

BỘ CÔNG THƯƠNG  
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP. HỒ CHÍ MINH  
KHOA CÔNG NGHỆ THÔNG TIN

---□&□---



MÔN HỌC :  
**HỌC SÂU**

Đề tài : Xây dựng hệ thống nhận diện ngôn ngữ ký hiệu  
tay

<i>Giảng viên hướng dẫn</i>	: Đặng Thị Phúc
<i>Nhóm thực hiện</i>	: Nhóm 32
<i>Lớp</i>	: DHKHD17A

*TP.HCM, ngày 2 tháng 12 năm 2024*

## MỤC LỤC

DANH SÁCH THÀNH VIÊN .....	<b>Error! Bookmark not defined.</b>
I. Giới thiệu bài toán .....	3
II. Xây dựng hệ thống nhận diện .....	3
1. Bộ dữ liệu (Dataset) sử dụng .....	3
2. Tổng quan hệ thống và các thư viện hỗ trợ .....	3
3. Xây dựng hệ thống .....	4
3.1 Chuẩn bị và tiền xử lý dữ liệu .....	4
3.2 Xây dựng mô hình .....	6
3.2.1 ResNet50 .....	6
3.2.2 EfficientNet .....	7
3.2.3 Inception V3 .....	8
3.3 Huấn luyện mô hình: .....	10
3.3.1 Huấn luyện mô hình ResNet50 .....	10
3.3.2 Huấn luyện mô hình EfficientNet .....	10
3.3.3 Huấn luyện mô hình InceptionV3 .....	10
3.4 Fine-Tuning mô hình: .....	11
III. Kết quả .....	12
1. ResNet50 .....	12
2. EfficientNet .....	12
3. InceptionV3 .....	13
IV. Kết luận: .....	16

## I. Giới thiệu bài toán

Người khuyết tật nghe và nói thường gặp rất nhiều thách thức trong giao tiếp và tham gia vào xã hội. Theo Tổ chức Y tế Thế giới (WHO), có khoảng 466 triệu người trên toàn thế giới có mức độ khiếm thính có vấn đề về thính giác. Trong đó, có khoảng 34 triệu trẻ em dưới 15 tuổi. Người khuyết tật nghe và nói thường gặp rất nhiều thách thức trong giao tiếp và tham gia vào xã hội. Họ thường cần phải học cách sử dụng các phương tiện khác nhau để giao tiếp, chẳng hạn như ngôn ngữ ký hiệu.

Ngôn ngữ ký hiệu (sign language) là hệ thống giao tiếp bằng cách sử dụng các cử chỉ tay, các biểu hiện mặt và các cử chỉ khác của cơ thể để truyền đạt ý nghĩa và thông tin. Việc hỗ trợ và tạo điều kiện thuận lợi cho người khuyết tật nghe và nói là rất quan trọng để giúp họ tham gia đầy đủ vào xã hội và học tập.

Do đó, việc xây dựng ứng dụng hoặc công nghệ hỗ trợ cho ngôn ngữ ký hiệu có thể đóng một vai trò quan trọng trong việc cải thiện việc giao tiếp và tương tác giữa người khiếm thính và người nghe, giúp họ tiếp cận thông tin và cơ hội như những người khác trong xã hội, bởi vì họ có thể giao tiếp dễ dàng với ngay cả những người không hiểu ngôn ngữ ký hiệu. Ngôn ngữ ký hiệu không phụ thuộc vào ngôn ngữ nói của bất kỳ quốc gia nào, mà có thể tồn tại riêng và khác nhau ở mỗi vùng lãnh thổ. Ngôn ngữ ký hiệu có thể thực hiện bằng 1 tay hoặc cả 2 tay. Có 2 dạng chính đó là: Isolated sign language (ISL) và continuous sign language (CSL). ISL bao gồm cử chỉ đơn lẻ biểu thị nghĩa của 1 từ đơn, trong khi đó CSL là 1 chuỗi các cử chỉ để biểu thị 1 câu hoàn chỉnh.

Trong bài báo cáo này, nhóm sẽ xây dựng 1 hệ thống sử dụng học sâu cho phép nhận diện ISL ASL thông qua camera.

## II. Xây dựng hệ thống nhận diện

### 1. Bộ dữ liệu (Dataset) sử dụng

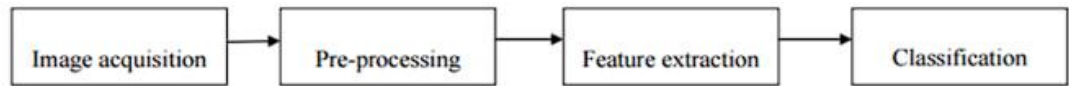
Bộ dữ liệu là tập hợp các hình ảnh về bảng chữ cái từ Ngôn ngữ ký hiệu của Mỹ (ASL), được lưu thành 29 thư mục đại diện cho các ký tự khác nhau.

