**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ

**BỘ MÔN TỰ ĐỘNG**

---------------o0o---------------

****

**BÁO CÁO BÀI TẬP NHÓM**

**NHẬN DẠNG CỬ CHỈ TAY**

**SỬ DỤNG MẠNG CNN**

**NHÓM 1**

1. **Mai Thiện Quang – 1512640**
2. **Nguyễn Trọng Phúc – 1512534**
3. **Lương Hữu Phú Lộc – 1511844**
4. **Bùi Tấn Phát – 1512396**
5. **Phạm Ngọc Khôi Nguyên – 1512221**

**Đề bài:** Sử dụng mạng CNN để giải quyết một bài toán cụ thể. Ở đây nhóm xin chọn bài toán nhận dạng cử chỉ tay bằng mạng CNN

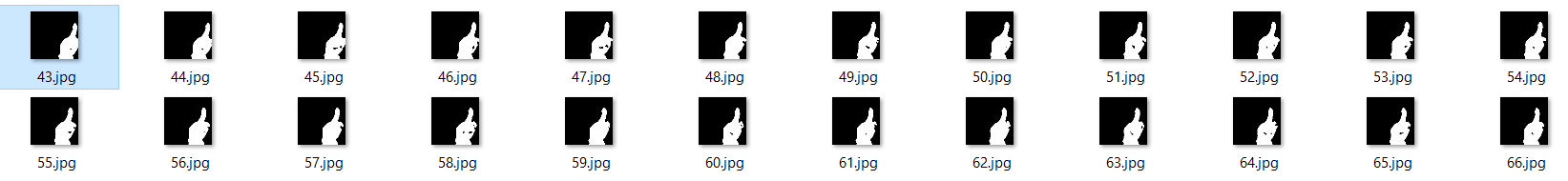
* Để tiết kiệm thời gian nhóm đã sử dụng bộ data lấy từ một project khác cũng nhận dạng cử chỉ bằng mạng CNN, bộ data này gồm có 5 loại là: punch, hand, one, two và none trong đó none nghĩa là không có cử chỉ gì. Mỗi loại data có khoảng gần 1000 tấm hình, riêng none chỉ có 533 tấm.
* Minh hoạ 5 loại dữ liệu như các hình sau:
* “Hand” data



