặc trưng, từ đó giảm số lượng tham số và tính toán trong mạng. Nó cũng giúp mô hình trở nên bất biến với các biến đổi nhỏ trong đầu vào (ví dụ: dịch chuyển).
    - **Output Shape**: (None, 128, 128, 32). Kích thước chiều rộng và chiều cao đã giảm đi một nửa (từ 256x256 xuống 128x128).
    - **Param #**: 0. Không có tham số nào được học.
- **dropout_16 (Dropout)**:
    - **Chức năng**: Trong quá trình huấn luyện, lớp Dropout ngẫu nhiên "tắt" (đặt về 0) một tỷ lệ nhất định các nơ-ron đầu ra của lớp trước đó.
    - **Ý nghĩa**: Dropout là một kỹ thuật điều hòa (regularization) giúp ngăn chặn mô hình học thuộc lòng dữ liệu huấn luyện (overfitting). Bằng cách loại bỏ ngẫu nhiên các nơ-ron, mô hình buộc phải học các đặc trưng mạnh mẽ hơn và ít phụ thuộc vào bất kỳ nơ-ron cụ thể nào.
    - **Output Shape**: (None, 128, 128, 32). Kích thước không thay đổi, chỉ một số giá trị bị đặt về 0.
    - **Param #**: 0. Không có tham số nào được học.

Các khối tiếp theo (conv2d_30 đến dropout_19) lặp lại một mẫu tương tự:

- **Conv2D**: Tiếp tục học các đặc trưng. Số lượng bộ lọc (và do đó số lượng bản đồ đặc trưng) thường tăng lên ở các lớp sâu hơn (32 -> 64 -> 128 -> 256), cho phép mô hình học các đặc trưng phức tạp hơn.
- **BatchNormalization**: Chuẩn hóa đầu ra để ổn định quá trình huấn luyện.
- **MaxPooling2D**: Giảm chiều không gian (kích thước hình ảnh) và giảm tham số. Kích thước hình ảnh giảm dần từ 128x128 xuống 16x16.
- **Dropout**: Ngăn ngừa overfitting.

Các lớp cuối cùng:

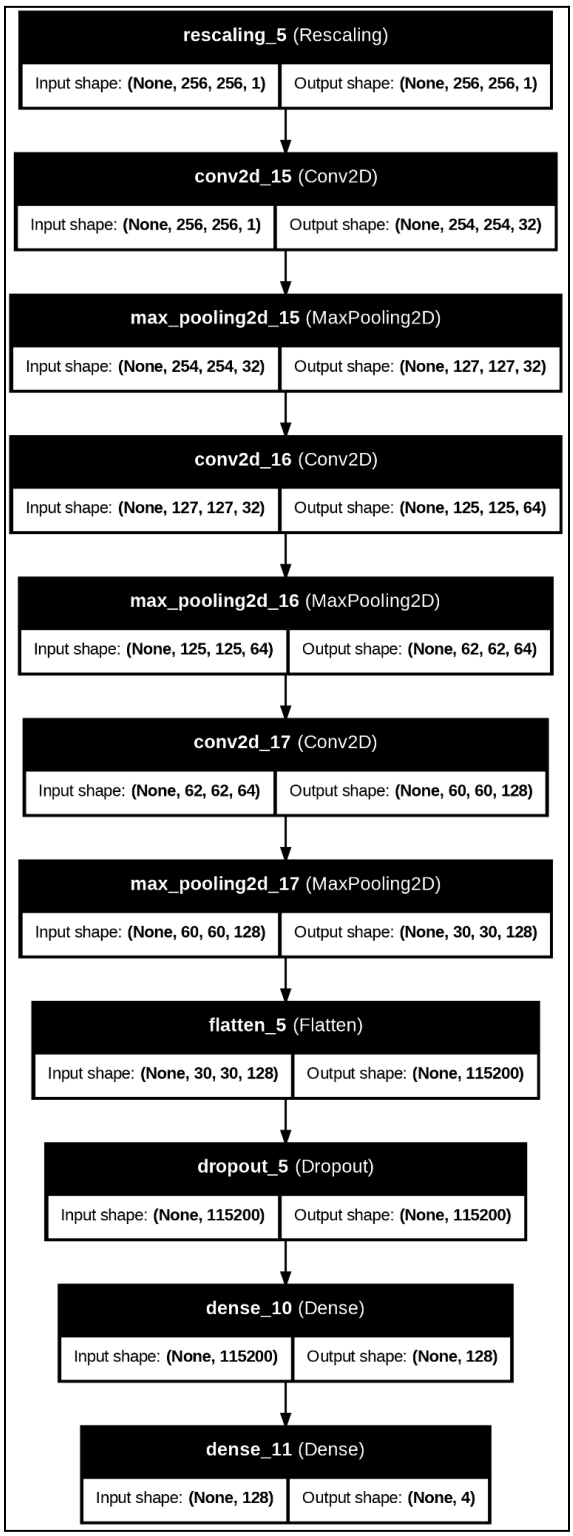
- **global_average_pooling2d_3 (GlobalAveragePooling2D)**:
    - **Chức năng**: Lớp này tính giá trị trung bình của từng bản đồ đặc trưng trên toàn bộ chiều rộng và chiều cao.
    - **Ý nghĩa**: Nó chuyển đổi đầu ra 3D của các lớp tích chập trước đó thành một vector 1D. Điều này giúp giảm đáng kể số lượng tham số so với việc làm phẳng (flatten) toàn bộ bản đồ đặc trưng và thường hoạt động tốt hơn trong việc giảm overfitting.
    - **Output Shape**: (None, 256). Đầu ra là một vector có 256 phần tử, tương ứng với số lượng bản đồ đặc trưng cuối cùng (256).
    - **Param #**: 0. Không có tham số nào được học.
- **dense_4 (Dense)**:
    - **Chức năng**: Đây là một lớp kết nối đầy đủ (Fully Connected Layer). Mỗi nơ-ron trong lớp này được kết nối với mọi nơ-ron của lớp trước đó.
    - **Ý nghĩa**: Sau khi các đặc trưng đã được trích xuất bởi các lớp tích chập, các lớp Dense sẽ sử dụng các đặc trưng này để thực hiện phân loại.
    - **Output Shape**: (None, 256). Có 256 nơ-ron trong lớp này.
    - **Param #**: 65,792. Số tham số lớn vì mỗi nơ-ron trong lớp này kết nối với 256 phần tử từ lớp trước đó.
- **batch_normalization_8 (BatchNormalization)**: Tương tự như các lớp Batch Normalization trước đó, chuẩn hóa đầu ra của lớp Dense.

- **dropout_20 (Dropout)**: Tương tự như các lớp Dropout trước đó, ngăn ngừa overfitting trên lớp Dense.

- **dense_5 (Dense)**:
    - **Chức năng**: Đây là lớp đầu ra cuối cùng của mô hình, cũng là một lớp kết nối đầy đủ.
    - **Ý nghĩa**: Số lượng nơ-ron trong lớp này bằng số lượng lớp muốn phân loại. Trong trường hợp này, 4 nơ-ron cho thấy mô hình đang phân loại thành 4 lớp khác nhau (bending, lying, sitting, standing). Các giá trị đầu ra từ lớp này (sau khi áp dụng hàm kích hoạt thích hợp như softmax) sẽ là xác suất cho mỗi lớp.
    - **Output Shape**: (None, 4). Đầu ra là một vector có 4 phần tử.
    - **Param #**: 1,028. Mỗi trong số 4 nơ-ron này kết nối với 256 nơ-ron từ lớp trước.