- Tập training bao gồm 43500 hình ảnh, mỗi ảnh có kích thước 200x200 pixels. Có 29 lớp trong đó 26 lớp là các chữ cái A-Z và 3 lớp còn lại là SPACE, DELETE, NOTHING. 3 lớp này thì hữu ích cho các ứng dụng thời gian thực(real time)

- Tập test bao gồm 29 hình ảnh thuộc từng lớp khác nhau

### 2. Tổng quan hệ thống và các thư viện hỗ trợ

Quá trình xây dựng hệ thống nhận diện ngôn ngữ ký hiệu được biểu diễn qua sơ đồ sau:

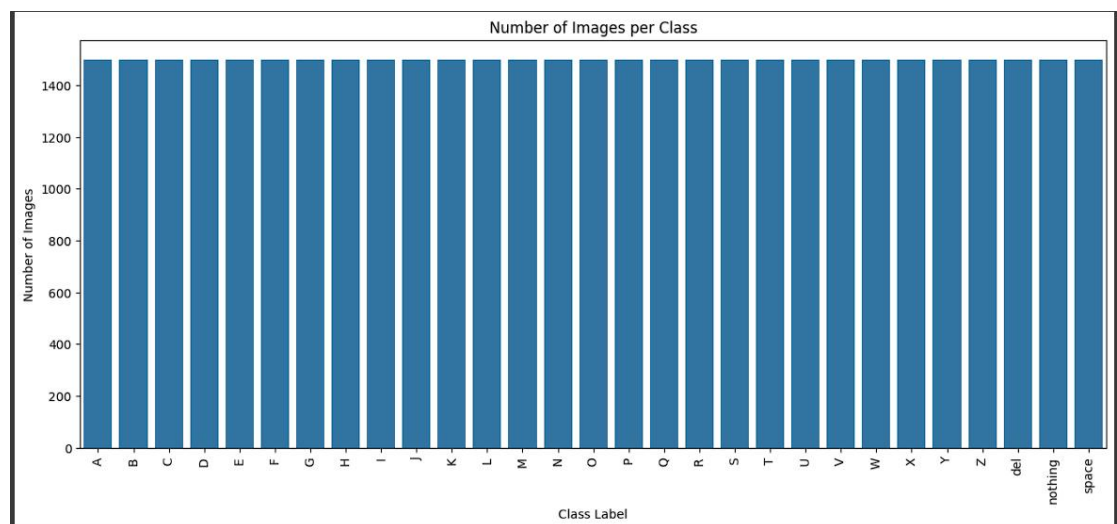


Các thư viện được dùng để xây dựng mô hình dự đoán: Tensorflow, OpenCV2.

### 3. Xây dựng hệ thống

#### 3.1 Chuẩn bị và tiền xử lý dữ liệu

Mỗi nhãn được lưu trong 1 thư mục, và có 1500 ảnh mỗi thư mục. Do các hình ảnh trong mỗi thư mục có 1 phần hình ảnh tương đối giống nhau nên ta sẽ thực hiện kỹ thuật Tăng cường dữ liệu hình ảnh (Data Augmentation) để tăng khả năng tổng quát của mô hình và giảm hiện tượng overfitting.



*Số lượng ảnh có trong mỗi thư mục*



*Fig. Các hình ảnh tương đối giống nhau*

Ta thực hiện việc tăng cường dữ liệu hình ảnh và chuẩn hóa dữ liệu trước khi đưa vào huấn luyện mô hình học sâu bằng lớp ImageDataGenerator trong TensorFlow.

Một số chức năng chính của ImageDataGenerator trong TensorFlow:

- **Tăng cường dữ liệu hình ảnh:** biến đổi ngẫu nhiên vào dữ liệu hình ảnh trong quá trình huấn luyện, bao gồm xoay, thu phóng, lật, cắt, chỉnh sáng, tăng cường màu sắc,.... Điều này giúp tăng độ đa dạng của dữ liệu huấn luyện mà không cần thu thập thêm dữ liệu.

- **Chuẩn bị dữ liệu cho huấn luyện:** hỗ trợ chia dữ liệu thành các batch nhỏ và đưa vào mô hình trong quá trình huấn luyện.

- **Chuẩn hóa dữ liệu:** chuẩn hóa giá trị pixel của hình ảnh thành khoảng (0, 1) để giúp quá trình huấn luyện diễn ra hiệu quả hơn.

