TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

———————— * ————————

BÁO CÁO MÔN HỌC

HỌC PHẦN: Project II

(Mã học phần: IT3930)

Đề tài:

Xây dựng hệ thống nhận diện ngôn ngữ ký hiệu tay sử dụng học sâu

Sinh viên thực hiện:	Lê Đức Quân
MSSV:	20204682
Mã lớp:	727526
Giảng viên hướng dẫn:	Ths. Nguyễn Duy Hiệp

Hà Nội, tháng 7 năm 2023

Mục lục

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 kí 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, em sẽ xây dựng 1 hệ thống sử dụng học sâu đơn giản cho phép nhận diện ISL ASL thông qua camera.

II. Kiến thức tổng quan

Học sâu (Deep Learning) là một lĩnh vực trong trí tuệ nhân tạo (AI) tập trung vào việc xây dựng các mạng nơ-ron nhân tạo có nhiều lớp ẩn, gọi là mạng nơ-ron sâu. Mục tiêu của học sâu là mô phỏng cấu trúc và hoạt động của hệ thống thần kinh sinh học trong não người để giải quyết các bài toán phức tạp trong nhiều lĩnh vực, bao gồm thị giác máy tính, xử lý ngôn ngữ tự nhiên, nhận dạng giọng nói, và nhiều ứng dụng khác.

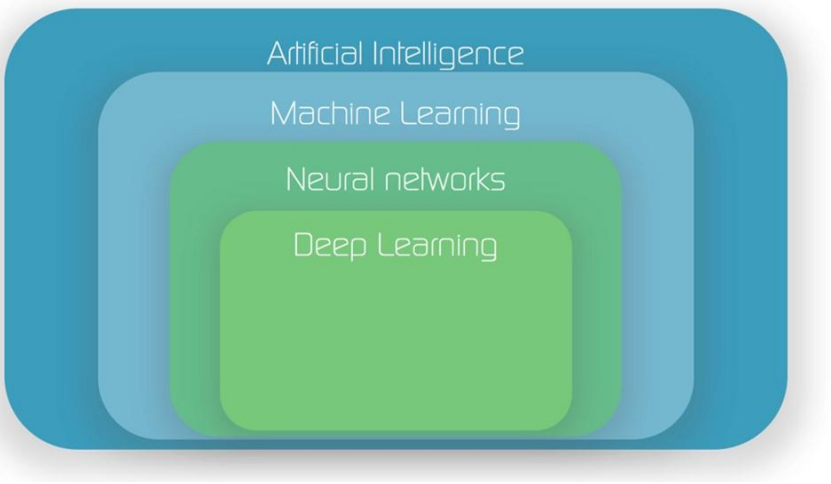


Fig. Mối quan hệ giữa DL và AI

Quá trình huấn luyện mạng nơ-ron sâu bao gồm việc điều chỉnh trọng số của các nơ-ron thông qua việc sử dụng các thuật toán tối ưu hóa, như gradient descent.

Điểm mạnh của học sâu là khả năng mô hình hóa các mô hình phức tạp và giải quyết các bài toán có tính chất phi tuyến. Nó đã mang lại những thành tựu đáng kể trong nhiều lĩnh vực, như xe tự lái, nhận dạng ảnh, dịch máy, xử lý ngôn ngữ tự nhiên, và nhiều ứng dụng khác. Tuy nhiên, học sâu cũng đòi hỏi lượng dữ liệu huấn luyện lớn và công nghệ tính toán mạnh mẽ để đạt được hiệu suất tốt.

Nhờ vào sự tiến bộ trong lĩnh vực học sâu, chúng ta đang chứng kiến những ứng dụng đáng kinh ngạc và tiềm năng tuyệt vời trong việc tạo ra các hệ thống thông minh có khả năng học và thích nghi tự động với môi trường xung quanh chúng.

1. Convolutional Neural Network (CNN)

Mạng CNN (Convolutional Neural Network) là một loại mạng nơ-ron học sâu đặc biệt được thiết kế để xử lý và phân tích dữ liệu hình ảnh và video. Nó là một trong những cấu trúc mạng phổ biến và mạnh mẽ nhất trong lĩnh vực thị giác máy tính và trí tuệ nhân tạo.

