TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH

KHOA ĐIỆN ĐIỆN TỬ

BỘ MÔN KỸ THUẬT MÁY TÍNH - VIỄN THÔNG



BÁO CÁO CUỐI KÌ

**PHÂN LOẠI TÍN HIỆU RADAR**

GVHD: **TS.Huỳnh Thế Thiện**

Sinh viên: **Hàn Anh Tú**

MSSV: 21161215

**Nguyễn Minh Nhựt**

MSSV: 2116xx66

TP. HỒ CHÍ MINH – 4/2024

BẢN NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

|  |  |  |
| --- | --- | --- |
| Stt | Nội dung thực hiện | Nhận xét |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Nhận xét tổng quát:

**TÓM TẮT**

Deep Learning là thuật toán dựa trên một số ý tưởng từ não bộ tới việc tiếp thu nhiều tầng biểu đạt, cả cụ thể lẫn trừu tượng, qua đó làm rõ nghĩa của các loại dữ liệu. Deep Learning được ứng dụng trong nhận diện hình ảnh, nhận diện giọng nói, xử lý ngôn ngữ tự nhiên. Hiện nay rất nhiều các bài toán nhận dạng sử dụng Deep Learning, vì nó có thể giải quyết các bài toán với số lượng lớn các biến, tham số kích thước đầu vào lớn với hiệu năng cũng như độ chính xác vượt trội so với các phương pháp phân lớp truyền thống, xây dựng những hệ thống thông minh với độ chính xác cao. Trong báo cáo này, chúng tôi nghiên cứu về đề tài “Phân loại tín hiệu radar” theo sự hướng dẫn của TS.Huỳnh Thế Thiện .

**mục lục**

[**DANH MỤC HÌNH** iv](#_Toc74751132)

[**CÁC TỪ VIẾT TẮT** v](#_Toc74751133)

[Chương 1: Giới thiệu 1](#_Toc74751134)

[1.1. Giới thiệu 1](#_Toc74751135)

[1.2. mục tiêu đề tài 1](#_Toc74751136)

[1.3. GIỚI HẠN ĐỀ TÀI 1](#_Toc74751137)

[1.4. PHƯƠNG PHÁP NGHIÊN CỨU 2](#_Toc74751138)

[1.5. ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU 2](#_Toc74751139)

[1.6. BỐ CỤC QUYỂN BÁO CÁO 2](#_Toc74751140)

[Chương 2: CƠ SỞ LÝ THUYẾT 3](#_Toc74751141)

[Giới thiệu mạng cnn 3](#_Toc74751142)

[Chương 3: xây dựng chương trình 9](#_Toc74751150)

[3.1. xây dựng mô hình 9](#_Toc74751151)

[3.2. Các bước thực hiện 9](#_Toc74751152)

[3.2.1. Load tập dữ liệu từ Kaggle 9](#_Toc74751153)

[3.2.2. Huấn luyện dữ liệu đưa vào 14](#_Toc74751154)

[3.2.3. Dữ liệu đầu ra được phân loại theo các lớp như sau 18](#_Toc74751155)

[Chương 4: Kết quả thí nghiệm 20](#_Toc74751156)

[4.1. Kết quả sau khi huấn luyện 20](#_Toc74751157)

[4.2. ĐỒ THỊ SỰ BIẾN THIÊN CỦA HÀM LOSS VÀ GIÁ TRỊ ACCURACY 21](#_Toc74751158)

[4.3. ĐƯA HÌNH ẢNH TỪ TỆP TEST ĐỂ PHÂN LOẠI 21](#_Toc74751158)

[KẾT LUẬN 24](#_Toc74751159)

[PHỤ LỤC 25](#_Toc74751160)

[TÀI LIỆU THAM KHẢO 28](#_Toc74751161)

DANH MỤC HÌNH

[Hình 2.1 Kiến trúc cơ bản của mạng CNN 3](#_Toc74668452)

[Hình 2.2 Bộ lọc tích chập được sử dụng trên ma trận điểm ảnh 4](#_Toc74668453)

[Hình 2.3 Phương thức Avarage Pooling và Max Pooling 5](#_Toc74668454)

[Hình 3.1 Sơ đồ quy trình của mô hình phân loại tín hiệu radar 9](#_Toc74668458)

[Hình 3.2 Hình ảnh 2 file test\_set và training\_set 11](#_Toc74668460)

[Hình 3.3 Hình ảnh các file tín hiệu được gán nhãn 11](#_Toc74668461)

[Hình 4.1 Tổng trọng số được ước tính trong mạng 20](#_Toc74668473)

[Hình 4.2 Độ chính và giá trị hàm lỗi theo từng epoch 21](#_Toc74668474)

[Hình 4.3 Đồ thị Training và Validation Accuracy/Loss 21](#_Toc74668475)

[Hình 4.4 Hình ảnh trước khi nhận diện (ảnh từ file B-FM) 22](#_Toc74668476)

[Hình 4.5 Hình ảnh sau khi phân loại (B-FM) 22](#_Toc74668477)

CÁC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| ReLu | Rectified Linear Unit |
| CNN | Convolutional Neural Network |
| MLL | Machine Learning Library |
| CONV | Convolution |

# Giới thiệu

## Giới thiệu

CNN là một trong những mô hình mạng Học sâu phổ biến nhất hiện nay, có khả năng nhận dạng và phân loại hình ảnh với độ chính xác rất cao, thậm chí còn tốt hơn con người trong nhiều trường hợp. Mô hình này đã và đang được phát triển, ứng dụng vào các hệ thống xử lý ảnh lớn của Facebook, Google hay Amazon… cho các mục đích khác nhau, như các thuật toán gắn thẻ tự động, tìm kiếm ảnh hoặc gợi ý sản phẩm cho người tiêu dùng.

Mạng CNN với kiến trúc thay đổi, có khả năng xây dựng liên kết chỉ sử dụng một phần cục bộ trong ảnh kết nối đến node trong lớp tiếp theo thay vì toàn bộ ảnh như trong mạng nơ ron truyền thẳng.

Nhờ sự phát triển mạng CNN, chúng tôi có cơ sở thực hiện đề tài “Nhận diện khuôn mặt”.

## mục tiêu đề tài

Đề tài “Phân loại các dạng tín hiệu radar” có thể áp dụng trong các hệ thống:

- Hệ thống thu phát sóng và tín hiệu

- Phân tích dữ liệu

## GIỚI HẠN ĐỀ TÀI

Phạm vi giới hạn của đề tài:

Đề tài chủ yếu sử dụng các thư viện có sẵn trong lập trình python, đồng thời ứng dụng các mô hình toán học trong mạng nơ ron nhân tạo. Điều kiện đầu vào của mô hình là các tệp hình ảnh có giới hạn, đầu ra phân loại các dạng sóng được đưa vào có độ chính xác tương đối. Đối với chức năng thì có sự giới hạn do hiểu biết về lập trình và cách ứng dụng của mô hình vào các hệ thống lớn.

## PHƯƠNG PHÁP NGHIÊN CỨU

Trong phần này, chúng tôi thiết kế mô hình phân loại tín hiệu dựa trên mạng CNN. Dựa trên các phương pháp nghiên cứu chính như phương pháp phân tích, tham khảo tài liệu, phương pháp tổng hợp tài liệu lý thuyết…

## ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU

Các đối tượng cần nghiên cứu và phạm vi nghiên cứu có thể giải quyết được đề tài:

- *Đối tượng nghiên cứu:* Trí tuệ nhân tạo, lập trình python.

- *Phạm vi nghiên cứu:* Mạng CNN, các module trong python: numpy, os, tensorflow, image.