Sau đây 1 số cấu hình (config) được sử dụng để tiền xử lý:

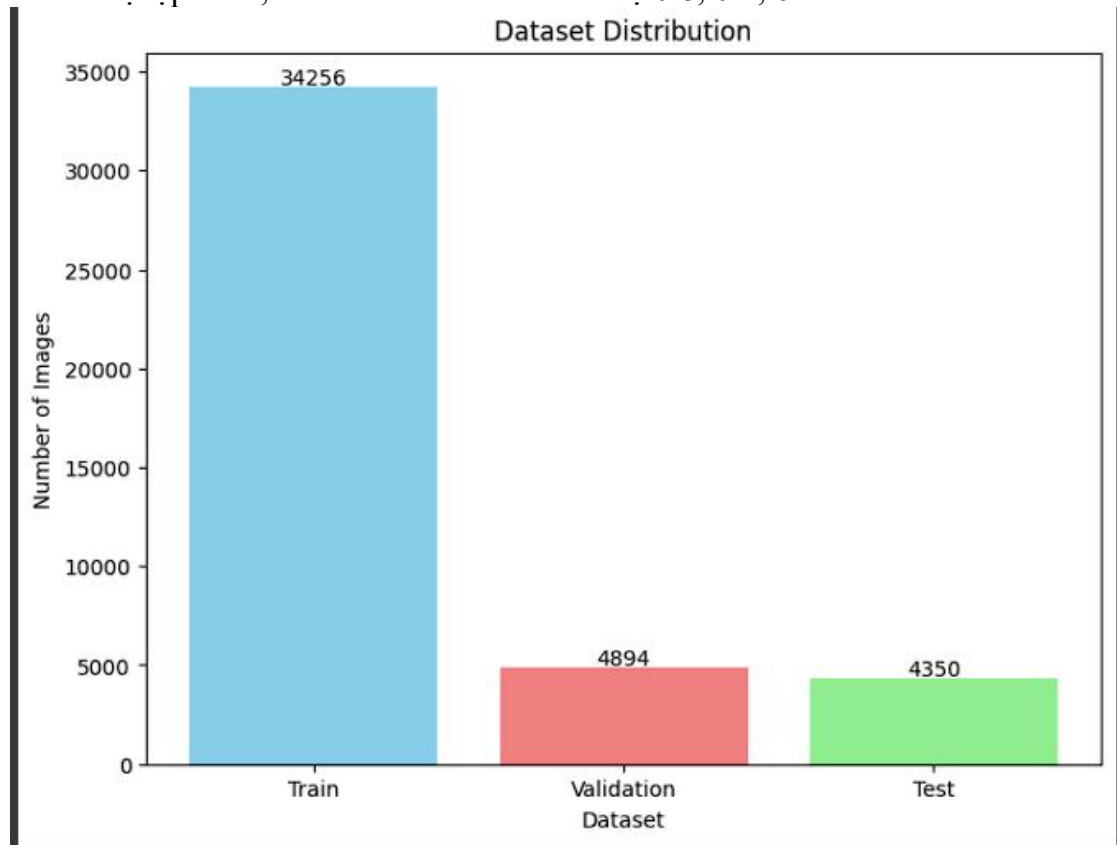
```
train_gen = ImageDataGenerator(
    rescale=1/255.,
    brightness_range=[0.8,1.2],
    zoom_range=[1.0,1.2],
    horizontal_flip=True)
```

*Fig. Config ImageDataGenerator*

Kích thước đầu vào của mô hình là 200\*200\*3 với mỗi batch = 64

- **brightness\_range:** tăng chỉnh ngẫu nhiên độ sáng của ảnh
- **zoom\_range:** thực hiện biến đổi zoom trên hình ảnh

- **horizontal\_flip=True**: Dữ liệu hình ảnh sẽ được lật ngang ngẫu nhiên
- Chia tỉ lệ tập train, validation và test theo tỉ lệ 0.8, 0.1, 0.1.



*Số lượng dữ liệu trong từng tập Train, Valid, Test*

Sử dụng cách lấy mẫu phân tầng (stratify): đảm bảo tỷ lệ phân phối của các lớp trong mẫu con giữ nguyên tỷ lệ phân phối của các lớp trong toàn bộ dữ liệu

## 3.2 Xây dựng mô hình

### 3.2.1 ResNet50

Mạng ResNet-50 là một trong những kiến trúc mạng nơ-ron tích chập (CNN) nổi tiếng, được giới thiệu bởi nhóm nghiên cứu của Microsoft Research trong bài báo "Deep Residual Learning for Image Recognition" tại hội nghị CVPR năm 2016. ResNet-50 đã đạt được thành tích xuất sắc tại cuộc thi ImageNet Large Scale Visual Recognition Challenge (ILSVRC), thiết lập tiêu chuẩn mới cho các mô hình mạng sâu trong lĩnh vực trí tuệ nhân tạo và thị giác máy tính.