2.4.2 CNN - 2 (VDT)

2.4.2.1 Kiến trúc mô hình CNN - 2



Hình 3. Sơ đồ kiến trúc CNN - 2

Mạng nơ-ron tích chập (CNN) được xây dựng với mục tiêu phân loại ảnh xám kích thước 256×256 thành 4 lớp. Kiến trúc bắt đầu bằng lớp **Rescaling**, có nhiệm vụ chuẩn hóa giá trị điểm ảnh từ [0, 255] về [0, 1] nhằm giúp quá trình huấn luyện ổn định hơn. Tiếp theo là ba khối gồm một lớp **Conv2D** và một lớp **MaxPooling2D** xen kẽ. Cụ thể, lớp Conv2D đầu tiên sử dụng 32 bộ lọc kích thước 3×3 để trích xuất các đặc trưng cục bộ ban đầu từ ảnh. Sau đó, lớp MaxPooling2D với kích thước 2×2 được áp dụng để giảm chiều không gian, qua đó làm giảm số lượng tham số và giúp mô hình tránh overfitting.

Khối thứ hai tiếp tục với lớp Conv2D sử dụng 64 bộ lọc, theo sau bởi một lớp pooling tương tự để tiếp tục cô đọng thông tin. Khối thứ ba tăng số lượng bộ lọc lên 128, nhằm học được các đặc trưng trừu tượng hơn từ dữ liệu, sau đó tiếp tục giảm kích thước không gian với một lớp pooling cuối cùng. Đầu ra của các tầng tích chập được chuyển thành vector 1 chiều bằng lớp **Flatten**, với kích thước rất lớn (115200 chiều), sau đó được đưa qua một lớp **Dropout** nhằm ngăn hiện tượng overfitting bằng cách ngẫu nhiên vô hiệu hóa một số nơ-ron trong quá trình huấn luyện.

Tiếp theo là hai lớp **Dense**, trong đó lớp đầu tiên gồm 128 nơ-ron đóng vai trò như một tầng ẩn học các mối liên hệ phi tuyến giữa các đặc trưng đã trích xuất, và lớp Dense cuối cùng gồm 4 nơ-ron tương ứng với 4 lớp phân loại. Lớp đầu ra này thường sử dụng hàm kích hoạt **softmax** để mô hình hóa xác suất dự đoán trên mỗi lớp.

2.4.2.2 Kiến trúc chi tiết mô hình CNN-2

Tên các lớp: ['bending', 'lying', 'sitting', 'standing']
Số lượng lớp: 4
Kích thước ảnh đầu vào cho mô hình: (256, 256, 1)
Model: "sequential_5"

Layer (type)	Output Shape	Param #
rescaling_5 (Rescaling)	(None, 256, 256, 1)	0
conv2d_15 (Conv2D)	(None, 254, 254, 32)	320
max_pooling2d_15 (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_16 (Conv2D)	(None, 125, 125, 64)	18,496
max_pooling2d_16 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_17 (Conv2D)	(None, 60, 60, 128)	73,856
max_pooling2d_17 (MaxPooling2D)	(None, 30, 30, 128)	0
flatten_5 (Flatten)	(None, 115200)	0
dropout_5 (Dropout)	(None, 115200)	0
dense_10 (Dense)	(None, 128)	14,745,728
dense_11 (Dense)	(None, 4)	516

Total params: 14,838,916 (56.61 MB)
Trainable params: 14,838,916 (56.61 MB)
Non-trainable params: 0 (0.00 B)

Hình 4. Kiến trúc CNN - 2

Mô hình gồm các tầng tích chập (Conv2D), pooling, dropout và dense được sắp xếp tuần tự, cụ thể như sau:

- **Lớp Rescaling**: Chuẩn hóa đầu vào về khoảng [0, 1], không có tham số huấn luyện.
- **Conv2D (32 filters)**: Trích xuất đặc trưng ban đầu, sử dụng 32 kernel 3×3, tạo ra 320 tham số.
- **MaxPooling2D**: Giảm kích thước ảnh còn một nửa, không có tham số.
- **Conv2D (64 filters)**: Học đặc trưng sâu hơn với 64 kernel, sinh ra 18,496 tham số.
- **MaxPooling2D**: Tiếp tục giảm kích thước không gian.

- **Conv2D (128 filters)**: Tăng cường học đặc trưng phức tạp hơn, sinh ra 73,856 tham số.
- **MaxPooling2D**: Giảm không gian ảnh còn (30, 30, 128).
- **Flatten**: Biến đầu ra thành vector 1 chiều với 115,200 phần tử.
- **Dropout**: Vô hiệu hóa ngẫu nhiên 50% số nơ-ron trong giai đoạn huấn luyện để giảm overfitting.
- **Dense (128 nơ-ron)**: Lớp kết nối đầy đủ, tạo ra 14,745,728 tham số.
- **Dense (4 nơ-ron)**: Lớp đầu ra, tương ứng với 4 lớp phân loại, với 516 tham số.

Tổng tham số của mô hình là 14,838,916 tham số.

2.4.3. Kiến trúc mô hình CNN - 3 (DTNY)

```python
# Xây dựng mô hình CNN
def build_cnn_model(input_shape, num_classes):
    model = models.Sequential([
        layers.Rescaling(1./255, input_shape=input_shape),

        layers.Conv2D(32, (3, 3), activation='relu', padding='same'),
        layers.BatchNormalization(),
        layers.Conv2D(32, (3, 3), activation='relu', padding='same'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),
        layers.Dropout(0.25),

        layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
        layers.BatchNormalization(),
        layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),
        layers.Dropout(0.3),

        layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
        layers.BatchNormalization(),
        layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),
        layers.Dropout(0.4),

        layers.Flatten(),
        layers.Dense(256, activation='relu'),
        layers.BatchNormalization(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation='softmax')
    ])
    return model
```

Hình 5. Code python kiến trúc CNN - 3

Layer (type)	Output Shape	Param #
rescaling (Rescaling)	(None, 256, 256, 1)	0
conv2d (Conv2D)	(None, 256, 256, 32)	320
batch_normalization (BatchNormalization)	(None, 256, 256, 32)	128
conv2d_1 (Conv2D)	(None, 256, 256, 32)	9,248
batch_normalization_1 (BatchNormalization)	(None, 256, 256, 32)	128
max_pooling2d (MaxPooling2D)	(None, 128, 128, 32)	0
dropout (Dropout)	(None, 128, 128, 32)	0
conv2d_2 (Conv2D)	(None, 128, 128, 64)	18,496
batch_normalization_2 (BatchNormalization)	(None, 128, 128, 64)	256
conv2d_3 (Conv2D)	(None, 128, 128, 64)	36,928
batch_normalization_3 (BatchNormalization)	(None, 128, 128, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 64)	0
dropout_1 (Dropout)	(None, 64, 64, 64)	0
conv2d_4 (Conv2D)	(None, 64, 64, 128)	73,856
batch_normalization_4 (BatchNormalization)	(None, 64, 64, 128)	512
conv2d_5 (Conv2D)	(None, 64, 64, 128)	147,584
batch_normalization_5 (BatchNormalization)	(None, 64, 64, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 32, 32, 128)	0
dropout_2 (Dropout)	(None, 32, 32, 128)	0
flatten (Flatten)	(None, 131072)	0
dense (Dense)	(None, 256)	33,554,688
batch_normalization_6 (BatchNormalization)	(None, 256)	1,024
dropout_3 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 4)	1,028

Hình 6. Sơ đồ kiến trúc CNN - 3

Mô hình CNN sử dụng kiến trúc Sequential trong Keras và được thiết kế với nhiều khối chức năng khác nhau để học đặc trưng cấp thấp đến cao.