## BỐ CỤC QUYỂN BÁO CÁO

Nội dung báo cáo:

CHƯƠNG 1: GIỚI THIỆU

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

CHƯƠNG 3: XÂY DỤNG CHƯƠNG TRÌNH

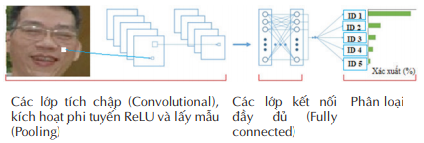
CHƯƠNG 4: KẾT QUẢ THÍ NGHIỆM

# CƠ SỞ LÝ THUYẾT

## Giới thiệu mạng cnn

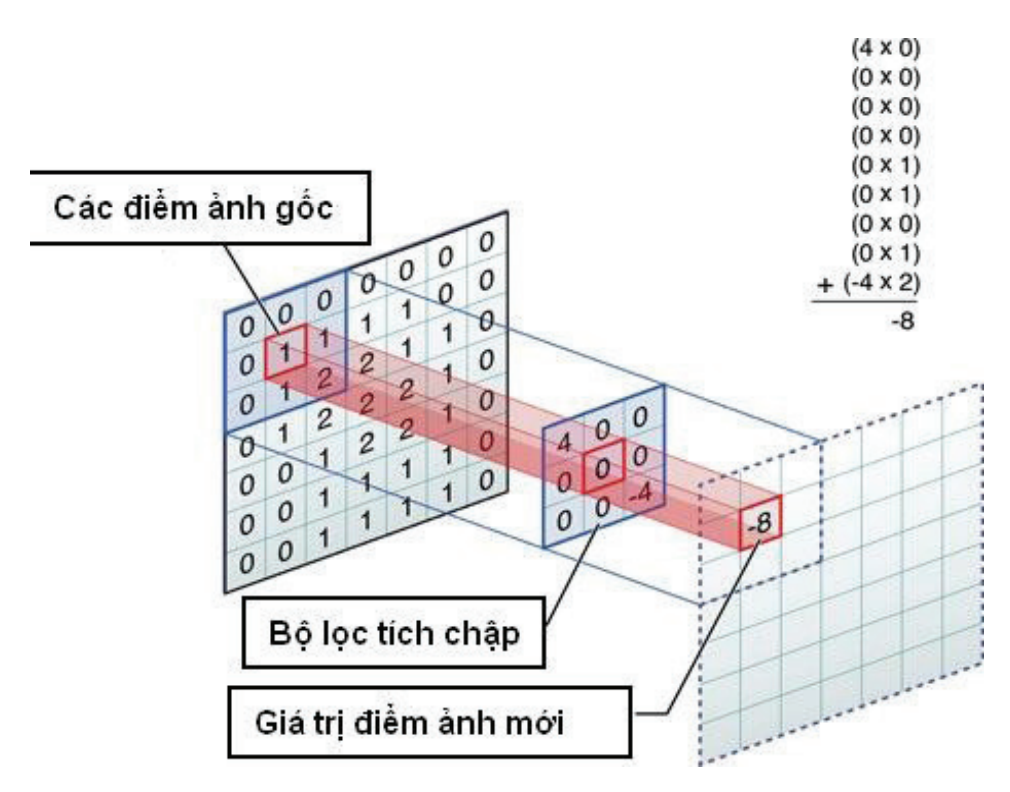
Hình 2.1 trình bày một kiến trúc mạng CNN, các lớp cơ bản trong một mạng CNN bao gồm: lớp tích chập (Convolutional); lớp kích hoạt phi tuyến ReLU (Rectified Linear Unit); lớp lấy mẫu (Pooling); lớp kết nối đầy đủ (Fully connected) được thay đổi về số lượng và cách sắp xếp để tạo ra các mô hình huấn luyện phù hợp cho từng bài toán khác nhau. Các lớp tích chập (Convolutional), kích hoạt phi tuyến ReLU và lấy mẫu (Pooling) Các lớp kết nối đầy đủ (Fully connected) Phân loại Hình 1. Kiến trúc cơ bản của một mạng CNN.

Lớp tích chập: đây là thành phần quan trọng nhất trong mạng CNN, thể hiện sự liên kết cục bộ thay vì kết nối toàn bộ các điểm ảnh. Các liên kết cục bộ được tính toán bằng phép tích chập giữa các giá trị điểm ảnh trong một vùng ảnh cục bộ với các bộ lọc filters có kích thước nhỏ.



Hình 2.1 Kiến trúc cơ bản của mạng CNN

Lớp tích chập: đây là thành phần quan trọng nhất trong mạng CNN, thể hiện sự liên kết cục bộ thay vì kết nối toàn bộ các điểm ảnh. Các liên kết cục bộ được tính toán bằng phép tích chập giữa các giá trị điểm ảnh trong một vùng ảnh cục bộ với các bộ lọc filters có kích thước nhỏ.



Hình 2.2 Bộ lọc tích chập được sử dụng trên ma trận điểm ảnh

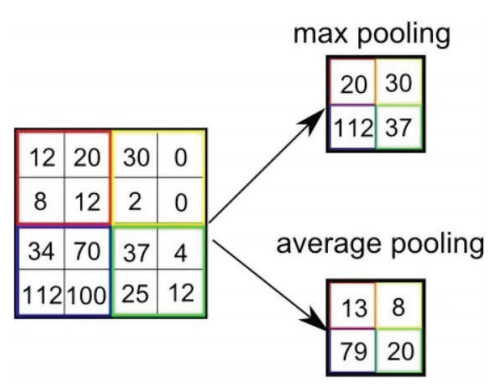
Bộ lọc tích chập được sử dụng trên ma trận điểm ảnh. Trong hình 2.2, bộ lọc được sử dụng là một ma trận có kích thước 3x3, bộ lọc này dịch chuyển lần lượt qua từng vùng ảnh đến khi hoàn thành quét toàn bộ bức ảnh, tạo ra một bức ảnh mới có kích thước nhỏ hơn hoặc bằng với kích thước ảnh đầu vào. Kích thước này được quyết định tùy theo kích thước các khoảng trắng được thêm ở viền bức ảnh gốc và được tính theo công thức sau:

Trong đó: O: kích thước ảnh đầu ra; i: kích thước ảnh đầu vào; p: kích thước khoảng trắng phía ngoài viền của ảnh gốc; k: kích thước bộ lọc; s: bước trượt của bộ lọc. Như vậy, sau khi đưa một bức ảnh đầu vào cho lớp tích chập nhận được kết quả đầu ra là một loạt ảnh tương ứng với các bộ lọc đã được sử dụng để thực hiện phép tích chập. Các trọng số của các bộ lọc này được khởi tạo ngẫu nhiên trong lần đầu tiên và sẽ được cập nhật trong quá trình huấn luyện. - Lớp kích hoạt phi tuyến ReLU: được xây dựng để đảm bảo tính phi tuyến của mô hình huấn luyện sau khi đã thực hiện một loạt các phép tính toán tuyến tính qua các lớp tích chập. Lớp kích hoạt phi tuyến sử dụng các hàm kích hoạt phi tuyến như ReLU hoặc sigmoid, tanh… để giới hạn phạm vi biên độ cho phép của giá trị đầu ra. Trong số các hàm kích hoạt này, hàm ReLU được chọn do cài đặt đơn giản, tốc độ xử lý nhanh mà vẫn đảm bảo được tính toán hiệu quả. Phép tính toán của hàm ReLU chỉ đơn giản là chuyển tất cả các giá trị âm thành giá trị 0. Lớp ReLU được áp dụng ngay phía sau lớp tích chập, với đầu ra là một ảnh mới có kích thước giống với ảnh đầu vào, các giá trị điểm ảnh cũng hoàn toàn tương tự, trừ các giá trị âm đã bị loại bỏ.

𝑓(𝑥) = 𝑚𝑎𝑥(0, 𝑥) (2.2)

Lớp lấy mẫu: được đặt sau lớp tích chập và lớp ReLU để làm giảm kích thước ảnh đầu ra trong khi vẫn giữ được các thông tin quan trọng của ảnh đầu vào. Việc giảm kích thước dữ liệu có tác dụng làm giảm được số lượng tham số cũng như tăng hiệu quả tính toán. Lớp lấy mẫu cũng sử dụng một cửa sổ trượt để quét toàn bộ các vùng trong ảnh như lớp tích chập, và thực hiện phép lấy mẫu thay vì phép tích chập, sẽ chọn lưu lại một giá trị duy nhất đại diện cho toàn bộ thông tin của vùng ảnh đó.

Hình 3 thể hiện các phương thức lấy mẫu thường được sử dụng nhất hiện nay, đó là Max Pooling (lấy giá trị điểm ảnh lớn nhất) và Avarage Pooling (lấy giá trị trung bình của các điểm ảnh trong vùng ảnh cục bộ).



Hình 2.3 Phương thức Avarage Pooling và Max Pooling

Như vậy, với mỗi ảnh đầu vào được đưa qua lấy mẫu sẽ thu được một ảnh đầu ra tương ứng, có kích thước giảm xuống đáng kể nhưng vẫn giữ được các đặc trưng cần thiết cho quá trình tính toán và nhận dạng.

Lớp kết nối đầy đủ: được thiết kế tương tự như trong mạng nơ ron truyền thống, tất cả các điểm ảnh được kết nối đầy đủ với node trong lớp tiếp theo.

So với mạng nơ ron truyền thống, các ảnh đầu vào của lớp này đã có kích thước được giảm bớt rất nhiều, đồng thời vẫn đảm bảo các thông tin quan trọng của ảnh cho việc nhận dạng. Do vậy, việc tính toán nhận dạng sử dụng mô hình truyền thẳng đã không còn phức tạp và tốn nhiều thời gian như trong mạng nơ ron truyền thống.