ResNet-50 thuộc dòng ResNet (Residual Network), nổi bật với việc sử dụng kiến trúc Residual Block - một cải tiến giúp giảm vấn đề suy giảm gradient (vanishing gradient) thường gặp khi huấn luyện các mạng rất sâu. Mạng ResNet-50 bao gồm 50 lớp, với các lớp chính là tích chập, pooling, batch normalization, và các lớp fully connected.

```

base_model_res = ResNet50(
    input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3),
    include_top=False,
    weights='imagenet'
)
for layer in base_model_res.layers:
    layer.trainable = False
x = base_model_res.output
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(1024, activation='relu')(x)
predictions = layers.Dense(29, activation='softmax')(x)

model_res = Model(inputs=base_model_res.input, outputs=predictions)

model_res.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

```

*Khởi tạo ResNet50 Model*

### 3.2.2 EfficientNet

Mạng EfficientNet là một trong những kiến trúc mạng nơ-ron tiên tiến và nổi tiếng trong lĩnh vực trí tuệ nhân tạo và thị giác máy tính. Nó được giới thiệu bởi nhóm nghiên cứu của Google AI vào năm 2019 trong bài báo "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". EfficientNet đã nhanh chóng trở thành một trong những mô hình hàng đầu nhờ khả năng cân bằng giữa độ chính xác và hiệu suất tính toán, vượt trội trên nhiều bài kiểm tra phân loại hình ảnh như ImageNet.

Mạng này sử dụng kiến trúc CNN sâu với nhiều lớp tích chập và lớp kết nối đầy đủ. Điểm đặc biệt của EfficientNet nằm ở phương pháp Compound Scaling, cho phép mở rộng mạng một cách tối ưu bằng cách đồng thời tăng chiều sâu (depth), chiều rộng (width) và độ phân giải ảnh đầu vào (resolution). Cách tiếp cận này giúp mạng tăng cường khả năng học đặc trưng mà không làm tăng quá nhiều chi phí tính toán.

EfficientNet được xây dựng dựa trên kiến trúc gốc MobileNetV3, trong đó các lớp tích chập được tổ chức một cách hiệu quả. Mạng bắt đầu với các lớp tích chập đơn giản và mở rộng theo cách tối ưu hóa hiệu suất nhờ việc kết hợp các phép tích chập  $1 \times 1$ ,  $11 \times 1$  và  $3 \times 3$ , cùng các kỹ thuật như Swish Activation và Squeeze-and-Excitation. Kết quả là mạng học được các đặc trưng đa cấp độ một cách hiệu quả, từ các chi tiết nhỏ đến các đặc trưng tổng quát.



```

base_model_eff = EfficientNetB0(
    input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3),
    include_top=False,
    weights='imagenet'
)
for layer in base_model_eff.layers:
    layer.trainable = False
x = base_model_eff.output
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dropout(0.5)(x)
x = layers.Dense(512, activation='relu')(x)
x = layers.Dropout(0.5)(x)
predictions = layers.Dense(29, activation='softmax')(x)

# Tạo mô hình hoàn chỉnh
model_eff = Model(inputs=base_model_eff.input, outputs=predictions)
model_eff = Model(inputs=base_model_eff.input, outputs=predictions)

model_eff.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

```

*Khởi tạo EfficientNetB0 Model*

### 3.2.3 Inception V3

Mạng Inception (hay còn gọi là GoogLeNet) là một trong những kiến trúc mạng nơ-ron nổi tiếng trong lĩnh vực trí tuệ nhân tạo và thị giác máy tính. Nó được giới thiệu bởi nhóm nghiên cứu của Google Research trong cuộc thi ImageNet Large Scale Visual Recognition Challenge (ILSVRC) năm 2014 và đạt được kết quả xuất sắc, giành giải thứ nhất trong các bài kiểm tra phân loại hình ảnh.

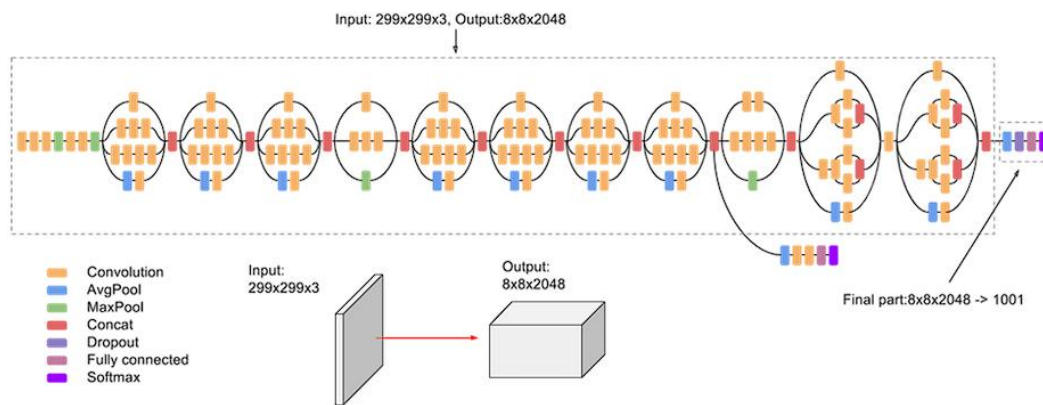
Mạng này sử dụng kiến trúc mạng CNN sâu với nhiều lớp tích chập và lớp kết nối đầy đủ. Tuy nhiên, điểm đặc biệt của Inception là việc sử dụng các module inception, cho phép mạng học các đặc trưng tại nhiều mức khác nhau của hình ảnh một cách hiệu quả.

