TRƯỜNG ĐẠI HỌC NGOẠI NGỮ - TIN HỌC TP. HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN HỌC PHẦN

Phần mềm nhận dạng 5 đối tượng dụng cụ học tập

GIẢNG VIÊN HƯỚNG DẪN: ThS. Tôn Quang Toại

SINH VIÊN THỰC HIỆN: Huỳnh Như Ý - 21DH114339

Lê Minh Toàn - 21DH113392

Nguyễn Minh Hiền - 21DH110542

Nguyễn Tuyết Trinh - 21DH113050

TP. HỒ CHÍ MINH – 03/2024

PHIẾU CHẤM ĐIỂM

Họ tên sinh viên 1 (SV1): Huỳnh Như Ý _____ Mã SV: 21DH114339

Họ tên sinh viên 2 (SV2): Lê Minh Toàn _____ Mã SV: 21DH113392

Họ tên sinh viên 3 (SV3): Nguyễn Minh Hiền _____ Mã SV: 21DH110542

Họ tên sinh viên 4 (SV4): Nguyễn Tuyết Trinh_____ Mã SV: 21DH113050

CLO	Nội dung/Chuẩn đầu ra	ĐIỂM CỦA SV 1	ĐIỂM CỦA SV 2	ĐIỂM CỦA SV 3	ĐIỂM CỦA SV 4
1	Xây dựng và huấn luyện mạng neuron (Bài toán phân lớp ảnh, huấn luyện mô hình phân lớp…)				
2	Tinh chỉnh mô hình và thuật toán huấn luyện (Bài toán phân đoạn, tìm gốc quay, phát hiện vị trí, …)				
3	Vận dụng mạng neuron tích chập và mạng hồi quy (Triển khai mô hình trên web, desktop, mobile, …)				
4	Có khả năng giải quyết một số vấn đề thực tế (Thu thập dữ liệu, xử lý dữ liệu, …)				
5	Có năng lực trình bày giải pháp kỹ thuật (Thuyết trình, trình bày báo cáo, …)				
	Tổng				

Họ tên GV 1: _____ Ký tên: _____

Họ tên GV 2: _____ Ký tên: _____

TÓM TẮT ĐỒ ÁN

Phần mềm cần nhận dạng và xác định vị trí của 5 đối tượng dụng cụ học tập trong một ảnh. Đối tượng này có thể bao gồm bút chì, bút mực, compa, máy tính, và thước kẻ. Mục tiêu là xác định đối tượng trong một ảnh.

Thu thập dữ liệu: Thu thập một tập dữ liệu lớn chứa hình ảnh của các đối tượng dụng cụ học tập. Tập dữ liệu này sẽ được sử dụng để huấn luyện mô hình nhận dạng đối tượng.

Phân lớp các đối tượng: Sử dụng một mô hình học máy để phân lớp các đối tượng trong tập dữ liệu. Mô hình này sẽ được huấn luyện để nhận biết và phân loại các đối tượng dựa trên các đặc trưng hình ảnh.

Xác định vị trí đối tượng trong ảnh: Sử dụng một mô hình nhận dạng đối tượng để xác định vị trí của các đối tượng trong ảnh. Mô hình này sẽ được huấn luyện để xác định vị trí của các đối tượng và vẽ một hộp chứa (bounding box) xung quanh chúng.

Triển khai mô hình: Sau khi mô hình đã được huấn luyện, nó sẽ được triển khai để nhận dạng và xác định vị trí của các đối tượng dụng cụ học tập trong các ảnh mới.

Kết quả là một phần mềm có khả năng nhận dạng và xác định vị trí của tối đa 2 đối tượng dụng cụ học tập trong một ảnh. Phần mềm này có thể được sử dụng trong các ứng dụng học tập trực tuyến, giúp học sinh dễ dàng tìm kiếm và sử dụng các dụng cụ học tập cần thiết.

MỤC LỤC

Chương 1. **Giới thiệu bài toán**

1.1 Câu hỏi nghiên cứu

- **Phát biểu nội dung bài toán:** Bài toán nhằm xây dựng một hệ thống nhận dạng đối tượng trong hình ảnh để phân loại các dụng cụ học tập như bút, viết, thước kẻ, compa, máy tính... từ hình ảnh chụp. Mục tiêu là tạo ra một mô hình máy học có khả năng tự động phân loại các đối tượng trong ảnh với độ chính xác cao.
- **Mô tả Input của bài toán:** Input của bài toán là các hình ảnh chứa các dụng cụ học tập. Các hình ảnh này có thể được chụp từ nhiều góc độ và điều kiện khác nhau, có thể có nền phức tạp hoặc không đồng nhất. Các hình ảnh có độ phân giải và kích thước khác nhau, và có thể chứa một hoặc nhiều đối tượng cần phân loại.
- **Mô tả Output của bài toán:** Output là nhãn (label) của đối tượng trong hình ảnh, cho biết đây là dụng cụ học tập gì. Đối với mỗi hình ảnh đầu vào, mô hình sẽ xuất ra một nhãn duy nhất để mô tả đối tượng trong hình ảnh đó.
- **Lấy một số ví dụ minh họa:** Một ví dụ minh họa có thể là quá trình nhận dạng một hình ảnh chứa một thước kẻ. Input của bài toán là hình ảnh chứa thước kẻ, có thể chụp từ nhiều góc độ và điều kiện ánh sáng khác nhau. Output của mô hình sẽ là nhãn "thước kẻ", xác định rằng trong hình ảnh đó có một thước kẻ.
- **Giới thiệu Dataset được sử dụng để giải quyết bài toán:** Dataset được sử dụng bao gồm các thư mục chứa hình ảnh được chia thành ba tập dữ liệu: train, validation và test. Mỗi hình ảnh được gắn nhãn với đối tượng trong ảnh, ví dụ: bút, viết, thước kẻ, compa, máy tính. Dataset này cung cấp đủ độ phức tạp và biến động về góc chụp, ánh sáng, nền, giúp mô hình có thể học được các điều kiện đa dạng của các đối tượng.