# xây dựng chương trình

## xây dựng mô hình

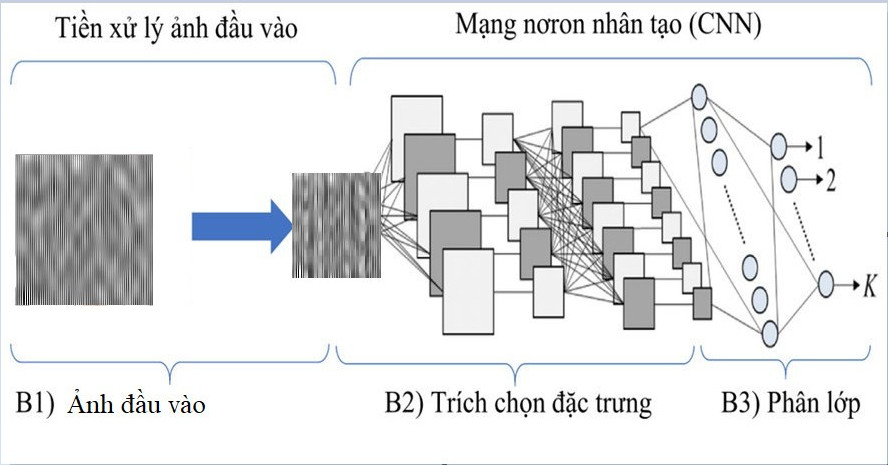
**Mô tả các bước xây dựng mạng**

Mô hình nhận dạng được chia thành 3 bước chính (Hình 3.1), bao gồm:

Bước 1: Load tập dữ liệu từ Kaggle .

Bước 2: Huấn luyện dữ liệu đưa vào và trích chọn các đặc trưng.

Bước 3: Phân loại ảnh dựa trên đặc trưng được trích chọn và đưa ra kết quả.

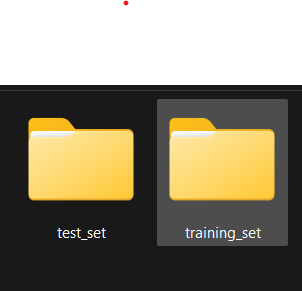


Hình 3.1 Sơ đồ quy trình của mô hình phân loại tín hiệu radar

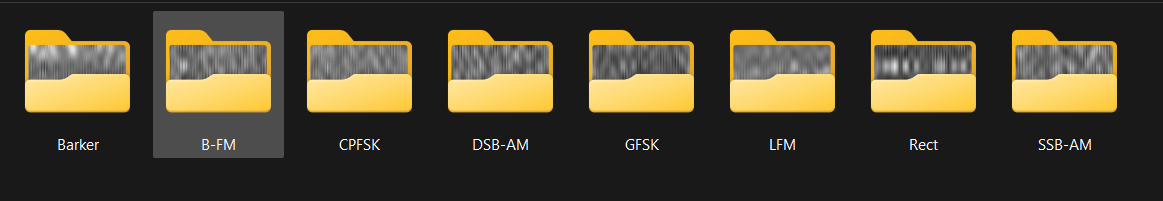
## Các bước thực hiện

### Load tập dữ liệu từ Kaggle

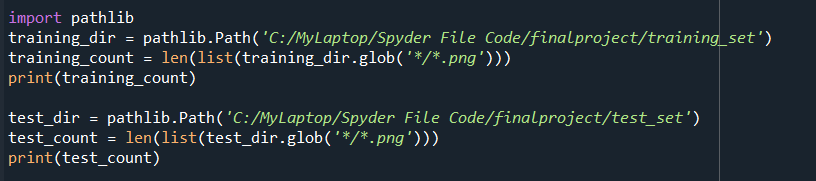
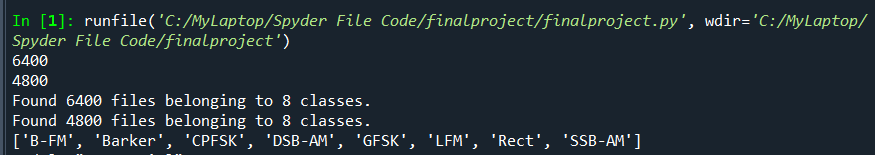
* Gồm 2 File : train\_set và test\_set được tải từ Kaggle



Hình 3.2 Hình ảnh 2 file test\_set và training\_set

**-** Trong mỗi 1 file là train\_set hay test\_set đều chứa 8 file là các hình ảnh từ tín hiệu radar đã được gán nhãn

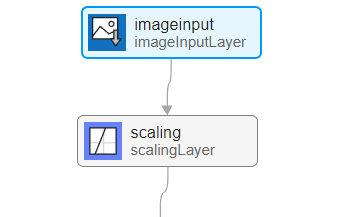
Hình 3.3 Hình ảnh các file tín hiệu được gán nhãn

 - Load dữ liệu trên file code

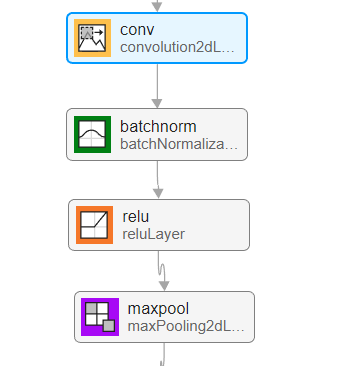
- Kết quả in ra ở màn hình Console cho thấy tệp train gồm 6400 mẫu và tệp test gồm 4800 mẫu , được gán nhãn phân thành 8 loại trong hình trên .

### Huấn luyện dữ liệu đưa vào

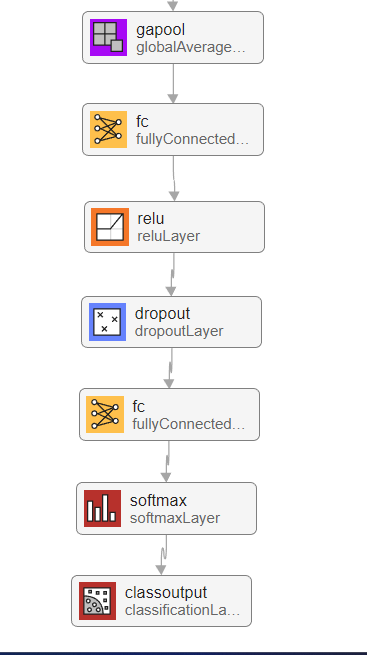
- Mô hình mạng CNN được thiết kế bằng Matlab như sau :

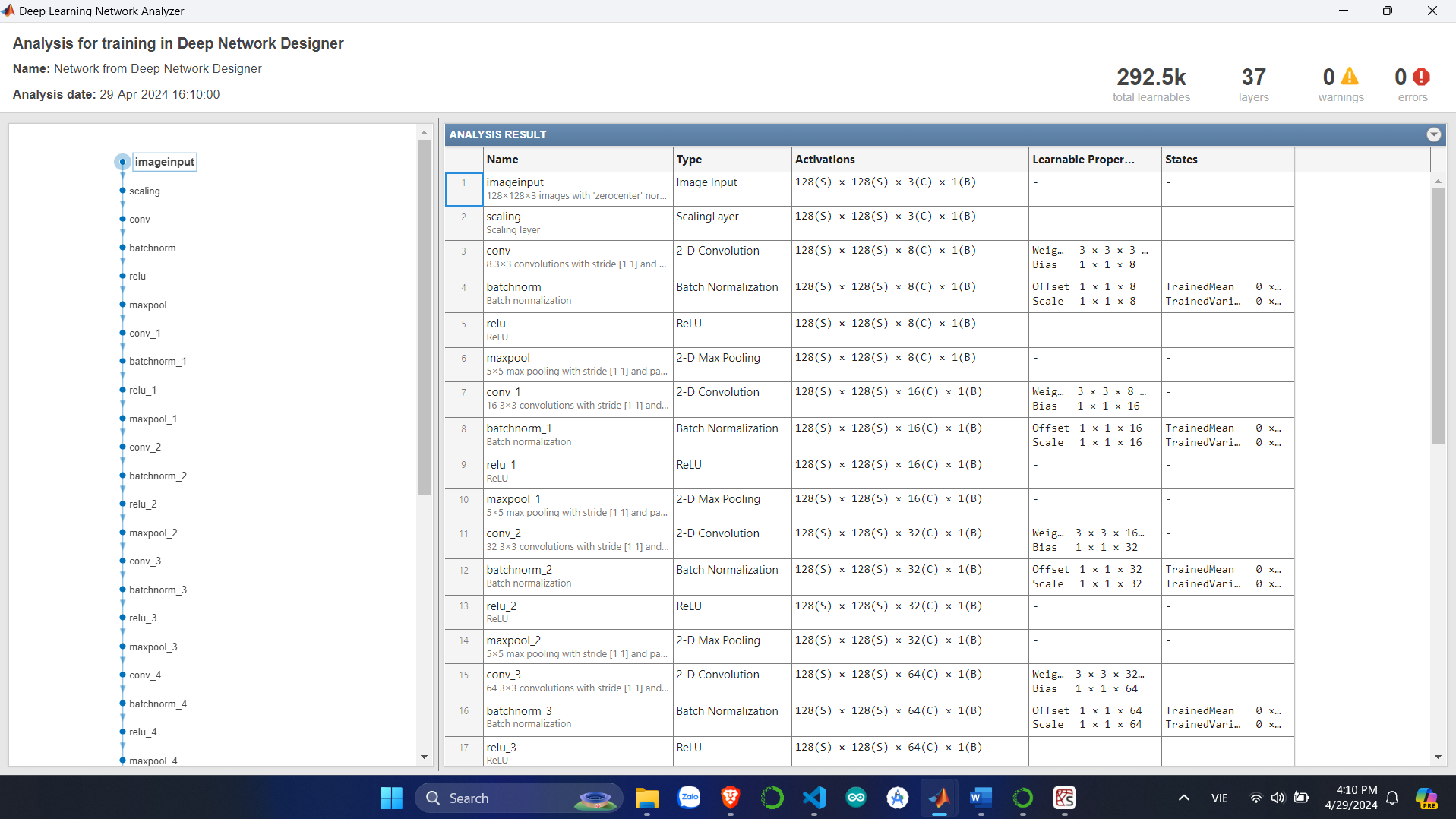
* Đây là lớp tiền xử lý dữ liệu

* Tiếp theo là các lớp tích chập 3x3 với số lớp liên tiếp là: 8,16,32,64,64,128,128



* Quá trình xử lý dữ liệu sau khi đã hoàn thành một quá trình xử lý chính.