Mạng CNN đã đạt được rất nhiều thành tựu đáng kể trong các ứng dụng thị giác máy tính, như nhận diện đối tượng, phân loại ảnh, phát hiện khuôn mặt, ô tô tự lái, và nhiều lĩnh vực khác. Sự thành công của CNN đến từ khả năng tự động học các đặc trưng từ dữ liệu và khả năng học cách tối ưu hóa thông qua việc sử dụng lượng lớn dữ liệu huấn luyện.

Do đó, đây là mô hình mạng học sâu phù hợp để giải quyết bài toán nhận diện ngôn ngữ kí hiệu của chúng ta

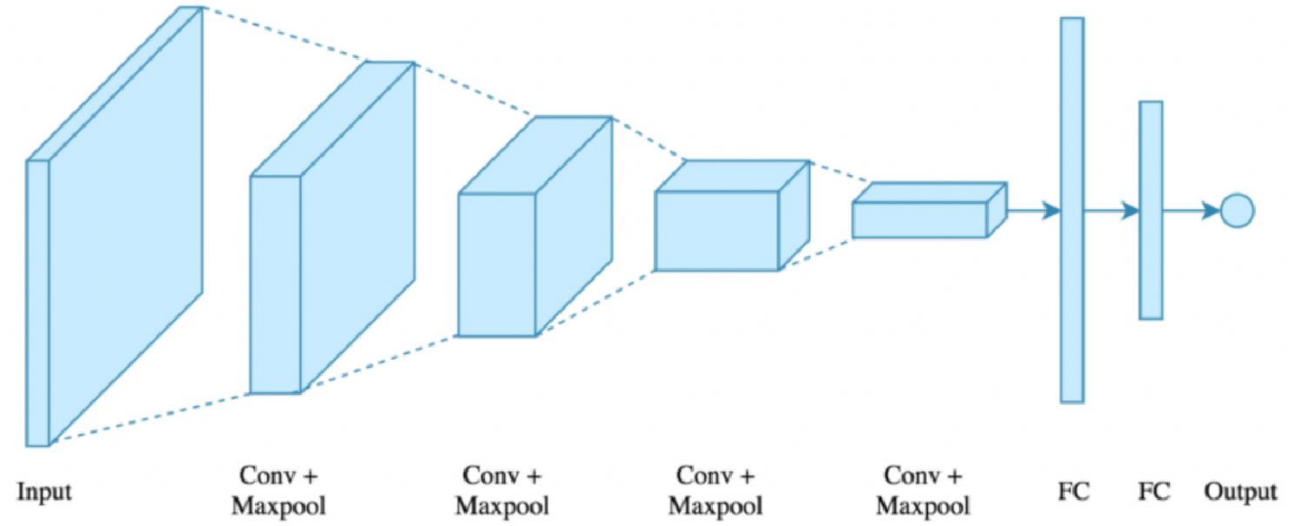


Fig. Sơ đồ hệ thống CNN

Dưới đây, là các thành phần cơ bản được sử dụng trong mạng CNN:

1.1 Lớp tích chập (Convolutional Layer)

Lớp tích chập (Convolutional Layer) là một trong những lớp quan trọng trong mạng nơ-ron học sâu Convolutional Neural Network (CNN). Nó là lớp đặc biệt được sử dụng để xử lý dữ liệu hình ảnh và giúp mạng CNN tự động học các đặc trưng từ hình ảnh đầu vào.

Cách hoạt động của lớp tích chập tương tự như cách não người thực hiện việc nhận diện các đặc trưng trong hình ảnh. Lớp tích chập sử dụng các bộ lọc (filters) hoặc bộ nhân (kernels) nhỏ có kích thước nhỏ và trượt chúng qua toàn bộ hình ảnh. Khi trượt qua, bộ lọc thực hiện phép tích chập giữa chính nó và các vùng tương ứng trong hình ảnh, tạo ra một ma trận kết quả gọi là feature map hoặc bản đồ đặc trưng.