1.2 Giới hạn nghiên cứu

- **Giới hạn về đối tượng nhận dạng:** Nghiên cứu này chỉ tập trung vào việc nhận dạng và phân loại các đối tượng cụ thể trong hình ảnh, bao gồm bút, viết, thước kẻ, compa, máy tính. Các đối tượng khác ngoài phạm vi này sẽ không được xem xét trong bài toán.
- **Giới hạn về độ phức tạp của hình ảnh:** Mặc dù cố gắng tạo ra mô hình có khả năng nhận dạng trong các điều kiện chụp khác nhau, nhưng hình ảnh quá phức tạp, có nhiều đối tượng hoặc có nhiều nhiễu có thể làm giảm hiệu suất của mô hình. Do đó, giới hạn về độ phức tạp của hình ảnh có thể ảnh hưởng đến khả năng nhận dạng của mô hình.
- **Giới hạn về số lượng lớp đối tượng:** Bài toán này chỉ tập trung vào phân loại các đối tượng vào một trong số năm lớp đã đề ra. Các lớp đối tượng khác ngoài danh sách này sẽ không được xem xét.
- **Giới hạn về số lượng và độ đa dạng của dữ liệu:** Mặc dù cố gắng sử dụng một dataset đa dạng, nhưng số lượng ảnh và độ đa dạng của các đối tượng trong dataset có thể bị hạn chế. Điều này có thể ảnh hưởng đến khả năng tổng quát hóa của mô hình đối với dữ liệu mới.

1.3 Bố cục đồ án

Hướng dẫn: Trình bày một đoạn văn cho biết đồ án có bao nhiêu chương, bao nhiêu mục, tóm tắt ngắn gọn nội dung của từng chương, các mục.

Chương 2. Giải pháp đề xuất

2.1 Phân tích dữ liệu

1. Số lượng mẫu:

- Tập dataset bao gồm:

  - Train: 2392 ảnh.
  - Test: 125 ảnh.
  - Validation: 249 ảnh.

2. Kích thước dữ liệu:

- Kích thước ảnh: 224x224 pixel.

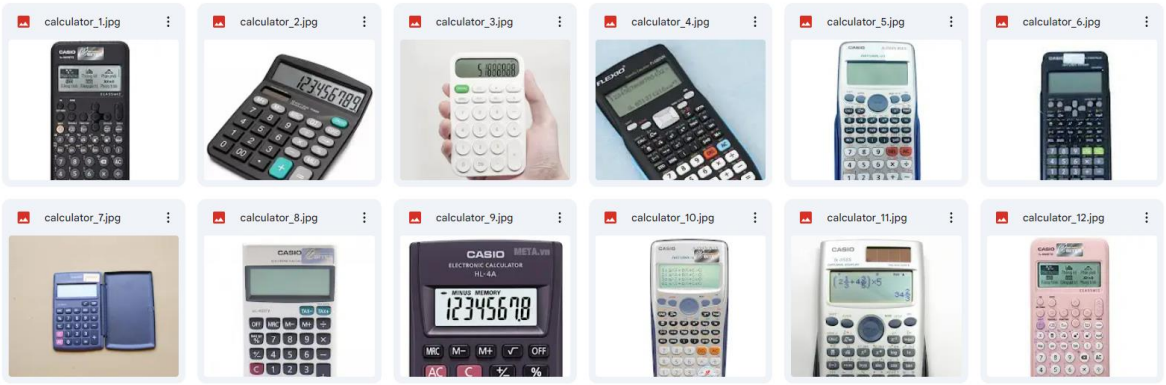
3. Phân loại ảnh:

- Số lượng ảnh trong từng danh mục:

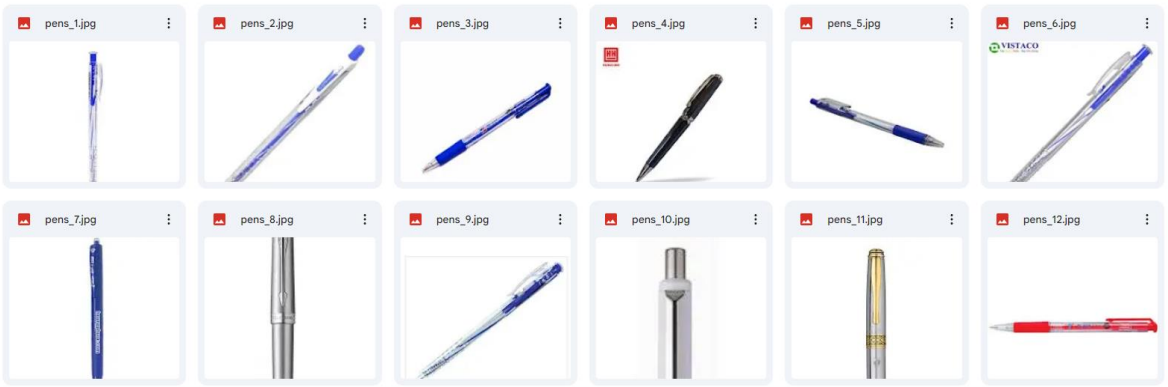
  - Calculator: 479 ảnh.
  - Pens: 490 ảnh.
  - Pencil: 483 ảnh.
  - Compa: 465 ảnh.
  - Ruler: 475 ảnh.

Dưới đây là một số ví dụ về hình ảnh từ tập dataset:

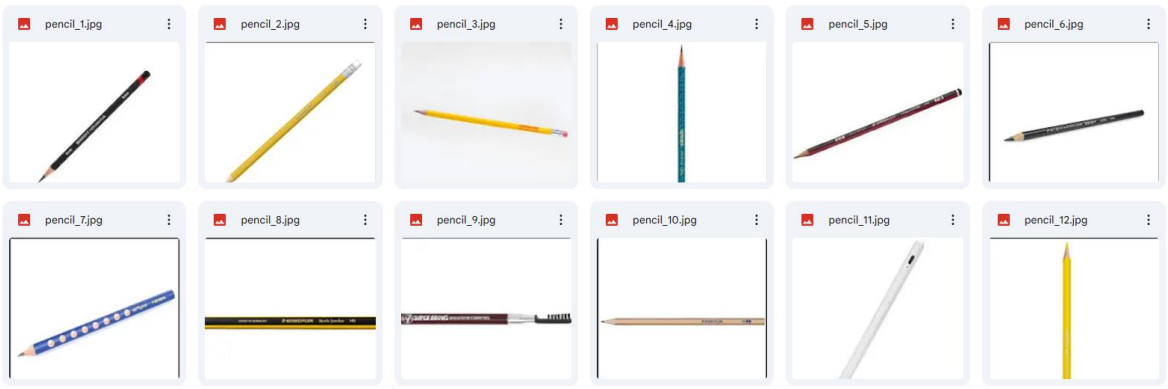
Hình ảnh calculator:



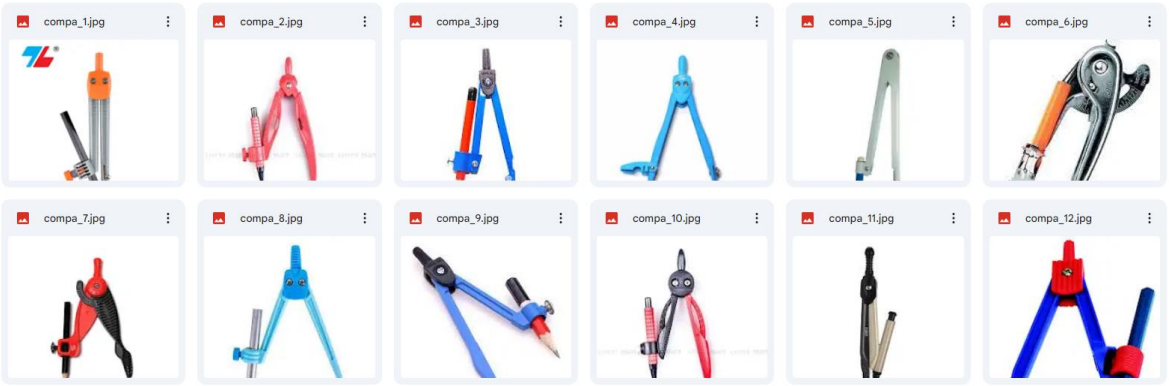
Hình ảnh pen:



Hình ảnh pencil:



Hình ảnh compa:



Hình ảnh ruler:

4. Nhận xét:

- Tập dataset khá cân bằng về số lượng ảnh giữa các danh mục.

- Kích thước ảnh đồng nhất (224x224 pixel), giúp cho việc xử lý và huấn luyện mô hình trở nên thuận tiện.

2.2 Mô hình

2.2.1 Mô hình CNN (Convolutional Neural Network)

- **Kiến trúc mô hình**

- **Input layer:** Nhận đầu vào là các hình ảnh có kích thước (224, 224, 3).
- **Convolutional layers:** Gồm 3 lớp Convolutional, mỗi lớp được kích hoạt bởi hàm activation ReLU (Rectified Linear Unit) để học các đặc trưng của ảnh.
- **MaxPooling layers:** Sử dụng lớp MaxPooling2D để giảm kích thước của đầu ra và tăng cường tính invariance của mô hình đối với biến dạng và dịch chuyển trong dữ liệu.
- **Flatten layer:** Lớp này được sử dụng để chuyển đổi đầu ra ở dạng ma trận 2D thành một vector 1D trước khi đưa vào các lớp fully connected.
- **Fully connected layers:** Gồm 2 lớp fully connected, được kích hoạt bởi hàm activation ReLU.
- **Output layer:** Sử dụng lớp fully connected với hàm activation softmax để đầu ra là xác suất dự đoán của mỗi lớp.

- **Bảng mô tả chi tiết kiến trúc**

Layer (type)	**Output Shape**	**Param #**
Conv2D	(None, 224, 224, 32)	896
MaxPooling2D	(None, 112, 112, 32)	0
Conv2D	(None, 112, 112, 64)	18,496
MaxPooling2D	(None, 56, 56, 64)	0
Conv2D	(None, 56, 56, 128)	36,928
Flatten	(None, 173056)	0
Dense	(None, 64)	11,075,684
Dropout	(None, 64)	0
Dense	(None, 5)	325

- **Hàm activation của tầng cuối:** Sử dụng hàm activation softmax tại tầng cuối để đầu ra là xác suất dự đoán của mỗi lớp.

```
# Thêm tầng cuối cùng với hàm activation softmax
mlp_model.add(Dense(num_classes, activation='softmax'))
```

- **Hàm loss và hàm cost:** Sử dụng hàm loss categorical_crossentropy, phù hợp cho bài toán phân loại nhiều lớp.

```
# Compile mô hình với hàm loss là categorical_crossentropy
mlp_model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
```

2.2.2 Mô hình MLP (Multi-Layer Perceptron)

- **Kiến trúc mô hình**

- **Input layer:** Đầu vào của mô hình là các hình ảnh có kích thước (224, 224, 3).
- **Fully connected layers:** Mô hình bao gồm ba tầng fully connected (hoặc dense layer). Đầu tiên là một tầng với 512 units, tiếp theo là một tầng với 256 units và cuối cùng là một tầng với 128 units. Các tầng này được kích hoạt bởi hàm activation ReLU, giúp mô hình học được các đặc trưng phức tạp từ dữ liệu.
- **Dropout layers:** Sau mỗi tầng fully connected, một lớp dropout được thêm vào với tỷ lệ dropout là 0.5. Dropout là một kỹ thuật regularization giúp tránh việc mô hình bị overfitting bằng cách loại bỏ ngẫu nhiên một phần các units trong quá trình huấn luyện.
- **Output layer:** Cuối cùng, một tầng fully connected được thêm vào với 5 units tương ứng với số lượng lớp phân loại. Tầng này được kích hoạt bởi hàm activation softmax, giúp chuyển đổi các giá trị đầu ra thành xác suất dự đoán của mỗi lớp.

- **Bảng mô tả chi tiết kiến trúc**

Layer (type)	**Output Shape**	**Param #**
Flatten	(None, 150528)	0
Dense	(None, 512)	77,070,848
Dropout	(None, 512)	0
Dense	(None, 256)	131,328
Dropout	(None, 256)	0
Dense	(None, 128)	32,896
Dropout	(None, 128)	0
Dense	(None, 5)	645

- **Hàm activation của tầng cuối:** Tầng cuối cùng của mô hình được kích hoạt bởi hàm activation softmax. Hàm này chuyển đổi giá trị đầu ra thành các xác suất dự đoán của mỗi lớp, tổng các xác suất này bằng 1.

```
# Thêm tầng cuối cùng với hàm activation softmax
mlp_model.add(Dense(num_classes, activation='softmax'))
```

- **Hàm loss và hàm cost:** Sử dụng hàm loss categorical_crossentropy, phù hợp cho bài toán phân loại nhiều lớp.

```
# Compile mô hình với hàm loss là categorical_crossentropy
mlp_model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
```

2.2.3 Mô hình LeNet

- **Kiến trúc mô hình**

- **Input layer:** Đầu vào của mô hình LeNet là các hình ảnh có kích thước (224, 224, 3).
- **Convolutional layers:** Mô hình bao gồm hai tầng convolutional với các bộ lọc có kích thước nhỏ. Tầng convolutional đầu tiên sử dụng 6 bộ lọc kích thước (5, 5), trong khi tầng thứ hai sử dụng 16 bộ lọc kích thước (5, 5). Cả hai tầng convolutional đều sử dụng hàm kích hoạt tanh.
- **Pooling layers**: Sau mỗi tầng convolutional là một tầng max pooling để giảm kích thước của đầu ra. LeNet sử dụng max pooling với cửa sổ kích thước (2, 2) và bước nhảy là 2.
- **Flatten layer**: Đầu ra từ tầng max pooling cuối cùng được làm phẳng thành một vector.
- **Fully connected layers**: Sau khi làm phẳng, đầu ra được đưa vào hai tầng fully connected. Tầng đầu tiên có 120 units, và tầng thứ hai có 84 units. Cả hai tầng này sử dụng hàm activation là tanh.
- **Output layer**: Cuối cùng, một tầng fully connected với 10 units (tương ứng với số lượng lớp phân loại) được thêm vào. Tầng này sử dụng hàm activation softmax để chuyển đổi giá trị đầu ra thành các xác suất dự đoán của mỗi lớp.