- Phân tích tổng số lượng trọng số trong mạng bằng Matlab

Hình 3.4 Bảng phân tích mạng CNN được design trên Matlab

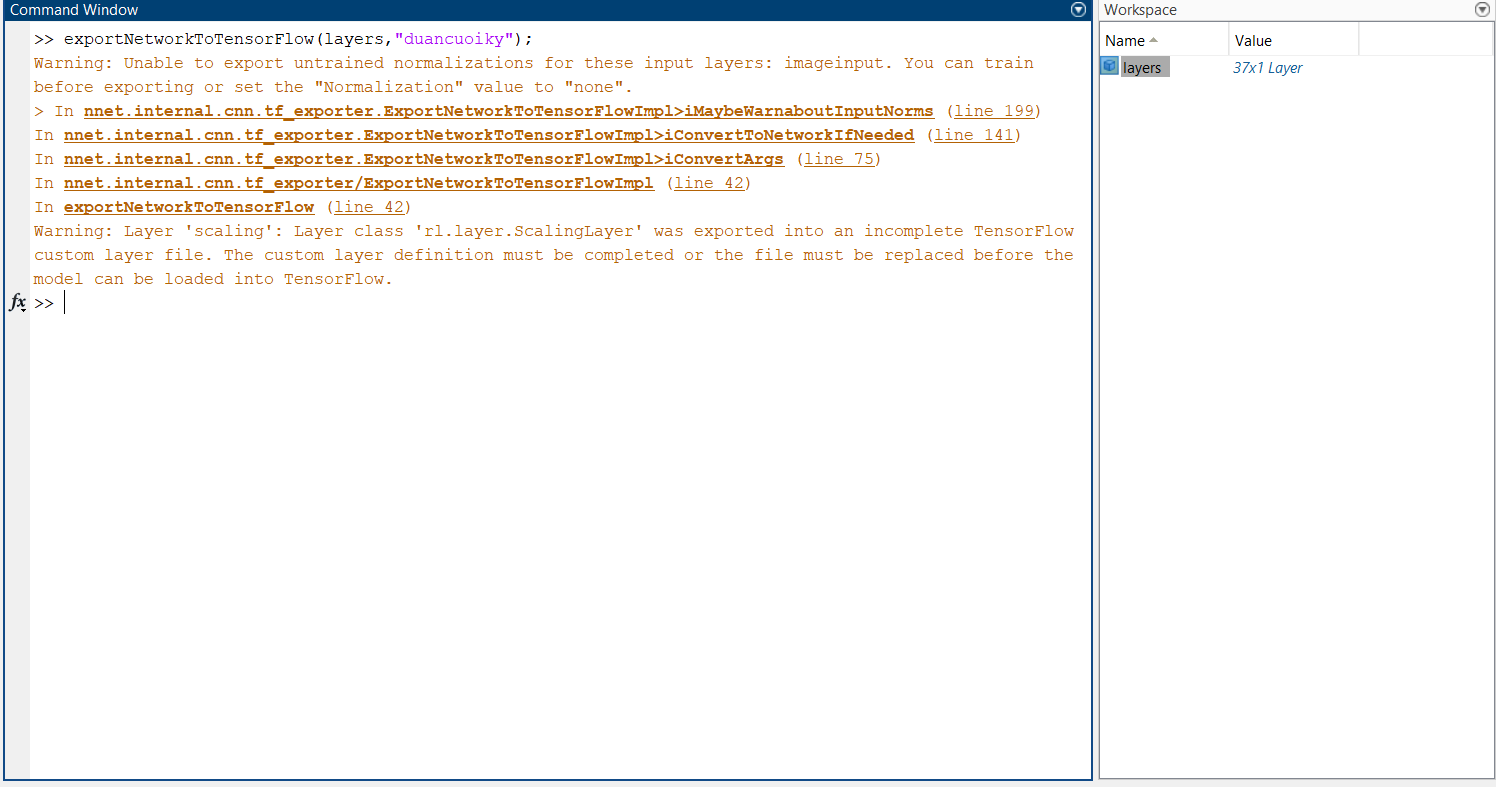
Như hình vẽ , Matlab đã thống kê được mạng CNN trên đã sử dụng tổng cộng 292.5K trọng số. Cụ thể như sau :

|  |  |
| --- | --- |
| Lớp | Tổng trọng số |
| Conv | 3x3x3x8 + 1x1x8 =224 |
| Batchnorm | 1x1x8 + 1x1x8 = 16 |
| Conv1 | 3x3x8x16 + 1x1x16 =1168 |
| Batchnorm1 | 1x1x16 + 1x1x16 = 32 |
| Conv2 | 3x3x16x32 + 1x1x32 = 4640 |
| Batchnorm2 | 1x1x32 + 1x1x32 = 64 |
| Conv3 | 3x3x32x64 + 1x1x64 = 18496 |
| Batchnorm3 | 1x1x64 + 1x1x64 = 128 |
| Conv4 | 3x3x64x64 + 1x1x64 = 36928 |
| Batchnorm4 | 1x1x64 + 1x1x64 = 128 |
| Conv5 | 3x3x64x128 + 1x1x128 = 73856 |
| Batchnorm5 | 1x1x128 + 1x1x128 = 256 |
| Conv6 | 3x3x128x128 + 1x1x128 = 147584 |
| Batchnorm6 | 1x1x128 + 1x1x128 = 256 |
| Fc | 64x128 + 64x1 = 8256 |
| Fc\_1 | 8x64 + 8x1 =520 |

- Phân tích các lớp trong mạng đã thiết kế

1. **Rescaling**: Lớp này điều chỉnh tỷ lệ giá trị pixel của ảnh về khoảng từ 0 đến 1 bằng cách chia mỗi giá trị pixel cho 255. Điều này giúp cải thiện hiệu suất của mô hình bằng cách làm cho dữ liệu đầu vào có dạng phân phối thích hợp hơn.
2. **Conv2D**: Lớp convolution 2D thực hiện phép tích chập trên đầu vào, sử dụng một tập hợp các bộ lọc (kernels) để trích xuất các đặc trưng từ ảnh đầu vào. Trong mạng này, lớp Conv2D được sử dụng với các kernel có kích thước (3, 3) và hàm kích hoạt là ReLU.
3. **BatchNormalization**: Lớp này chuẩn hóa đầu vào bằng cách áp dụng một phép chuẩn hóa batch, giúp giảm thiểu hiện tượng biến thiên (covariate shift) và tăng cường ổn định và tốc độ học của mô hình.
4. **Activation (ReLU)**: Lớp kích hoạt ReLU thực hiện hàm kích hoạt ReLU (Rectified Linear Unit) trên đầu ra từ các lớp convolution và batch normalization, giúp giải quyết vấn đề về tính toán phi tuyến tính và tăng tính phi tuyến tính của mô hình.
5. **MaxPooling2D**: Lớp này thực hiện phép gộp giảm kích thước của đầu ra từ các lớp convolution bằng cách chọn giá trị lớn nhất từ một vùng cụ thể trong đầu vào.
6. **GlobalAveragePooling2D**: Lớp này thực hiện phép gộp trung bình trên toàn bộ các đặc trưng trong mỗi kênh của đầu ra từ các lớp convolution trước đó, giảm kích thước của tensor đầu ra xuống một vector duy nhất.
7. **Dense**: Lớp này thực hiện phép kết nối đầy đủ giữa các neuron, trong đó mỗi neuron trong lớp này kết nối với tất cả các neuron trong lớp trước đó. Lớp Dense cuối cùng có hàm kích hoạt softmax để tạo ra xác suất dự đoán cho các lớp đầu ra.
8. **Dropout**: Lớp này áp dụng kỹ thuật dropout để ngẫu nhiên "tắt" một số lượng neuron trong quá trình huấn luyện, giúp tránh overfitting bằng cách ngăn chặn sự phụ thuộc quá mức vào một số neuron cụ thể. Trong mạng này, dropout được sử dụng với tỷ lệ dropout là 0.2.