Ví dụ, giả sử chúng ta tính tích chập của một hình ảnh 32x32x3 (hình ảnh 32x32 với 3 kênh màu R, G và B tương ứng) với một bộ lọc 5x5x3, với bước nhảy **stride=1**, và **không có padding**. Chúng ta lấy bộ lọc 5x5x3 và trượt nó qua toàn bộ hình ảnh và trong quá trình đó tính tích chập (dot product) giữa bộ lọc và từng phần nhỏ của hình ảnh đầu vào. Ứng với 1 bộ lọc sẽ sinh ra 1 feature map kích thước 28x28

Quá trình này giúp tách biệt các đặc trưng cơ bản từ dữ liệu hình ảnh, chẳng hạn như các đường viền, góc, sắc thái, và các mẫu đặc biệt. Nhiều bộ lọc được sử dụng song song nhau để học nhiều đặc trưng và tạo ra nhiều feature map khác nhau.

Lớp tích chập giúp giảm số lượng tham số của mạng, làm giảm tính toán và giúp mô hình trở nên hiệu quả hơn. Nó cũng giúp mô hình học cách chia sẻ các đặc trưng giữa các vùng của hình ảnh, làm cho mạng có khả năng tổng quát hóa tốt hơn và giảm thiểu overfitting.

1.2 Lớp gộp (Pooling layer)

Lớp pooling sẽ giảm bớt số lượng tham số khi hình ảnh quá lớn. Không gian pooling còn được gọi là lấy mẫu con hoặc lấy mẫu xuống làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng. Lớp gộp hoạt động độc lập trên từng feature map. Các pooling có thể có nhiều loại khác nhau:

- Max Pooling
- Average Pooling
- Sum Pooling

Max Pooling là lấy phần tử lớn nhất từ ma trận đối tượng, là lớp gộp hay được dùng nhiều nhất trong mạng CNN

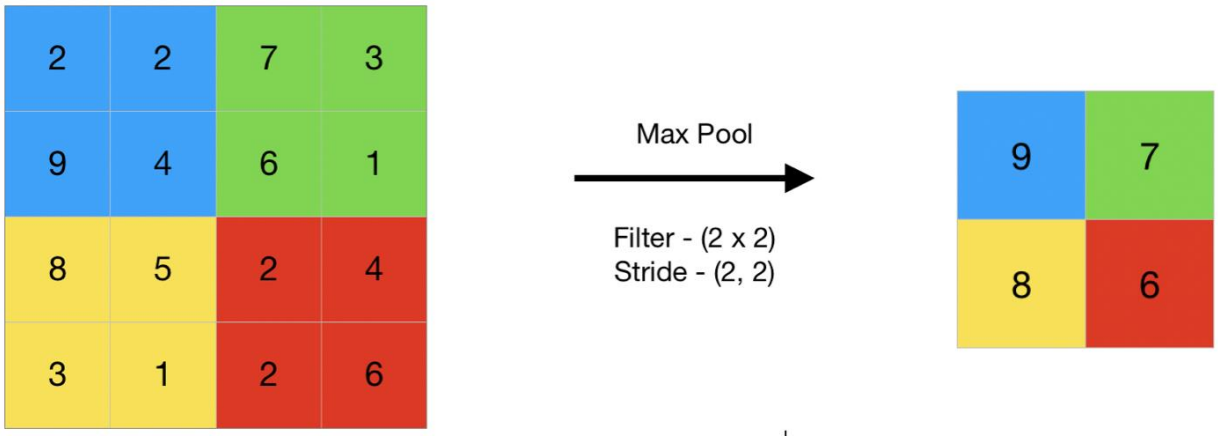


Fig. Max Pooling

1.3 Lớp kích hoạt

Hàm kích hoạt (activation function) mô phỏng tỷ lệ truyền xung qua axon của một neuron thần kinh. Trong một mạng nơ-ron nhân tạo, hàm kích hoạt đóng vai trò là thành phần phi tuyến tại output của các nơ-ron.