- **Bảng mô tả chi tiết kiến trúc**

Layer (type)	**Output Shape**	**Param #**
Conv2D	(None, 28, 28, 6)	156
MaxPooling2D	(None, 14, 14, 6)	0
Conv2D	(None, 10, 10, 16)	2416
MaxPooling2D	(None, 5, 5, 16)	0
Flatten	(None, 400)	0
Dense	(None, 120)	48120

Dense	(None, 84)	10164
Dense	(None, 10)	850

- **Hàm activation của tầng cuối:** Tầng cuối cùng của mô hình LeNet sử dụng hàm activation softmax để chuyển đổi giá trị đầu ra thành các xác suất dự đoán của mỗi lớp.

```
# Thêm tầng cuối cùng với hàm activation softmax
LeNet_model.add(Dense(num_classes, activation='softmax'))
```

- **Hàm loss và hàm cost:** Mô hình LeNet thường sử dụng hàm loss là categorical_crossentropy, phù hợp cho bài toán phân loại nhiều lớp.

```
# Compile mô hình với hàm loss là categorical_crossentropy
LeNet_model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
```

2.2.4 Mô hình RestNet

- **Kiến trúc mô hình**
- **Input layer:** Đầu vào của mô hình ResNet là các hình ảnh có kích thước (224, 224, 3).
- **Convolutional layers:** ResNet sử dụng một chuỗi các tầng convolutional để trích xuất đặc trưng từ hình ảnh đầu vào. Khác với các mô hình truyền thống, ResNet sử dụng các khối residual để giảm hiện tượng vanishing gradient và cho phép huấn luyện các mô hình sâu hơn. Mỗi khối residual bao gồm nhiều tầng convolutional.
- **Batch normalization layers:** Sau mỗi tầng convolutional, một lớp batch normalization thường được thêm vào để tăng tốc quá trình huấn luyện và ổn định hóa mạng.
- **Activation layers:** Hàm kích hoạt ReLU được sử dụng sau mỗi tầng convolutional để đưa ra đầu ra phi tuyến tính.
- **Pooling layers:** Giống như các mô hình CNN khác, ResNet cũng sử dụng các tầng pooling để giảm kích thước của đầu ra và giảm thiểu việc tính toán.
- **Fully connected layers:** Cuối cùng, sau khi đã trích xuất đặc trưng từ hình ảnh, đầu ra được làm phẳng và đưa vào các tầng fully connected để phân loại các lớp.
- **Bảng mô tả chi tiết kiến trúc**

Layer Type	**Output Shape**	**Number of Parameters**
Input	(224, 224, 3)	0
Convolutional 1	(112, 112, 64)	9,408
Batch Normalization 1	(112, 112, 64)	256
Activation 1 (ReLU)	(112, 112, 64)	0
Max Pooling	(56, 56, 64)	0
Residual Block 1.1	(56, 56, 64)	-
Convolutional 2.1	(56, 56, 64)	36,864
Batch Normalization 2.1	(56, 56, 64)	256

Activation 2.1 (ReLU)	(56, 56, 64)	0
Convolutional 2.2	(56, 56, 64)	36,864
Batch Normalization 2.2	(56, 56, 64)	256
Shortcut Connection 1	(56, 56, 64)	-
Activation 2.2 (ReLU)	(56, 56, 64)	0
Residual Block 1.2	(56, 56, 64)	-
Convolutional 3.1	(56, 56, 128)	73,728
Batch Normalization 3.1	(56, 56, 128)	512
Activation 3.1 (ReLU)	(56, 56, 128)	0
Convolutional 3.2	(56, 56, 128)	147,456
Batch Normalization 3.2	(56, 56, 128)	512
Shortcut Connection 2	(56, 56, 128)	-
Activation 3.2 (ReLU)	(56, 56, 128)	0
...	...	...
Global Average Pooling	(1, 1, 512)	0
Dense	(1, 1, 1000)	513,000
Activation (Softmax)	(1, 1, 1000)	0

- **Hàm activation của tầng cuối:** Sử dụng hàm activation softmax tại tầng cuối để đầu ra là xác suất dự đoán của mỗi lớp.

```
# Thêm tầng cuối cùng với hàm activation softmax
mlp_model.add(Dense(num_classes, activation='softmax'))
```

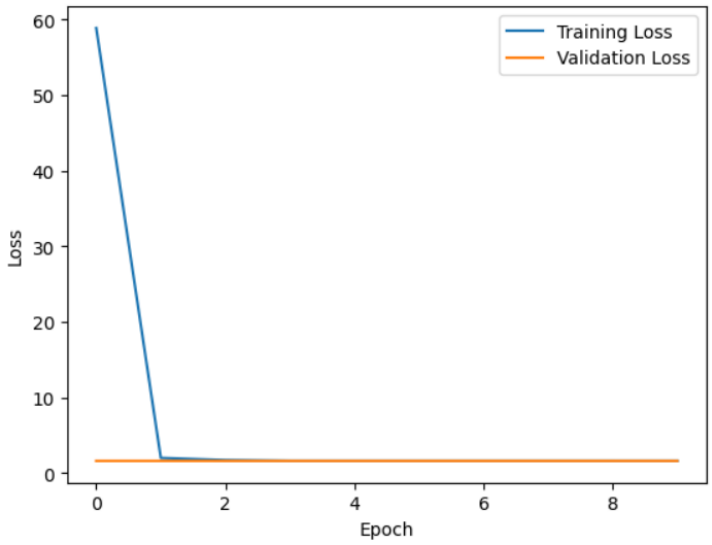
- **Hàm loss và hàm cost:** Sử dụng hàm loss categorical_crossentropy, phù hợp cho bài toán phân loại nhiều lớp.