- Chuyển sang code python và dùng thư viện Keras

 Chuyển đổi mô hình mạng CNN đã xây dựng ở trên từ Matlab sang tensorflow bằng cách sử dụng hàm Matlab:  [exportNetworkToTensorFlow](https://www.mathworks.com/help/deeplearning/ref/exportnetworktotensorflow.html).

Hình 3.5 Chuyển đổi mô hình từ Matlab sang file code python

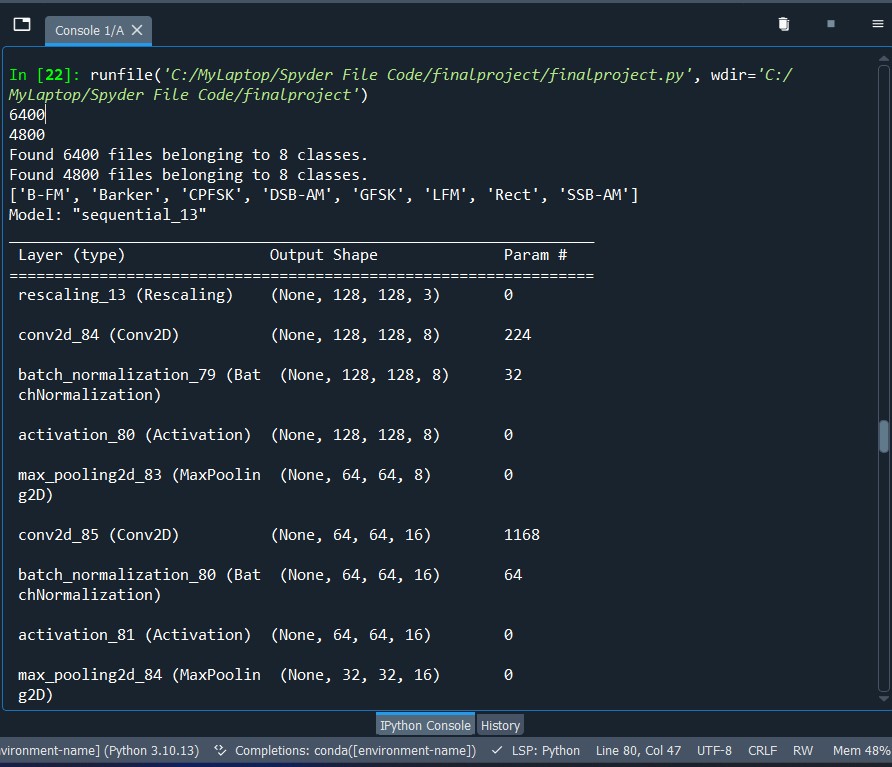
### Dữ liệu đầu ra được phân loại theo các lớp như sau

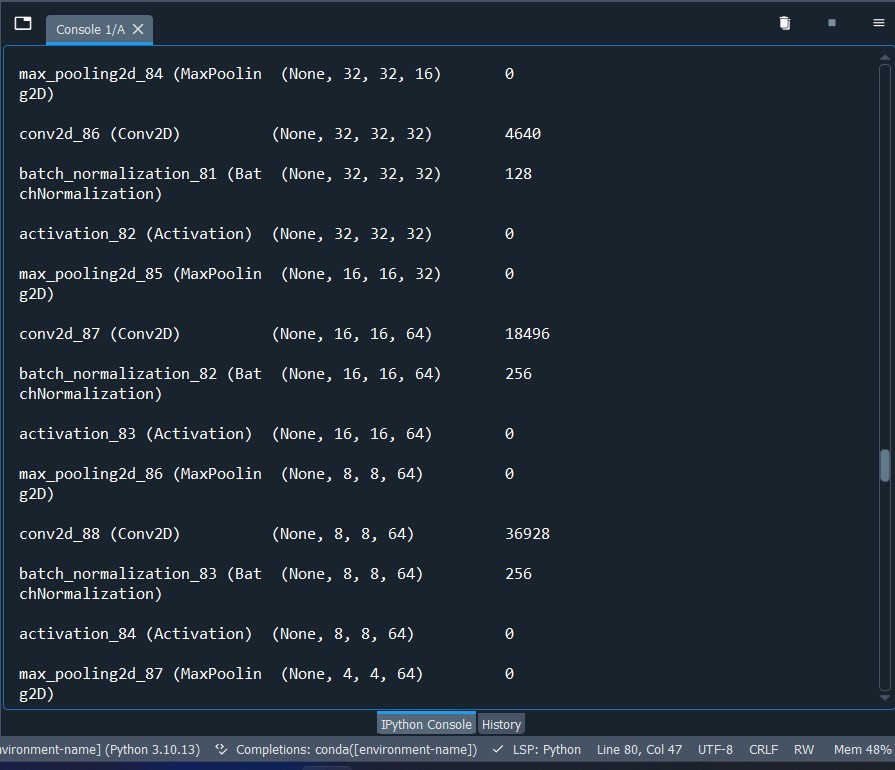
|  |  |  |
| --- | --- | --- |
| STT | Ảnh đầu vào | Tên phân loại |
| 1      2    3  4 |  | CPFSK  LFM  SSB-AM  Rect |
| 5      6  7  8 |  | GFSK  DSB-AM  B-FM    Barker |

Hình 3.6 Bảng phân tích đầu ra phân loại tín hiệu

# Kết quả thí nghiệm

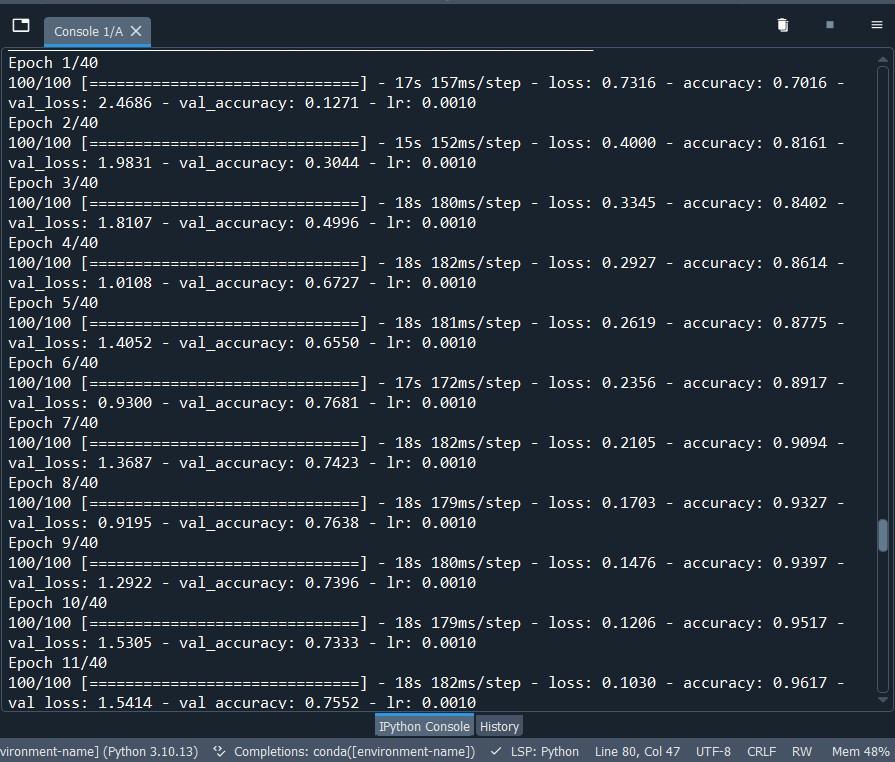
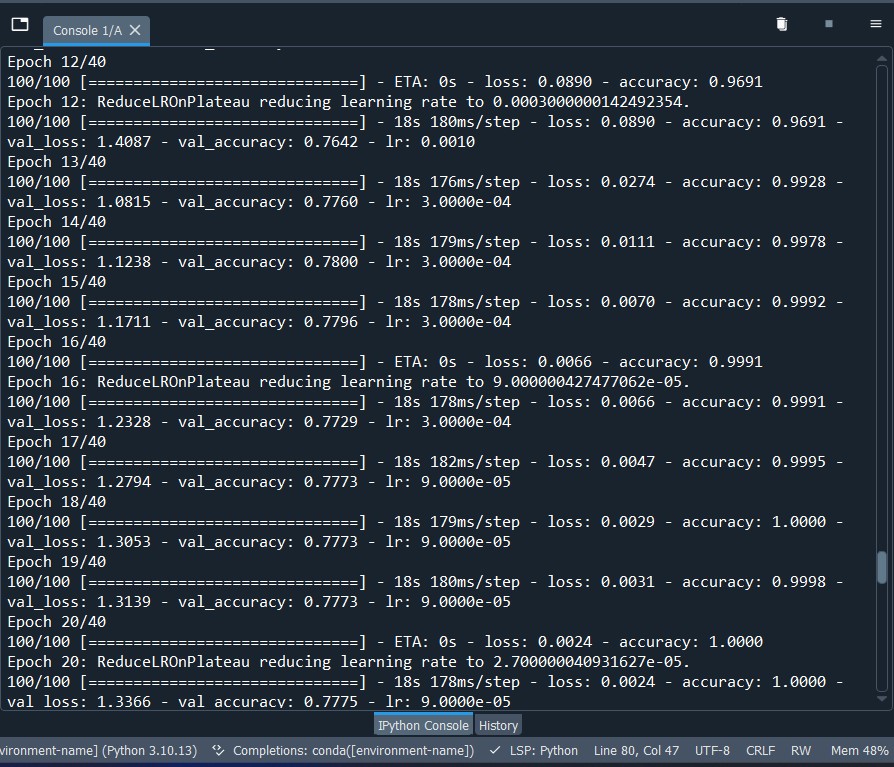
## Kết quả sau khi huấn luyện