2.4.3.1. Lớp Tiền xử lý: layers.Rescaling(1./255, input_shape=input_shape)

Công dụng: Chuẩn hóa giá trị pixel từ [0,255] về [0, 1] giúp:

- Giảm sai số khi tính toán gradient.
- Tăng tốc hội tụ.

2.4.3.2. Các Khối Tích chập (Convolutional Blocks)

2.4.3.2.1. Lớp Tích chập: layers.Conv2D(filters, (3, 3), activation='relu', padding='same')

Công dụng:

- Trích xuất đặc trưng ở nhiều cấp độ:
    + Filters: Tăng dần từ 32 => 64 => 128.
    + Kernel (3 x 3): Học đặc trưng cục bộ.
    + ReLu: Kích hoạt phi tuyến tính, giảm nguy cơ mất gradient.
    + Padding ‘same’: Duy trì kích thước đầu ra.

Tác động:

- Lớp đầu: Học cạnh, góc.
- Lớp sau: Học phần của hình thể, dáng người.

2.4.3.2.2. Lớp Chuẩn hóa Hàng loạt: layers.BatchNormalization()

Công dụng: Chuẩn hóa đầu ra của Conv2D về trung bình = 0, độ lệch chuẩn = 1.

Tác động:

- Giúp huấn luyện nhanh hơn.
- Giảm phụ thuộc vào khởi tạo.
- Cải thiện khả năng tổng quát.

2.4.3.2.3. Lớp Gộp tối đa: layers.MaxPooling2D((2, 2))

Công dụng: Lấy giá trị lớn nhất trong vùng 2 x 2.

Tác động:

- Giảm kích thước đầu ra, tiết kiệm bộ nhớ/ tính toán.
- Tăng tính bất biến dịch chuyển.

2.4.3.2.4. Lớp Dropout: layers.Dropout(rate)

Công dụng: Ngẫu nhiên "tắt" tỉ lệ % neural trong khi huấn luyện.

Tác động:

- Ngăn overfitting.
- Tăng đa dạng trong huấn luyện (ensemble ngầm).

2.4.3.3. Các Lớp Kết nối Đầy đủ (Fully Connected Layers)

2.4.3.3.1. Lớp làm phẳng: layers.Flatten()

- Chuyển 3D => 1D vertor.
- Chuẩn bị dữ liệu cho Dense.

2.4.3.3.2. Lớp Kết nối Đầy đủ: layers.Dense(256, activation='relu')

- Học các tổ hợp đặc trưng.
- Kết hợp đặc trưng để phân loại.

2.4.3.3.3.. Lớp Chuẩn hóa Hàng loạt: layers.BatchNormalization() + Dropout(0.5)

- Giống ở trên, áp dụng cho Fully Connected.
- Dropout tỷ lệ cao để điều hòa khi gần đầu ra.

2.4.3.3.4. Lớp Đầu ra: layers.Dense(num_classes, activation='softmax')

- Đầu ra cuối: Xác suất cho mỗi lớp.
- Chọn lớp có xác suất cao nhất.

2.4.3.4. Kết luận

Kiến trúc mô hình CNN được thiết kế một cách hệ thống và hợp lý:

- Conv2D: Học đặc trưng cơ bản và phức tạp.
- BatchNormalization + Dropout: Củng cố tốc độ huấn luyện và giảm overfitting.
- MaxPooling: Giảm thiểu chi phí tính toán và tăng tính bền vững.
- Dense: Kết hợp đặc trưng và dự đoán lớp.

Tổng thể, mô hình đáp ứng tốt cho nhiệm vụ nhận dạng tư thế người với dữ liệu ảnh silhouette đơn giản nhưng thách thức.

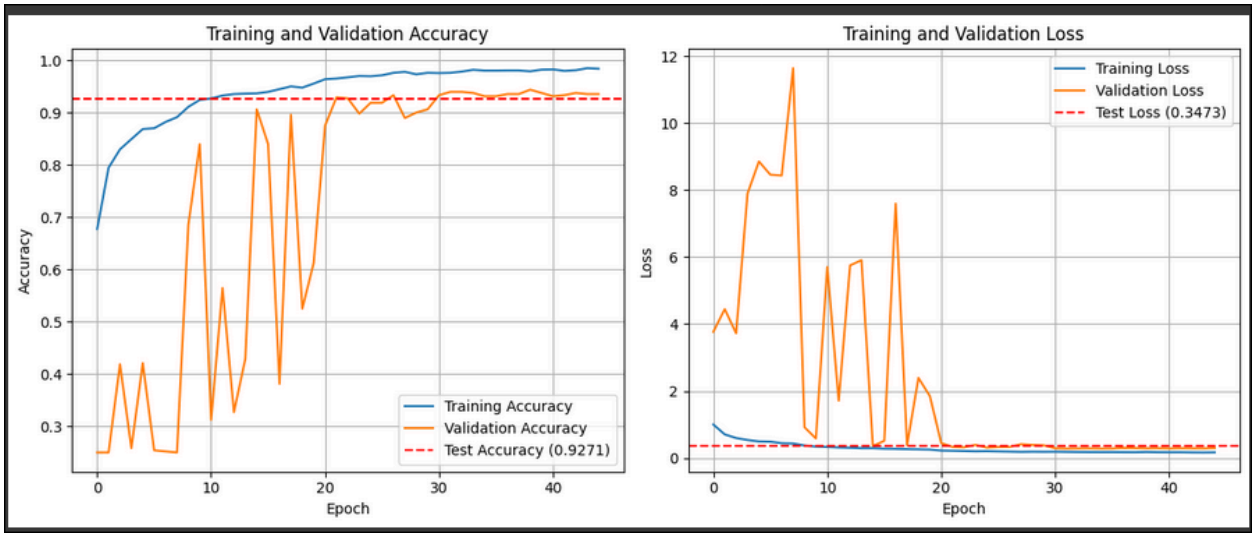
Chương 3 Kết quả và Thực nghiệm

Chương này trình bày các kết quả thực nghiệm đạt được từ việc huấn luyện và đánh giá các mô hình CNN đã thiết kế. Các kết quả bao gồm biểu đồ độ chính xác (Accuracy) và hàm mất mát (Loss) trong quá trình huấn luyện và kiểm định, ma trận nhầm lẫn (Confusion Matrix) và báo cáo phân loại (Classification Report) trên tập dữ liệu kiểm tra.

3.1 Đánh giá các mô hình CNN tự xây dựng

3.1.1 Mô hình CNN-1

Đồ thị huấn luyện:

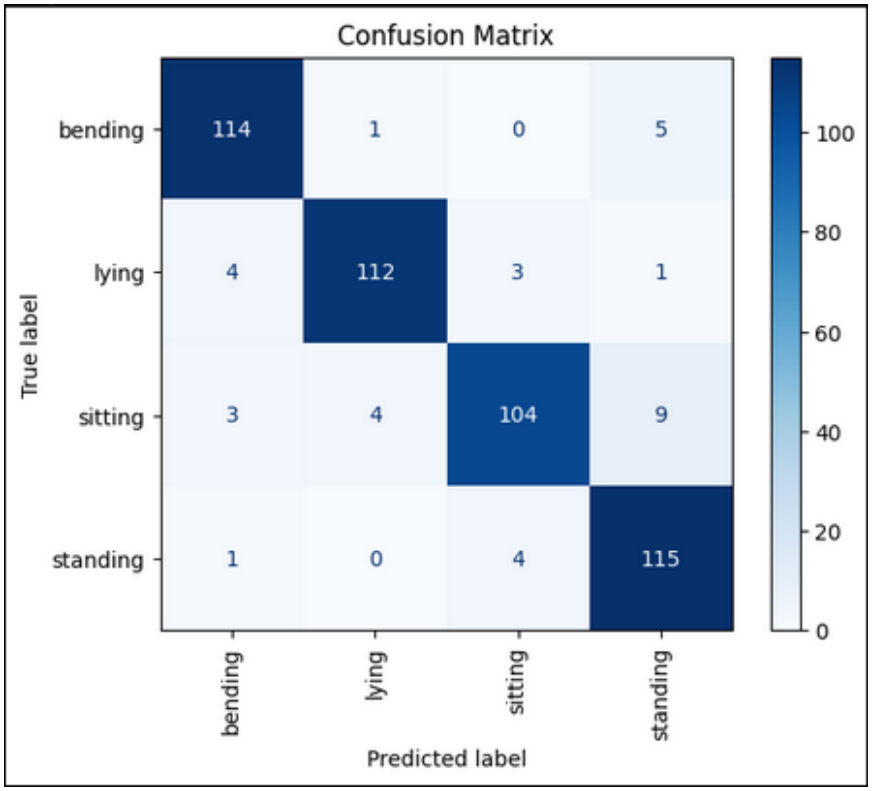


Hình 7. Đồ thị huấn luyện CNN - 1

Biểu đồ "Training and Validation Accuracy" và "Training and Validation Loss"

- **Độ chính xác tăng trưởng ổn định**: Biểu đồ "Training and Validation Accuracy" cho thấy cả độ chính xác huấn luyện (Training Accuracy) và độ chính xác kiểm định (Validation Accuracy) đều tăng lên theo số epoch. Quan trọng hơn, cả hai đường cong đều gần nhau và đạt mức cao (trên 0.90 cho độ chính xác kiểm định) và khá ổn định sau khoảng 30 epoch. Điều này cho thấy mô hình đang học tốt và tổng quát hóa được trên dữ liệu chưa thấy.
- **Loss giảm và ổn định**: Biểu đồ "Training and Validation Loss" cho thấy Training Loss giảm rất nhanh và duy trì ở mức thấp. Validation Loss ban đầu có nhiều biến động nhưng cuối cùng cũng giảm xuống và ổn định ở mức thấp, gần với Test Loss (0.3473) được hiển thị bằng đường đứt nét màu đỏ. Mặc dù có những đỉnh nhiễu trong Validation Loss ở các epoch đầu (có thể do tốc độ học, kích thước lô, hoặc tính chất dữ liệu), nhưng xu hướng tổng thể là sự hội tụ tốt.
- **Không Overfitting đáng kể**: Khoảng cách giữa Training Accuracy và Validation Accuracy không quá lớn ở các epoch cuối, và Validation Loss không tăng đột biến sau khi Training Loss tiếp tục giảm. Điều này cho thấy mô hình không bị overfitting nghiêm trọng. Các lớp Dropout trong kiến trúc đóng vai trò quan trọng trong việc ngăn chặn overfitting.
- **Độ chính xác trên tập test cao**: Test Accuracy (độ chính xác trên tập test) là 0.9271, rất gần với Validation Accuracy cuối cùng. Điều này củng cố rằng mô hình hoạt động tốt trên dữ liệu thực tế và không bị "học thuộc" dữ liệu huấn luyện.

Báo cáo phân loại:



Hình 8. Ma trận nhầm lẫn CNN - 1

Ma trận nhầm lẫn cung cấp cái nhìn chi tiết về hiệu suất của mô hình đối với từng lớp:

- **Số lượng dự đoán đúng cao**: Các giá trị trên đường chéo chính của ma trận (114, 112, 104, 115) là số lượng mẫu được phân loại đúng cho mỗi lớp. Các giá trị này rất cao so với tổng số mẫu cho mỗi lớp (khoảng 120 mẫu mỗi lớp, tổng cộng 480 mẫu).
- **Số lượng dự đoán sai thấp**: Các giá trị ngoài đường chéo chính (số lỗi phân loại) rất nhỏ. Ví dụ:
    - "bending" bị nhầm lẫn là "standing" 5 lần.
    - "lying" bị nhầm lẫn là "bending" 4 lần và "sitting" 3 lần.
    - "sitting" bị nhầm lẫn là "lying" 4 lần và "standing" 9 lần.
    - "standing" bị nhầm lẫn là "bending" 1 lần và "sitting" 4 lần.

- **Đặc biệt, sự nhầm lẫn giữa "sitting" và "standing" là cao nhất (9 lần). Điều này khá hợp lý vì trong một số trường hợp, tư thế ngồi và đứng có thể có những đặc trưng tương đồng hoặc khó phân biệt hơn so với các tư thế khác.**

	precision	recall	f1-score	support
bending	0.93	0.95	0.94	120
lying	0.96	0.93	0.91	120
sitting	0.94	0.87	0.85	120
standing	0.88	0.97	0.92	120
accuracy			0.93	480
macro avg	0.93	0.93	0.93	480
weighted avg	0.93	0.93	0.93	480

Bảng 2. Tóm tắt hiệu xuất phân loại CNN - 1

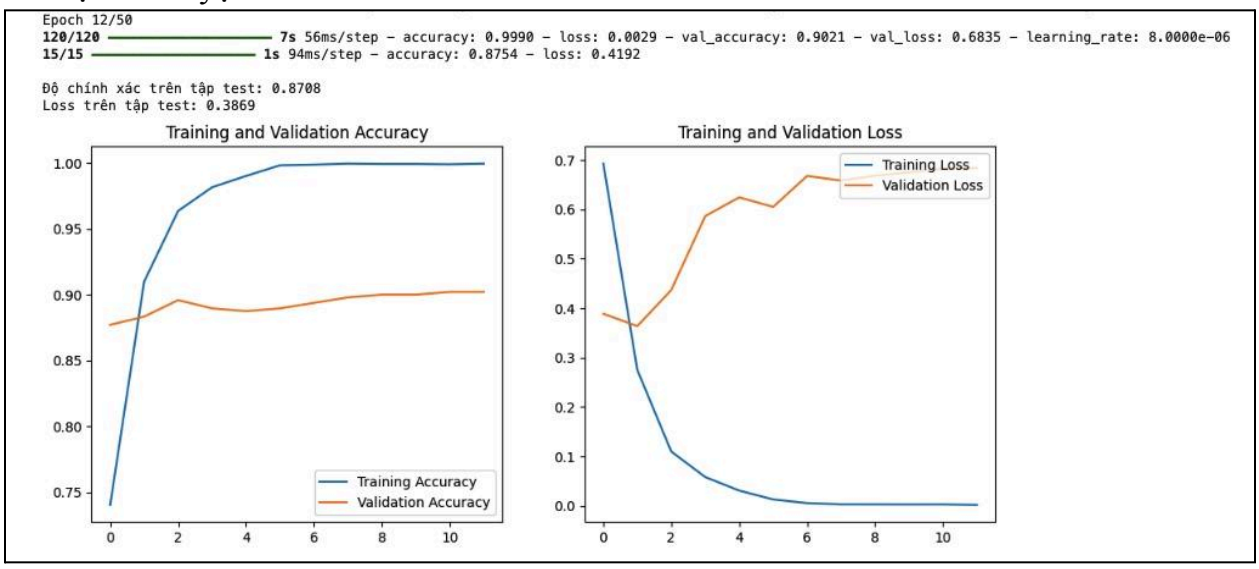
Báo cáo phân loại cung cấp các chỉ số precision, recall, và f1-score cho từng lớp:

- **Precision (Độ chính xác)**: Tỷ lệ các dự đoán tích cực đúng trong số tất cả các dự đoán tích cực. Giá trị precision cao (0.93 - 0.96) cho các lớp "bending" và "lying" cho thấy khi mô hình dự đoán là "bending" hoặc "lying", khả năng cao là đúng.
- **Recall (Độ phủ)**: Tỷ lệ các trường hợp tích cực đúng được dự đoán là tích cực. Giá trị recall cao (0.93 - 0.96) cho các lớp "bending", "lying" và "standing" cho thấy mô hình có khả năng tìm thấy hầu hết các trường hợp thực sự thuộc về các lớp đó. Lớp "sitting" có recall thấp nhất (0.87), điều này phù hợp với việc nó bị nhầm lẫn nhiều nhất trong ma trận nhầm lẫn.
- **F1-score**: Là trung bình điều hòa của precision và recall, cung cấp một cái nhìn cân bằng về hiệu suất. F1-score cao (0.90 - 0.95) cho thấy hiệu suất tổng thể tốt cho tất cả các lớp.