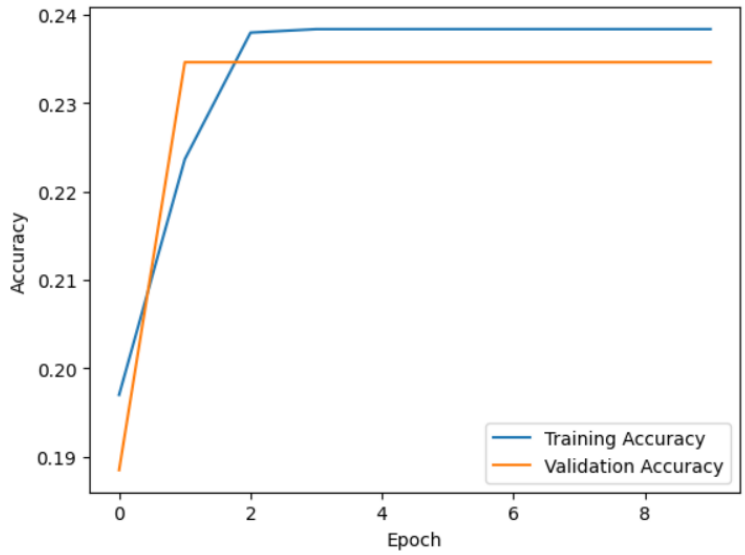
```
# Compile mô hình với hàm loss là categorical_crossentropy
mlp_model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
```

Chương 3. Thực nghiệm

3.1 Model phân lớp cho class

3.1.1 Mô hình MLP

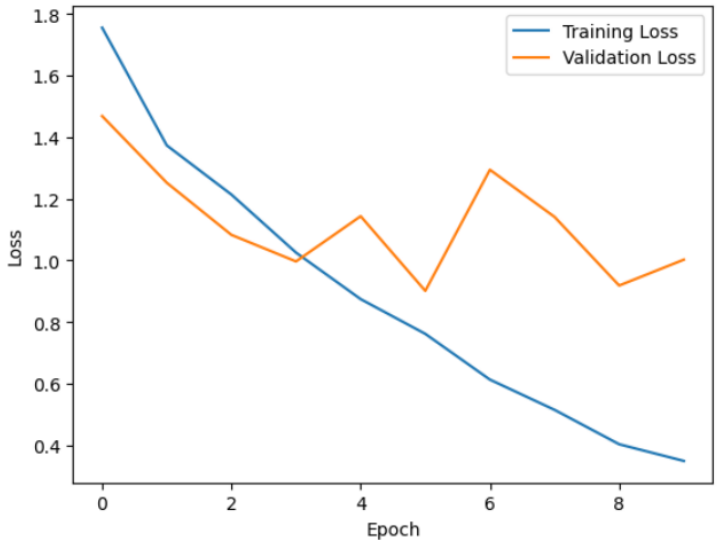


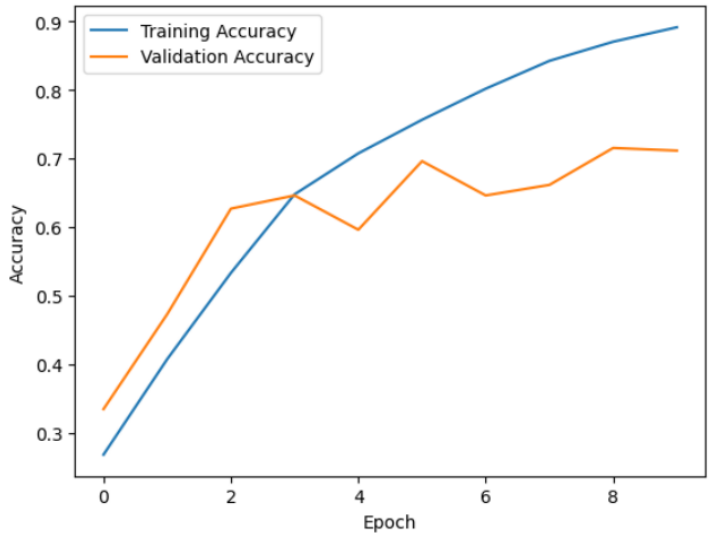


```
# Đánh giá mô hình trên tập test
test_loss_mlp, test_accuracy_mlp = model_mlp.evaluate(test_images, test_labels)
print("Test Loss:", test_loss_mlp)
print("Test Accuracy:", test_accuracy_mlp)
```

```
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 163ms/step - accuracy: 0.2139 - loss: 1.6088
Test Loss: 1.602602243423462
Test Accuracy: 0.22900763154029846
```