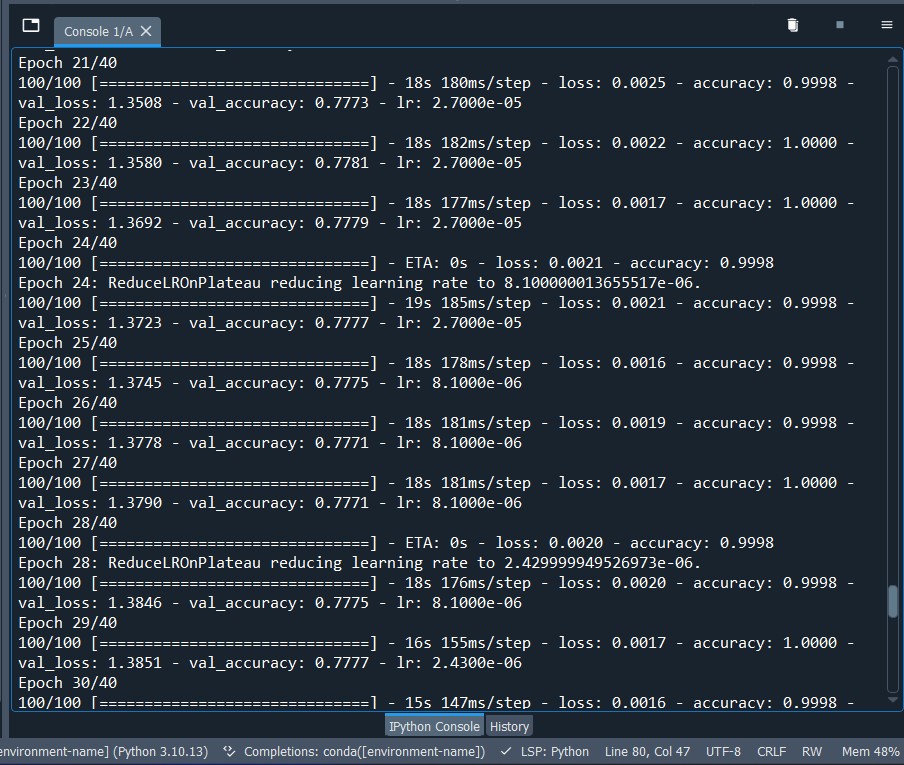
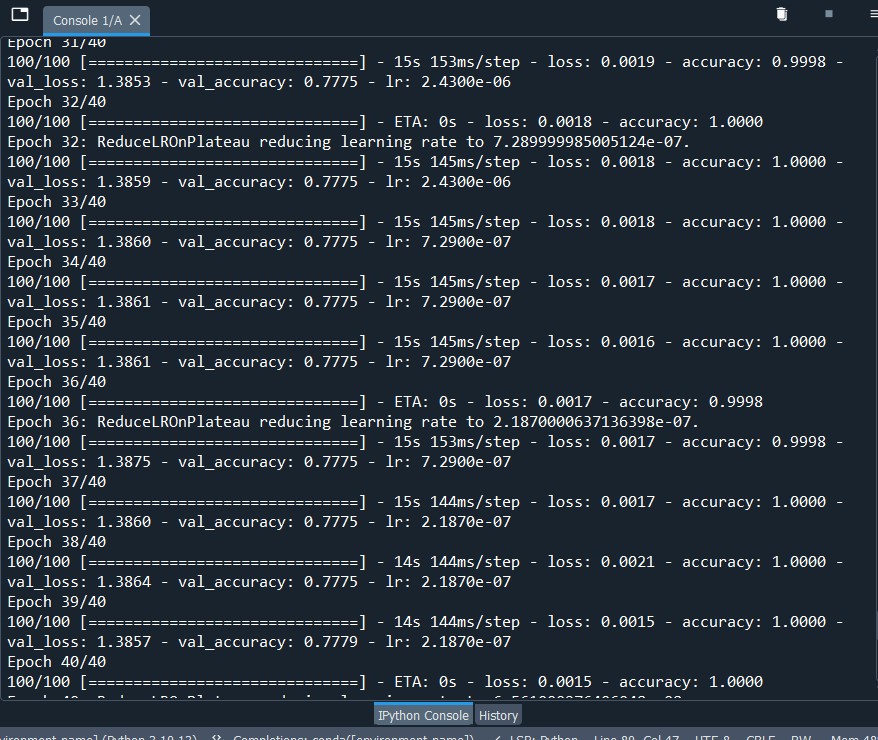
Quá trình huấn luyện tập dữ liệu sau khi chạy code Python trên spyder



Hình 4.1 Tổng trọng số được ước tính trong mạng

**- Nhận xét :** Con số khá giống những gì đã phân tích ở phần trước



 Hình 4.2 Độ chính và giá trị hàm lỗi theo từng epoch

**- Nhận xét:**

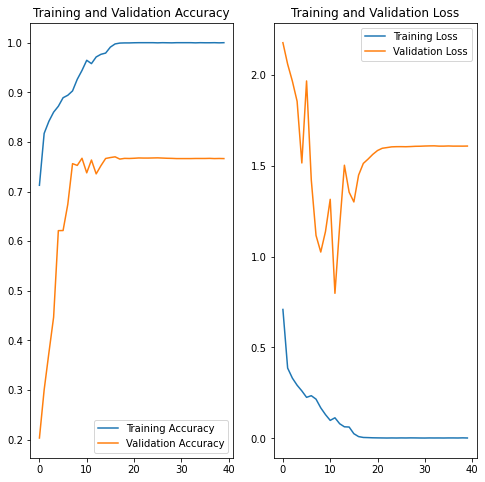
* Giá trị của hàm lỗi giảm dần theo từng epoch. Giá trị của hàm lỗi ở epoch 1 là 0.73, giá trị hàm lỗi ở epoch 10 giảm xuống còn 0.1206 và đến epoch 40 là 0.015 .
* Độ chính xác tăng dần qua từng epoch, tăng lên tối đa 100% ở epoch thứ 18 và duy trì mức ngưỡng 99%-100% đến epoch 40 .

**Với yêu cầu là chạy 3 lần code trên và ghi lại giá trị accuracy cho mỗi epoch ta có bảng tổng hợp sau đây:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Lần chạy**  **Epoch** | **Lần 1** | **Lần 2** | **Lần 3** |
| **1** | **0.7016** | **0.7127** | **0.7048** |
| **2** | **0.8161** | **0.8172** | **0.8148** |
| **3** | **0.8402** | **0.8416** | **0.8372** |
| **4** | **0.8614** | **0.8600** | **0.8587** |
| **5** | **0.8775** | **0.8719** | **0.8648** |
| **6** | **0.8917** | **0.8891** | **0.8756** |
| **7** | **0.9094** | **0.8941** | **0.8963** |
| **8** | **0.9327** | **0.9028** | **0.9408** |
| **9** | **0.9397** | **0.9262** | **0.9586** |
| **10** | **0.9517** | **0.9441** | **0.9744** |
| **11** | **0.9617** | **0.9645** | **0.9912** |
| **12** | **0.9691** | **0.9778** | **0.9956** |
| **13** | **0.9928** | **0.9712** | **0.9983** |
| **14** | **0.9978** | **0.9766** | **0.9987** |
| **15** | **0.9992** | **0.9791** | **0.9992** |
| **16** | **0.9991** | **0.9911** | **0.9989** |
| **17** | **0.9995** | **0.9975** | **0.9984** |
| **18** | **1.0000** | **0.9992** | **0.9992** |
| **19** | **0.9998** | **0.9995** | **0.9991** |
| **20** | **1.0000** | **0.9995** | **0.9991** |
| **21** | **0.9998** | **0.9998** | **0.9994** |
| **22** | **1.0000** | **1.0000** | **0.9986** |
| **23** | **1.0000** | **1.0000** | **0.9997** |
| **24** | **0.9998** | **1.0000** | **0.9995** |
| **25** | **0.9998** | **1.0000** | **0.9994** |
| **26** | **0.9998** | **0.9997** | **1.0000** |
| **27** | **1.0000** | **1.0000** | **1.0000** |
| **28** | **1.0000** | **0.9998** | **0.9991** |
| **29** | **1.0000** | **0.9997** | **0.9996** |
| **30** | **0.9998** | **1.0000** | **1.0000** |
| **31** | **0.9998** | **1.0000** | **0.9989** |
| **32** | **1.0000** | **1.0000** | **1.0000** |
| **33** | **1.0000** | **1.0000** | **1.0000** |
| **34** | **1.0000** | **0.9997** | **1.0000** |
| **35** | **1.0000** | **1.0000** | **1.0000** |
| **36** | **0.9998** | **0.9998** | **0.9996** |
| **37** | **1.0000** | **0.9998** | **0.9997** |
| **38** | **1.0000** | **1.0000** | **1.0000** |
| **39** | **1.0000** | **0.9997** | **1.0000** |
| **40** | **1.0000** | **1.0000** | **1.0000** |
| **Tổng** | **38.6396** | **38.5132** | **38.6972** |