Module inception bao gồm nhiều lớp tích chập với các kích thước bộ lọc khác nhau (1x1, 3x3, 5x5) và lớp pooling (max-pooling) song song nhau. Việc kết hợp các lớp tích chập và pooling này giúp mạng học được các đặc trưng cấu trúc và không cấu trúc của hình ảnh một cách toàn diện và hiệu quả. Sau đó, các đặc trưng này được kết hợp lại với nhau và đưa vào lớp fully connected để thực hiện phân loại.

Inception v3 là phiên bản tiếp theo của GoogLeNet, được giới thiệu vào năm 2015. Mạng này đã thực hiện nhiều cải tiến để tăng cường độ chính xác, bao gồm việc sử dụng kỹ thuật "Factorization into small convolutions" và "Auxiliary Classifiers" giúp giảm vấn đề vanishing gradient và cải thiện quá trình huấn luyện. Mô hình Inception V3 có 42 lớp, nhiều hơn Inception V1 và V2.

Sử dụng mô hình **Inception v3** từ thư viện Tensor Flow





*Kiến trúc tổng quan của Inception V3*

TYPE	PATCH / STRIDE SIZE	INPUT SIZE
Conv	3×3/2	299×299×3
Conv	3×3/1	149×149×32
Conv padded	3×3/1	147×147×32
Pool	3×3/2	147×147×64
Conv	3×3/1	73×73×64
Conv	3×3/2	71×71×80
Conv	3×3/1	35×35×192
3 × Inception	Module 1	35×35×288
5 × Inception	Module 2	17×17×768
2 × Inception	Module 3	8×8×1280
Pool	8 × 8	8 × 8 × 2048
Linear	Logits	1 × 1 × 2048
Softmax	Classifier	1 × 1 × 1000

*Chi tiết về các lớp trong Inception V3*

Hệ thống nhận diện của nhóm sẽ sử dụng kỹ thuật học tái sử dụng (transfer learning) mô hình Inception đã được huấn luyện trước (pre-train) trên bộ dữ liệu của ImageNet (bao gồm 1000 nhãn), sau đó thêm các lớp fully connected ở các lớp cuối cùng để phù hợp với số lượng lớp của bài toán nhận diện.

```
base_model = InceptionV3(
    input_shape = (IMAGE_SIZE, IMAGE_SIZE, 3),
    include_top = False,
    weights = 'imagenet'
)
```

*Khởi tạo InceptionV3 Model*

Đóng băng (freeze) một số lớp đầu tiên trong mô hình, sau đó thực hiện thêm các lớp fully connected ở cuối mạng Inception V3 phục vụ cho việc phân loại:

```
for layer in base_model.layers:
    layer.trainable = False
x = base_model.output
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(1024, activation='relu')(x)
predictions = layers.Dense(29, activation='softmax')(x)
```

### 3.3 Huấn luyện mô hình:

#### 3.3.1 Huấn luyện mô hình ResNet50

**Tổng quan về số lượng tham số của mô hình ResNet50** (trainable parameter và non-trainable parameter) :

```
Total params: 25,715,613 (98.10 MB)
Trainable params: 2,127,901 (8.12 MB)
Non-trainable params: 23,587,712 (89.98 MB)
```

#### 3.3.2 Huấn luyện mô hình EfficientNet

**Tổng quan về số lượng tham số của mô hình EfficientNet**(trainable parameter và non-trainable parameter) :

```
Total params: 4,720,320 (18.01 MB)
Trainable params: 670,749 (2.56 MB)
Non-trainable params: 4,049,571 (15.45 MB)
```

#### 3.3.3 Huấn luyện mô hình InceptionV3

Sau khi định nghĩa mô hình, ta bắt đầu cấu hình mô hình trước khi bắt đầu quá trình huấn luyện.