- **Accuracy (Độ chính xác tổng thể)**: 0.93. Đây là độ chính xác tổng thể của mô hình trên tập test, xác nhận hiệu suất cao.

3.1.2 Mô hình CNN-2

Đồ thị huấn luyện:



Hình 9. Đồ thị huấn luyện CNN - 2

Training and Validation Accuracy

- **Training Accuracy** tăng rất nhanh và đạt gần 1.0 chỉ sau khoảng 5 epoch, sau đó duy trì ổn định.

- **Validation Accuracy** ban đầu tăng nhẹ, dao động quanh mức 0.88 đến 0.90 từ khoảng epoch 2 đến epoch 11, nhưng không có xu hướng cải thiện rõ rệt sau các epoch đầu tiên.

- **Nhận xét**: Đường Accuracy của validation không theo kịp training và dần ổn định sớm, điều này có thể cho thấy mô hình bắt đầu học quá tốt dữ liệu huấn luyện mà không cải thiện khả năng tổng quát hóa thêm.

Training and Validation Loss

- **Training Loss** giảm rất nhanh và về gần 0, điều này cho thấy mô hình học rất tốt trên dữ liệu huấn luyện.

- **Validation Loss** có xu hướng tăng dần từ epoch 2 trở đi, dao động khá mạnh và đạt đỉnh ở các epoch sau (khoảng trên 0.65).

- **Nhận xét**: Validation loss tăng trong khi training loss tiếp tục giảm cho thấy hiện tượng **overfitting** – mô hình bắt đầu ghi nhớ quá mức dữ liệu huấn luyện thay vì học cách tổng quát hóa.

Báo cáo phân loại:

	precision	recall	f1-score	support
bending	0.85	0.88	0.86	120
lying	0.90	0.93	0.91	120
sitting	0.82	0.88	0.85	120
standing	0.92	0.81	0.86	120
accuracy			0.87	480
Macro avg	0.87	0.87	0.87	480
Weighted avg	0.87	0.87	0.87	480

Bảng 3. Tóm tắt hiệu xuất phân loại CNN - 2

Precision (Độ chính xác):

- Các giá trị precision dao động từ **0.82 đến 0.92**.

- Lớp **standing** đạt precision cao nhất (0.92), cho thấy khi mô hình dự đoán một hành động là "standing", khả năng đúng là rất cao.

- Lớp **sitting** có precision thấp nhất (0.82), cho thấy mô hình đôi khi nhầm lẫn khi dự đoán hành động này.

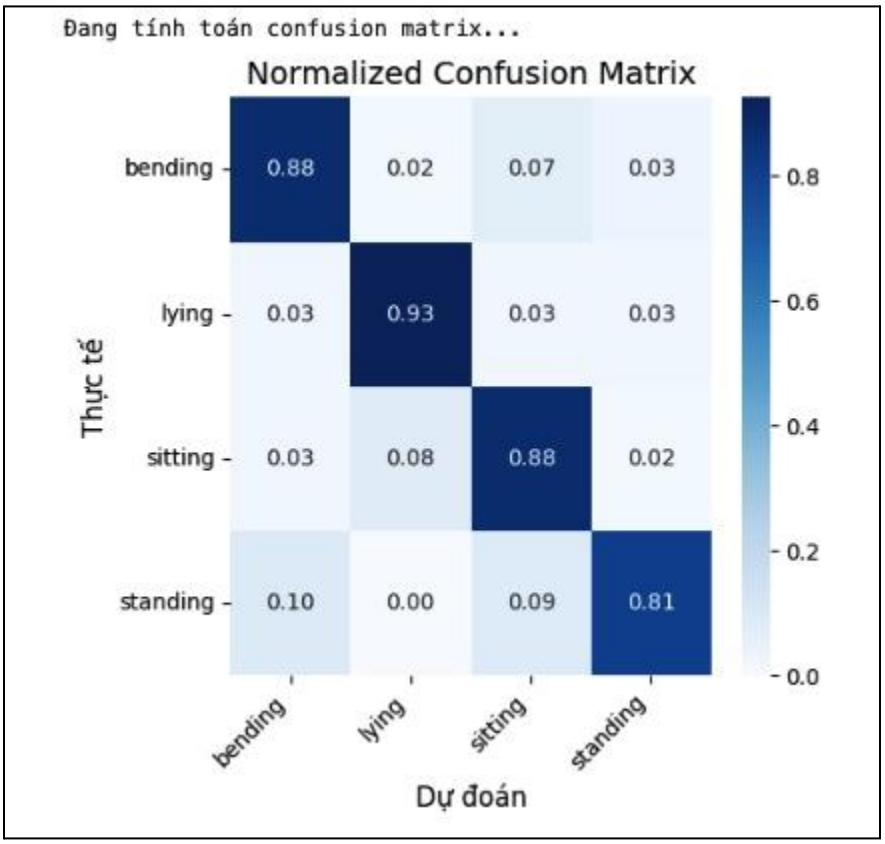
Recall (Độ phủ):

- Các giá trị recall dao động từ **0.81 đến 0.93**.
- Lớp **lying** đạt recall cao nhất (0.93), chứng tỏ mô hình rất tốt trong việc nhận diện đúng các trường hợp "lying".
- Lớp **standing** có recall thấp nhất (0.81), có thể do bị nhầm với các lớp tương tự như "sitting" hoặc "bending".

F1-Score (Điểm F1):

- Giá trị F1 dao động từ **0.85 đến 0.91**.
- Lớp **lying** nổi bật với F1-score cao nhất (0.91), phản ánh hiệu suất cân bằng tốt nhất giữa precision và recall.
- Các lớp khác như **bending**, **sitting**, và **standing** có F1-score khá tương đồng (0.85–0.86), cho thấy mô hình xử lý tương đối đồng đều giữa các lớp.

Ma trận nhầm lẫn:



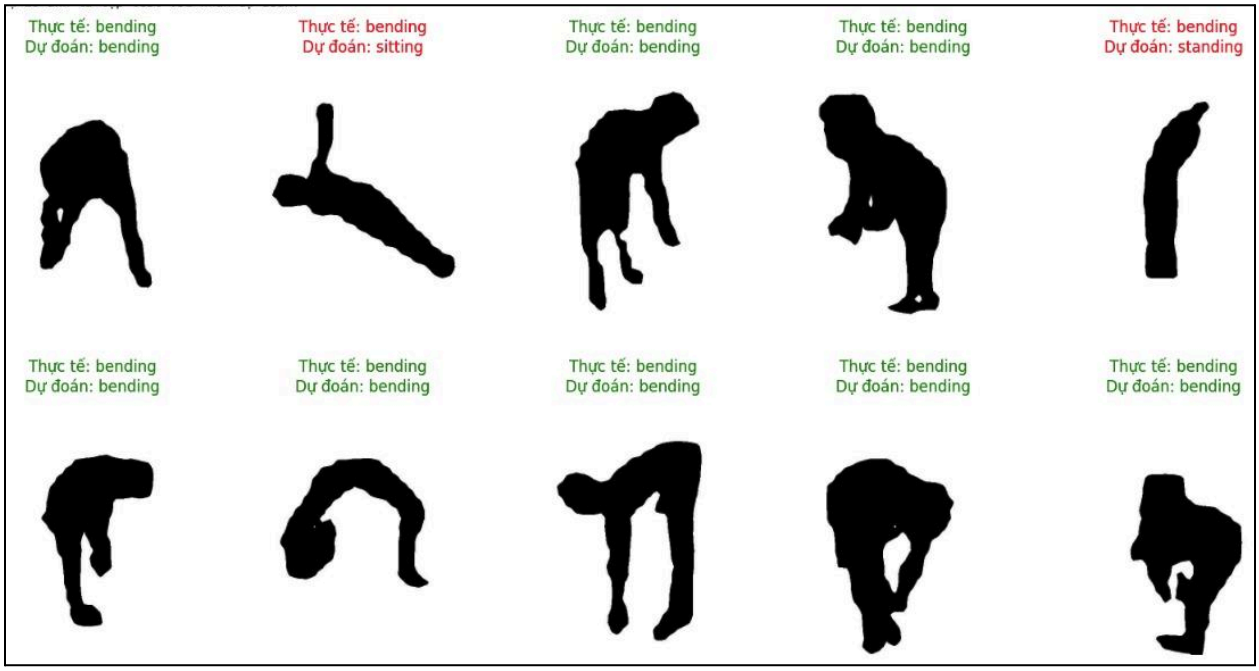
Hình 10. Ma trận nhầm lẫn CNN - 2.

Ma trận nhầm lẫn được chuẩn hóa cho thấy mô hình CNN đạt độ chính xác cao trong nhiệm vụ phân loại các tư thế của con người, bao gồm: **bending**, **lying**, **sitting** và **standing**. Kết quả phân loại cho thấy phần lớn các mẫu thuộc từng lớp đều được mô hình nhận diện chính xác, với độ chính xác từng lớp đều lớn hơn 80%.

Trong đó, lớp **lying** đạt độ chính xác cao nhất, với tỷ lệ dự đoán đúng là **93%**, cho thấy tư thế này có đặc trưng nổi bật và dễ dàng được phân biệt so với các tư thế còn lại. Lớp **bending** và **sitting** đều có độ chính xác **88%**, tuy nhiên vẫn tồn tại sự nhầm lẫn nhỏ giữa hai tư thế này, chủ yếu do sự tương đồng về hình dạng trong một số tình huống cụ thể. Đáng chú ý, lớp **standing** có độ chính xác thấp nhất (**81%**) và bị nhầm lẫn đáng kể với các tư thế **bending (10%)** và **sitting (9%)**. Điều này có thể xuất phát từ việc đặc trưng hình ảnh của các tư thế này có sự chồng lấn, hoặc do dữ liệu huấn luyện chưa bao phủ đầy đủ các biến thể của tư thế đứng.

Từ kết quả này có thể nhận định rằng mô hình có khả năng phân loại tốt đối với các tư thế có đặc trưng rõ ràng. Tuy nhiên, để nâng cao hiệu quả nhận dạng ở các tư thế dễ gây nhầm lẫn, cần xem xét bổ sung dữ liệu huấn luyện với mức đa dạng cao hơn (về góc nhìn, đối tượng, bối cảnh), đồng thời áp dụng các biện pháp tăng cường dữ liệu (data augmentation) hoặc điều chỉnh cấu trúc mô hình nhằm cải thiện khả năng phân biệt giữa các lớp.

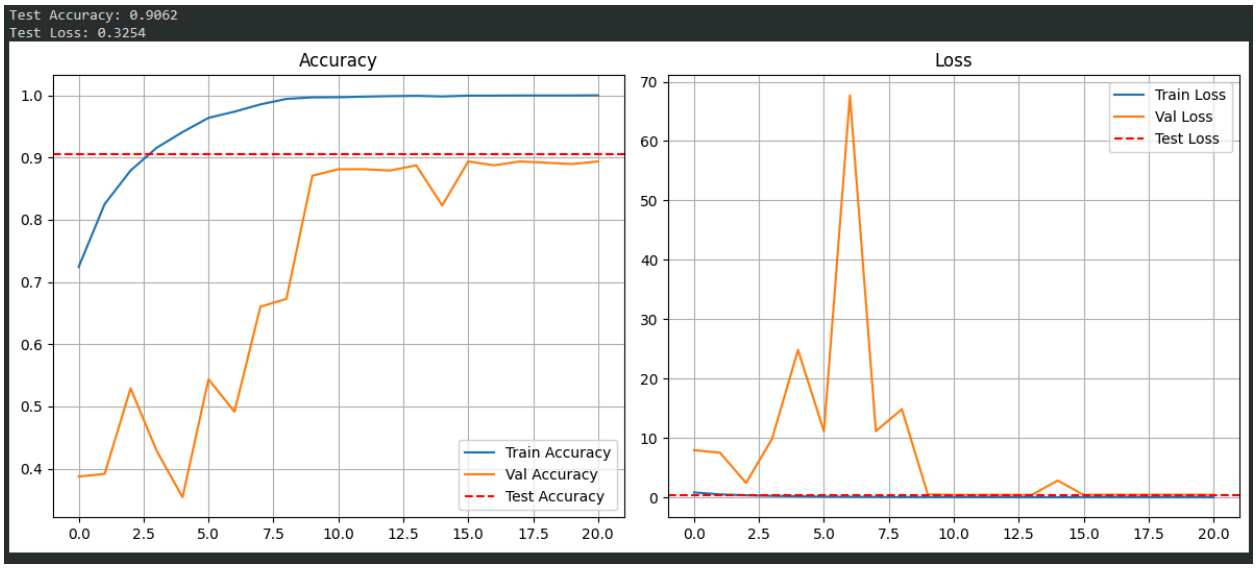
Hiển thị 10 ảnh từ tập test với nhãn dự đoán



Hình 11. Dự đoán 10 ảnh từ tập test CNN - 2

3.1.3. Mô hình CNN-3

Đồ thị huấn luyện:



Hình 12. Đồ thị huấn luyện CNN - 3

3.1.3.1. Độ chính xác (Accuracy)

- Train Accuracy: Tăng rất nhanh và đạt xấp xỉ 100% sau khoảng 10 epoch, cho thấy mô hình học rất tốt trên tập huấn luyện.
- Validation Accuracy: Dao động và tăng từ từ, đạt khoảng 89-90% vào cuối quá trình huấn luyện, tiệm cận với Test Accuracy = 90.62% (đường kẻ đỏ).
- Sự chênh lệch giữa train và val không lớn → **ít xảy ra overfitting**, hoặc overfitting đã được khắc phục sau những epoch đầu.

3.1.3.2. Hàm mất mát (Loss)

- Training Loss: giảm đều và gần như chạm về 0, biểu hiện mô hình khớp cực tốt với dữ liệu huấn luyện.
- Validation Loss: có nhiều dao động mạnh, đặc biệt tại epoch 6 (tăng đột biến), cho thấy dữ liệu validation có thể chứa nhiều nhiễu hoặc mô hình đang có dấu hiệu overfitting nhẹ ở giai đoạn đầu, nhưng đã cải thiện sau đó.
- Sau khoảng epoch 10 → validation loss tiến về gần 0, thể hiện sự ổn định của mô hình về sau.

Nhận xét:

- Mô hình ban đầu gặp dao động, nhưng khả năng học tốt và tổng quát ổn định sau giai đoạn đầu.
- Không bị overfit rõ rệt, và test loss thấp (0.3254) cho thấy mô hình có thể áp dụng tốt với dữ liệu mới.