Lớp kích hoạt (Activation Layer) là một trong các lớp quan trọng trong mạng nơ-ron học sâu, bao gồm cả mạng CNN. Lớp kích hoạt được đặt sau mỗi lớp tích chập hoặc lớp fully connected trong mạng

ReLU (Rectified Linear Unit) là một hàm kích hoạt phổ biến và quan trọng trong mạng nơ-ron học sâu, bao gồm cả mạng CNN và các mạng nơ-ron khác do thực hiện tính toán nhanh

Công thức của hàm ReLU là: $f(x) = \max(0, x)$

1.4 Lớp kết nối đầy đủ (Fully Connected)

Lớp fully connected đặc trưng bởi việc tất cả các nơ-ron trong lớp này đều kết nối với tất cả các nơ-ron trong lớp trước đó, tạo thành một mạng liên kết đầy đủ giữa các lớp. Khi các dữ liệu từ các lớp trước đó đã được rút trích đặc trưng thông qua các lớp tích chập và lớp pooling trong mạng CNN (hoặc các lớp ẩn trong các mạng nơ-ron khác), lớp fully connected thực hiện công việc phân loại và dự đoán dựa trên các đặc trưng đã được học.

Tuy nhiên, số lượng trọng số trong lớp này thường rất lớn, đòi hỏi khá nhiều dữ liệu huấn luyện và tài nguyên tính toán. Do đó, trong mạng CNN, lớp fully connected thường chỉ được sử dụng trong các lớp cuối cùng để thực hiện phân loại hoặc dự đoán, trong khi các lớp trước đó thường sử dụng lớp tích chập và lớp pooling.

2. Mạng Inception

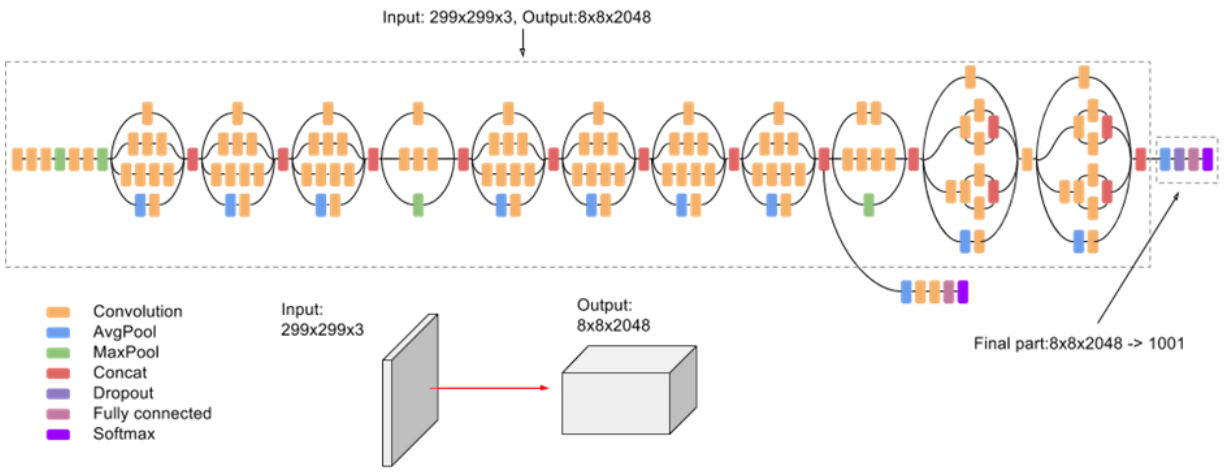
Mạng Inception (hay còn gọi là GoogLeNet) là một trong những kiến trúc mạng nơ-ron nổi tiếng trong lĩnh vực trí tuệ nhân tạo và thị giác máy tính. Nó được giới thiệu bởi nhóm nghiên cứu của Google Research trong cuộc thi ImageNet Large Scale Visual Recognition Challenge (ILSVRC) năm 2014 và đạt được 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 em sẽ sử dụng kĩ thuật học tái sử dụng (transfer learning) mô hình Inception đã được huyến 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

III. Xây dựng hệ thống nhận diện

1. Bộ dữ liệu (Dataset) sử dụng

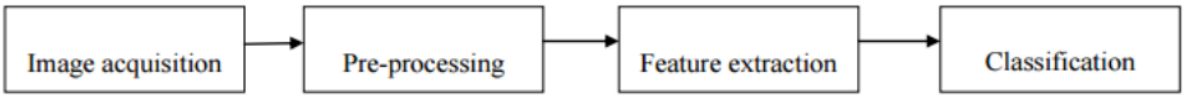
Bộ dữ liệu được lấy từ: [ASL Alphabet | Kaggle]

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 kí tự khác nhau.