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

Giải thích qua 1 số tham số:

- **optimizer:** Tham số này xác định thuật toán tối ưu hóa được sử dụng trong quá trình huấn luyện mô hình. sử dụng `optimizer='adam'`, đây là một thuật toán tối ưu hóa sử dụng giới hạn tỉ lệ học (learning rate) thích ứng cho từng tham số.
- **loss:** Là hàm mất mát (loss function) được sử dụng để đo lường mức độ sai lệch giữa dự đoán và nhãn thực tế của mô hình. sử dụng hàm mất mát "`categorical_crossentropy`", phù hợp với bài toán phân loại đa lớp (multi-class classification).
- **metrics:** Là danh sách các độ đo (metrics) sẽ được sử dụng để đánh giá hiệu suất của mô hình trong quá trình huấn luyện. Ta sử dụng độ đo "`acc`" (accuracy), tức là tỷ lệ dự đoán chính xác của mô hình trên tập dữ liệu huấn luyện.

**Tổng quan về số lượng tham số** (trainable parameter và non-trainable parameter) :

```
Total params: 23,930,685 (91.29 MB)
Trainable params: 2,127,901 (8.12 MB)
Non-trainable params: 21,802,784 (83.17 MB)
```

Cấu hình 1 số tham số khác trong quá trình training:

**BATCH\_SIZE** = 64 : đại diện cho số lượng mẫu dữ liệu được xử lý cùng một lúc trong mỗi lượt lặp trong quá trình huấn luyện

Sau khi cấu hình mô hình, bắt đầu quá trình huấn luyện bằng cách sử dụng phương thức `fit` của mô hình. Quá trình huấn luyện mô hình sử dụng 5 epoch (vòng lặp) trong quá trình huấn luyện. Mỗi epoch tương ứng với việc chạy qua toàn bộ tập dữ liệu huấn luyện một lần.

### 3.4 Fine-Tuning mô hình:

Vì sau khi Training top layers của 2 mô hình ResNet50 và EfficientNet cho ra kết quả không cao nên nhóm quyết định chỉ Fine-Tuning đối với InceptionV3.

Sau khi training top layers, ta bắt đầu fine-tune các lớp tích chập của mạng từ InceptionV3. Để phù hợp với bài toán, ta cho 2 inception blocks ở lớp trên cùng được huấn luyện và đóng băng (freeze) một số lớp đầu tiên trong mô hình, đặc biệt là các lớp tích chập sớm, để giữ cho các trọng số của chúng không bị thay đổi trong quá trình huấn luyện.

```
for layer in model.layers[:249]:
    layer.trainable = False
for layer in model.layers[249:]:
    layer.trainable = True
```

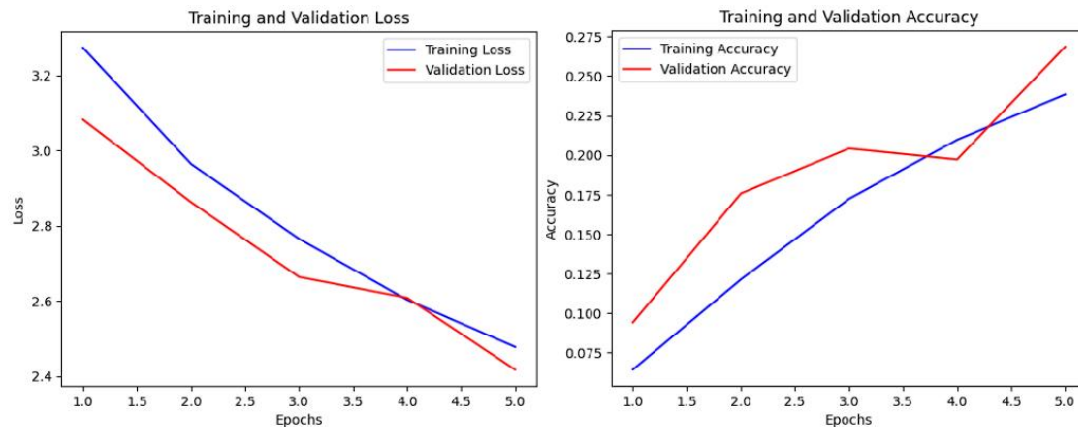
Recompile lại mô hình sử dụng SGD với tốc độ học nhỏ:

```
model.compile(optimizer=SGD(learning_rate=0.0001, momentum=0.9), loss='categorical_crossentropy', metrics=['accuracy'])
```

### III. Kết quả

#### 1. ResNet50

Training loss và accuracy trong quá trình huấn luyện của 5 epochs:



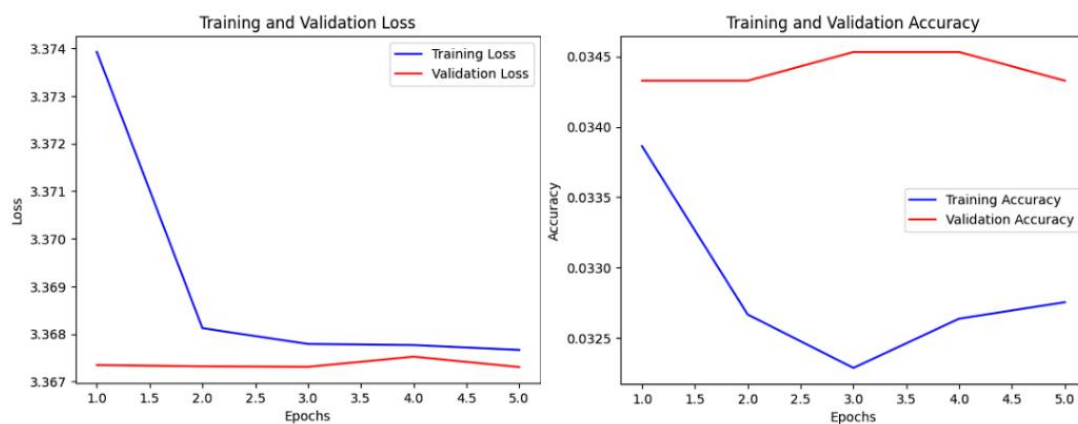
*Training Loss và Accuracy mô hình ResNet50*

Kết quả accuracy trên tập test chỉ đạt độ chính xác 27%:

```
/usr/local/lib/python3.10/dist-packages/keras/src/  
self._warn_if_super_not_called()  
4894/4894 ————— 42s 8ms/step  
Evaluate Test Accuracy: 0.27%
```

#### 2. EfficientNet

Training loss và accuracy trong quá trình huấn luyện của 5 epochs:



*Training Loss và Accuracy mô hình EfficientNet*

Kết quả accuracy trên tập test chỉ đạt độ chính xác 3%:

```
4894/4894 ————— 49s 7ms/step
Evaluate Test Accuracy: 0.03%
```

### 3. InceptionV3

Training loss và accuracy trong quá trình huấn luyện của 5 epochs:



*Training Loss và Accuracy mô hình InceptionV3*

Training loss và accuracy trong quá trình fine-tuning sau 5 epochs:





*Fine Tuning mô hình InceptionV3*

Kết quả accuracy trên tập test đạt độ chính xác 98%:

```
[ ] predictions = model.predict(test_generator)
    predictions = predictions.argmax(axis=1)
    true_labels = test_generator.classes

    from sklearn.metrics import accuracy_score
    accuracy = accuracy_score(predictions, true_labels)
    print("%s: %.2f%%" % ("Evaluate Test Accuracy", accuracy))
```

 4894/4894  47s 8ms/step  
Evaluate Test Accuracy: 0.98%

Confusion matrix:

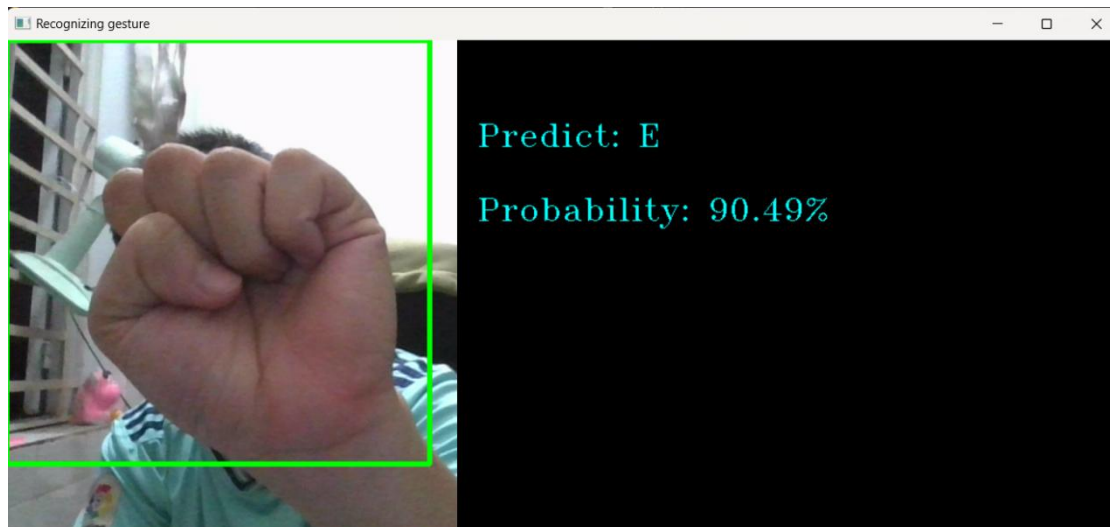


### Confusion Matrix

Actual Values	/space	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	169			
	/nothing	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	169	0		
	/del	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	168	0	0		
	/Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	165	1	0	0	
	/Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	1	0	0	0	166	0	0	0	0	
	/X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	1	1	1	0	158	1	0	0	0	0	0	0		
	/W	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2	165	0	1	0	0	0	0	
	/V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	2	164	0	1	0	0	0	0	0	0	0	0	
	/U	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	163	0	0	4	0	0	0	0	0	0	0	
	/T	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2	163	0	0	0	0	1	2	0	0	0	0	0	0	
	/S	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	162	2	0	2	0	1	0	1	0	0	0	0	0	
	/R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	167	1	0	1	0	0	0	0	0	0	0	0	0	0	
	/Q	0	0	0	0	0	0	0	0	0	0	0	0	0	1	165	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	
	/P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	164	1	0	0	0	1	0	0	0	0	0	0	0	2	0	0
	/O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	164	0	0	0	0	0	1	0	0	0	1	0	1	0	2		
	/N	0	0	0	0	0	0	0	0	0	0	0	2	167	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	/M	0	0	0	0	0	0	0	0	0	0	0	162	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	/L	0	0	0	0	0	0	0	0	0	0	168	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	/K	0	0	0	0	0	2	0	5	0	158	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	0	0	0
	/J	0	0	0	0	0	0	0	0	168	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	/I	0	0	1	0	0	0	0	0	163	0	0	0	0	0	0	0	1	0	0	1	2	0	0	0	0	0	0	0	1	0	0
	/H	0	0	0	0	0	0	0	168	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	/G	0	0	0	2	1	0	165	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	/F	0	0	0	0	0	169	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	/E	0	0	0	0	165	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	/D	0	0	0	168	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	/C	0	0	167	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
	/B	0	169	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	/A	164	0	0	0	2	0	0	0	1	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		/A	/B	/C	/D	/E	/F	/G	/H	/I	/J	/K	/L	/M	/N	/O	/P	/Q	/R	/S	/T	/U	/V	/W	/X	/Y	/Z	/del	/nothing	/space		
	Predicted Values																															

Triển khai dự đoán trên webcam, nhận các khung hình từ khung màu xanh bên góc trái màn hình, kết quả được dự đoán nếu thu được 10 frame dự đoán liên tục ra 1 nhãn.





#### IV. Kết luận:

Qua quá trình tìm hiểu và xây dựng mô hình nhận diện ngôn ngữ ký hiệu bằng webcam, nhóm đã bước đầu hoàn thiện một hệ thống dựa vào Deep Learning. Dù hệ thống đã đạt được những kết quả nhất định, vẫn còn nhiều tiềm năng để cải thiện và phát triển.

Trong quá trình triển khai, nhóm nhận thấy InceptionV3 là một lựa chọn phù hợp hơn so với ResNet và EfficientNet cho bài toán nhận diện ngôn ngữ ký hiệu.

##### **Ưu điểm của InceptionV3:**

Kiến trúc module Inception với nhiều kích thước bộ lọc (1x1, 3x3, 5x5) giúp mô hình học được các đặc trưng đa cấp độ từ dữ liệu ngôn ngữ ký hiệu, nơi các hành động tay có thể rất đa dạng về hình dáng và ngữ cảnh. Hiệu quả cao trong trích xuất đặc trưng mà không yêu cầu quá nhiều tài nguyên tính toán như EfficientNet.

Khả năng tổng quát hóa tốt hơn so với ResNet trong các bài toán có dữ liệu không đồng nhất hoặc số lượng dữ liệu huấn luyện hạn chế.

Hướng phát triển tiếp theo, nhóm sẽ tập trung vào các mục tiêu chính như sau:

**Tăng cường khả năng biểu diễn ngữ nghĩa:** Nghiên cứu và tích hợp mạng hồi quy RNN (Recurrent Neural Network) hoặc các biến thể như LSTM/GRU để mô hình hóa chuỗi hành động, từ đó giúp hệ thống có khả năng hiểu và biểu diễn các câu có ngữ nghĩa.

**Mở rộng dữ liệu và ngôn ngữ ký hiệu:** Tiếp tục thu thập và bổ sung các bộ dữ liệu ngôn ngữ ký hiệu mới, phong phú và đa dạng hơn nhằm cải thiện độ chính xác và khả năng nhận diện của mô hình.

Tuy nhiên, do hạn chế về dữ liệu, mô hình hiện tại vẫn còn tồn tại nhiều sai sót. Một số lỗi nhận diện xuất phát từ việc dữ liệu huấn luyện chưa đủ phong phú, hoặc do sự khác biệt về cách biểu diễn của từng người. Việc xử lý những hạn chế này sẽ là ưu tiên hàng đầu trong các bước nghiên cứu và cải tiến tiếp theo.