Báo cáo phân loại:

	precision	recall	f1-score	support
bending	0.90	0.89	0.90	120
lying	0.97	0.91	0.94	120
sitting	0.90	0.88	0.89	120
standing	0.86	0.94	0.90	120
accuracy			0.91	480
Macro avg	0.91	0.91	0.91	480
Weighted avg	0.91	0.91	0.91	480

Bảng 4. Tóm tắt hiệu xuất CNN - 3

Độ chính xác tổng thể:

- Accuracy đạt 91% trên tập kiểm tra (480 mẫu) → cho thấy mô hình có hiệu suất cao, đáng tin cậy.
- Macro average / Weighted average của precision, recall và f1-score đều đạt 0.91, chứng tỏ mô hình hoạt động ổn định và cân bằng giữa các lớp (không thiên lệch).

Đánh giá hiệu quả mô hình CNN qua các chỉ số phân loại:

Dựa trên kết quả phân loại thu được, mô hình CNN đạt độ chính xác tổng thể (accuracy) là 91% trên tập dữ liệu kiểm tra. Các chỉ số precision, recall và f1-score đều duy trì ở mức cao và ổn định trên cả 4 lớp tư thế (đứng, ngồi, nằm, cúi người), cho thấy mô hình có năng lực phân loại đồng đều và không thiên lệch về lớp.

- Lớp "lying" (nằm) có hiệu suất cao nhất, với precision = 0.97 và f1-score = 0.94. Điều này cho thấy tư thế "nằm" có đặc trưng hình học rõ ràng, dễ được mô hình phân biệt và rất ít bị nhầm lẫn với các tư thế khác.

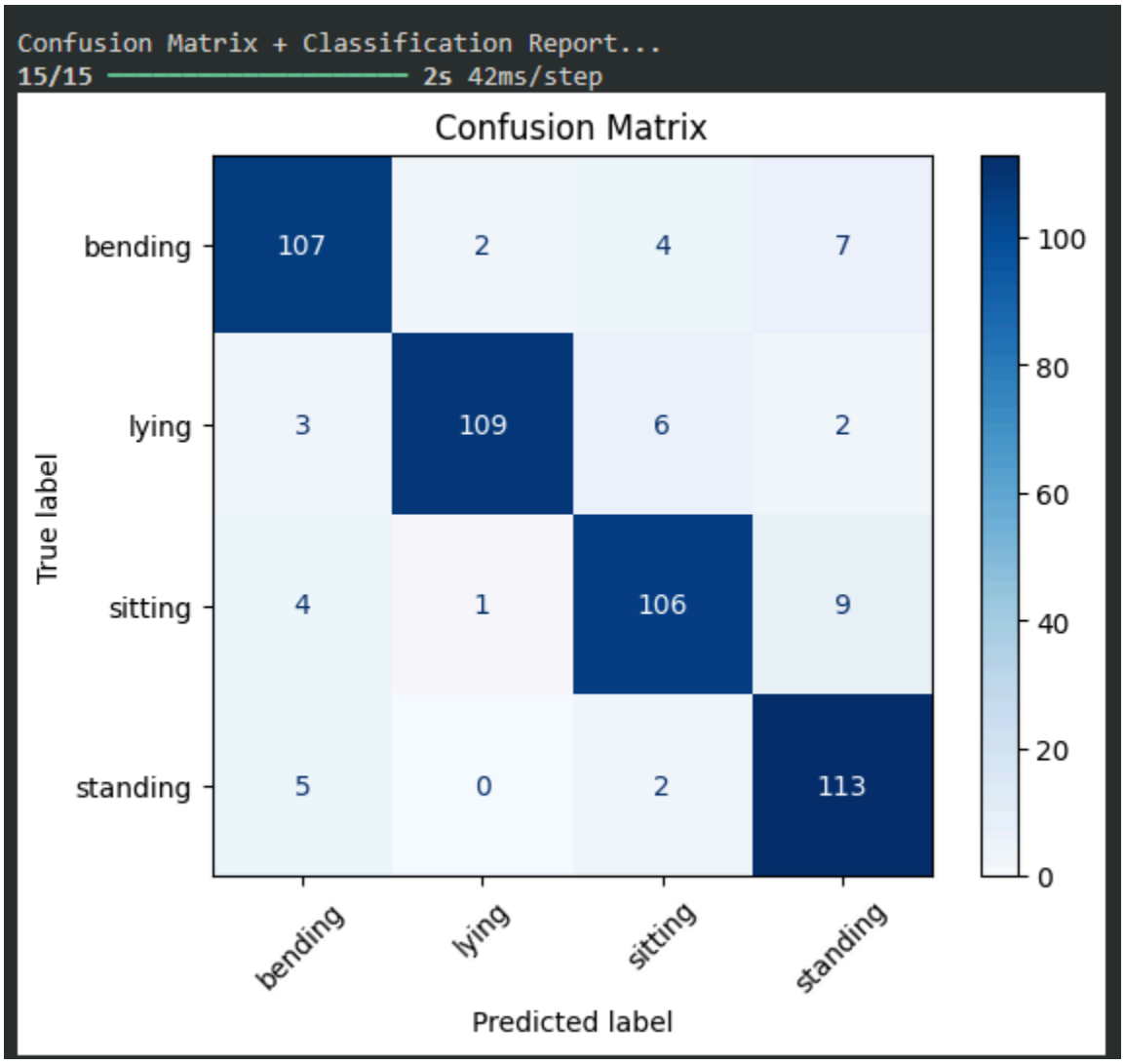
- Lớp "standing" (đứng) có recall cao nhất (0.94), tức là phần lớn các mẫu tư thế đứng được nhận diện đúng. Tuy nhiên, precision chỉ đạt 0.86, cho thấy mô hình vẫn có xu hướng gán nhầm một số tư thế khác (đặc biệt là bending và sitting) thành standing. Điều này phản ánh sự tương đồng hình dáng giữa các tư thế khi nhìn từ một số góc nhất định.

- Lớp "bending" (cúi người) đạt precision = 0.90 và recall = 0.89, phản ánh khả năng phân loại tốt. Tuy nhiên vẫn tồn tại nhầm lẫn nhỏ với tư thế standing, có thể do hình dạng phần thân gần giống trong một số trường hợp.

- Lớp "sitting" (ngồi) cũng có hiệu suất khá tốt (f1-score = 0.89), tuy nhiên vẫn bị nhầm lẫn sang standing và ngược lại, phản ánh thách thức trong việc phân biệt hai tư thế này ở dạng ảnh silhouette.

Trung bình cộng (macro average) và trung bình có trọng số (weighted average) của các chỉ số đều đạt 0.91, cho thấy mô hình giữ được hiệu suất ổn định trên tất cả các lớp và không bị lệch về một lớp c**ụ thể nào.**

Ma trận nhầm lẫn:



Hình 13. Ma trận nhầm lẫn CNN - 3

Ma trận nhầm lẫn cho thấy mô hình CNN phân loại tư thế con người với độ chính xác cao trên cả bốn lớp: **bending, lying, sitting, standing**. Cụ thể:

- **Lớp "bending" (cúi người)**:
    - Có 107/120 mẫu được dự đoán đúng, đạt độ chính xác ~89%.

    - Mô hình chủ yếu nhầm lẫn với “standing” (7 mẫu) và “sitting” (4 mẫu), phản ánh đặc điểm hình học tương tự khi cúi hoặc ngồi thẳng người.

    - Số mẫu bị nhầm sang “lying” rất ít (2 mẫu), cho thấy sự khác biệt rõ giữa hai tư thế này.

- **Lớp “lying” (nằm)**:

    - Mô hình nhận diện 109/120 mẫu chính xác (~91%), là một trong những lớp có hiệu suất tốt nhất.