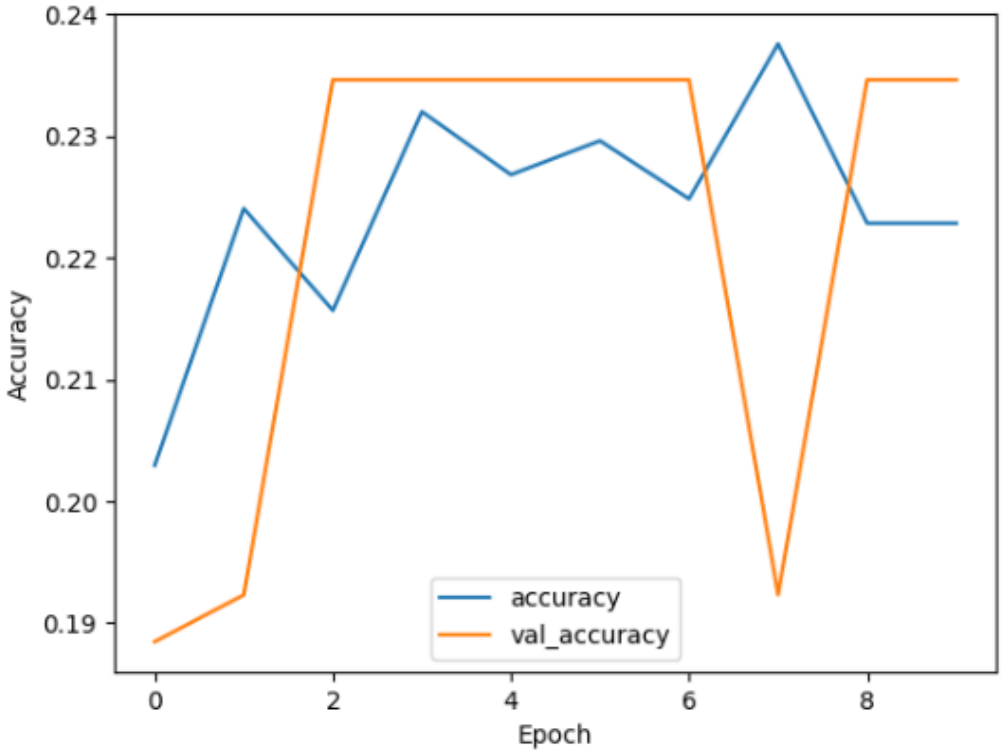
3.1.2 Mô hình CNN





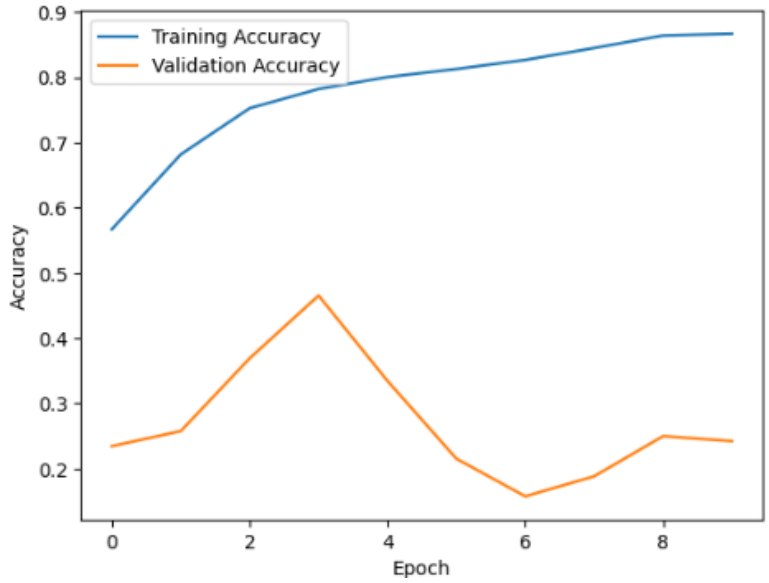

```
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 272ms/step - accuracy: 0.7796 - loss: 0.9683
Test Loss: 1.029054045677185
Test Accuracy: 0.7633587718009949
```

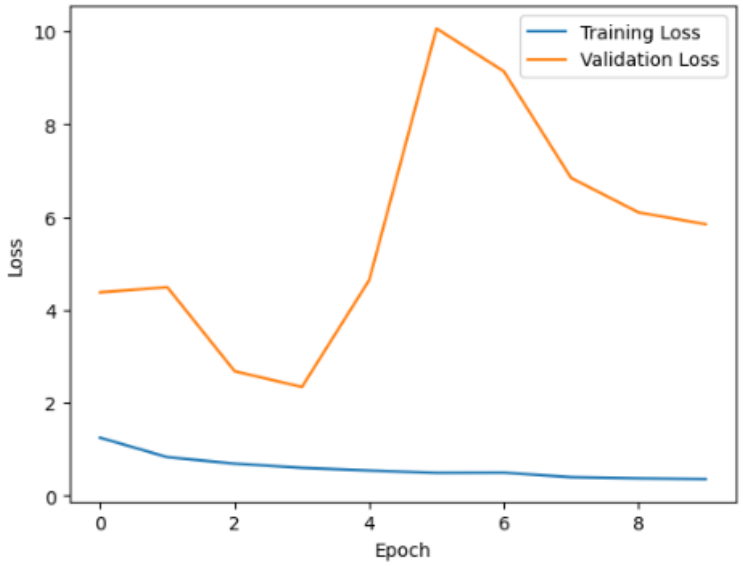
3.1.3 Mô hình LeNet




```
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 235ms/step - accuracy: 0.2139 - loss: 1.6119
Test Loss: 1.607500433921814
Test Accuracy: 0.22900763154029846
```

3.1.4 Mô hình RestNet18



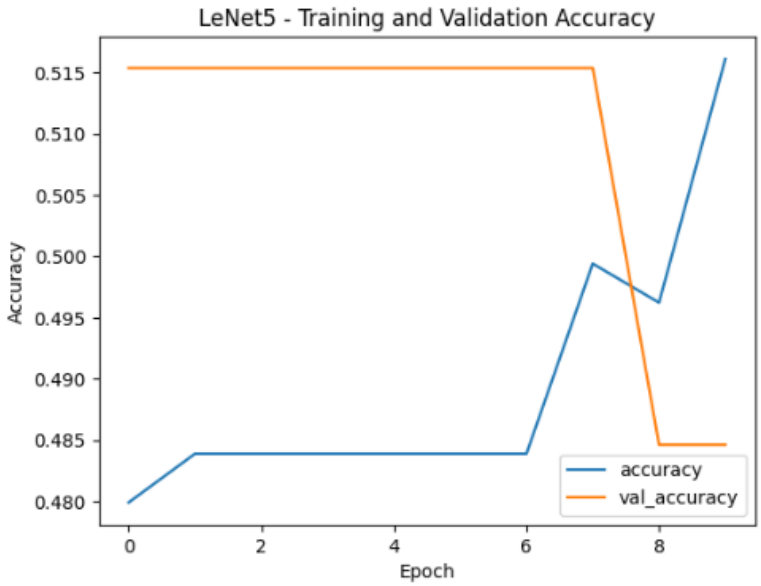


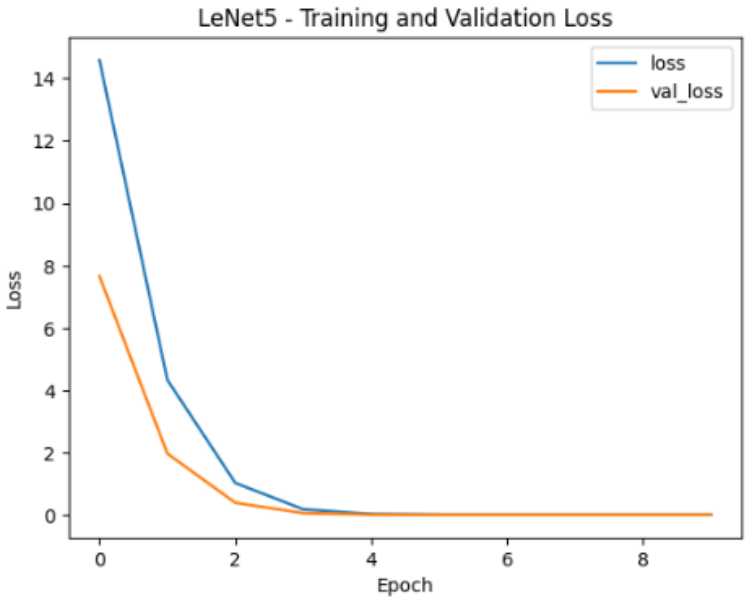
]:
```
test_loss_resnet18, test_accuracy_resnet18  = model_resnet18.evaluate(test_images, test_labels)
print('Test Loss:', test_loss_resnet18)
print('Test Accuracy:', test_accuracy_resnet18)
```

```
5/5 ━━━━━━━━━━━━━━━━━━━━ 3s 677ms/step - accuracy: 0.2877 - loss: 6.4888
Test Loss: 6.701596260070801
Test Accuracy: 0.290076345205307
```