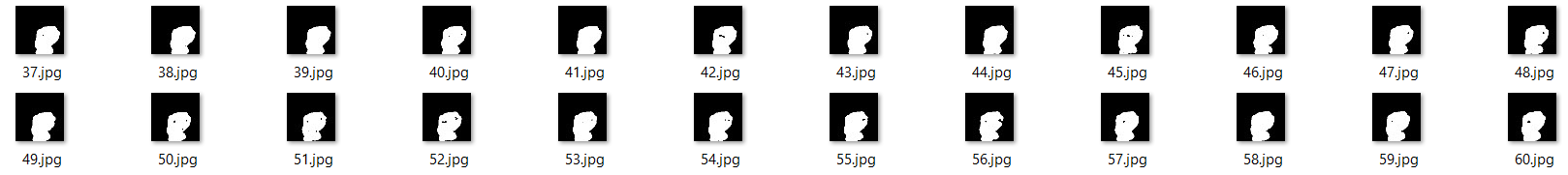
* “None” data

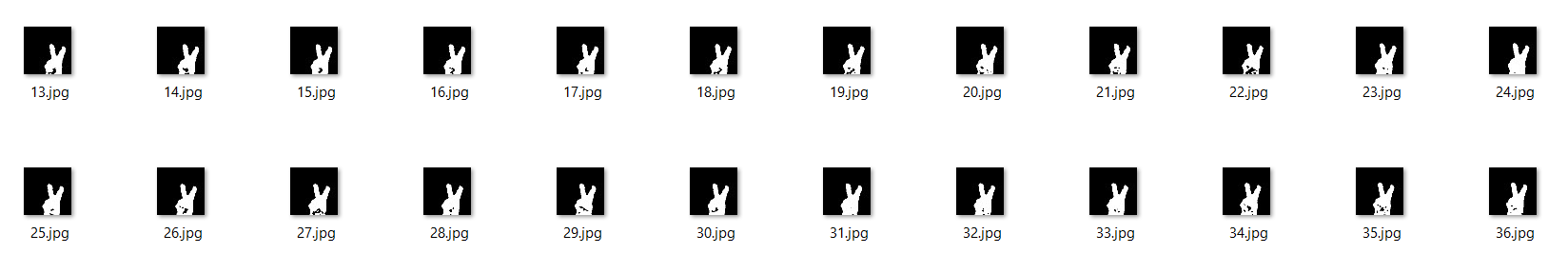


* “One” data



* “Punch” data



* “Two” data

## **Training cho mạng CNN:**

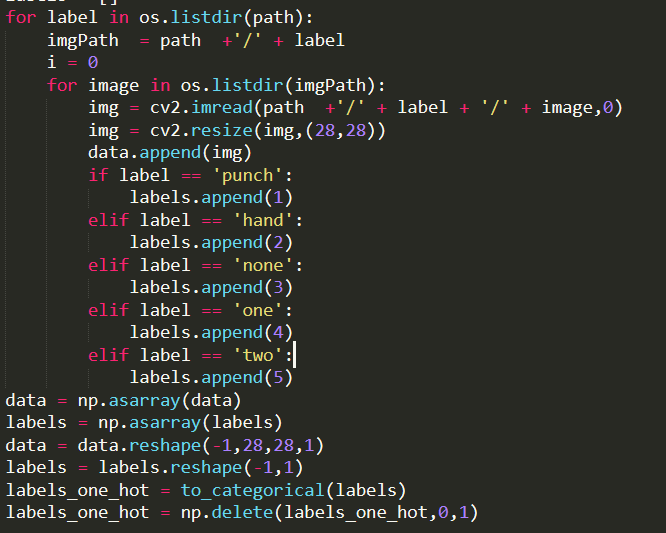
Nhóm sử dụng thư viện Keras của tensorflow

Thông số khởi tạo:

* batch size =128, thể hiện số lượng mẫu mà Mini-batch GD sử dụng cho một lần cập nhật trọng số.
* epochs = 10 với mỗi epoch là một lần “duyệt” qua hết số lượng mẫu trong tập huấn luyện.
* Số ngõ ra của mạng = 5, tương ứng với 5 cử chỉ trên

Từ folder hình ảnh ở trên, nhóm tạo data để train cho mạng như sau:

* Truy cập vào từng folder, sau đó lưu mỗi hình vào mảng data,còn nhãn tương ứng sẽ là tên của folder
* Để đơn giản ngõ ra, khi tạo data để train nhóm sử dụng label là số thay cho tên cử chỉ,vd: “PUNCH” thay bằng 1, “HAND” là 2, “NONE” là 3, “ONE” là 4 và “TWO” là 5