**Accuracy Final = ( 38.6396 + 38.5132 + 38.6972 ) / 120 = 0.9654**

**Vậy ta có thể đưa ra nhận xét tổng kết rằng độ chính xác của mạng CNN này là 96,54 %**

 **4.2 ĐỒ THỊ SỰ BIẾN THIÊN CỦA HÀM LOSS VÀ GIÁ TRỊ ACCURACY**

Hình 4.3 Đồ thị Training và Validation Accuracy/Loss

**Nhận xét**

- Training Accuracy và Validation Accuracy:

+ Đường Training Accuracy tăng dần và cuối cùng đạt mức gần 1, cho thấy mô hình học được tốt trên dữ liệu huấn luyện.

+ Đường Validation Accuracy tăng lên đáng kể ở đầu và sau đó giữ ổn định ở một mức độ cao. Sự ổn định này cho thấy mô hình đang hoạt động tốt trên dữ liệu kiểm tra, không có dấu hiệu của overfitting.

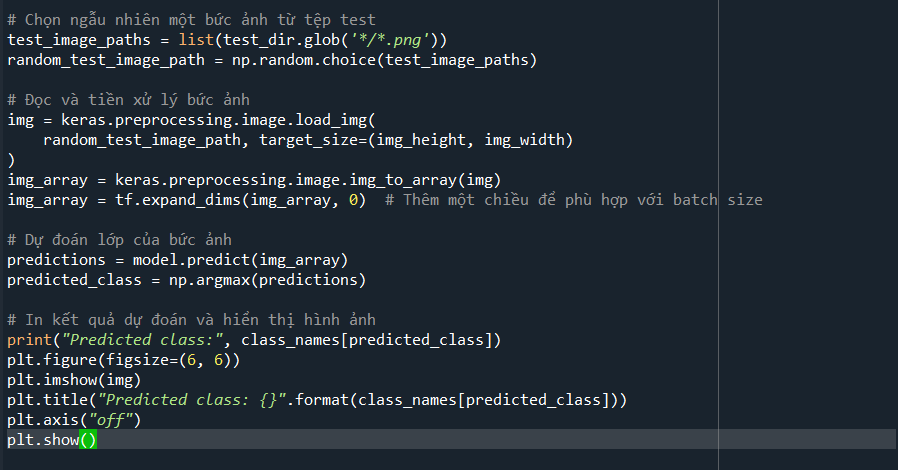
- Training Loss và Validation Loss:

+ Đường Training Loss giảm dần và cuối cùng hội tụ ở mức thấp, cho thấy mô hình đang học được từ dữ liệu huấn luyện.

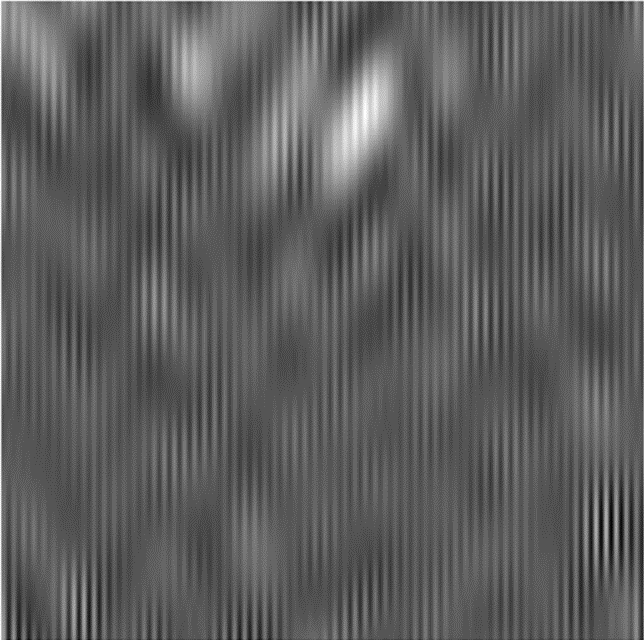
+ Đường Validation Loss giảm lên đầu và sau đó bắt đầu tăng trở lại.

=> Tổng quan, mô hình có vẻ hoạt động tốt trên cả dữ liệu huấn luyện và kiểm tra, với độ chính xác và sự mất mát ổn định.

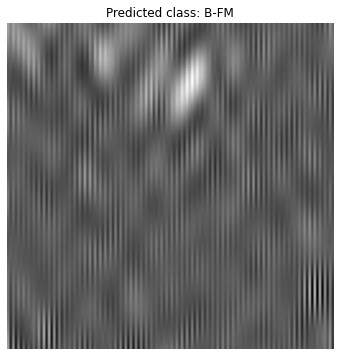
## 4.3 Đưa hình ảnh TỪ TỆP TEST ĐỂ PHÂN LOẠI

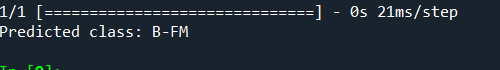
- Đoạn code dùng model đã xây dựng để phân loại ảnh từ tệp test\_set

- So sánh kết quả trước và sau phân loại tín hiệu :



Hình 4.4 Hình ảnh trước khi nhận diện (ảnh từ file B-FM)





Hình 4.5 Hình ảnh sau khi phân loại (B-FM)

**Nhận xét:**

- Kết quả cho ra là chính xác đồng thời khi test với nhiều trường hợp khác thì độ chính xác cũng rất cao.