- Tập training bao gồm 87000 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 kí 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 3000 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 kĩ thuật Tăng cường dữ liệu hình ảnh (**Data Augmentation**) để tăng khả năng tổng quát hóa của mô hình và giảm hiện tượng overfitting.

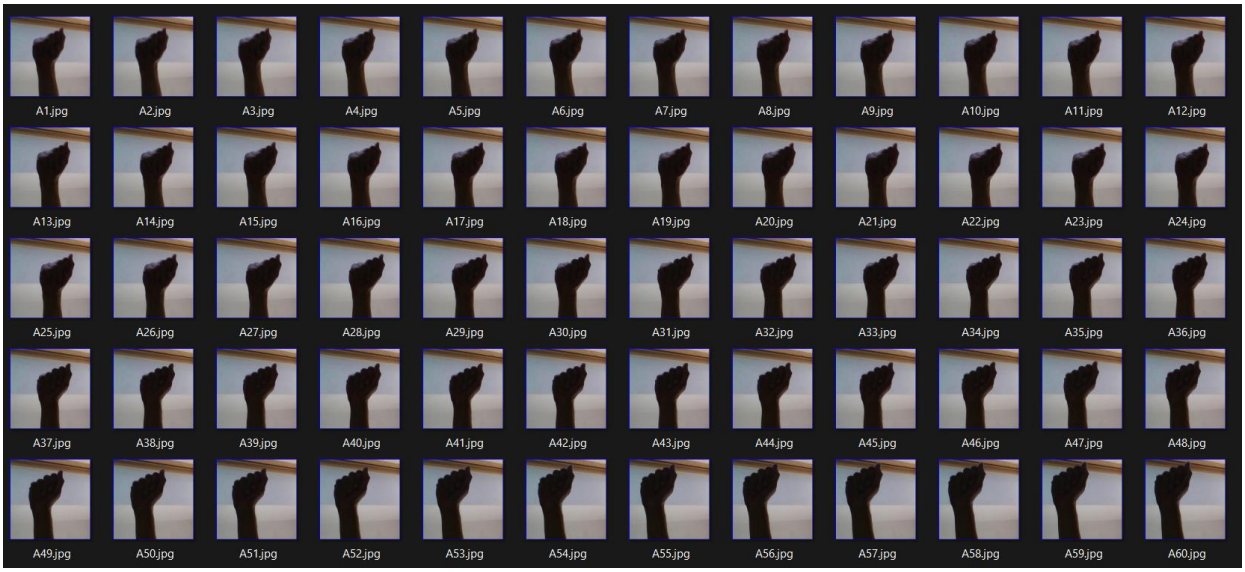


Fig. Các hình ảnh tương đối giống nhau

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 0.6, 0.15, 0.25

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 bộ dữ liệu

3.2 Xây dựng mô hình

Sử dụng kĩ thuật học tái sử dụng (transfer learning) mô hình Inception đã được huyến luyện trước (pre-train) trên bộ dữ liệu của ImageNet (bao gồm 1000 nhãn).

```
inception_v3_model = InceptionV3(
    input_shape = (IMAGE_SIZE, IMAGE_SIZE, 3),
    include_top = False,
    weights = 'imagenet'
)
```

Fig. Khởi tạo Inception V3 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)
```

Sơ đồ của mô hình dưới dạng hình ảnh được lưu tại file: ./code/visualize_model.png

3.3 Huấn luyện mô hình

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
Trainable params: 2,127,901
Non-trainable params: 21,802,784
```

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 10 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