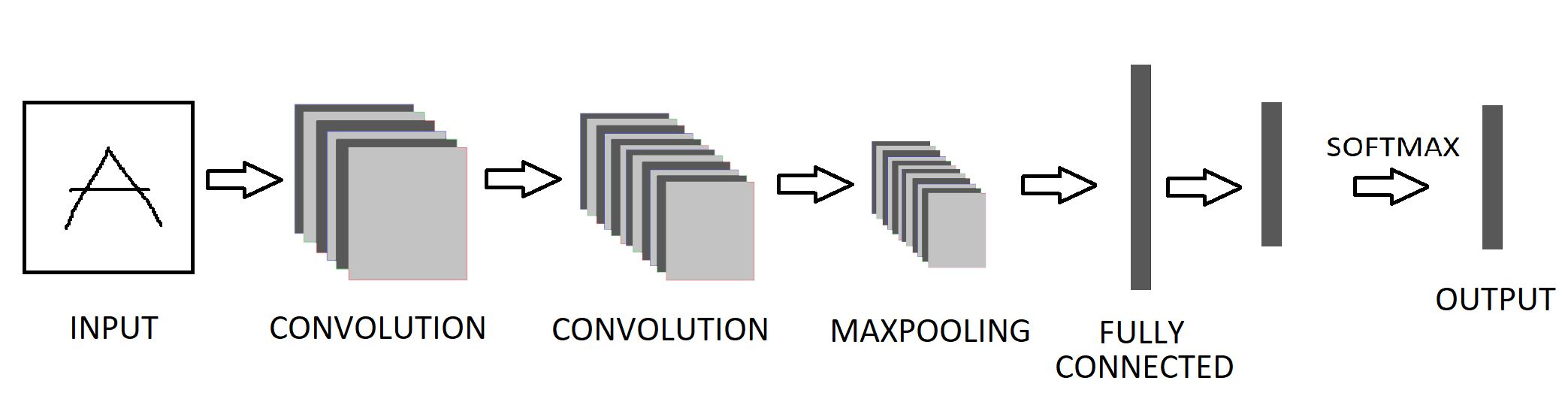


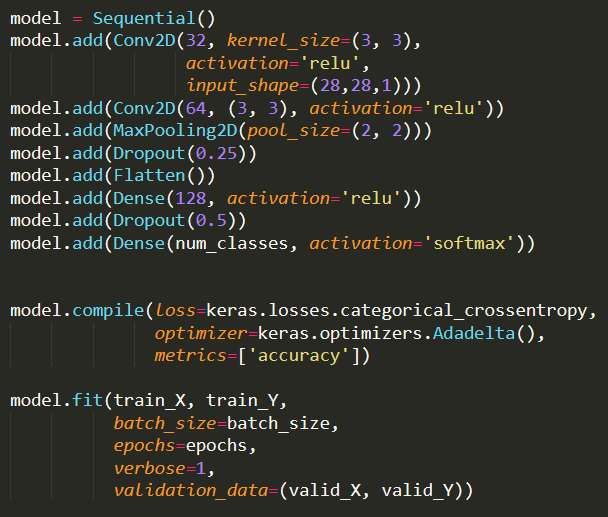
Tiếp đến tạo ma trận one\_hot sử dụng hàm **to\_categorical()**, ví dụ 2 ảnh được đánh số 1 vs 3:

tương đường với

Trong tổng số ảnh ta chia ra làm 2 phần với tỷ lệ 80-20 dùng để train và để test sau quá trình train với hàm **train\_test\_split()**.

Ta tiếp tục xây dựng sơ đồ cho mạng CNN gồm 2 lớp tích chập, lớp MaxPooling và lớp Softmax:





Ta thực hiện quá trình train với hàm **fit()** và compile lại mạng CNN vừa tạo bằng hàm **compile()** khi thực hiện hàm **compile()** các trọng số của mạng không hề thay đổi, mà chỉ giúp tối ưu hóa cho mạng CNN vừa tạo.

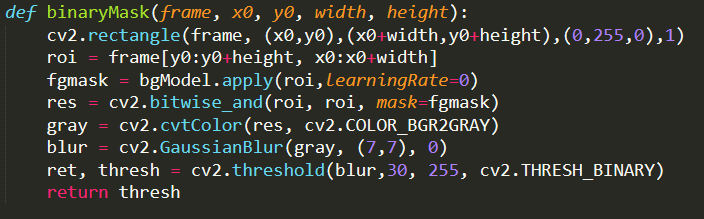
Sau khi training, ta lưu lại mạng CNN với tên **cnn\_model.h5**. Model này có thể load lại để sử dụng sau này.

## **Thực hiện quá trình nhận dạng:**

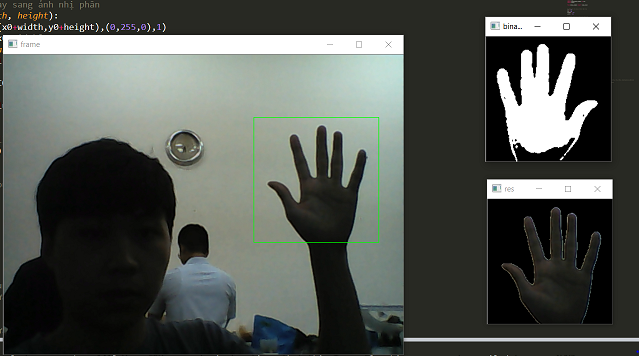
Để tách bàn tay từ khung hình, nhóm sử dụng phương pháp **backgroundSubtractorMOG2** của thư viện OpenCV, đầu tiên là lấy khung hình không có bàn tay làm background, sau đó với những khung hình sau khi đưa bàn tay vào sẽ trừ đi background, từ đó ta tách được bàn tay