# KẾT LUẬN

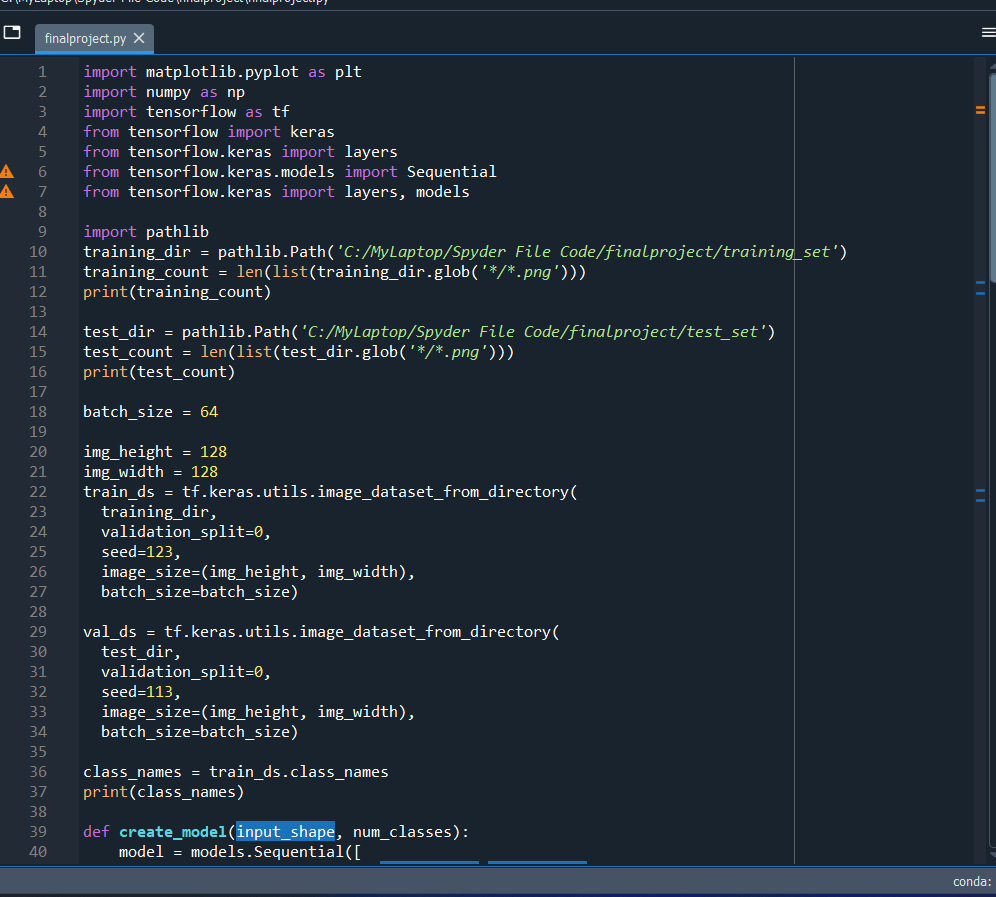
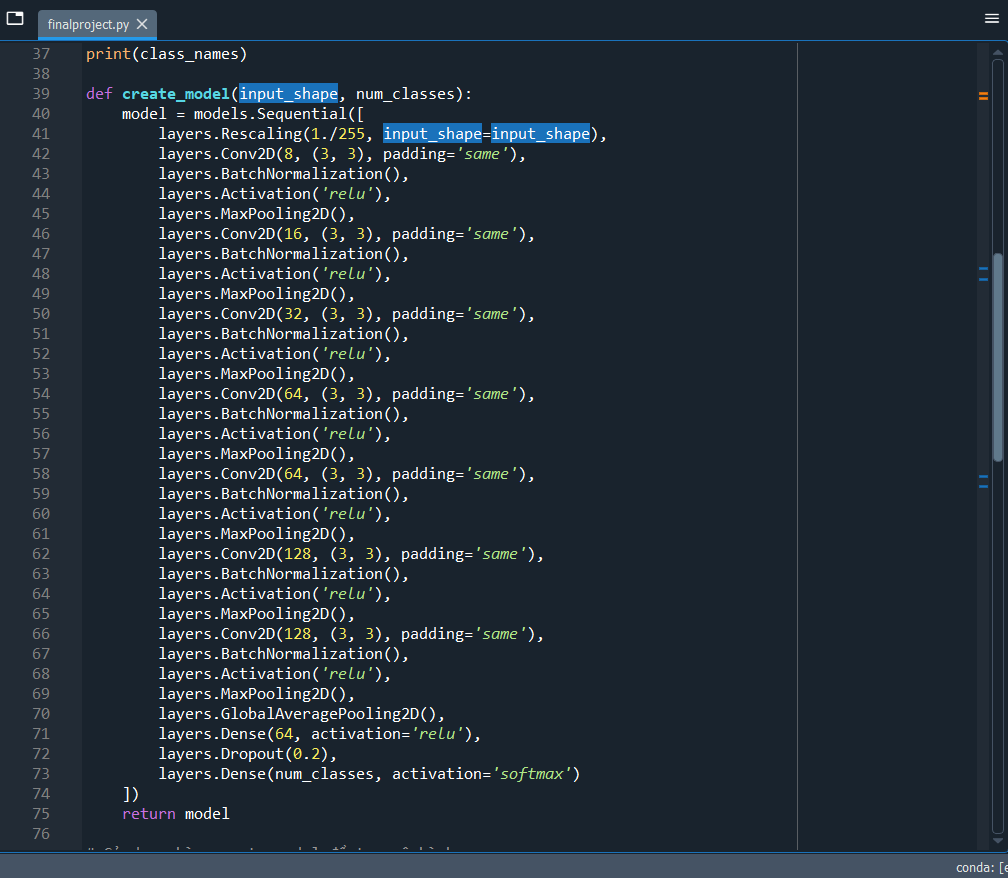
Trong báo cáo này, chúng tôi đã đề xuất một mô hình dựa trên mạng nơron tích chập (CNN) để phân loại các dạng tín hiệu radar . Mô hình này có 6 lớp nơron tích chập (Convolution) và 2 lớp nơron liên kết đầy đủ (Fully Connected), tổng số tham số là khoảng hơn 293 ngàn.

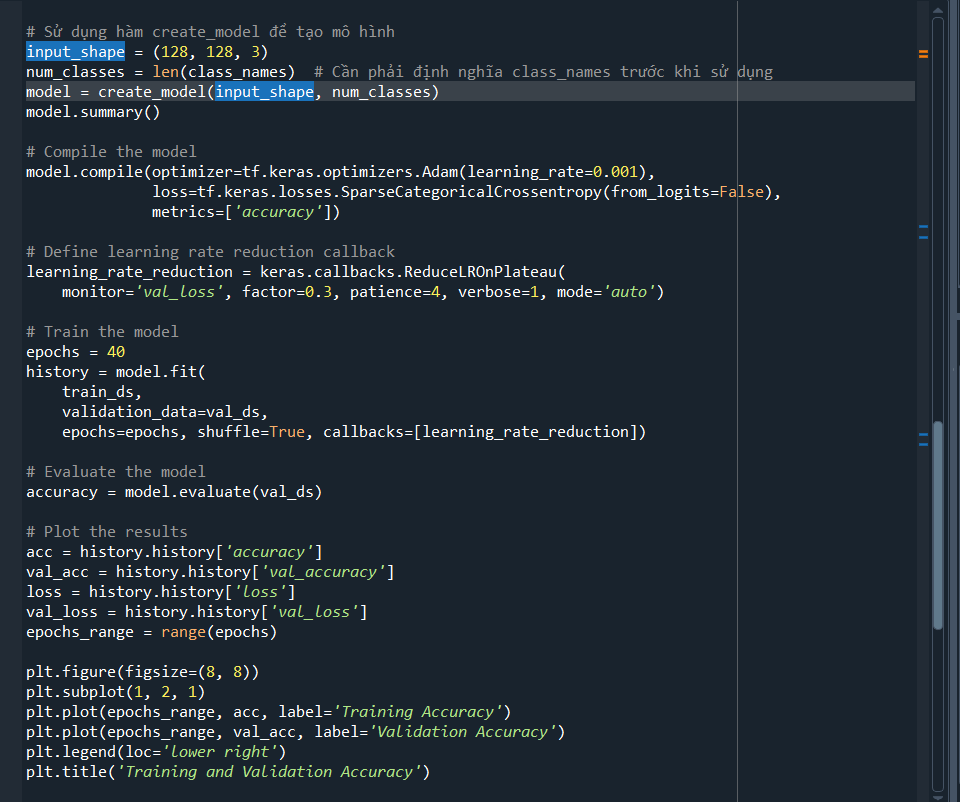
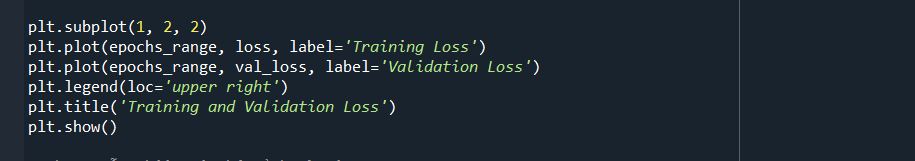
Như vậy, có thể khẳng định mô hình của chúng tôi có độ phức tạp ở mức vừa phải, phù hợp với các hệ thống xử lý ở mức trung bình và đem lại tiềm năng khả thi trong ứng dụng thực tiễn.

Mặc dù độ phức tạp của mô hình ở mức thấp so với các mô hình khác, nhưng kết quả thử nghiệm cho thấy tính hiệu quả của phân lớp khá cao. Hiện nay do điều kiện tính toán nên chỉ áp dụng số lần huấn luyện còn thấp, nếu được huấn luyện ở mức độ sâu hơn thì kỳ vọng sẽ đem lại kết quả cao hơn nữa.

Để phát triển thêm cho mô hình, chúng tôi sẽ tìm hiểu và thiết kế một hệ thống thu thập dữ liệu hình ảnh để tạo bộ dữ liệu huấn luyện đa dạng cho mô hình, từ đó xây dựng một ứng dụng cho bài toán thực tiễn như hệ thống đánh giá và giám sát tín hiệu radar ở trạm thu ,trong lĩnh vực quân sự, các hệ thống radar được sử dụng để phát hiện, theo dõi và phân loại các mục tiêu như máy bay, tàu thuyền, và phương tiện di động khác hoặc trong y tế và cứu thương, radar y tế có thể được sử dụng để phát hiện các biểu hiện bất thường trong cơ thể con người .

# PHỤ LỤC

Chương trình được thực hiện trên môi trường Spyder (Anaconda) với chương trình như sau:

+ File code : được gửi kèm theo file pdf báo cáo

**TÀI LIỆU THAM KHẢO**

[1] PGS. TS Trương Ngọc Sơn, *Trí tuệ nhân tạo cơ sở và ứng dụng,* Đại học quốc gia TPHCM, 2020.

[2] Tài liệu online.

[3] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, and B. Yu, “Recent advances in convolutional neural network acceleration,” *Neurocomputing*, vol. 323, pp. 0–38, 2019, doi: 10.1016/j.neucom.2018.09.038.