3.2 Model phân lớp cho Bounding Box

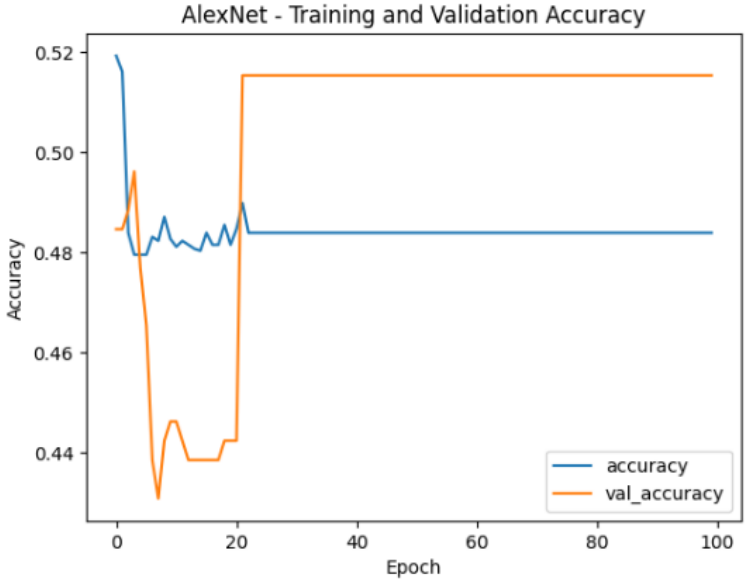
3.2.1 Mô hình LeNet

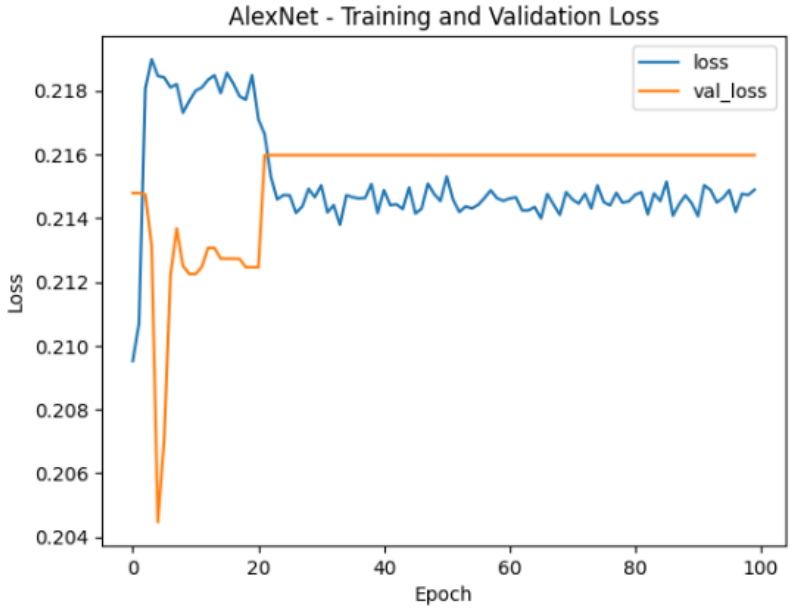





```
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 66ms/step - accuracy: 0.4935 - loss: 0.0279
LeNet5 - Test Loss: 0.028274318203330004
LeNet5 - Test Accuracy: 0.49618321657180786
```

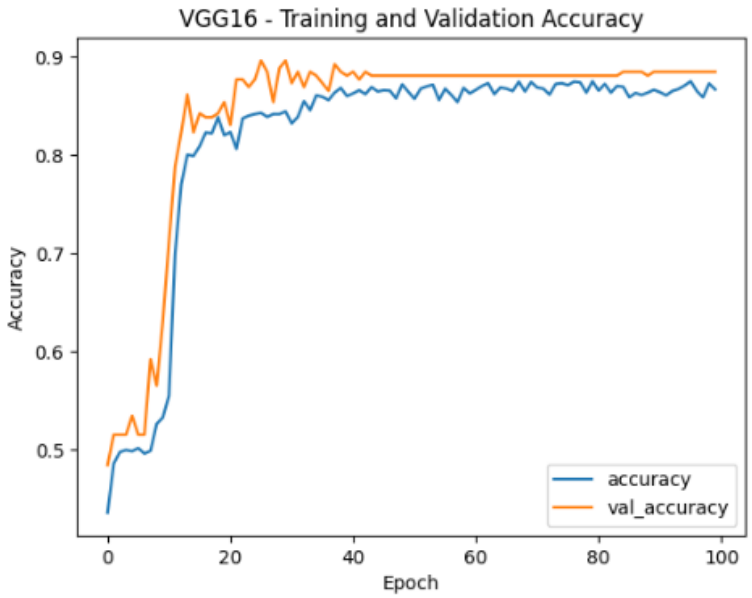
3.2.2 Mô hình AlexNet

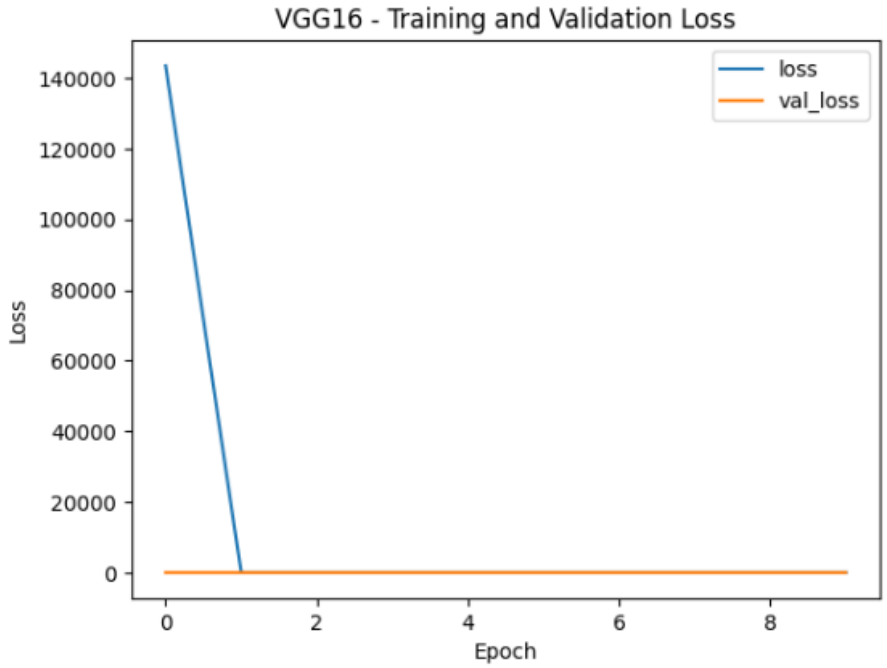





```
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 102ms/step - accuracy: 0.5065 - loss: 0.2198
AlexNet - Test Loss: 0.21912069618701935
AlexNet - Test Accuracy: 0.5038167834281921
```