    - Một số mẫu bị nhầm sang “sitting” (6 mẫu), nhưng tỷ lệ này thấp.

    - Điều này cho thấy tư thế “nằm” có đặc trưng hình bóng nổi bật và ít gây nhầm lẫn.

- **Lớp “sitting” (ngồi)**:

    - Có 106 mẫu dự đoán đúng, chiếm gần 88%.

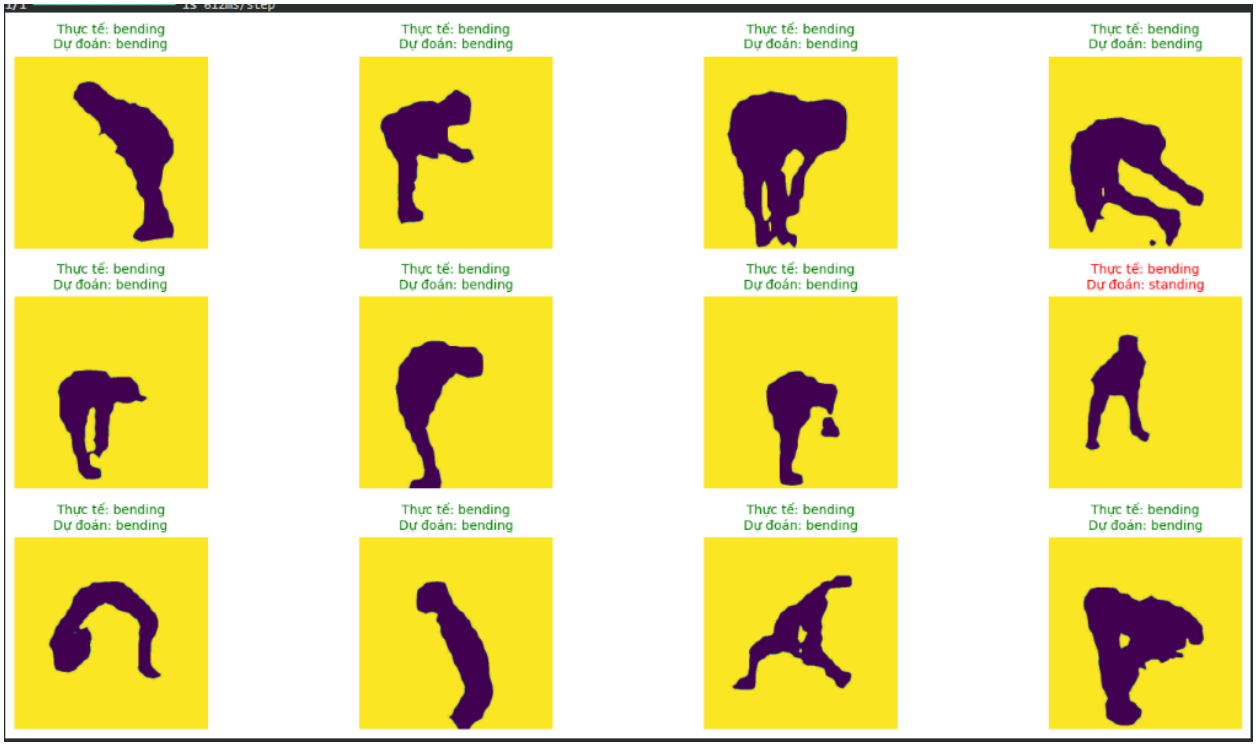
    - Mô hình nhầm lẫn “sitting” nhiều nhất với “standing” (9 mẫu) và “bending” (4 mẫu), điều này là hợp lý vì ba tư thế này dễ gây nhầm lẫn nếu nhìn từ phía trước hoặc có góc nhìn tương tự.

- **Lớp “standing” (đứng)**:

    - Mô hình dự đoán đúng 113/120 mẫu, đạt độ chính xác khoảng 94%, là lớp có recall cao nhất.

    - Một số nhầm lẫn nhỏ xảy ra với “bending” (5 mẫu), “sitting” (2 mẫu), cho thấy mô hình vẫn chưa phân tách hoàn toàn rõ ràng giữa ba tư thế có hình dáng cơ thể gần giống nhau.

Hiển thị 16 ảnh từ tập test với nhãn dự đoán:



Hình 14. Dự đoán 16 ảnh từ tập test CNN - 3

Chương 4 Kết luận và Hướng phát triển

4.1 Kết luận

Trong đề tài này, ba mô hình mạng nơ-ron tích chập (CNN) đã được xây dựng và đánh giá nhằm giải quyết bài toán phân loại tư thế con người từ ảnh silhouette với bốn lớp: đứng, ngồi, nằm và cúi người.

Các kết quả đạt được có thể tổng kết như sau:

- **Mô hình CNN-1** cho thấy hiệu suất cao nhất với độ chính xác lên đến **92.71%** trên tập test, cùng các chỉ số precision, recall và f1-score cao và ổn định trên cả bốn lớp. Mô hình thể hiện khả năng học sâu và tổng quát hóa tốt nhờ cấu trúc phức tạp với nhiều lớp tích chập, batch normalization và dropout.

- **Mô hình CNN-2**, dù có thiết kế đơn giản hơn, vẫn đạt được độ chính xác **87%**. Tuy nhiên, mô hình có xu hướng overfitting khi training accuracy tăng nhanh trong khi validation accuracy và loss dao động mạnh. Điều này cho thấy mô hình phù hợp hơn với bài toán nhỏ hoặc cần tối ưu thêm về cấu trúc.

- **Mô hình CNN-3** đạt độ chính xác **90.62%**, gần ngang ngửa CNN-1, đồng thời thể hiện sự ổn định trong suốt quá trình huấn luyện. Kiến trúc này kết hợp hiệu quả giữa các khối convolutional, chuẩn hóa và dropout để cân bằng giữa hiệu năng và khả năng tổng quát hóa.

Từ các kết quả trên, có thể rút ra kết luận rằng **mạng CNN hoàn toàn có khả năng phân biệt chính xác các tư thế con người từ ảnh silhouette**, với độ chính xác cao và phù hợp triển khai thực tế trong các hệ thống giám sát, hỗ trợ người cao tuổi, hoặc các ứng dụng chăm sóc sức khỏe thông minh.

4.2 Hướng phát triển

Để nâng cao hiệu quả mô hình và mở rộng ứng dụng trong tương lai, một số hướng phát triển được đề xuất:

- **Tăng cường dữ liệu (Data Augmentation):** Sử dụng xoay ảnh, dịch chuyển, nhiễu hoặc biến đổi hình dạng để tăng độ đa dạng của tập huấn luyện và tăng khả năng tổng quát hóa.

- **Thử nghiệm với các mô hình tiền huấn luyện (Pre-trained Models)** mạnh hơn như EfficientNet, DenseNet hoặc Vision Transformer để cải thiện hiệu suất mà không cần tăng số lượng dữ liệu.

- **Tối ưu hóa siêu tham số tự động:** Áp dụng Grid Search, Random Search hoặc Keras Tuner để tìm cấu hình tốt nhất cho từng kiến trúc.

- **Phân tích theo thời gian (temporal analysis):** Tích hợp thêm dữ liệu chuỗi hình ảnh (video hoặc nhiều khung hình liên tiếp) để nhận diện chuyển động, từ đó phân biệt rõ ràng hơn các tư thế tương đồng.

- **Triển khai thực tế:** Xây dựng demo hệ thống tích hợp mô hình CNN với camera thời gian thực để ứng dụng trong các môi trường như bệnh viện, trung tâm chăm sóc người cao tuổi, hoặc hệ thống an ninh.