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(lr=0.0001, momentum=0.9), loss='categorical_crossentropy', metrics=['accuracy'])
```

Sau khi huấn luyện xong, lưu lại mô hình trong file:
./code/model/asl_alphabet_model.h5

IV. Kết quả

Training loss và accuracy trong quá trình huấn luyện của 10 epochs đầu tiên:

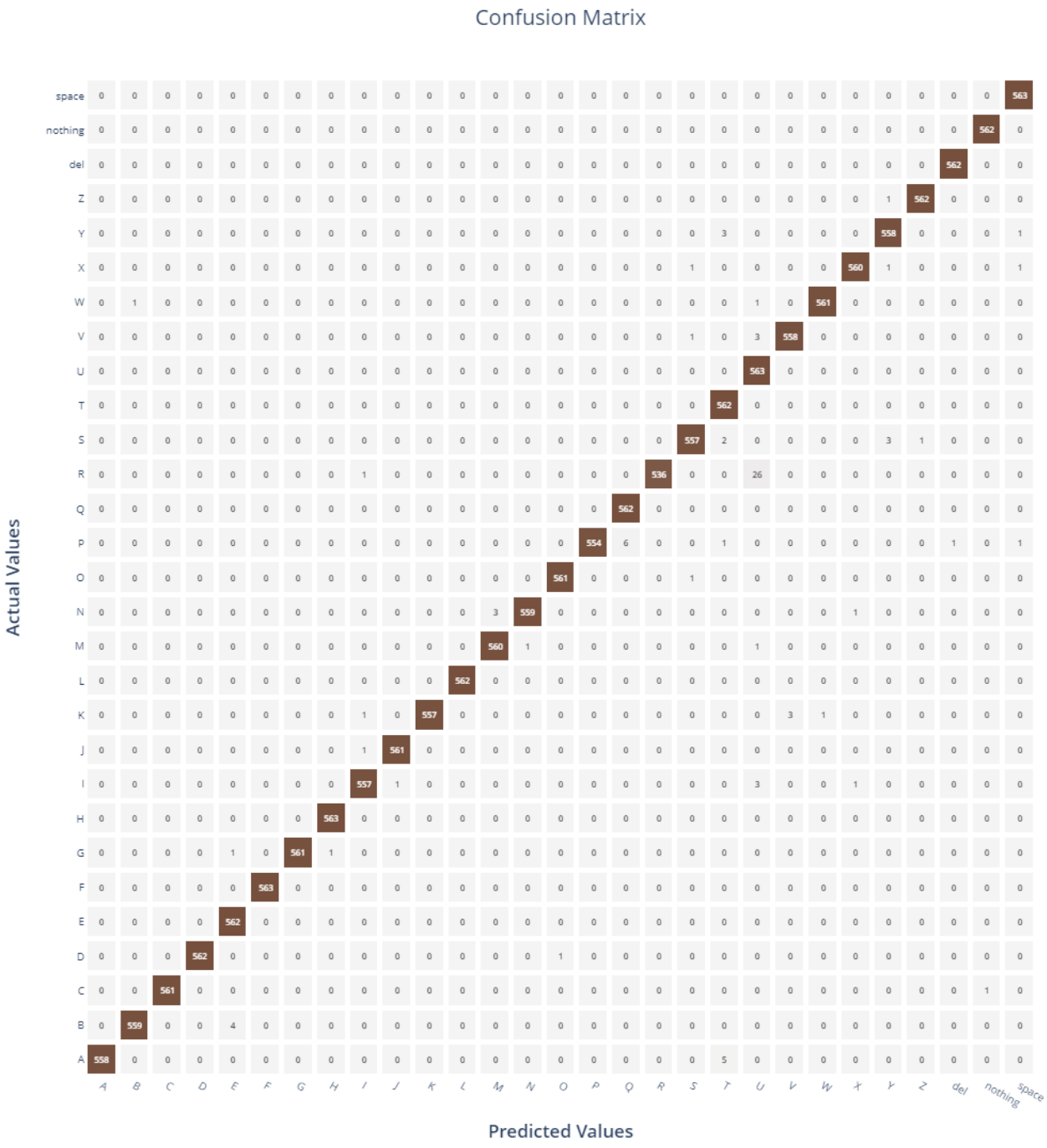
Training loss và accuracy trong quá trình fine-tuning sau 5 epochs đầu:

Kết quả accuracy trên tập test đạt độ chính xác 99%

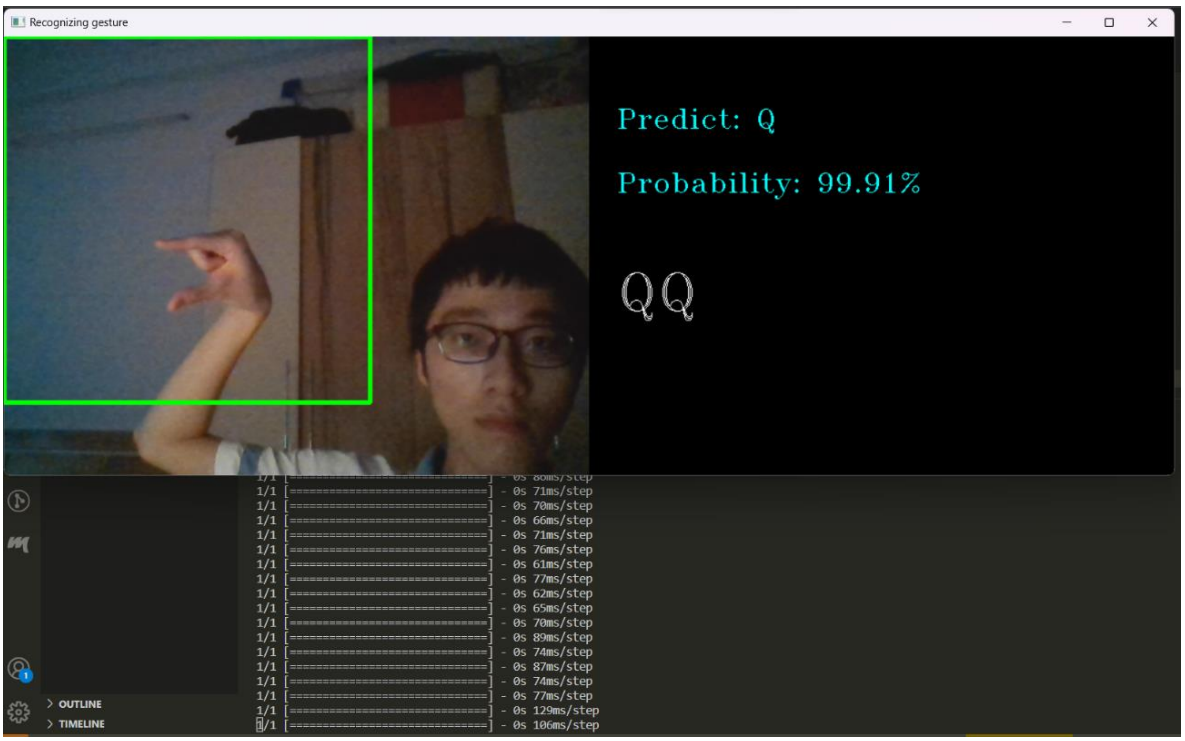
```
predictions = model.predict(test_generator)
predictions = predictions.argmax(axis=1)
true_labels = test_generator.classes

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

```
16313/16313 [==============================] - 183s 11ms/step
Evaluate Test Accuracy: 0.99%
```

Confusion matrix:


Confusion Matrix
Actual Values
Predicted Values
space nothing del Z Y X W V U T S R Q P O N M L K J I H G F E D C B A
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

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.



V. Kết luận

Qua việc tìm hiểu mô hình học sâu và mạng CNN, em đã xây dựng được 1 mô hình để nhận diện các ngôn ngữ kí hiệu bằng webcam 1 cách đơn giản. Do thời gian thời gian chưa có nhiều nên em trình 1 hệ thống cơ bản, hướng phát triển tiếp theo của em sẽ tìm hiểu cách biểu diễn các câu có nghĩa bằng chuỗi các hành động và sử dụng mạng hồi quy RNN, thêm số lượng ngôn ngữ kí hiệu có ngữ nghĩa mới vào mô hình. Cuối cùng cảm ơn thầy Nguyễn Duy Hiệp đã có những góp ý để em hoàn thiện hệ thống !

Tài liệu tham khảo

[1] Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition

[2] Keras API reference

[3] [1409.4842] Going Deeper with Convolutions (arxiv.org)