Trên mỗi frame ta tạo một khung chữ nhật – nơi sẽ được dùng để xác định cử chỉ tay. Khung hình này sẽ được cắt ra và xử lí bằng hàm con ***binaryMask*** như sau:



Kết quả sau khi tách như sau:

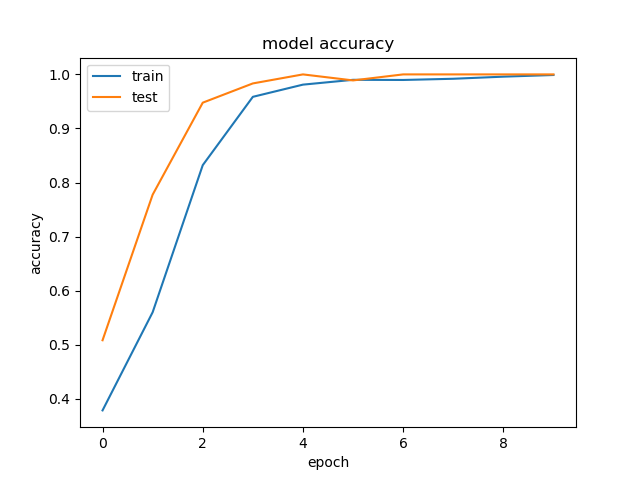


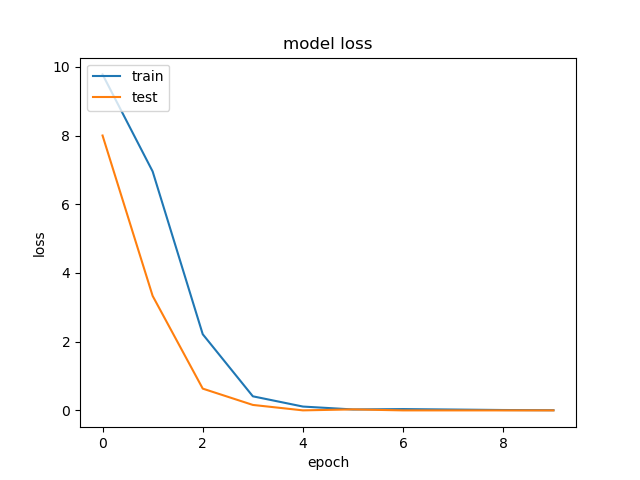
Ảnh nhị phân trên sẽ được resize về kích thước 28x28x1 để phù hợp với ngõ vào của mạng CNN.

Ta dùng hàm **predict(ảnh nhị phân)** sau đó dùng **argmax()** để tìm label có phần trăm lớn nhất và lấy đó làm kết quả của mạng.

## **Kết quả:**

Biểu đồ khi training mạng CNN:





Ngoài ra, nhóm còn có một video demo đính kèm trong folder báo cáo.

## **Nhận xét:**

Từ biểu đồ trên có thể thấy độ chính xác của mô hình là rất cao (gần bằng 1). Tuy nhiên khi áp dụng mô hình, nhóm nhận thấy để chương trình hoạt động tốt cần phải tìm một Background tốt (nền màu sáng) để dễ dàng hơn trong việc nhận dạng cử chỉ tay, nếu Background được chọn tệ có thể dẫn đến việc dự đoán không chính xác do không thể tách được bàn tay hoặc bàn tay bị khuyết,…

Nhưng nhìn chung code vẫn đưa ra cùng 1 kết quả rất chính xác, quá trình training cũng diễn ra khá nhanh và thuận lợi do việc số label cần xác định còn khá ít (là 5)