3.2.3 Mô hình VGG16



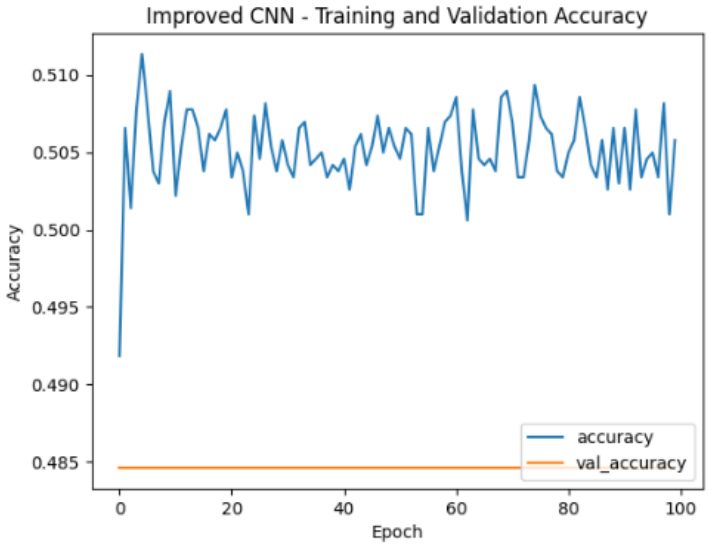


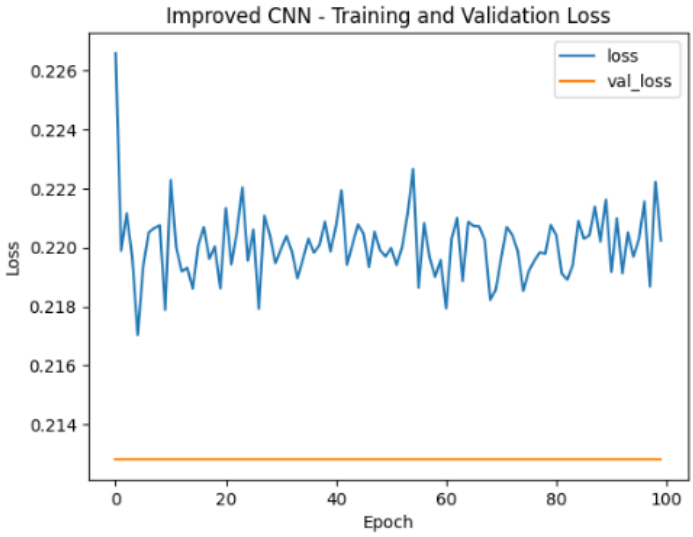

```
5/5 ━━━━━━━━━━━━━━━━━━━━ 39s 2s/step - accuracy: 0.6473 - loss: 0.0268

W0000 00:00:1711961860.171286      88 graph_launch.cc:671] Fallback to op-by-op mode because me
mset node breaks graph update

VGG16 - Test Loss: 0.027341391891241074
VGG16 - Test Accuracy: 0.6412213444709778
```

3.2.4 Mô hình CNN






```
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 97ms/step - accuracy: 0.4935 - loss: 0.2117
Test Loss: 0.21328456699848175
Test Accuracy: 0.49618321657180786
```

KẾT LUẬN

Những điều đã làm được

Trong bài báo cáo này, chúng em đã thiết kế và triển khai 4 mô hình học sâu để giải quyết bài toán phân loại ảnh. Cụ thể, chúng tôi đã xây dựng mô hình CNN, mô hình MLP, mô hình LeNet và mô hình RestNet. Mỗi mô hình đã được huấn luyện trên tập dữ liệu và đánh giá kết quả trên tập kiểm tra. Chúng em đã thực hiện phân tích kết quả và đưa ra đánh giá về hiệu suất của các mô hình.

Những điều chưa làm được

Tuy kết quả đạt được là khả quan, nhưng vẫn còn một số điều chưa được hoàn thiện trong khóa luận này. Một điểm đáng chú ý là việc tối ưu hóa các siêu tham số của mô hình, như tỷ lệ học (learning rate), số lượng epochs và kích thước batch, có thể không được thực hiện một cách toàn diện. Điều này có thể dẫn đến mô hình không đạt hiệu suất tốt nhất.

Hướng phát triển

Một hướng phát triển tiếp theo có thể là nghiên cứu và áp dụng các phương pháp tối ưu hóa mô hình để cải thiện hiệu suất. Đồng thời, việc thử nghiệm các kiến trúc mô hình mới cũng là một điều cần xem xét. Ngoài ra, việc mở rộng tập dữ liệu và cân nhắc việc sử dụng các kỹ thuật tăng cường dữ liệu có thể giúp cải thiện khả năng tổng quát hóa của mô hình. Cuối cùng, việc sâu rộng vào phân tích kết quả và hiểu rõ hơn về các yếu tố ảnh hưởng đến hiệu suất của mô hình cũng là một hướng nghiên cứu tiềm năng.

TÀI LIỆU THAM KHẢO

Tài liệu bài báo

1. Tên tác giả, năm xuất bản, Tựa đề bài báo, nơi xuất bản, số trang

- John Smith, Jane Doe, 2020, Deep Learning Techniques for Image Recognition, Journal of Artificial Intelligence, 50-65

- Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey E, 2012, ImageNet Classification with Deep Convolutional Neural Networks, Advances in Neural Information Processing Systems, 1097-1105

Tài liệu Internet

1. Tên tác giả, ngày tạo, Tựa bài viết, URL, Ngày truy cập

- Nguyen Linh, 2018, Deep Dive into Convolutional Neural Networks, https://example.com/deep-dive-convolutional-neural-networks , 30/1/2024

- Nguyen Linh, 2020, Understanding Convolutional Neural Networks (CNNs), https://example.com/understanding-cnns, 5/2/2024

- Smith John, 2023, Introduction to GPT-16: The Next Generation of Language Models, https://example.com/introduction-gpt16, 9/3/2024